International Journal of Computer Vision (2022) 130:669–695
https://doi.org/10.1007/s11263-021-01551-y

Learning Scene Dynamics from Point Cloud Sequences

Pan He[1] · **Patrick Emami**[1] · **Sanjay Ranka**[1] · **Anand Rangarajan**[1]

Received: 19 February 2021 / Accepted: 26 October 2021 / Published online: 23 January 2022

Abstract
Understanding 3D scenes is a critical prerequisite for autonomous agents. Recently, LiDAR and other sensors have made large amounts of data available in the form of temporal sequences of point cloud frames. In this work, we propose a novel problem—*sequential* scene flow estimation (SSFE)—that aims to predict 3D scene flow for all pairs of point clouds in a given sequence. This is unlike the previously studied problem of scene flow estimation which focuses on two frames. We introduce the SPCM-Net architecture, which solves this problem by computing multi-scale spatiotemporal correlations between neighboring point clouds and then aggregating the correlation across time with an order-invariant recurrent unit. Our experimental evaluation confirms that recurrent processing of point cloud sequences results in significantly better SSFE compared to using only two frames. Additionally, we demonstrate that this approach can be effectively modified for sequential point cloud forecasting (SPF), a related problem that demands forecasting future point cloud frames. Our experimental results are evaluated using a new benchmark for both SSFE and SPF consisting of synthetic and real datasets. Previously, datasets for scene flow estimation have been limited to two frames. We provide non-trivial extensions to these datasets for multi-frame estimation and prediction. Due to the difficulty of obtaining ground truth motion for real-world datasets, we use self-supervised training and evaluation metrics. We believe that this benchmark will be pivotal to future research in this area. All code for benchmark and models will be made accessible at (https://github.com/BestSonny/SPCM).

Keywords 3D deep learning · Scene dynamics · Point cloud processing · Scene flow estimation · Spatiotemporal learning · Self-supervised learning

1 Introduction

Autonomous agents need to understand 3D environments to ensure safe planning and navigation. A critical step is to perceive and predict the actions of entities such as vehicles, pedestrians, and cyclists. This requires learning rich embeddings of recorded data. Among various 3D geometric data representations, point clouds can accurately preserve the original geometric information in 3D environments with less information loss compared to other representations such as voxels (Maturana and Scherer 2015), or projected images (Wu et al. 2018). This has led to the explosive growth in developing point cloud-based deep architectures, as evidenced in (Landrieu and Simonovsky 2018; Liu et al. 2019b; Wang et al. 2019b; Qi et al. 2017a, b; Su et al. 2018) and other tasks such as 3D semantic and instance segmentation (Hou et al. 2019; Pham et al. 2019; Wang et al. 2018b; Yi et al. 2019; Zhao and Tao 2020), and 3D object detection (Gwak et al. 2020; Qi et al. 2018, 2020; Shi et al. 2020; Tang et al. 2020; Yin et al. 2021). However, the community has paid less attention to the processing of dynamic point clouds in a spatiotemporal scene. Unlike grid-based RGB images or videos, dynamic point clouds are unordered and irregular in the spatial dimension and can change drastically in the temporal dimension. The spatiotemporal processing of raw point cloud sequences remains an open challenge.

Communicated by Zuzana Kukelova.

✉ Pan He
pan.he@ufl.edu

✉ Anand Rangarajan
anand@cise.ufl.edu

Patrick Emami
pemami@ufl.edu

Sanjay Ranka
ranka@cise.ufl.edu

[1] Modern Artificial Intelligence and Learning Technologies Lab (MALT-Lab), Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL, USA

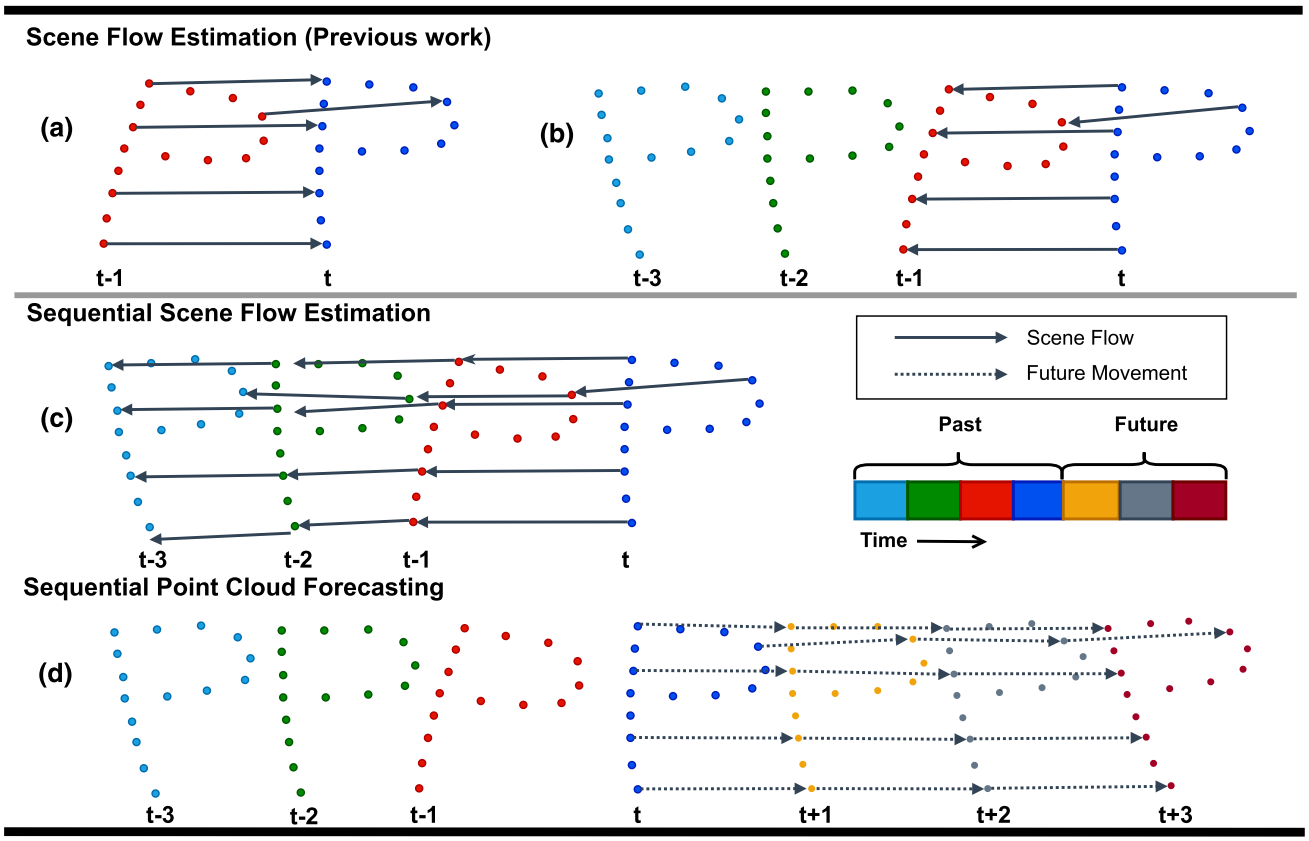

Fig. 1 Contrasting the proposed sequential scene flow estimation (SSFE) problem and standard scene flow estimation (SFE). **a** SFE predicts relative motion between a single pair of frames and has been widely evaluated in prior work such as Liu et al. (2019b), Wu et al. (2020b) and Puy et al. (2020). **b** The problem can be further elevated by utilizing preceding frames, as evidenced in MeteorNet (Liu et al. 2019c). **c** Our proposed SSFE problem requires estimating 3D scene flow between *multiple* adjacent frames. It requires processing an entire point cloud sequence, implying that multi-step spatiotemporal information is relevant for solving this task. It has been unexplored until now due to the lack of an appropriate benchmark with supervision for point cloud sequences and standardized training and evaluation protocols. Our proposed benchmark addresses this gap. **d** We also include both supervised and self-supervised variants of the closely related task of sequential point cloud forecasting (SPF) (Fan and Yang 2019; Weng et al. 2020) in the new benchmark, which has likewise been difficult to study for the same reasons. This enables investigating whether, e.g., pre-training on SSFE aids SPF

One fundamental 3D task is to understand the motion of a dynamically changing scene by estimating the scene flow between two consecutive point clouds (Fig. 1a). Despite receiving significant attention from the 3D community (Gu et al. 2019; Liu et al. 2019b; Mittal et al. 2020; Puy et al. 2020; Wu et al. 2020b), most scene flow estimation (SFE) approaches have focused on inferring the relative motion of a given frame pair. Only one known study has considered using an input *sequence* of point clouds (Liu et al. 2019c) (Fig. 1b), but they still only predict scene flow for a single pair of point clouds. Unlike previous work, we instead consider a new sequence-to-sequence problem of obtaining a sequence of flow estimation or future movement conditioned on an input sequence of point clouds and conduct a through investigation with several contributions.

First, we introduce the sequential scene flow estimation (SSFE) task. Different from the standard SFE problem, models solving SSFE are evaluated on their ability to predict $T-1$ consecutive scene flows conditioned on an input sequence of T point clouds. In SFE, models need only predict a single scene flow, whether the input is a pair of frames (Fig. 1a) or a sequence (Fig. 1b). SSFE is a non-trivial extension of SFE because current SFE methods are not equipped to extract multi-step spatiotemporal information from sequences and because SFE benchmarks do not support the training and evaluation of $T-1$ consecutive scene flows.

We also propose SSFE to help solve the challenging and related task of sequential point cloud forecasting (SPF, Fig. 1d) (Weng et al. 2020). Unlike models for SSFE, models trained on SPF must predict a sequence of *future* point clouds conditioned on a sequence of *past* point clouds. SPF is still relatively unexplored. One study proposed a self-supervised architecture for this task but failed to adequately formalize the new task and evaluate the method (Fan and

Yang 2019). Recently, another self-supervised architecture SPFNet (Weng et al. 2020) was proposed as well as a formalization of self-supervised SPF. They focus on trajectory forecasting and only provide a limited evaluation of the self-supervised SPF task. In this work, we study the connection between our new SSFE problem and the supervised and self-supervised variants of SPF.

Second, we establish a method for solving SSFE called Sequential Point Cloud Modeling Network (SPCM-Net). SPCM-Net extends a state-of-the-art coarse-to-fine SFE architecture (Wu et al. 2020b) to exploit multi-step information, and differs from related models for sequential processing of point clouds by using a set-to-set cost volume layer. Specifically, SPCM-Net embeds the set-to-set cost volume layer within a recurrent cell at each scale of a feature pyramid. This provides multi-scale spatiotemporal correlation information between neighboring point clouds which gets aggregated over time by an order-invariant recurrent unit. SPCM-Net can be directly used to also solve SPF by appending a similarly-designed decoder to the architecture. Furthermore, we explore how pre-training on SSFE impacts performance when fine-tuning on SPF.

Our third contribution is to standardize training and evaluation protocols by introducing a rigorous benchmark consisting of several datasets for both the SSFE and SPF problems. No dataset, to the best of our knowledge, has been proposed to train and evaluate frame-wise scene flow estimation in point cloud sequences of lengths longer than two frames. To overcome this limitation, we take the popular synthetic FlyingThings3D dataset (Mayer et al. 2016) and reconstruct point cloud sequences with multi-step ground truth scene flow to support the SSFE task. We repeat the same process on the newly released Virtual KITTI dataset (Gaidon et al. 2016) for synthetic evaluation on traffic scenes. We also process raw LIDAR sequences collected from the Argoverse dataset (Chang et al. 2019) for the self-supervised variant of the SPF task. We define suitable metrics for the new SSFE problem as well as for SPF and adapt appropriate prior work (Fan and Yang 2019; Liu et al. 2019b; Puy et al. 2020; Qi et al. 2017a; Wu et al. 2020b) for comparison.

Experimental results on the new benchmark confirm the effectiveness of SPCM-Net on the new SSFE problem. We demonstrate a clear advantage over pre-existing SFE methods due to the recurrent processing of point cloud sequences for learning scene dynamics. We also show that pre-training on SSFE followed by fine-tuning on the SPF task improves SPF performance significantly and establishes state-of-the-art performance compared to training from scratch. Without additional pre-training, SPCM-Net achieves competitive performance on the SPF task compared to relevant prior work. In a control study on the popular KITTI SFE benchmark (Liu et al. 2019c; Menze and Geiger 2015), we find that SPCM-Net's recurrent cost volume approach provides a stronger inductive bias for sequential point cloud processing than 4D convolution proposed by the state-of-the-art MeteorNet (Liu et al. 2019c).

1.1 Contributions

We summarize the contributions of this paper as follows:

- To the best of our knowledge, this is the first work to formally define the problem of sequential scene flow estimation (SSFE) for point cloud sequences.
- We propose the new SPCM-Net architecture for solving SSFE. SPCM-Net establishes the state-of-the-art performance on the new SSFE task by combining a set-to-set cost volume layer within a recurrent point cloud processing architecture.
- We show that pre-training SPCM-Net on the proposed SSFE task improves the downstream performance on sequential point cloud forecasting.
- To aid future research we present a sequential point cloud benchmark consisting of two synthetic datasets and one real-world dataset. The benchmark standardizes metrics for both supervised and self-supervised task variants and provides multi-step ground truth motion annotations.

In what follows, we describe SSFE and SPF problems (Sect. 2) and then present our proposed method (Sect. 3). Then, we introduce the proposed benchmark consisting of three new datasets along with appropriate evaluation metrics (Sect. 4). Experimental results are presented and discussed in Sect. 5. Related work is discussed in Sect. 6. We discuss findings, limitations, and future work in Sect. 7 and draw conclusions in Sect. 8.

2 Problem Definitions

2.1 Sequential Scene Flow Estimation

SSFE requires capturing spatiotemporal interaction to estimate frame-wise motions of points in different frames (Fig. 1c). Formally, the input is a sequence of T consecutive point clouds $\boldsymbol{P}_{1:T} = \{(\boldsymbol{C}_t, \boldsymbol{X}_t) \mid t = 1, \ldots, T\}$ with 3D point coordinates $\boldsymbol{C}_t \in \mathrm{R}^{N\times 3}$ and their corresponding features $\boldsymbol{X}_t \in \mathrm{R}^{N\times d}$, where N and d denote the number of points and feature dimensions, respectively. Given the sequence $\boldsymbol{P}_{1:T}$, the goal is to estimate the scene flow associated to each frame of the sequence (starting from the second frame). Denoting the predicted flows as $\widehat{\boldsymbol{S}}_{2:T}$ and the ground

truth scene flows as $\boldsymbol{S}_{2:T}$,[1] we want to find a function f to compute $\widehat{\boldsymbol{S}}_{2:T} = f_{\text{ssfe}}(\boldsymbol{P}_{1:T})$ that minimizes the error defined as

$$E(f_{\text{ssfe}}, \boldsymbol{P}_{1:T}) = D_e(f_{\text{ssfe}}(\boldsymbol{P}_{1:T}), \boldsymbol{S}_{2:T}). \quad (1)$$

D_e is generally instantiated as a mean square error between $\widehat{\boldsymbol{S}}_{2:T}$ and $\boldsymbol{S}_{2:T}$. The function f_{ssfe} is modeled with a neural network suitable to point cloud sequences.

Prior work (Gu et al. 2019; Liu et al. 2019b; Wang et al. 2020c; Wu et al. 2020b) has focused on the standard scene flow estimation problem between two consecutive frames. By contrast, SSFE requires processing sequences longer than two frames, which encourages the extraction of contextual information from all frames to achieve more accurate and robust estimation.

2.2 Sequential Point Cloud Forecasting

The SPF task is to process a given T-length sequence $\boldsymbol{P}_{1:T}$ and predict the most probable future point cloud sequence of length K, given by $\widehat{\boldsymbol{P}}_{T+1:T+K} = \{(\widehat{\boldsymbol{C}}_{T+k}, \widehat{\boldsymbol{X}}_{T+k}) \mid k = 1, \ldots, K\}$ (Fig. 1d):

$$\widehat{\boldsymbol{P}}_{T+1:T+K} = arg\,max_{\tilde{\boldsymbol{P}}_{T+1:T+K}} \Pr(\tilde{\boldsymbol{P}}_{T+1:T+K} \mid \boldsymbol{P}_{1:T}). \quad (2)$$

The problem is highly non-trivial due to the complexity inherent in point cloud sequences, e.g., partial or full object occlusion, shape deformation, and scale variations.

When ground truth point-wise motion is available, ground truth future frames can be generated from the input sequence $\boldsymbol{P}_{1:T}$ by adding 3D motion to the last point cloud of the sequence $\boldsymbol{P}_T$. In this setting, the goal is to find a function f_{spf} that minimizes the error between the predicted frames $\tilde{\boldsymbol{P}}_{T+1:T+K} = f_{\text{spf}}(\boldsymbol{P}_{1:T})$ and the ground truth future frames $\boldsymbol{P}_{T+1:T+K}$ defined as

$$E(f_{\text{spf}}, \boldsymbol{P}_{1:T}) = D_p(f_{\text{spf}}(\boldsymbol{P}_{1:T}), \boldsymbol{P}_{T+1:T+K}). \quad (3)$$

In this supervised setting, D_p can be implemented as the mean square error.

When ground truth point-wise motion is not available, the task can be approached in a self-supervised manner (Fan and Yang 2019; Weng et al. 2020). Here, pseudo-ground truth is generated by estimating nearest points to predicted points from the point cloud frame in the next timestep, and vice versa. Then, D_p can be implemented as the Chamfer distance (CD), which is applied to every pair of predicted and future frames at each timestep (see Sect. 3.3 for a formal definition of CD).

[1] In this paper, we define the ground truth scene flow as the motion from frame T to frame $T-1$, a backward flow. It mostly follows the setting in Liu et al. (2019c) for a convenient comparison.

The formulation of SPF above is a generalization of the same task considered in prior work (Fan and Yang 2019; Weng et al. 2020) as it admits both supervised and self-supervised approaches. This eases the study of this problem within the context of our proposed benchmark which provides multi-step ground-truth motion annotations.

3 Proposed Method

In this section, we describe our proposed method for sequential scene flow estimation *and* sequential point cloud forecasting. Our model solves the defined tasks by exploiting several properties of point cloud sequences (Liu et al. 2019c; Zhang et al. 2019):

- *Intra-frame order invariance* Points within the same frame are arranged without a specific order. Any permutation applied to the points should not change the output of the model.
- *Inter-frame location variance* Points at different timestamps may carry different spatial correlations. Such dynamic changes of spatial correlation should be captured by a model, i.e., changing the timestamp of a point should result in a different feature vector.
- *Spatiotemporal interaction between points* Points that are close spatially and temporally should be considered as neighboring points, from where local dependencies should be modeled.

3.1 Network Design

Due to the nature of SSFE and SPF, models must have a capability of *accurately capturing multi-step spatiotemporal information from point cloud sequences*. Existing SFE approaches will not adapt to these tasks because they are originally designed to handle frame pairs. Without a temporal receptive field spanning the input point cloud sequence, they are likely to fail to exploit multi-step information. Inspired by this, we propose SPCM-Net to recurrently process each pair of frames in a sequence and aggregate features in a spatiotemporal fashion.

The architecture of SPCM-Net for the SSFE task is visualized in Fig. 2. It consists of multiple modules including an intra-frame feature pyramid (IFFP) module, an inter-frame spatiotemporal correlation (IFSC) module, and a multi-scale coarse-to-fine prediction (MCP) module. The IFFP module encodes each point cloud frame into a feature pyramid that captures local spatial information at different scales. The IFSC module recurrently processes each pair of frames in the sequence and fuses features with past information, which we will describe in Sect. 3.1.2. The MCP module generates multi-scale prediction at each timestep based on features

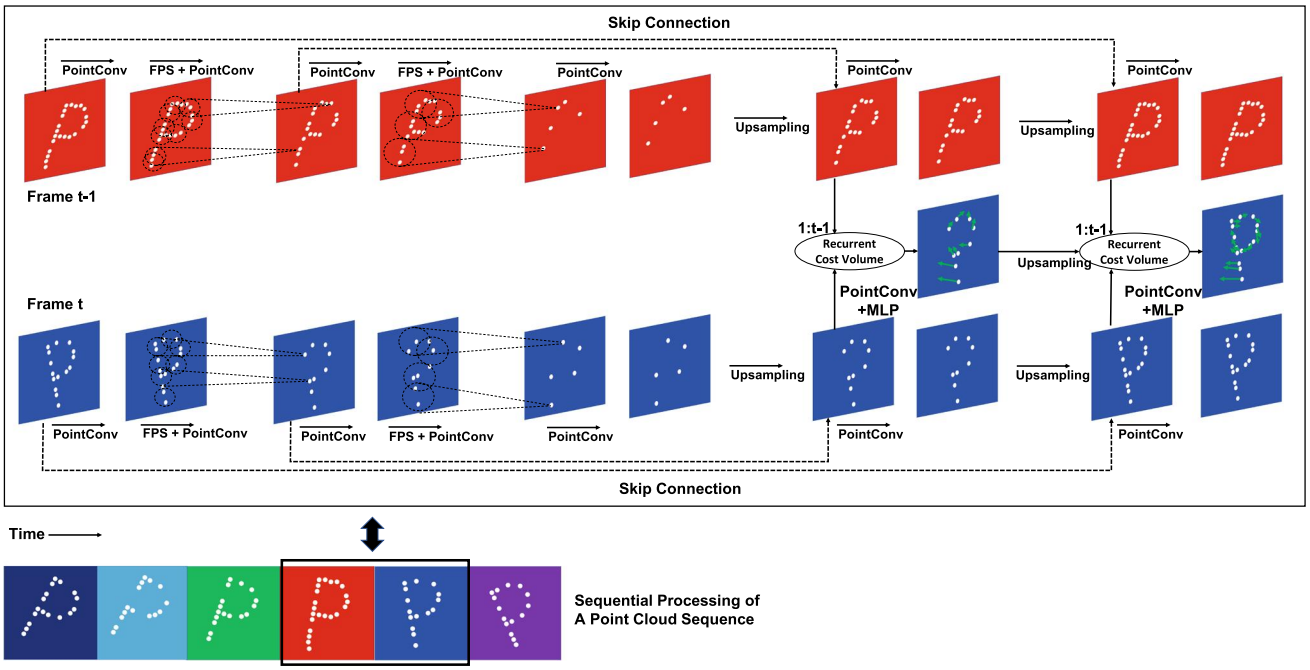

Fig. 2 The SPCM-Net architecture for sequential scene flow estimation. We only show the operation of two frames here although SPCM-Net is designed to recurrently process a point cloud sequence. For each point cloud frame of the current frame fair, we first encode it into a feature pyramid via a stack of PointConv (Wu et al. 2019), multilayer perceptron (MLP), and Farthest Point Sampling (FPS) layers (Eldar et al. 1997; Moenning and Dodgson 2003; Qi et al. 2017b). Then at each pyramid level l, its recurrent cost volume layer takes features of the current point cloud (F_l) as the input and updates its hidden states from $t-1$ to t. We concatenate the updated states, the upsampled coarse flow from pyramid level $l-1$, and feature F_l, followed by multiple PointConv (Wu et al. 2019) and MLP layers, to generate a finer scene flow and the intermediate features. For simplicity, we omit one level and only visualize a three-level pyramid architecture. The future predictor is designed similarly and is described accordingly in Sect. 3.1.3

from IFFP and IFSC modules. Depending on the specific task, the MCP module is to either estimate scene flows or predict future point movements.

We now describe each module of the SPCM-Net architecture for the SSFE task and discuss key ingredients to our approach. The architecture for the SPF task is similarly designed and described in Sect. 3.1.3.

3.1.1 Intra-Frame Feature Pyramid

We can not directly apply conventional convolution operators to point clouds because they are irregular and orderless. Therefore, we follow PointPWC-Net architecture (Wu et al. 2020b) to utilize the PointConv layer (Wu et al. 2019) to capture local spatial information within each point cloud. This generates pyramidal features by hierarchically sampling each point cloud via multiple Farthest Point Sampling (FPS) (Eldar et al. 1997; Moenning and Dodgson 2003; Qi et al. 2017b) layers. Each pyramid captures the local geometric structure within its receptive field. The pyramid features are further aggregated into larger units to generate higher-level features. We repeat the process until reaching a demanding number of pyramid levels, e.g., five (Wu et al. 2020b).

3.1.2 Inter-Frame Spatiotemporal Correlation

To model spatiotemporal correlation between point cloud frames, we would like to exploit the local dependencies of neighboring frames to capture a larger temporal receptive field across the whole sequence length. A straightforward solution would be directly applying 4D convolutions on the voxelized sequence following (Choy et al. 2019). However, this requires extensive computation. Furthermore, quantization errors during voxelization may cause performance drops when tackling problems requiring precise measurement, e.g., point-wise scene flow estimation. A promising solution would be instead borrowing designs from sequence modeling to construct a recurrent model customized for point cloud sequences.

PointPWC-Net uses a cost volume for computing the scene flow between two consecutive frames. In our model, within each pyramid level, the pyramid features generated from two frames are combined to compute a cost volume to obtain spatiotemporal correlation. Although we could simply repeat this computation between every two adjacent frames to predict scene flow for a given point cloud sequence, this ignores multi-step information in preceding frames which

could lead to more accurate estimation. Instead, we combine the cost volume for pairwise frame modeling with a recurrent neural unit similar to long short-term memory (LSTM) (Graves 2012; Hochreiter and Schmidhuber 1997). The spatiotemporal features produced by the cost volume are fused with past information, producing a smoother estimation of 3D scene flow for each point. This is important for both the SSFE and SPF tasks. The detailed description of the *recurrent* cost volume (RCV) layer can be found in Sect. 3.2.

3.1.3 Multi-scale Coarse-to-Fine Prediction

Inspired by scene flow approaches (Gu et al. 2019; Revaud et al. 2015; Sun et al. 2018; Wu et al. 2020b), we adopt a *coarse-to-fine* prediction approach where the current scene flow prediction is initialized with estimated flows from a preceding prediction. We establish this by plugging RCV layers into the feature pyramid defined in Sect. 3.1.1. At each pyramid level, the RCV layer builds a spatiotemporal correlation between downsampled versions of the current pair of point cloud frames in the sequence via FPS (Eldar et al. 1997; Li et al. 2018b; Moenning and Dodgson 2003; Qi et al. 2017a, 2019; Yu et al. 2018). At the top level is the original input point clouds. The bottom level contains the fewest points, which generates the coarsest scene flows. We upsample these flows with respect to points in the higher level via the widely used inverse distance weighted interpolation (Qi et al. 2017b) and make a further refinement.

Sequential flow estimator At each pyramid level l of the current timestep t, the RCV takes features of the current point cloud (F_l^t) as the input and updates its hidden states from $t-1$ to t. We concatenate the updated states, the upsampled coarse flow from the pyramid level $l-1$, and features F_l, followed by a stack of PointConv (Wu et al. 2019) and multilayer perceptron (MLP) layers, to generate a finer scene flow and the intermediate features.

Formally, let $SF_{l,t}$ be the estimated flow at level l of timestep t, $\boldsymbol{P}_{t,l}$ be the point cloud at level l of timestep t, we upsample the estimated coarse flow $SF_{l-1,t}$ at level $l-1$ with respect to $\boldsymbol{P}_{t,l}$ to obtain the upsampled coarse flow. For each point $p_i^{l,t}$ in the fine level point cloud $\boldsymbol{P}_{t,l}$, we find its K nearest neighbors $N(p_i^{l,t})$ in the coarse level point cloud $\boldsymbol{P}_{t,l-1}$. Each scene flow $\hat{sf}_{l,t}^i$ in $\hat{SF}_{l,t}$ for finer level l is computed via inverse distance weighted interpolation:

$$\hat{sf}_{l,t}^i = \frac{\sum_{j=1}^{K} \omega(p_i^{l,t}, p_j^{l-1,t}) SF_j^{l-1,t}}{\sum_{j=1}^{K} \omega(p_i^{l,t}, p_j^{l-1,t})} \tag{4}$$

where $\omega(p_i^{l,t}, p_j^{l-1,t}) = \frac{1}{d(p_i^{l,t}, p_j^{l-1,t})}$ and $p_j^{l-1,t} \in N(p_i^{l,t})$. We used the Euclidean distance as the distance metric $d(p_i^{l,t}, p_j^{l-1,t})$. The estimated flow $SF_{l,t}$ is further obtained by concatenating $\hat{SF}_{l,t}$, F_l^t and the updated state of RCV ($\boldsymbol{H}_t$) and feeding them to a stack of PointConv (Wu et al. 2019) and MLP layers:

$$SF_{l,t} = MLP(PointConv(\widehat{SF}_{l,t}; F_l^t; \boldsymbol{H}_t). \tag{5}$$

We repeat the process for all pyramid levels. The final estimated scene flow at time t is $SF_{L,t}$. By doing so, we have modeled a point cloud sequence via multiple RCV layers at different pyramid levels and across different timesteps. This design allows exploiting stronger spatiotemporal correlation in sequences, which we verify in the experiment section.

Sequential future predictor The architecture from the SSFE task can be adapted to support the SPF task by treating it as an encoder and adding a decoder. The encoder digests input point cloud frames while the decoder predicts the future movement of the last input point cloud $\boldsymbol{P}_T = (\boldsymbol{C}_T, \boldsymbol{X}_T)$. Specifically, the encoder consumes the input point cloud sequence frame-by-frame and keeps updating the states of RCV layers till $\boldsymbol{P}_T$. The obtained states will initialize the states for the decoder. To simplify the problem, we predict the displacements $\Delta \boldsymbol{P}$ between points of the current timestep and the next timestep rather than directly reconstructing future point coordinates from scratch, which is generally more difficult. We feed the predicted point cloud frame into the model to interact with the states of the RCV layers and generate the next point cloud. We repeat this operation until the prediction step reaches K.

3.2 Recurrent Cost Volume Layer

This section describes the recurrent cost volume in detail. We begin by providing a preliminary introduction to the cost volume to build the necessary background.

We first introduce the learnable matching cost between two consecutive point clouds following PointPWC-Net (Wu et al. 2020b). Formally, given two points $\boldsymbol{p}_{t-1}^i = (\boldsymbol{c}_{t-1}^i, \boldsymbol{x}_{t-1}^i) \in \boldsymbol{P}_{t-1}$ and $\boldsymbol{p}_t^j = (\boldsymbol{c}_t^j, \boldsymbol{x}_t^j) \in \boldsymbol{P}_t$ with 3D point coordinates and their corresponding features, the matching cost between $\boldsymbol{p}_t^j$ and $\boldsymbol{p}_{t-1}^i$ is defined as

$$\text{Cost}(\boldsymbol{p}_t^j, \boldsymbol{p}_{t-1}^i) = \phi_{\text{MLP}}(\boldsymbol{c}_{t-1}^i - \boldsymbol{c}_t^j, \boldsymbol{x}_{t-1}^i, \boldsymbol{x}_t^j). \tag{6}$$

Here, the feature vectors of two points and the directional difference between their spatiotemporal positions are passed to an MLP. Note that $\boldsymbol{p}_{t-1}^i$ comes from a neighboring point set of $\boldsymbol{p}_t^j$ based on spatiotemporal distance or feature similarity.

However, it has been shown that this pure point-to-point matching cost is sensitive to outliers (Wu et al. 2020b). The *flow embedding* layer proposed by Liu et al. (2019b) partly addresses it by aggregating flow votes from neighboring points. Specifically, for a given point $\boldsymbol{p}_t^j$, they finds its neigh-

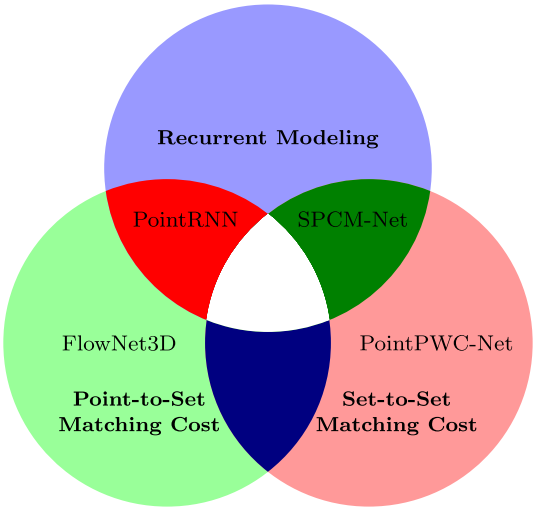

Fig. 3 A comparison between different model architectures. The key operation in PointPWC-Net (Wu et al. 2020b) is the *cost volume*, which we consider as *the set-to-set matching cost*, in contrast to *the point-to-set matching cost* (the *flow embedding* layer) proposed by FlowNet3D (Liu et al. 2019b). PointRNN (Fan and Yang 2019) incorporates the *flow embedding* layer of FlowNet3D (Liu et al. 2019b) into a recurrent unit for predicting future point clouds while the proposed SPCM-Net extends PointPWC-Net Wu et al. (2020b) by introducing the *recurrent cost volume* that combines the cost volume for pairwise frame modeling with a recurrent neural unit similar to long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997; Graves 2012)

boring points at timestep $t-1$ via ball query. These points are considered as multiple soft correspondence points for $\boldsymbol{p}_t^j$ and are utilized to obtain multiple matching costs defined in Eq. (6). Matching costs are further aggregated via the max-pooling. However, motion information can be lost due to the max-pooling operation. To obtain a more robust and stable matching cost, a preferable approach is to aggregate matching costs in a manner similar to the patch-to-patch approach in optical flow (Hosni et al. 2012; Sun et al. 2018). This motivates us to choose the *cost volume* in PointPWC-Net (Wu et al. 2020b) to describe point motion. We consider the cost volume as *the set-to-set matching cost*, in contrast to *the point-to-set matching cost* (the *flow embedding* layer). The leap from point-to-set to set-to-set matching costs is exactly prefigured in the move from the softmax to the softassign cost in earlier point matching (Chui and Rangarajan 2003). We provide a comparison between our proposed SPCM-Net and several model architectures including PointRNN (Fan and Yang 2019), FlowNet3D (Liu et al. 2019b), and PointPWC-Net (Wu et al. 2020b) in Fig. 3.

Formally, the cost volume for $\boldsymbol{p}_t^j$ is defined as

$$\begin{aligned}\mathrm{CV}(\boldsymbol{p}_t^j) = &\sum_{\boldsymbol{p}_t^k \in M(\boldsymbol{p}_t^j)} \omega_M(\boldsymbol{p}_t^k, \boldsymbol{p}_t^j) \\ &\times \sum_{\boldsymbol{p}_{t-1}^i \in N(\boldsymbol{p}_t^k)} \omega_N(\boldsymbol{p}_{t-1}^i, \boldsymbol{p}_t^k)\, \mathrm{Cost}(\boldsymbol{p}_{t-1}^i, \boldsymbol{p}_t^k)\end{aligned} \tag{7}$$

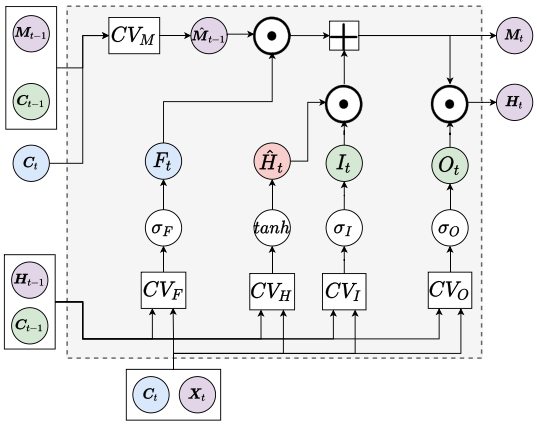

Fig. 4 Recurrent cost-volume layer. At each timestep t, it takes the current point locations $\boldsymbol{C}_t$ and the associated features $\boldsymbol{X}_t$ as the inputs. They will interact with recurrent cost volume memory states $\boldsymbol{C}_{t-1}$, $\boldsymbol{H}_{t-1}$, and $\boldsymbol{M}_{t-1}$ via multiple gates

$$\omega_M(\boldsymbol{p}_t^k, \boldsymbol{p}_t^j) = \mathrm{MLP}(\boldsymbol{c}_t^k - \boldsymbol{c}_t^j) \tag{8}$$

$$\omega_N(\boldsymbol{p}_{t-1}^i, \boldsymbol{p}_t^k) = \mathrm{MLP}(\boldsymbol{c}_{t-1}^i - \boldsymbol{c}_t^k) \tag{9}$$

This requires finding a spatial neighboring point set $M(\boldsymbol{p}_t^j)$ around $\boldsymbol{p}_t^j$ in $\boldsymbol{P}_t$. Then for each point $\boldsymbol{p}_t^k \in M(\boldsymbol{p}_t^j)$, we find a spatiotemporal neighboring point set $N(\boldsymbol{p}_t^k)$ around $\boldsymbol{p}_t^k$ in $\boldsymbol{P}_{t-1}$ (across time). The interaction between these points is modeled by two directional vectors obtained via convolutional operations $\omega_M(\boldsymbol{p}_t^k, \boldsymbol{p}_t^j)$ and $\omega_N(\boldsymbol{p}_{t-1}^i, \boldsymbol{p}_t^k)$, and their matching costs. Both the spatial neighboring point set $M(\boldsymbol{p}_t^j)$ and $N(\boldsymbol{p}_t^k)$ can be obtained by conducting an efficient GPU-based ball query that finds all points within a radius to the query point or the K-nearest neighbor search that finds a fixed number of points that are the closest.

Now we show how to embed the cost volume into a recurrent unit for point cloud sequences (Fig. 4).

Inputs To maintain spatial structure, our hidden states maintain both the point coordinates and the associated features. At timestep t, both $\boldsymbol{C}_t$ and $\boldsymbol{X}_t$ will be fed into the RCV layer as the input.

Initialization Accordingly, the states of the RCV layer are extended to $\boldsymbol{C}_{t-1}$, $\boldsymbol{H}_{t-1}$ and $\boldsymbol{M}_{t-1}$ to track the most recent historic point locations and memory states. For notation clarity, since we already are using $\boldsymbol{C}_t$ to denote the coordinates of points in the point cloud, we will use $\boldsymbol{M}_t$ to refer to the recurrent cell state. The hidden and cell states are zero-initialized at time $t=0$.

Order-invariance Since the point sets can be ordered completely differently across time and objects may appear or disappear from view, there is no guarantee that a one-to-one mapping between neighboring frames exists. Therefore, we use the cross-frame neighborhood query within the cost vol-

ume operation to perform updates to the hidden and cell states ($\boldsymbol{H}_{t-1}$ and $\boldsymbol{M}_{t-1}$), making them invariant to any changes to the order of points or to the addition of new points.

Update Let

$$\mathrm{CV}(\boldsymbol{P}_t; \boldsymbol{P}_{t-1}) = \mathrm{CV}(\boldsymbol{C}_t, \boldsymbol{X}_t; \boldsymbol{C}_{t-1}, \{\boldsymbol{H}_{t-1}, \boldsymbol{M}_{t-1}\}) \quad (10)$$

be the cost volume for all points in timestep t with respect to the point cloud at timestep $t-1$. The core component of the update operator is a recurrent unit similar to the LSTM cell, where we replace the fully connected layers with the cost volume. The relevant update equations are:

$$I_t = \sigma_I(CV_I(\boldsymbol{C}_t, \boldsymbol{X}_t; \boldsymbol{C}_{t-1}, \boldsymbol{H}_{t-1}), \quad (11)$$

$$F_t = \sigma_F(CV_F(\boldsymbol{C}_t, \boldsymbol{X}_t; \boldsymbol{C}_{t-1}, \boldsymbol{H}_{t-1}), \quad (12)$$

$$O_t = \sigma_O(CV_O(\boldsymbol{C}_t, \boldsymbol{X}_t; \boldsymbol{C}_{t-1}, \boldsymbol{H}_{t-1}), \quad (13)$$

$$\hat{\boldsymbol{M}}_{t-1} = CV_M(\boldsymbol{C}_t, \mathrm{None}; \boldsymbol{C}_{t-1}, \boldsymbol{M}_{t-1}), \quad (14)$$

$$\hat{\boldsymbol{H}}_t = \tanh(CV_H(\boldsymbol{C}_t, \boldsymbol{X}_t; \boldsymbol{C}_{t-1}, \boldsymbol{H}_{t-1})), \quad (15)$$

$$\boldsymbol{M}_t = F_t \odot \hat{\boldsymbol{M}}_{t-1} + I_t \odot \hat{\boldsymbol{H}}_t, \quad (16)$$

$$\boldsymbol{H}_t = O_t \odot \boldsymbol{M}_t, \quad (17)$$

where the operator $\odot$ denotes the Hadamard product. CV_I, CV_F, and CV_O denote the cost volume operations for the $input$, $forget$ and $output$ gates, respectively. The "None" in the cost volume operation applied to the cell state $\boldsymbol{M}_{t-1}$ represents that we do not use any input features when performing the neighborhood query on the cell state (x_t^j in Eq. (6) is ignored). Then $\hat{\boldsymbol{M}}_{t-1}$ is modulated via the forget gate F_t and is further aggregated with the new memory $\hat{\boldsymbol{H}}_t$ passed to the input gate I_t to obtain the latest memory state $\boldsymbol{M}_t$. The latest hidden state $\hat{\boldsymbol{H}}_t$ is obtained by conditionally deciding what to output from $\boldsymbol{M}_t$ controlled by the output gate O_t. Note that $\boldsymbol{C}_t$ and $\boldsymbol{H}_t$ can be used together as the input features of downstream tasks.

3.3 Learning Objectives

SSFE When ground truth scene flows are available, we adapt the multi-scale loss function used in PWC-Net (Sun et al. 2018) and PointPWC-Net (Wu et al. 2020b) and extend it to handle point cloud sequences. Given the predicted scene flow $\boldsymbol{SF}_{t,l}$ at the pyramid level l from timestep t and its ground truth scene flow $\boldsymbol{GF}_{t,l}$. The objective function is specified as

$$L_{\text{supervised}}^{\text{SSFE}} = \sum_{t=2}^{T} \sum_{l=1}^{L} \alpha_l ||\boldsymbol{SF}_{t,l} - \boldsymbol{GF}_{t,l}||_2^2. \quad (18)$$

Occluded points are not considered by masking them out from gradient computation and weight updating. As done previously (Wu et al. 2020b), we use a set of hyper-parameters $\{\alpha_l \mid l = 1, \ldots, L\}$ to balance the importance of losses from different pyramid levels.

In this study, we do not explore self-supervised SSFE as this would require non-trivial innovation to develop a suitable training objective. Recent work (Mittal et al. 2020) has shown progress on self-supervised SFE which suggests that an extension to SSFE is possible.

SPF When ground truth point-wise motion is available, the learning objective is similar to Eq. (18). We compute the difference between the predicted future frames and the future ground truth frames derived from ground truth scene flow. Denote $\boldsymbol{P}_{t,l}$ and $\widehat{\boldsymbol{P}}_{t,l}$ as ground truth and predicted frames at the pyramid level l from timestep t. The objective function is specified as

$$L_{\text{supervised}}^{\text{SPF}} = \sum_{t=T+1}^{T+K} \sum_{l=1}^{L} \alpha_l ||\widehat{\boldsymbol{P}}_{t,l} - \boldsymbol{P}_{t,l}||_2^2. \quad (19)$$

In reality, it is often difficult and expensive to obtain ground truth scene flows for real-world point cloud sequences and therefore few scene flow datasets are available. To avoid relying on the availability of ground truth scene flows, we can also define a self-supervised learning objective to train the model. We adopt the Chamfer Distance (CD) to compute the difference between predicted sequences and actual future sequences (Weng et al. 2020), which allows us to approximate the ground truth scene flow and guide the model learning. The CD is defined as

$$D_{CD}(\boldsymbol{P}_t, \widehat{\boldsymbol{P}}_t) = \sum_{\boldsymbol{p} \in \boldsymbol{P}_t} \min_{\hat{\boldsymbol{p}} \in \widehat{\boldsymbol{P}}_t} \|\boldsymbol{p} - \hat{\boldsymbol{p}}\|^2 + \sum_{\hat{\boldsymbol{p}} \in \widehat{\boldsymbol{P}}_t} \min_{\boldsymbol{p} \in \boldsymbol{P}_t} \|\hat{\boldsymbol{p}} - \boldsymbol{p}\|^2, \quad (20)$$

where $\boldsymbol{P}_t$ and $\widehat{\boldsymbol{P}}_t$ are ground truth and predicted frames. We apply Eq. (20) to all the future frames:

$$L_{\text{self-supervised}}^{\text{SPF}} = \sum_{t=T+1}^{T+K} \sum_{l=1}^{L} \alpha_l \mathcal{D}_{CD}(\widehat{\boldsymbol{P}}_{t,l}, \boldsymbol{P}_{t,l}). \quad (21)$$

We do not explore advanced techniques widely used in the scene flow community for self-supervised SPF, e.g., Laplacian regularization or local smoothness (Pontes et al. 2020; Wu et al. 2020b), to further improve the performance. It guarantees a relatively fair comparison to baseline methods with simple learning objectives.

4 Datasets and Metrics

In this section, we will describe datasets and the evaluation metrics designed for new tasks.

Limitations of existing benchmarks Most existing benchmarks (Mayer et al. 2016; Menze and Geiger 2015) focus on SFE between two consecutive point cloud frames with ground truth annotations, which are widely adopted in recent state-of-the-art approaches (Gu et al. 2019; Liu et al. 2019b; Wu et al. 2020b). An extension of the KITTI scene flow dataset to short sequences for flow estimation of the last input frame has been considered (Liu et al. 2019c). However, this does not meet the requirement of multi-step scene flow annotations necessary for supervised SSFE and SPF.

New benchmarks Therefore, to evaluate SSFE and SPF, new datasets are needed to help systematically analyze novel methods. For supervised SSFE and SPF we adapt two synthetic yet challenging datasets: FlyingThings3D (Mayer et al. 2016) and Virtual KITTI (Gaidon et al. 2016). We generate ground truth annotations for point cloud sequences that could be useful to both SSFE and SPF tasks. For self-supervised SPF we extract sequences from the real-world Argoverse dataset (Chang et al. 2019).

4.1 Sequential FlyingThings3D (SFT3D) Dataset

FlyingThings3D (Mayer et al. 2016) is the first large-scale synthetic dataset proposed for training deep learning models on scene flow estimation. It contains videos all with a frame length of 10, rendered from scenes by randomly moving objects from the ShapeNet dataset (Chang et al. 2015). However, it does not provide point cloud sequences directly. Therefore, we reconstructed point clouds and 3D scene flows based on the ground-truth disparity maps, maps of disparity change, optical flows, and the provided camera parameters. We follow (Gu et al. 2019; Liu et al. 2019b) and only maintain points with a depth of less than 35 m.

Our SFT3D dataset can be used for evaluating both SSFE and SPF. In detail:

- *SSFE* We use the first six frames as the input for SSFE. Models must predict all the scene flows starting from frame 2 to frame 6. All these frames are provided with ground truth scene flows. For input frames, we randomly sample a fixed number of points (e.g., 2048 points) for each frame in a non-corresponding manner, meaning a point of a certain frame may not necessarily find its corresponding point in the subsequent frame.
- *SPF* We take the first six frames as the input while using the rest of the frames as ground truth. For all points in future frames, we find and track the future movement of points sampled in the last input frames (the 6th input frames) along the whole prediction period to obtain the ground truth motions.

The proposed SFT3D dataset is challenging, e.g., it contains points of occluded scene flows. Similar to the preparation of Liu et al. (2019b), occluded points are present in both the input and output of a designed model. However, they are not considered during performance evaluation or included in training losses. We remove sequences where all points are completely occluded in any frame. A visual example can be found in Fig. 5.

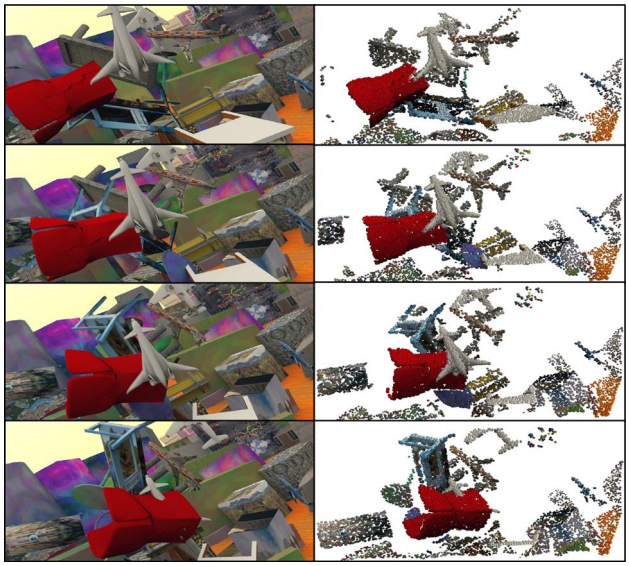

Fig. 5 One example sequence from our created SFT3D dataset (the frame number increases from top to bottom). Left: The original frames of FlyingThings3D dataset (Mayer et al. 2016). Right: The reconstructed point cloud sequences. Notice that existing scene flow approaches only take a pair of frames as the input while in this paper we focus on handling point cloud sequences. Best viewed in color

4.2 Virtual KITTI Sequence (VKS) Dataset

The Virtual KITTI dataset uses a game engine to recreate real-world videos from the KITTI tracking benchmark (Geiger et al. 2012). Due to recent improvement in lighting and post-processing of the Unity game engine, an improved dataset called the Virtual KITTI 2 dataset is released to be more photo-realistic and better-featured. It provides images from a stereo camera with new supports for forward and backward optical flow, forward and backward scene flow, and available camera parameters. We consider vehicles as objects of interest as they are the main dynamic objects in a traffic scene, i.e., trucks, cars, and vans. We obtain 3D point locations by projecting 2D pixel positions (in meters) to the 3D space based on the camera parameters.

As shown in Fig. 6, the original virtual KITTI contains five scenes of crowded urban area (Scene01), busy intersections (Scene02, Scene06), long road in the forest (Scene18), and highway driving scene (Scene20). For each sequence, we repeatedly sample consecutive frames with a length of 10 and select the starting frame number every five frames. For all videos, we use their first 60% of frames for training and

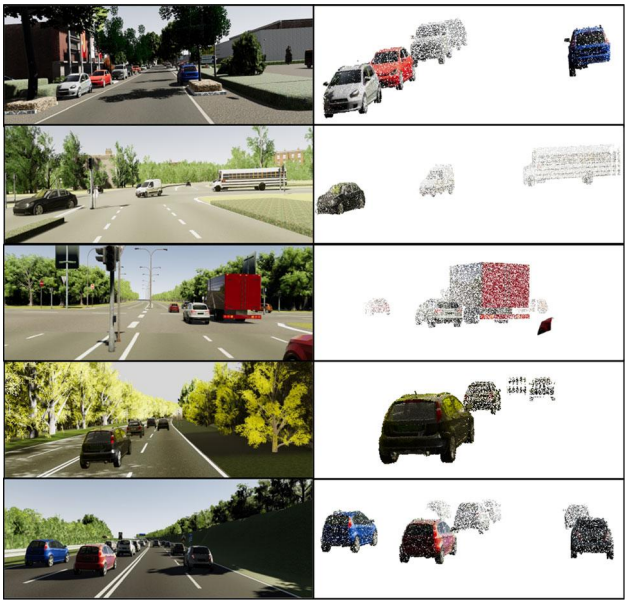

Fig. 6 Sample snapshot frames from the VKS dataset (sequence 1, 2, 6, 18, 20 from top to bottom). We consider vehicles as objects of interests as they are the major dynamic objects in a traffic scene. (Left) Original image frames from the Virtual KITTI dataset. (Right) Our created point clouds. Best viewed in color

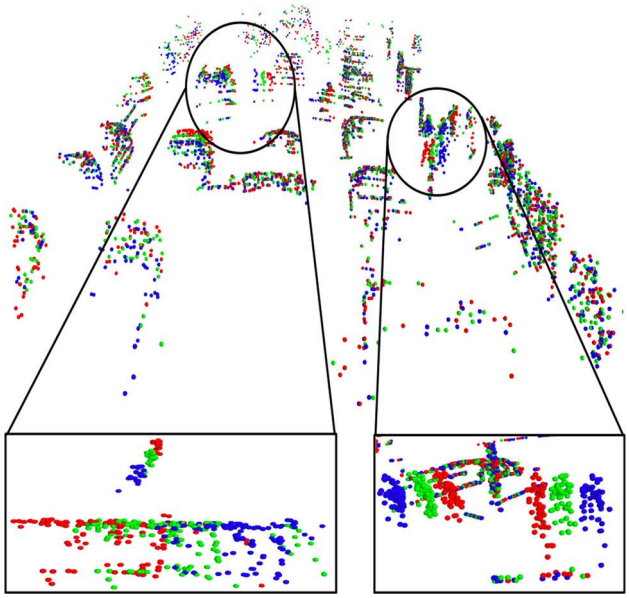

Fig. 7 Visualization of one example from our created SAG dataset. Different colors denote points from a different frame number: first input frame 1, last input frame 5, and last future frame 10. Ground plane points are removed using a heuristic algorithm. Best viewed in color

the remaining 40% for testing. We sample 2048 points for each frame and only consider points with a depth less than 35 m.

Our VKS dataset is used for evaluating both SSFE and SPF tasks:

- *SSFE* The first five frames are used as the input. Models estimate flows from frame two to frame five.
- *SPF* Models predict the future movement of points starting from the last input frame (frame five) for five steps. Similar to our SFT3D dataset, occluded points are not considered during the evaluation.

4.3 Sequential Argoverse (SAG) Dataset

The Argoverse dataset (Chang et al. 2019) is collected in Pittsburgh, Pennsylvania, USA and Miami, Florida, USA by a fleet of autonomous vehicles. The collected dataset captures different seasons, weather conditions, and times of the day. We use the raw LiDAR data from *Argoverse-Tracking* consisting of 113 log segments varying in length from 15 to 30 s. Among them, 89 logs are used for training and the rest is for testing.

The Argoverse dataset does not provide ground-truth point-wise motion and therefore we adopt metrics that do not require annotations (Sect. 4.5). Hence, we only train and evaluate models on self-supervised SPF with this dataset. Forecasting future point clouds on real-world datasets is challenging with rapid changes in the vehicle's surroundings. We focus on short-term prediction in this work. We repeatedly sample from each driving log by randomly choosing 10 consecutive frames and creating the corresponding point cloud sequences. To achieve a reasonable computation, we sample a fixed number of points for each frame, i.e., 2048. We remove the ground points to reduce the bias caused by the flattened geometry of the ground. Similarly, we follow the practice of Wu et al. (2020a) and crop the point clouds to extract region defined by $[-32, 32] \times [-8, 8] \times [-\infty, 2]$ m, which corresponds to the XYZ range. A sample data is shown in Fig. 7.

4.4 Overview of Dataset Splits

Here we provide detailed information on the dataset splits (see Table 1 as well).

- *SFT3D* A total of 5337 sequences are created. Among them, 4020 videos are used for training and the rest of 446 and 871 videos are held out for validation and testing, respectively. The validation set is utilized to select the best training model.
- *VKS* We have collected a total of 3350 point cloud sequences, where 2175 of them are used to train the model and the rest of the sequences are for testing.
- *SAG* The SAG dataset contains a total of 1200 test point cloud sequences, where the first five frames are the input and the rest are the ground truth future frames.

Table 1 A summary of the created datasets with X forward, Y left, and Z up

Dataset	Frame sampling	X–Y–Z point range	Points per frame	#Train	#Validation	#Test	SSFE	SPF
STF3D	N/A	$X \leq 35$	2048	4020	446	871	✓	✓
VKS	Sample 10 frames per second	$X \leq 35$	2048	2175	N/A	1175	✓	✓
SAG	Sample 10 frames per second	$[-32, 32] \times [-8, 8] \times [-\infty, 2]$	2048	On-the-fly	N/A	1200	X	✓

Training samples are generated on-the-fly similar to test sequences.

4.5 Evaluation Metrics

To adapt several standard evaluation metrics in point cloud processing to fit our task format, we analyzed the usefulness of scene flow estimation metrics as well as metrics for future point cloud prediction and identified which are best suited. Our evaluation is mainly divided into three types: supervised SSFE metrics, supervised SPF metrics, and self-supervised SPF metrics.

Supervised SSFE metrics If the ground truth scene flow annotations are available, we adapt the evaluation protocol of 3D scene flow estimation (Gu et al. 2019; Liu et al. 2019b, c) and extend to sequences. Specifically, the 3D end point error (EPE3D) and accuracy (ACC3D) are used as the metrics. The EPE3D measures the average ℓ_2 distance between the predicted scene flow vector $\widehat{\boldsymbol{s}}_t^i$ and ground truth scene flow vector $\boldsymbol{s}_t^i$ for all points in the sequence of length $T-1$, which is computed as

$$\text{EPE3D} = \frac{1}{\sum_{t,i} m_t^i} \sum_{t=2}^{T} \sum_{i=1}^{N} m_t^i ||\widehat{\boldsymbol{s}}_t^i - \boldsymbol{s}_t^i||_2 \tag{22}$$

where m is a binary mask and $m_t^i = 0$ denotes an invalid scene flow of the point $\boldsymbol{p}_t^i$. This is possible in reality due to the viewpoint shift and occlusion. Taking the average over all valid points reflects the overall performance of flow estimation over sequences. The ACC3D reflects the portion of estimated flows that are below a specified end point error threshold among all points. Following Gu et al. (2019), both strict and relaxed ACC3D are used:

- The *strict ACC3D (Acc3DS)* considers the percentage of points whose $EPE3D < 0.05$ m or relative error $< 5\%$.
- The *relaxed ACC3D (Acc3DR)* considers the percentage of points whose $EPE3D < 0.1$ m or relative error $< 10\%$.

To measure outlier prediction, we use the *Outliers3D* to compute the percentage of points whose $EPE3D > 0.3$ m or relative error $> 10\%$. We noticed that Outliers3D is invalid when the ground truth flows are near $\mathbf{0}$. The reason is that computing this value requires dividing by the norm of the ground truth flow, which is sensitive to values near zero. Therefore, we fix it by proposing *a rectified version of Outliers3D (RectOutliers3D)* that only depends on the condition $EPE3D > 0.3$ m when the ground truth scene flow is small (e.g., its ℓ_2 norm is lower than a threshold value 0.1). Otherwise, it remains the same to Outliers3D.

We use two additional metrics that involve projecting point clouds back to the image plane. We obtain EPE2D by computing the 2D end point error in the image plane and Acc2D by calculating the percentage of points whose EPE2D $< 3px$ or relative error $< 5\%$.

We highlight that although all the SSFE metrics are SFE metrics that have been extended to sequences, we maintain the same names for simplicity.

Supervised SPF metrics We propose to use the standard evaluation from the trajectory forecasting community (Alahi et al. 2016). We consider two common metrics: the average displacement error (ADE) and final displacement error (FDE).

The ADE measures the average Euclidean distance between the estimated point $\boldsymbol{p}_{T+t}^i$ and ground truth point $\boldsymbol{g}_{T+t}^i$ for all points in each prediction step. Considering the occluded points, it is defined as

$$\text{ADE} = \frac{1}{\sum_{t,i} m_{T+t}^i} \sum_{t=1}^{K} \sum_{i=1}^{N} m_{T+t}^i ||\boldsymbol{p}_{T+t}^i - \boldsymbol{g}_{T+t}^i||_2 \tag{23}$$

where m_{T+t}^i denotes that the point has disappeared due to occlusion or view shift if the value is 0, otherwise 1. The FDE computes the average Euclidean distance between estimated point $\boldsymbol{p}_{T+K}^i$ and ground truth point $\boldsymbol{g}_{T+K}^i$ for all points at end of the prediction period K.

Self-supervised SPF metrics ADE and FDE are suitable when ground truth annotations are available. In most real-world datasets where ground truth point correspondences between frames are difficult to obtain, we cannot compute them anymore. We could have approximated the true correspondence by solving a weighted bipartite matching problem (Jonker and Volgenant 1987) or via softassign (Chui and Ran-

garajan 2000, 2003) but instead adopted a simpler nearest neighbor approach (Barrow et al. 1977; Besl and McKay 1992; Fan and Yang 2019; Li et al. 2018b; Weng et al. 2020). Specifically, we generate the pseudo-ground truth points by considering nearest points to predicted points from the point cloud frame in the next timestep, and vice versa. This could be implemented as a Chamfer distance, which is previously defined in Sect. 3, Eq. (20). We apply Chamfer distance to all future frames and sum the errors up by enumerating t from $T+1$ to $T+K$.

Similarly, we also use Earth Mover's Distance (EMD) (Rubner et al. 2000) defined as

$$\mathcal{D}_{EMD}(\boldsymbol{P}_t, \widehat{\boldsymbol{P}}_t) = \min_{\phi:\boldsymbol{P}_t \to \widehat{\boldsymbol{P}}_t} \sum_{\boldsymbol{p}\in\boldsymbol{P}_t} \|\boldsymbol{p} - \phi(\boldsymbol{p})\|^2, \tag{24}$$

where $\phi : \boldsymbol{P}_t \to \widehat{\boldsymbol{P}}_t$ is a bijection. We divide both EMD and CD by the total number of points.

Both CD and EMD lack a mechanism to handle outliers which are likely to exist due to occlusions, noise, and sampling patterns in LiDAR point clouds. For example, in CD, noisy points or isolated points that are far from the others might substantially increase the CD by introducing large distance values between them and their nearest-neighbours, leading to noisy evaluation. In EMD, the constraint of one-to-one mapping (a bijective mapping) is usually too harsh for LiDAR point clouds, which are sampled randomly.

Inspired by Gojcic et al. (2021) and Yew and Lee (2020), we introduce an evaluation method built upon optimal transport (Peyré and Cuturi 2019). Our goal is to find the corresponding point of the estimated point $\boldsymbol{p}_t^i$ with respect to the ground truth point cloud $\widehat{\boldsymbol{P}}_t$. Denoting φ as the hard correspondence mapping, we define an evaluation metric as

$$\mathcal{D}_{CORR}(\boldsymbol{P}_t, \widehat{\boldsymbol{P}}_t) = \sum_{\boldsymbol{p}\in\boldsymbol{P}_t} \|\boldsymbol{p} - \varphi(\boldsymbol{p}, \widehat{\boldsymbol{P}}_t)\|^2. \tag{25}$$

To obtain φ, we firstly obtain the optimal soft assignment (called the softassign in an homage to the softmax nonlinearity) φ_{soft} via the Sinkhorn algorithm (Sinkhorn 1964), and then follow Li et al. (2021) to obtain hard correspondence. Specifically, we use the point coordinates of $\boldsymbol{p}_t^i$ and $\widehat{\boldsymbol{p}}_t^j$ to construct an affinity matrix $\mathcal{M}$ where $\mathcal{M}_{\boldsymbol{p}_t^i, \widehat{\boldsymbol{p}}_t^j}$ is defined as

$$\mathcal{M}_{\boldsymbol{p}_t^i, \widehat{\boldsymbol{p}}_t^j} = exp(\frac{\gamma}{d(\boldsymbol{p}_t^i, \widehat{\boldsymbol{p}}_t^j) + \epsilon}). \tag{26}$$

Here, γ is empirically set to 10, $\epsilon = 1e^{-8}$, and $d(\boldsymbol{p}_t^i, \widehat{\boldsymbol{p}}_t^j)$ is the Euclidean distance. Given $\mathcal{M}$, we perform an alternating row and column normalization for a few iterations (e.g., five iterations), which yields a doubly stochastic assignment matrix $\mathcal{A}$. The soft correspondence function φ_{soft} then reads $\varphi_{soft}(\boldsymbol{p}_t^i, \widehat{\boldsymbol{P}}_t) = a_i / |a_i|_1 \widehat{\boldsymbol{P}}_t$ where a_i is the i-th row of $\mathcal{A}$.

To handle outliers, we follow Chui and Rangarajan (2003), Gojcic et al. (2021), Yew and Lee (2020) and add an additional row and column of ones (a slack row and column) to the original input of Sinkhorn normalization (the matrix $\mathcal{M}$) while only performing the alternating row and column normalization on non-slack rows and columns to obtain a resulting matrix $\hat{\mathcal{A}}$. To generate φ, we follow Li et al. (2021) to modify each row of $\hat{\mathcal{A}}$ by setting its column element with the maximum value to 1 and the remaining element to 0 such that the point with the highest transport score is selected as the corresponding point in this row.

We average all the ℓ_2 distances between all pairs of points except those which are assigned to slack columns. These are denoted as $\boldsymbol{P}_t^{valid} \subset \boldsymbol{P}_t$ respectively and then the Sinkhorn Distance (SD) evaluation metric is defined as

$$\mathcal{D}_{SD}(\boldsymbol{P}_t, \widehat{\boldsymbol{P}}_t) = \frac{1}{|\boldsymbol{P}_t^{valid}|} \sum_{\boldsymbol{p}\in\boldsymbol{P}_t^{valid}} \|\boldsymbol{p} - \varphi(\boldsymbol{p}, \widehat{\boldsymbol{P}}_t)\|^2. \tag{27}$$

We use the ADE, FDE, CD, and EMD metrics for evaluating SPF tasks on SFT3D and VKS datasets. When we move to the SAG dataset, we report both CD, EMD, and SD metrics because no ground truth annotation is available. Also, the introduced SD is used to downweight outliers. Note that SPCM-Net uses the CD as the learning objective on the SAG dataset.

5 Experiments

In this section, our main goal is to present experimental results evaluating SPCM-Net and relevant baselines on the proposed SFT3D, VKS, and SAG datasets.

To that end, we first evaluate SPCM-Net and relevant models on supervised SSFE and supervised SPF on the SFT3D dataset (Sect. 5.2). Then, we evaluate the two taskson the VKS dataset with the same models (Sect. 5.3). Next, we evaluate SPCM-Net on the self-supervised SPF task with the SAG dataset (Sect. 5.4). Finally, we demonstrate the benefit of our recurrent cost volume approach to modeling point cloud sequences in a controlled experiment using the standard KITTI scene flow dataset (Sect. 5.5).

5.1 Implementation Details

We implemented all the developed models in PyTorch (Paszke et al. 2019) using distributed training with 8 GPUs. Training each model generally takes 1-2 days. For SFE models (such as FlowNet3D) built upon TensorFlow (Abadi et al. 2016), we converted their pre-trained model weights into Pytorch and achieved identical performance. For most of the experiments, we set the learning rate and weight decay as

0.001 and 0.0001, respectively. We trained the models for 400 epochs while decaying the learning rate of each parameter group by 0.1 every 100 epochs. The gradient clip technique was applied to normalize the gradients. We didn't use any data augmentation strategy (such as rotation and scaling). We will release training and evaluation code for the new benchmark and models to facilitate future research at https://github.com/BestSonny/SPCM.

5.2 Sequential FlyingThings3D Dataset (SFT3D)

5.2.1 Supervised SSFE

Models We created several models to compare against the proposed SPCM-Net by directly adapting prior SFE approaches for frame pairs. We selected three representative state-of-the-art architectures that are publicly available for a comprehensive evaluation, namely FlowNet3D (Liu et al. 2019b), PointPWC-Net (Wu et al. 2020b), and FLOT (Puy et al. 2020). Originally, these models only support scene flow estimation between two consecutive frames. FlowNet3D and FLOT were trained with the FlyingThings3D dataset (FT3D) prepared by Liu et al. (2019b). We train PointPWC-Net using the same dataset. These models are denoted as **MODEL_NAME + FT3D**.

To report the performance for these models on our new SFT3D dataset, we pass every two consecutive frames of each point cloud sequence to obtain the predicted scene flows (e.g., four-step scene flow estimation for a point cloud sequence of length five). To ensure a fair comparison, we customized these methods to support SSFE by making n-step predictions and retraining them with the training split of the SFT3D dataset. We used the validation split to select the best models. These models are denoted as **MODEL_NAME + SFT3D**. The same setting of training and evaluation ensures a fair comparison between SPCM-Net and these models.

Results All results are aggregated and shown in Table 2. All models trained with the pair-wise FT3D dataset achieve limited performance on the SFT3D dataset. After customizing these models and re-training them on SFT3D, we observe consistent improvement in performance. The improvement reflects that the extra supervision from multiple frame pairs of a sequence provides a stronger learning signal compared to single-frame-pair supervision. However, these models independently conduct standard scene flow estimation on all frame pairs in point cloud sequences, ignoring multi-step spatiotemporal information. This has been addressed by SPCM-Net, which can recurrently process point cloud sequences.

SPCM-Net surpasses baseline methods significantly on various metrics. On both validation and test splits of the SFT3D dataset, SPCM-Net shows a clear improvement over the best models. It obtains EPE3D scores of 0.108 and 0.157 on the validation and test split, respectively, improving the best results of relevant models, i.e., **PointPWC-Net + SFT3D** and **FlowNet3D + SFT3D**. The improvements become larger when we measure the accuracy of scene flow prediction and the number of outlier predictions. For example, on the test split, SPCM-Net achieves a Acc3DS score of 0.380 and a Acc3DR score of 0.659, largely outperforming FlowNet3D, FLOT, and PointPWC-Net.

Specifically, it surpasses the previous best result on every single metric: $> 6\%$ on Acc3DS (higher is better), $> 2\%$ on Acc3DR (higher is better), and $> 5\%$ on Outliers3D (lower is better). This indicates that our framework can handle sequences in a more principled way by recurrently processing a point cloud sequence, thus capturing scene dynamics across a longer temporal length. Compared to baselines designed for pair-wise frames, the capability to utilize multi-step information reduces outlier predictions. Overall, our model demonstrates an ability to utilize historical point-wise motion patterns derived from spatiotemporal neighborhoods of points across frames. We further verify this by visualizing model predictions on the SFT3D dataset, as shown in Fig. 8.

5.2.2 Supervised SPF

Models We evaluate the prediction task by comparing against several prediction baselines: (1) **PointNet++** (Qi et al. 2017b) **+ LSTM**. We established a simple baseline by converting each point cloud into a global feature vector via a pooling layer. To learn the temporal dynamics and propagate it to the future, we use a standard fully-connected LSTM network to process the global feature vectors of the past input frames. At each timestep, the output feature of the LSTM will be broadcasted to each point and combined with the local point feature similar to the segmentation network in PointNet++. The model output will be the motion offsets of future points. (2) We used a recent preprint work called **PointRNN** (Fan and Yang 2019), which essentially extends the *flow embedding* layer in FlowNet3D (Liu et al. 2019b) to a recurrent model to support future prediction. We are unable to include the SPFNet architecture (Weng et al. 2020) in our evaluation as code has not been released for it at the current time, but we aim to add it to our benchmark in the near future.

We evaluate two variants of the proposed SPCM-Net—one trained from scratch (**SPCM-Net**) and one fine-tuned on SPF after pre-training on SSFE (**SPCM-Net + Pretrained**). The first one ensures a fair comparison to the baselines while the second one explores whether pretraining on the SSFE task helps the SPF task. All the models were trained with our proposed SFT3D dataset.

Results Table 3 reports the future prediction results on validation and test splits. Compared to other baseline approaches, our SPCM-Net achieves lower ADE, FDE, CD, and EMD under the same setting of training from scratch. Addition-

Table 2 Multi-step representation achieves improved performance on the SSFE task for point cloud sequences longer than two frames

SFT3D validation split							
Method	EPE3D↓	Acc3DS↑	Acc3DR↑	Outliers3D↓	RectOutliers3D↓	EPE2D↓	Acc2D↑
FlowNet3D (Liu et al. 2019b) + FT3D	0.200	0.173	0.494	0.783	0.764	11.803	0.319
FLOT (Puy et al. 2020) + FT3D	0.173	0.304	0.606	0.649	0.630	10.145	0.412
PointPWC-Net (Wu et al. 2020b) + FT3D	0.190	0.318	0.615	0.635	0.625	11.178	0.415
FlowNet3D (Liu et al. 2019b) + SFT3D	0.136	0.314	0.692	0.614	0.595	7.939	0.466
FLOT (Puy et al. 2020) + SFT3D	0.159	0.331	0.639	0.619	0.600	9.513	0.440
PointPWC-Net (Wu et al. 2020b) + SFT3D	0.115	0.455	0.760	0.502	0.483	7.210	0.548
SPCM-Net (Ours)	**0.108**	**0.484**	**0.782**	**0.468**	**0.450**	**6.709**	**0.567**
SFT3D test split							
Method	EPE3D↓	Acc3DS↑	Acc3DR↑	Outliers3D↓	RectOutliers3D↓	EPE2D↓	Acc2D↑
FlowNet3D (Liu et al. 2019b) + FT3D	0.191	0.169	0.494	0.792	0.769	11.743	0.320
FLOT (Puy et al. 2020) + FT3D	0.183	0.287	0.583	0.676	0.653	11.364	0.398
PointPWC-Net (Wu et al. 2020b) + FT3D	0.178	0.321	0.606	0.654	0.640	11.313	0.418
FlowNet3D (Liu et al. 2019b) + SFT3D	**0.150**	0.278	0.636	0.676	0.652	**9.327**	0.432
FLOT (Puy et al. 2020) + SFT3D	0.172	0.310	0.612	0.651	0.628	10.763	0.422
PointPWC-Net (Wu et al. 2020b) + SFT3D	0.174	0.320	0.609	0.656	0.633	11.087	0.420
SPCM-Net (Ours)	0.157	**0.380**	**0.659**	**0.597**	**0.575**	10.204	**0.473**
VKS dataset							
Method	EPE3D↓	Acc3DS↑	Acc3DR↑	Outliers3D↓	RectOutliers3D↓	EPE2D↓	Acc2D↑
FlowNet3D (Liu et al. 2019b)	0.0584	0.7178	0.8819	0.4628	0.2622	2.4355	0.8119
FLOT (Puy et al. 2020)	0.0672	0.7622	0.8638	0.3944	0.2095	2.5508	0.8354
PointPWC-Net (Wu et al. 2020b)	0.0458	0.8085	0.8990	0.3700	0.1793	1.9018	0.8591
SPCM-Net (Ours)	**0.0454**	**0.8330**	**0.9121**	**0.3520**	**0.1634**	**1.7578**	**0.8836**

We verify it by comparing supervised SSFE results against the adapted prior arts on our SFT3D validation and test splits and VKS dataset. 'FT3D' denotes training with the pair-wise FlyingThings3D dataset (FT3D) prepared by Liu et al. (2019b). Bold values indicate the best performance

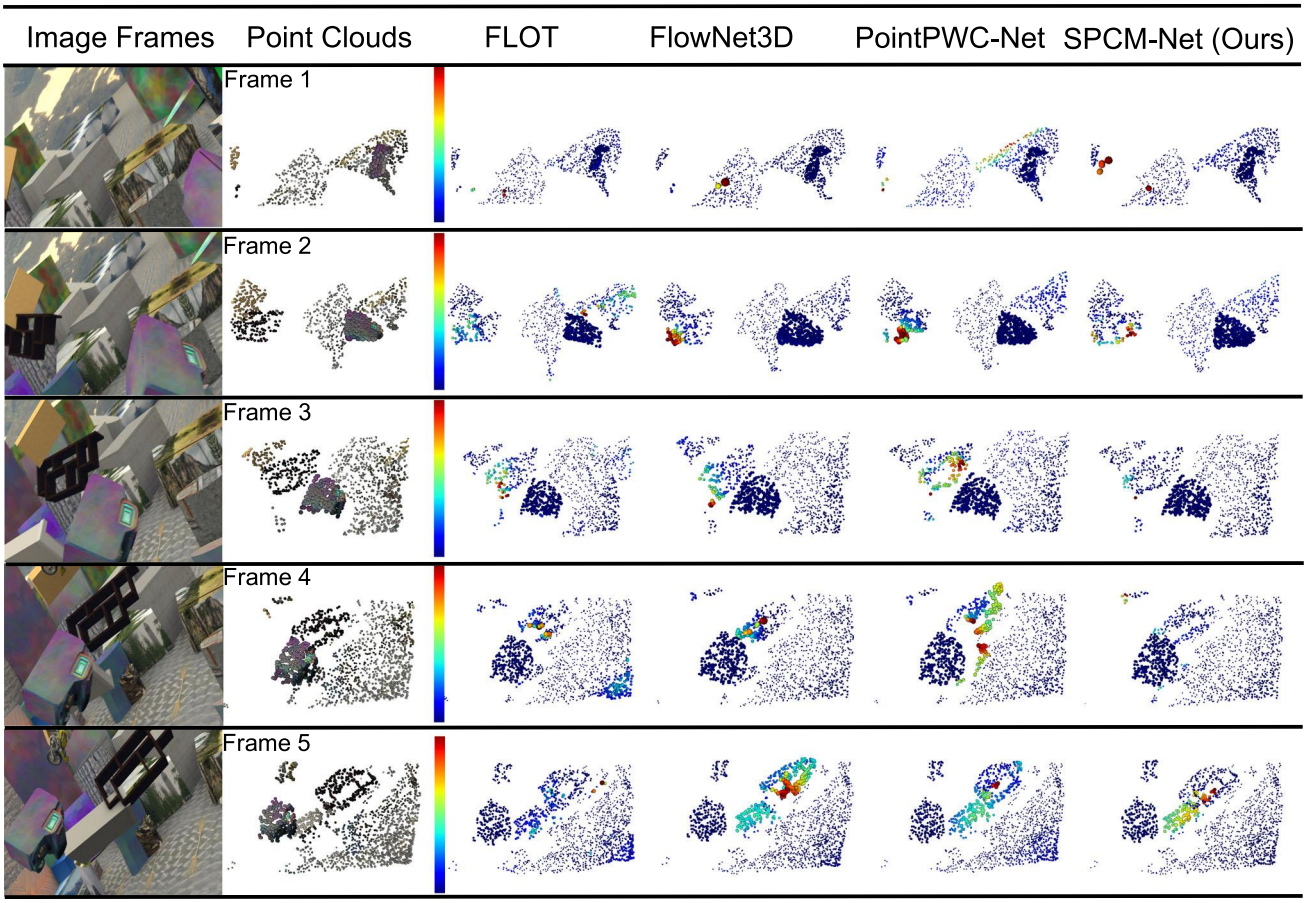

Fig. 8 The SPCM-Net can generate accurate scene flow estimation while largely reducing the outlier prediction, due to its capability of extracting multi-step information from sequences. Points can leverage their historical motion patterns in nearby locations to estimate more accurate and robust scene flow. Qualitative comparisons between SSFE results of different models on our SFT3D dataset. The 'Image Frames' column visualizes one sample sequence of the original FlyingThings3D dataset. The 'Point Clouds' column shows the corresponding point cloud sequences created after adding ground truth scene flows to themselves. All the models are trained with our SFT3D dataset. The error heatmaps illustrate the estimation results. The errors gradually increase from dark blue to dark red. Best viewed in color

Table 3 SPCM-Net can be adapted to support the SPF task while achieving a superior performance

Method	SFT3D validation split				SFT3D test split			
	ADE ↓	FDE↓	CD↓	EMD↓	ADE ↓	FDE↓	CD↓	EMD↓
PointNet++(Qi et al. 2017a) + LSTM	0.6740	1.0045	0.4603	0.9495	1.4017	2.1717	0.8878	2.0773
PointRNN (Fan and Yang 2019)	0.5605	0.8022	0.3715	0.7998	0.7607	1.1275	0.4783	1.0996
SPCM-Net (Ours)	**0.5069**	**0.7893**	**0.3300**	**0.7364**	**0.6286**	**0.9769**	**0.3848**	**0.9155**
SPCM-Net (Ours) + Pretrained	**0.2488**	**0.3886**	**0.1737**	**0.3408**	**0.3978**	**0.6373**	**0.2514**	**0.5684**

Pretraining on the SSFE task helps the SPF task. We show supervised SPF results on our SFT3D datasets. All models are trained from scratch except the model (SPCM-Net (Ours) + Pretrained). 'Pretrained' denotes that it is finetuned on the model pretrained in the task of supervised SSFE. Bold values indicate the best performance

ally, we verify that using the pre-trained weights obtained from the SSFE task to initialize the model is beneficial. It further reduces the displacement errors and obtains better future predictions. For example, on both validation and testing splits, compared to the best result of SPCM-Net, SPCM-Net + Pretrained further reduces the ADE and FDE to 0.2488 and 0.3978, achieving a significant decrease of 50.92% and 36.72%, respectively. The CD and EMD also reflect that a significant improvement has been achieved.

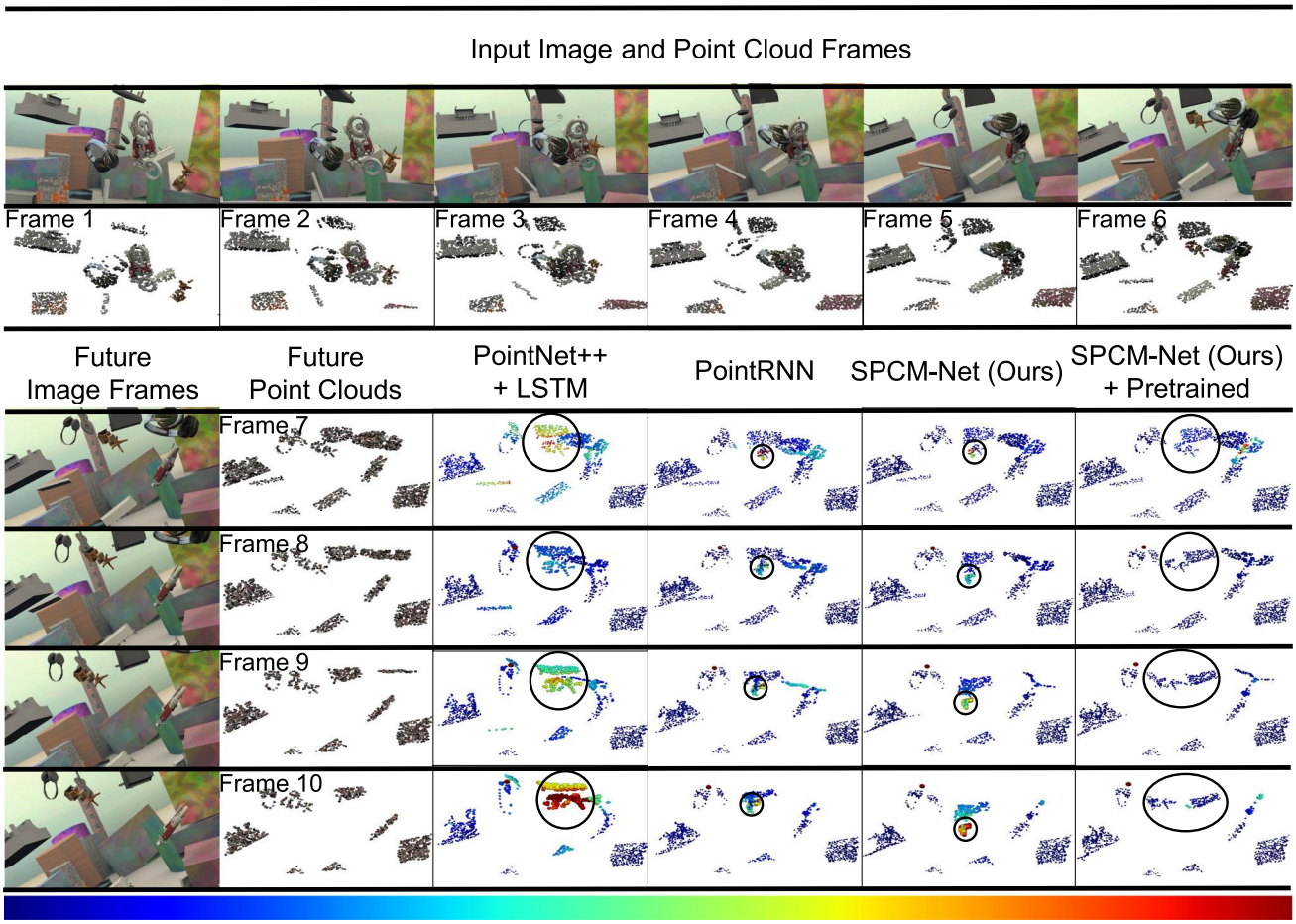

Fig. 9 Qualitative comparisons between SPF results of different models on our SFT3D dataset. Top rows show the original input image frames and our constructed point cloud frames. The bottom rows show the prediction results. The error heatmaps reflect the average displacement errors (ADE) of points. The errors gradually increase from dark blue to dark red. We show that performing pre-training on SSFE, followed by fine-tuning on the SPF task, helps boost the performance of SPF compared to training from scratch, which is verified by the circled results. 'SPCM-Net (ours) + Pretrained' successfully predicts the future motion of the lamp with the wooden mount while all models trained from scratch have failed. Best viewed in color

We visualize a sample generated by each model in Fig. 9. We see that SPCM-Net + Pretrained achieves the best future prediction. In particular, it successfully predicts the motion of the lamp with the wooden mount while all other models have failed.

5.3 Virtual KITTI Sequence (VKS) Dataset

The VKS dataset further allows us to evaluate models on simulated environments for the real-world, providing a preliminary prototype evaluation for traffic scenes. It provides accurate ground truth scene flows and future movements of multiple moving objects that are rather difficult to obtain in real-world settings.

5.3.1 Supervised SSFE

Baselines The evaluated models are very similar to the models used in the SFT3D dataset except for one difference. Both PointPWC-Net and SPCM-Net have changed to ball-query-based neighbor search instead of K-nearest neighbor search for better performance. This follows the practice of Qi et al. (Qi et al. 2017b) to maintain a fixed region scale and highlight local region patterns. All models are first pre-trained on the SFT3D dataset then fine-tuned on the VKS dataset.

Results The results are shown in Table 2. We find that SPCM-Net outperforms other baseline methods on all the metrics including EPE3D, Acc3DS, Acc3DR, Outliers3D, RectOutliers3D, and EPE2D, and Acc2D. The main improvements are seen in the accuracy scores (reflected by Acc3DS and Acc3DR) and reductions in outlier predictions. FLOT (Puy et al. 2020) performs slightly worse than we

Table 4 Our general *SPCM-Net* architecture achieves competitive SPF results compared to tailored state-of-the-art models under both supervised and self-supervised settings

Method	VKS (supervised)				SAG (self-supervised)		
	ADE ↓	FDE↓	CD↓	EMD↓	CD↓	EMD↓	SD↓
PointNet++ (Qi et al. 2017b) + LSTM	1.1747	1.9278	0.7260	1.4071	2.0718	2.5574	2.4363
PointRNN (Fan and Yang 2019)	0.2856	**0.4655**	0.1551	0.3575	**1.2322**	2.3160	**1.3630**
SPCM-Net (Ours)	**0.2768**	0.4799	**0.1400**	**0.3418**	1.3453	**2.2992**	1.4845

Bold values indicate the best performance

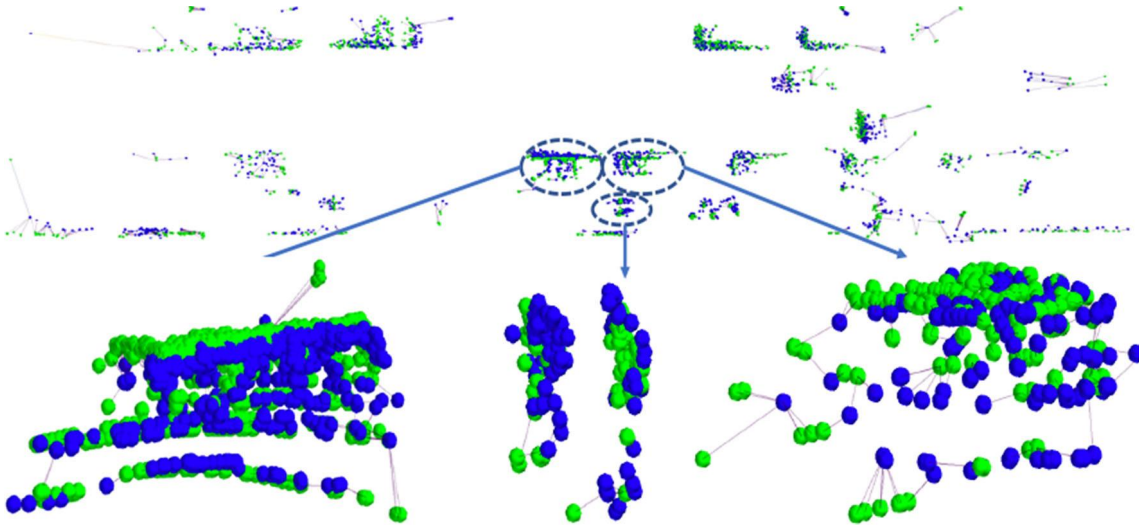

Fig. 10 Visualization of SAG future prediction with our SPCM-Net. Top: Top view of the whole scene. Bottom: Zoomed in areas. green points are future points at frame 10 while blue points are the predicted point cloud. Lines connect the nearest neighboring points between green points and blue points, which are obtained by the Chamfer distance. Shorter-length lines are preferred since they reflect lower errors. Best viewed in color

initially expected. We suspect that the K-nearest neighbor search used in FLOT could potentially degrade the performance, due to its learned features that are less generalized in space.

5.3.2 Supervised SPF

Models The baselines are the same as for the SFT3D dataset. Our SPCM-Net uses a ball-query-based neighbor search. All models are first pre-trained on the supervised SPF task with the SFT3D dataset and then fine-tuned on the VKS dataset.

Results Quantitative results are listed in Table 4. We show that SPCM-Net achieves a comparable performance compared to PointRNN (Fan and Yang 2019). Interestingly, PointNet++ (Qi et al. 2017b) + LSTM performs significantly worse. Because it uses the global fully-connected feature as the spatiotemporal representation, it struggles to capture the dynamics of local regions in the VKS dataset.

A visualization of predictions made by our SPCM-Net is provided in Fig. 11. It makes reasonably good predictions, and overall we achieve competitive performance to the prior art PointRNN. We show some typical errors made

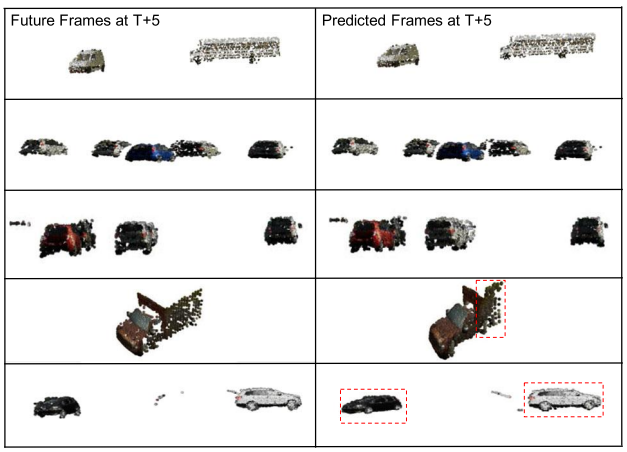

Fig. 11 Visualization of VKS future prediction with our SPCM-Net. Left: the ground truth future frames at the end of prediction period (the 5th future frames). Right: the predicted frames. Best viewed in color

by SPCM-Net. When predicting the future movement of the truck objects, the model shows that it lacks awareness of the physical constraints. Another challenge is that when a vehicle makes a turn, the model struggles to capture the precise movement (i.e., orientation, speed) of the vehicle. This could be found in the black car example in Fig. 11. We encourage the community to further explore this problem by investigating advanced topics in generative models (Achlioptas et al. 2018; Wang et al. 2020a) and physical scene understanding (Yao et al. 2018).

5.4 Sequential Argoverse (SAG) Dataset

To explore the performance in real-world datasets, we train and evaluate models on the SAG dataset. The baselines are the same as models used in the supervised SPF VKS experiment, except that we use the self-supervised SPF objective to guide model learning.

Results Results are shown in the right section of Table 4. SPCM-Net achieves a competitive result compared to other baselines. Figure 10 shows visual results. The overall prediction is relatively good with high fidelity. However, we notice that the ground truth points formulated by randomly sampling points in a scene contain outliers (lines of long length in Fig. 10). This can explain high CD and EMD scores. Both PointRNN and SPCM-Net achieve competitive SD results. We encourage future work to revisit the point sampling process by focusing on object points. However, this likely requires applying extra unsupervised object segmentation (Landrieu and Simonovsky 2018) or object discovery techniques (Karpathy et al. 2013). Also, it is promising to map point clouds to a latent space where a robust distance metric can be computed, as evidenced in a recent work (Zuanazzi et al. 2020) using adversarial learning.

5.5 KITTI Scene Flow Dataset

We conduct an experiment on standard scene flow estimation using the KITTI Scene Flow benchmark with the multi-frame setup of MeteorNet (Liu et al. 2019c) to demonstrate a direct comparison between their architecture and SPCM-Net. We consider MeteorNet as the 4D extension of PointNet++ (Qi et al. 2017b) since it appends a 1D temporal coordinate to the 3D spatial coordinates. The resulting 4D coordinates help find spatiotemporal neighbors and extract features in 4D space to handle point cloud sequences.

Experimental setup The KITTI scene flow dataset (Menze and Geiger 2015) provides ground truth disparity maps and optical flows for 200 frame pairs, from where the 3D ground truth scene flow can be constructed. Among 200 frame pairs, only 142 provides the corresponding mapping to laser point clouds. MeteorNet (Liu et al. 2019c) further extends the dataset to use preceding point cloud frames. As a result, the task aims to predict one-step scene flow of each frame pair while taking a point cloud sequence as the input (recall from Fig. 1b). The first 100 of the 142 frames are used to fine-tune the models while the remaining 42 sequences are used for testing. We used the same dataset prepared and released by Liu et al. (2019c) for training and evaluation.

Results We train and evaluate two model variants of SPCM-Net with three and four preceding frames as input to match MeteorNet's results. Increasing the number of frames

Table 5 Flow estimation results on KITTI sceneflow dataset

Method	Frames	Mean	SD
FlowNet3D	2	0.287	0.250
MeteorNet (direct)	3	0.282	0.204
MeteorNet (direct)	4	0.263	0.210
MeteorNet (chained-flow)	3	0.277	0.244
MeteorNet (chained-flow)	4	0.251	0.227
SPCM-Net (ours)	3	**0.229**	**0.184**
SPCM-Net (ours)	4	**0.194**	**0.174**

Metrics are the mean and standard deviation of the end-point-error (EPE) of scene flow. Bold values indicate the best performance

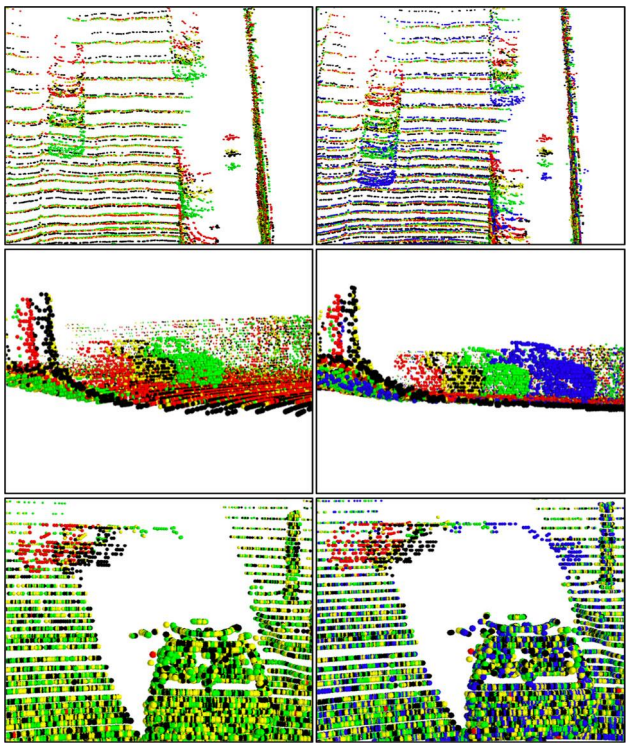

Fig. 12 Visualization of SPCM-Net example results on the KITTI scene flow dataset with three (left) and four (right) preceding frames as input. Different colors denotes points from a different frame number: frame $t-3$, frame $t-2$, frame $t-1$, frame t, and the translated points in black (frame $t-3$ + predicted scene flows). The translated points should highly overlap to points of frame $t-2$ for a good estimation. Left-column and right-column images show the results of using 4 and 3 input frames, respectively. Best viewed in color

from three to four achieves a consistent performance gain as expected. SPCM-Net significantly improves the results compared to the previous state-of-the-art method MeteorNet. Specifically, SPCM-Net achieves lower mean errors of 0.229 and 0.194 with three and four preceding frames, decreasing the errors of MeteorNet by 17.33% and 22.71%, respectively. The use of a recurrent cost volume to processes a point cloud sequence shows a better capability of extracting motion patterns for scene flow estimation than the 4D convolution approach of MeteorNet. Figure 12 visualizes some examples of the predicted scene flows.

6 Related Work

6.1 Sequential Point Cloud Processing

Our work is related to techniques for sequential point cloud processing. This area is fairly new but is a critical step towards general learning of scene dynamics. To effectively process point cloud sequences, Fast and Furious (FaF) (Luo et al. 2018) proposes a network that jointly tackles 3D detection, tracking, and motion forecasting with a birds-eye view representation of point clouds and 3D convolutions. MinkowskiNet (Choy et al. 2019) invents a generalized sparse tensor-based computing framework that allows handling point cloud sequences with 4D sparse convolutions. However, directly applying 4D convolutions on a voxelized sequence requires extensive computation. Furthermore, quantization errors during voxelization may cause performance drops when tackling problems requiring precise measurement, e.g., point-wise scene flow estimation. Occupancy Flow (Niemeyer et al. 2019) learns a temporally and spatially continuous vector field to perform the 4D reconstruction. More recently, PSTNet (Fan et al. 2021b) designs a point spatiotemporal convolution to compute features from point cloud sequences with a hierarchical design. They apply their architecture to 3D action recognition. 3DV (Wang et al. 2020b) proposes to use 3D dynamic voxels as the motion representation for depth videos and utilizes PointNet++ (Qi et al. 2017b) for feature abstraction. Prantl et al. (2019) presented a new deep learning method that aims at capturing stable and temporally coherent features from point cloud sequences, via a novel temporal loss that extends the EMD loss to minimize the difference between estimated and ground-truth super-resolution point clouds in higher orders, i.e., positions, velocities, and accelerations. They introduced an additional *mingling* loss term to push the individual points of a group apart, avoiding temporal mode collapse. Their method can be potentially adapted for our defined tasks as it could produce smooth motion for point cloud sequences. Rempe et al. (2020) made an important step to aggregate and encode spatio-temporal changes of objects from point cloud sequences and learn Canonical Spatiotemporal Point Cloud Representation (CaSPR) via a Latent ODE approach. Their technique has been applied to various applications such as reconstruction, camera pose estimation, and correspondence estimation. P4Transformer (Fan et al. 2021a) introduced a new transformer-like network to model raw point cloud videos, where a novel point 4D convolution has been proposed to efficiently encode spatio-temporal local structures. P4Transformer shows superior performance on various benchmarks including 3D action recognition (Li et al. 2010; Liu et al. 2019; Shahroudy et al. 2016) and 4D semantic segmentation (Choy et al. 2019).

Recently, a collection of work has focused on learning scene dynamics from sequences of point clouds. We consider MeteorNet (Liu et al. 2019c) as a 4D extension of PointNet++ (Qi et al. 2017b) to handle temporal point cloud sequences by appending a 1D temporal coordinate to the 3D spatial coordinates. The resulting 4D coordinates help find spatiotemporal neighborhoods in 4D space. Both direct and chained-flow grouping are proposed to consider the spatiotemporal interaction of points such as the maximum travel distance and the motion direction of points. We show that SPCM-Net's recurrent processing of sequences provides a stronger inductive bias compared to the 4D convolution of MeteorNet. MeteorNet is only evaluated on last-frame scene flow estimation at a relatively small scale, which is clearly distinct from the SSFE task proposed in this work. Their work also does not consider future prediction. PointRNN (Fan and Yang 2019) has been proposed to incorporate the *flow embedding* layer of FlowNet3D (Liu et al. 2019b) into a recurrent unit for predicting future point clouds. We consider the *flow embedding* layer as the *point-to-set* matching cost, in contrast to SPCM-Net's recurrent cost volume that adopts a *set-to-set* matching cost which is more robust to outliers (see Sect. 3.2). PointRNN failed to adequately formalize the SPF task which limited their evaluation. They only focus on self-supervised training and evaluation, whereas we consider both supervised and self-supervised SPF in addition to supervised SSFE. Moreover, our experiments confirmed that SPCM-Net's recurrent cost volume for propagating point-wise spatiotemporal features across time is a more robust matching cost than PointRNN's flow embedding cost. Another self-supervised architecture, SPFNet, is proposed to tackle the SPF problem on self-driving datasets (Weng et al. 2020). They investigated both point-based and range-map-based encoders to extract useful information from past frames, followed by LSTM-based decoders to predict future scene point clouds. The point-based encoder is similar to PointNet (Qi et al. 2017a) to extract a global feature from each point cloud frame while the range-map-based encoder is only suitable for processing of LIDAR point clouds due to its range map representation. Their paper focused on tra-

jectory forecasting and provided limited evaluation of the self-supervised SPF task.

6.2 Scene Flow Estimation

To estimate motion between frames, previous work (Dosovitskiy et al. 2015; Hui et al. 2018; Teed and Deng 2020) tends to follow traditional optical flow approaches such as energy minimization (Horn and Schunck 1981) or warping-based methods (Brox et al. 2004; Bruhn et al. 2005). FlowNet is a generic deep learning architecture for optical flow estimation (Dosovitskiy et al. 2015) that correlates feature vectors of image pairs at different image locations. Since then, new work has been proposed to further improve its performance (Ilg et al. 2017; Ranjan and Black 2017; Sun et al. 2018).

Prior to the study of 3D scene flow estimation, flow estimation was concerned with motion across pairs of images. The three-dimensional scene flow is initially described in (Vedula et al. 1999) as a 3D extension of 2D optical flow, where they formulate the estimation problem as a factor-graph-based energy minimization problem with hand-crafted SHOT descriptors (Tombari et al. 2010) for correspondence. Early work on scene flow estimation used multi-view geometry to associate salient image key points (Vedula et al. 1999). The problem has also been addressed by jointly optimizing registration and motion (Huguet and Devernay 2007; Pons et al. 2007).

Recently, deep learning models have been proposed to estimate LiDAR flow with parametric continuous convolution layers (Wang et al. 2018a). The *flow embedding* layer, introduced in FlowNet3D (Liu et al. 2019b), encodes motion between two consecutive point clouds and has achieved competitive performance. HPLFlowNet (Gu et al. 2019) instead estimates motion by converting point clouds into permutohedral lattices with bilateral convolutions to aggregate features. PointPWC-Net (Wu et al. 2020b) presents an end-to-end deep scene flow model to conduct scene flow estimation in a coarse-to-fine fashion. FLOT (Puy et al. 2020) finds the point correspondences between two points by adapting optimal transport with relaxed transport constraints to handle real-world imperfections. In Mittal et al. (2020), the authors present a self-supervised approach based on nearest neighbors and cycle consistency with competitive results compared to supervised scene flow approaches. In Pontes et al. (2020), scene flow from point clouds is recovered and regularized with graph Laplacian (Bobenko and Springborn 2007). Our work draws upon model designs of the scene flow approaches while making further innovations to support recurrent processing of point cloud sequences to solve the SSFE and SPF tasks.

6.3 Deep Learning on 3D Point Clouds

Extensive research projects have been undertaken to develop modeling techniques aimed at automatically understanding 3D scenes and objects for numerous applications, such as 3D object classification (Wang et al. 2019b; Klokov and Lempitsky 2017; Li et al. 2018a; Qi et al. 2017a, b), 3D object detection (Qi et al. 2018; Shi et al. 2019), 3D semantic labeling (Choy et al. 2019; Graham et al. 2018; Landrieu and Simonovsky 2018; Su et al. 2018), and 3D instance segmentation (Pan et al. 2020; Pham et al. 2019; Yang et al. 2019). Most prior work relies on transforming 3D data into regular representations such as voxels (Wu et al. 2015) or 2D grids (Su et al. 2015) for processing. Here, context aggregation can be achieved easily with convolutions at relatively low resolutions due to the expensive computational overhead and memory footprint. To mitigate the issue, we have seen architectures such as OctNet (Riegler et al. 2017) and permutohedral lattice representations (Su et al. 2018) being proposed to achieve efficient memory allocation and computation without compromising resolution.

Recently, new work has emerged that directly processes raw and irregular point clouds (Wang et al. 2019b; Qi et al. 2017a, b; Pham et al. 2019) by applying MLPs in a point-wise fashion. To further capture local structures, follow-ups (Wang et al. 2019b; Qi et al. 2017b; Shen et al. 2018; Thomas et al. 2019) have defined pseudo-convolutional operators where convolutions are instantiated as continuous kernels, assuming a continuous space for point clouds. However, this incurs an extra cost due to the use of greedy nearest neighbor search and point sampling algorithms for hierarchical processing. More recently, sparse tensor-based point cloud processing has been proposed to conduct sparse convolutions only on non-empty locations. Popular frameworks, such as SparseConvNet (Graham et al. 2018), MinkowskiEngine (Choy et al. 2019), and TorchSparse (Tang et al. 2020), can conduct the sparse convolutions very efficiently based on their fast indexing structure.

Up to now, the majority of this body of work is aimed at processing static point clouds. Less progress has been made on dynamic point cloud modeling, especially the motion estimation and future prediction of point cloud sequences, which is the focus of this paper.

6.4 Self-supervised Learning

Deep learning models have demonstrated the ability to obtain discriminative embeddings with unsupervised learning without providing any external supervision, i.e., supervision signals are generated from data itself (Doersch et al. 2015; Lee et al. 2009; Srivastava et al. 2015). These representations could be used in downstream tasks or as strong initialization for supervised tasks. In point cloud processing, we have

seen several works attempting to jointly learn multiple tasks including depth estimation, optical flow estimation, ego-motion estimation, and camera pose estimation based on 2D images (Lee and Fowlkes 2019; Yin and Shi 2018; Zou et al. 2018). Other self-supervised point cloud tasks include point set generation (Fan et al. 2017), point cloud auto-encoder (Yang et al. 2018), point set registration (Fitzgibbon 2003). We refer the interested reader to Guo et al. (2019) for a comprehensive examination. In this paper, we have utilized geometric losses, i.e., chamfer distance and earth mover's distance (Rubner et al. 2000), for self-supervised learning of future prediction from point cloud sequences. Advanced techniques such as Laplacian regularization or local smoothness (Pontes et al. 2020; Wu et al. 2020b) could be further utilized to regularize the learning and improve the performance.

6.5 Spatiotemporal Learning

RNNs and their variants are widely used in sequence prediction while having difficulty in applying directly to structured data such as videos due to the ignorance of handling the spatial arrangement of data. To address this, the convolutional LSTM (ConvLSTM) (Xingjian et al. 2015) is proposed to capture local spatial correlations and replace the fully connected layer in the recurrent state transition with a convolution operation. The spatiotemporal LSTM (Wang et al. 2017) further extends ConvLSTM by introducing a novel recurrent unit that can deliver memory states both vertically (across recurrent layers) and horizontally (across time). Numerous follow-up works have been proposed along this direction (Wang et al. 2018c, 2019a). In CubicLSTM (Fan et al. 2019), the authors extend the ConvLSTM by utilizing two states (the temporal state and the spatial state) with independent convolutions. All of these LSTMs can be applied to spatiotemporal data. However, additional designs (see Sect. 3) are required to apply them to 3D point cloud sequences because the point cloud sequences are unstructured and orderless. Our work demonstrates one way in which these methods can be applied to point cloud sequences.

7 Discussion

Our experimental results showed that SPCM-Net achieves superior performance compared to state-of-the-art SFE models on the SSFE task by leveraging temporal coherence of points over many frames. We attribute this to the recurrent cost volume layer, which effectively propagates point-wise spatiotemporal information across time. Empirically, we observed a large reduction in outlier predictions which helped improve the overall scene flow estimation performance. SPCM-Net also produces competitive SPF results under supervised and self-supervised settings compared to the best prior model. We achieved state-of-the-art SPF performance by first pre-training on the SSFE task before fine-tuning on SPF.

This evaluation is conducted on a newly introduced benchmark for SSFE and SPF. As shown by our qualitative results, the ground truth supervision provided for the two synthetic datasets SFT3D and VKS enables a rigorous and principled comparison between competing models. Both datasets offer unique challenges for future study; in particular, the VKS dataset contains multiple dynamic objects in each frame and requires learning physical properties of vehicles for accurate estimation and prediction. The real-world SAG dataset also contains its own set of challenges, particularly related to handling outliers during training. We expect this benchmark to be pivotal for standardizing training and evaluation protocols for future work on SSFE and SPF.

7.1 Limitations and Future Work

In this work, our studied tasks focus on relatively low-level dynamic point cloud processing that aims to predict the point-wise motion of sequences. We expect that object category information and motion smoothness priors could help improve performance. This, however, requires defining novel tasks and proposing new benchmarks. Also, the self-supervised Chamfer distance objective used in our paper is designed for point matching between two point sets at each timestep. Objective functions that take into account temporal correspondence *across* frames, like those used for multi-frame data association in multi-object tracking (Emami et al. 2020), are needed to provide stronger supervision signals.

We can suggest further promising directions for future research. First, better modeling of occlusion is needed to further improve the scene flow estimation, e.g., Ouyang and Raviv (2020). Second, the self-supervised SPF remains an open problem. Currently, models make an assumption that points in the current predicted future frame are translated from points in the previous predicted frame with a motion offset. In real-world applications, due to the nature of how each point cloud is generated, this assumption does not hold. This leads to isolated points being improperly matched, which then leads to noisy training signals. A promising alternative to nearest-neighbor losses like Chamfer distance is adversarial learning (Zuanazzi et al. 2020) where point clouds are mapped to a latent space in which a robust distance metric can be computed.

8 Conclusion

In this paper, we introduced *sequential scene flow estimation* (SSFE) for point cloud sequences, which is a novel exten-

sion of the well-studied scene flow estimation task to multiple frames. We proposed the SPCM-Net architecture to solve this task as well as the related sequential point cloud forecasting (SPF) task. To help advance future research, we collected and presented a new benchmark consisting of three point cloud sequence datasets containing diverse backgrounds and multiple object motions in synthetic and realistic environments. Our benchmark uniquely contains ground truth annotations for multi-step scene flow which current SFE datasets lack, which should be pivotal to future research.

Acknowledgements This work is supported by NSF CNS 1922782, by the Florida Dept. of Transportation (FDOT) and FDOT District 5. The opinions, findings and conclusions expressed in this publication are those of the author(s) and not necessarily those of the Florida Department of Transportation or the National Science Foundation.

Declaration

Conflict of interest The authors declare that they have no conflict of interest.

A Appendix

A.1 Additional Evaluations

This section provides additional results supporting the evaluations presented in the Experiments.

A.1.1 Additional SPF Results on nuScenes

The nuScenes dataset (Caesar et al. 2019) is a large-scale public autonomous driving dataset, which contains 850 publicly available scenes in total collected in both Boston and Singapore, which are known for dense traffic and highly challenging driving situations. 15 h of driving data (242 km traveled at an average of 16km/h) was collected with the dataset containing 68 driving logs for training and 15 driving logs for testing. The LiDAR data was captured by a Velodyne 32-beam LiDAR.

The nuScenes dataset does not provide ground-truth annotations for scene flow. Therefore, we adopt metrics without requiring any annotation. Given the fact that forecasting future point clouds on real-world datasets is challenging due to rapid changes in the vehicle's surroundings, we focus on short-term prediction for the nuScenes dataset. With each driving log providing a point cloud sequence, we repeatedly sample from it by randomly choosing 10 successive point clouds for training and testing (consecutively sampling every other frame and repeating 10 times). To achieve reasonable computation times while staying within memory limits, we sample a fixed number of points for each frame and remove the ground points to reduce the bias caused by the flattened geometry of the ground. Specifically, 2048 points are randomly sampled from every point cloud frame in the sequence. Following the practice of (Wu et al. 2020a), the point clouds are cropped to extract the region defined by $[-32, 32] \times [-8, 8] \times [-1.3, 2]$ m, which corresponds to the XYZ range.

The experimental results of nuScenes are summarized in Table 7. Our SPCM-Net achieves a competitive result compared to other baselines in all metrics. One interesting observation is that both SPCM-Net and PointNet++ (Qi et al. 2017b) + LSTM outperform PointRNN (Fan and Yang 2019) significantly. We suspect that static points dominate on the nuScenes dataset such that most of the points only contain ego-motion. Therefore, it prefers models with a better capability of extracting global information.

Table 6 Evaluation results on VKS for the SSFE task with the goal to test generalization to unseen driving scenarios

VKS train split							
Method	EPE3D↓	Acc3DS↑	Acc3DR↑	Outliers3D↓	RectOutliers3D↓	EPE2D↓	Acc2D↑
FlowNet3D (Liu et al. 2019b)	0.0422	0.8790	**0.9718**	0.2043	0.0855	3.9368	0.9159
FLOT (Puy et al. 2020)	**0.0396**	0.8625	0.9316	0.2174	0.1118	2.5706	0.8939
PointPWC-Net (Wu et al. 2020b)	0.0588	0.9202	0.9542	**0.1960**	**0.0824**	2.4082	**0.9306**
SPCM-Net (Ours)	0.0618	**0.9212**	0.9556	0.2061	0.0877	**2.4044**	0.9180
VKS test split							
Method	EPE3D↓	Acc3DS↑	Acc3DR↑	Outliers3D↓	RectOutliers3D↓	EPE2D↓	Acc2D↑
FlowNet3D (Liu et al. 2019b)	0.0550	0.6877	**0.9123**	0.3974	0.2888	2.3364	0.8368
FLOT (Puy et al. 2020)	0.0689	0.7471	0.8708	0.3484	0.2506	3.4343	0.8360
PointPWC-Net (Wu et al. 2020b)	0.0478	0.7972	0.8993	0.3458	0.2440	2.2210	0.8556
SPCM-Net (Ours)	**0.0477**	**0.8106**	0.9082	**0.3298**	**0.2268**	**2.0846**	**0.8678**

Bold values indicate the best performance

Table 7 Evaluation results on nuScenes for the SPF task

Method	NuScenes		
	CD↓	EMD↓	SD↓
PointNet++ (Qi et al. 2017b) + LSTM	**0.6176**	0.8334	**0.6920**
PointRNN (Fan and Yang 2019)	0.9750	0.9878	1.0969
SPCM-Net (Ours)	0.6339	**0.7858**	0.7113

Bold values indicate the best performance

Table 8 Evaluation results on VKS for the SPF task with the goal to test generalization to unseen driving scenarios

Method	VKS train split			
	ADE ↓	FDE↓	CD↓	EMD↓
PointNet++ (Qi et al. 2017b) + LSTM	0.9971	1.6726	0.5783	1.2601
PointRNN (Fan and Yang 2019)	**0.2201**	**0.3292**	**0.1242**	**0.2856**
SPCM-Net (Ours)	0.2535	0.4327	0.1418	0.3315
Method	VKS test split			
	ADE ↓	FDE↓	CD↓	EMD↓
PointNet++ (Qi et al. 2017b) + LSTM	1.4951	2.4608	0.8209	1.8308
PointRNN (Fan and Yang 2019)	0.3334	0.5562	0.1798	0.4095
SPCM-Net (Ours)	**0.3291**	0.5853	0.1824	**0.4068**

Bold values indicate the best performance

A.1.2 Additional SSFE Results on VKS

In Sect. 5.3, we have evaluated the performance of VKS, providing a preliminary prototype evaluation for traffic scenes. Table 6 further supplements it with the evaluation results on the VKS dataset with a new split to report the generalization capability of unseen driving scenarios. Recall from Sect. 4.2 that the original virtual KITTI contains five scenes of crowded urban area (Scene01), busy intersections (Scene02, Scene06), long road in the forest (Scene18), and highway driving scene (Scene20), and we initially have chosen the train and test splits both containing examples of all scenes. We made a further exploration to create the train and test splits such that they do not have any overlap on scenarios. To do so, we held out the highway driving scene (Scene20) as the test split and use other scenes as the train split. This leads a total of 1876 train and 1474 test point cloud sequences. We didn't try other possible split combinations as it would end up with excessive number of experiments to run. Instead, we will provide the possibility to try other combinations in our implementation for future research.

As expected, the SSFE performance of all fully supervised methods drops slightly when moving from the training scenes to the unseen scenes (Table 6). Remarkably though, obtaining comparable or worse results on the training split, our SPCM-Net outperforms other methods on the unseen scenes, implying that it has a better generalization capability on the SSFE task.

A.1.3 Additional SPF Results on VKS

Similarly, we also evaluate the prediction performance on VKS following the same split as done in previous Sect. A.1.2. Our SPCM-Net still achieves a comparable performance compared to PointRNN (Fan and Yang 2019), drawing a similar conclusion as in Sect. 5.3.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). TensorFlow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)* (pp. 265–283).

Achlioptas, P., Diamanti, O., Mitliagkas, I., & Guibas, L. (2018). Learning representations and generative models for 3D point clouds. In *International conference on machine learning, PMLR* (pp. 40–49).

Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., & Savarese, S. (2016). Social LSTM: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 961–971).

Barrow, H. G., Tenenbaum, J. M., Bolles, R. C., & Wolf, H. C. (1977). Parametric correspondence and chamfer matching: Two new techniques for image matching. In *International joint conferences on artificial intelligence*.

Besl, P. J., & McKay, N. D. (1992). Method for registration of 3-D shapes. *Sensor Fusion IV: Control Paradigms and Data Structures, International Society for Optics and Photonics, 1611*, 586–606.

Bobenko, A. I., & Springborn, B. A. (2007). A discrete Laplace–Beltrami operator for simplicial surfaces. *Discrete & Computational Geometry, 38*(4), 740–756.

Brox, T., Bruhn, A., Papenberg, N., & Weickert, J. (2004). High accuracy optical flow estimation based on a theory for warping. In *European conference on computer vision* (pp. 25–36). Springer.

Bruhn, A., Weickert, J., & Schnörr, C. (2005). Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision, 61*(3), 211–231.

Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., & Beijbom, O. (2019). nuScenes: A multimodal dataset for autonomous driving. arXiv preprint. arXiv:1903.11027.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. (2015). ShapeNet: An information-rich 3D model repository. arXiv preprint. arXiv:1512.03012.

Chang, M. F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., et al. (2019). Argoverse: 3D tracking and forecasting with rich maps. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 8748–8757).

Choy, C., Gwak, J., & Savarese, S. (2019). 4D spatio-temporal ConvNets: Minkowski convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3075–3084).

Chui, H., & Rangarajan, A. (2000). A new algorithm for non-rigid point matching. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2*, 44–51.

Chui, H., & Rangarajan, A. (2003). A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding, 89*(2–3), 114–141.

Doersch, C., Gupta, A., & Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision* (pp. 1422–1430).

Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., & Brox, T. (2015). FlowNet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 2758–2766).

Eldar, Y., Lindenbaum, M., Porat, M., & Zeevi, Y. Y. (1997). The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing, 6*(9), 1305–1315.

Emami, P., Pardalos, P. M., Elefteriadou, L., & Ranka, S. (2020). Machine learning methods for data association in multi-object tracking. *ACM Computing Surveys (CSUR), 53*(4), 1–34.

Fan, H., & Yang, Y. (2019). PointRNN: Point recurrent neural network for moving point cloud processing. arXiv preprint. arXiv:1910.08287.

Fan, H., Su, H., & Guibas, L. J. (2017). A point set generation network for 3D object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 605–613).

Fan, H., Zhu, L., & Yang, Y. (2019). Cubic LSTMs for video prediction. In *Proceedings of the thirty-third conference on artificial intelligence*.

Fan, H., Yang, Y., & Kankanhalli, M. (2021a). Point 4D transformer networks for spatio-temporal modeling in point cloud videos. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 14204–14213).

Fan, H., Yu, X., Ding, Y., Yang, Y., & Kankanhalli, M. (2021b). PSTNet: Point spatio-temporal convolution on point cloud sequences. In *International conference on learning representations*.

Fitzgibbon, A. W. (2003). Robust registration of 2D and 3D point sets. *Image and Vision Computing, 21*(13–14), 1145–1153.

Gaidon, A., Wang, Q., Cabon, Y., & Vig, E. (2016). Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4340–4349).

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.*

Gojcic, Z., Litany, O., Wieser, A., Guibas, L. J., & Birdal, T. (2021). Weakly supervised learning of rigid 3D scene flow. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 5692–5703).

Graham, B., Engelcke, M., & van der Maaten, L. (2018). 3D semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 9224–9232).

Graves, A. (2012). Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks* (pp. 5–13). Springer.

Gu, X., Wang, Y., Wu, C., Lee, Y. J., & Wang, P. (2019). HPLFlowNet: Hierarchical permutohedral lattice FlowNet for scene flow estimation on large-scale point clouds. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3254–3263).

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., & Bennamoun, M. (2019). Deep learning for 3D point clouds: A survey. arXiv preprint. arXiv:1912.12033.

Gwak, J., Choy, C. B., & Savarese, S. (2020). Generative sparse detection networks for 3D single-shot object detection. In *European conference on computer vision*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Horn, B. K., & Schunck, B. G. (1981). Determining optical flow. *Artificial Intelligence, 17*(1–3), 185–203.

Hosni, A., Rhemann, C., Bleyer, M., Rother, C., & Gelautz, M. (2012). Fast cost-volume filtering for visual correspondence and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35*(2), 504–511.

Hou, J., Dai, A., & Nießner, M. (2019). 3D-SIS: 3D semantic instance segmentation of RGB-D scans. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*.

Huguet, F., & Devernay, F. (2007). A variational method for scene flow estimation from stereo sequences. In *IEEE international conference on computer vision* (pp. 1–7). IEEE.

Hui, T. W., Tang, X., & Loy, C. C. (2018). LiteFlowNet: A lightweight convolutional neural network for optical flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 8981–8989).

Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., & Brox, T. (2017). FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2462–2470).

Jonker, R., & Volgenant, A. (1987). A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing, 38*(4), 325–340.

Karpathy, A., Miller, S., & Fei-Fei, L. (2013). Object discovery in 3D scenes via shape analysis. In *IEEE international conference on robotics and automation* (pp. 2088–2095). IEEE.

Klokov, R., & Lempitsky, V. (2017). Escape from cells: Deep Kd-networks for the recognition of 3D point cloud models. In *Proceedings of the IEEE international conference on computer vision* (pp. 863–872).

Landrieu, L., & Simonovsky, M. (2018). Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4558–4567).

Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of

hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning* (pp. 609–616).
Lee, M., & Fowlkes, C. C. (2019). CeMNet: Self-supervised learning for accurate continuous ego-motion estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*.
Li, J., Chen, B. M., & Hee Lee, G. (2018a). SO-Net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 9397–9406).
Li, R., Lin, G., & Xie, L. (2021). Self-point-flow: Self-supervised scene flow estimation from point clouds with optimal transport and random walk. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 15577–15586).
Li, W., Zhang, Z., & Liu, Z. (2010). Action recognition based on a bag of 3D points. In *IEEE computer society conference on computer vision and pattern recognition-workshops* (pp 9–14). IEEE.
Li, Y., Bu, R., Sun, M., Wu, W., Di, X., & Chen, B. (2018b). PointCNN: Convolution on X-transformed points. In *Advances in neural information processing systems* (pp. 820–830).
Liu, J., Shahroudy, A., Perez, M., Wang, G., Duan, L. Y., & Kot, A. C. (2019). NTU RGB+D 120: A large-scale benchmark for 3D human activity understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 42*(10), 2684–2701.
Liu, X., Qi, C. R., & Guibas, L. J. (2019b). FlowNet3D: Learning scene flow in 3D point clouds. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 529–537).
Liu, X., Yan, M., & Bohg, J. (2019c). MeteorNet: Deep learning on dynamic 3D point cloud sequences. In *Proceedings of the IEEE international conference on computer vision* (pp. 9246–9255).
Luo, W., Yang, B., & Urtasun, R. (2018). Fast and furious: Real time end-to-end 3D detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3569–3577).
Maturana, D., & Scherer, S. (2015). VoxNet: A 3D convolutional neural network for real-time object recognition. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 922–928). IEEE.
Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., & Brox, T. (2016). A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4040–4048).
Menze, M., & Geiger, A. (2015). Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3061–3070).
Mittal, H., Okorn, B., & Held, D. (2020). Just go with the flow: Self-supervised scene flow estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11177–11185).
Moenning, C., & Dodgson, N. A. (2003). Fast marching farthest point sampling. Technical report. University of Cambridge, Computer Laboratory.
Niemeyer, M., Mescheder, L., Oechsle, M., & Geiger, A. (2019). Occupancy flow: 4D reconstruction by learning particle dynamics. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 5379–5389).
Ouyang, B., & Raviv, D. (2020). Occlusion guided scene flow estimation on 3D point clouds. arXiv preprint. arXiv:2011.14880.
Pan, Y., Gao, B., Mei, J., Geng, S., Li, C., & Zhao, H. (2020). SemanticPOSS: A point cloud dataset with large quantity of dynamic instances. In *IEEE intelligent vehicles symposium (IV)* (pp. 687–693). IEEE.
Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems* (pp. 8026–8037).
Peyré, G., Cuturi, M., et al. (2019). Computational optimal transport: With applications to data science. *Foundations and TrendsR in Machine Learning, 11*(5–6), 355–607.
Pham, Q. H., Nguyen, T., Hua, B. S., Roig, G., & Yeung, S. K. (2019). JSIS3D: Joint semantic-instance segmentation of 3D point clouds with multi-task pointwise networks and multi-value conditional random fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 8827–8836).
Pons, J. P., Keriven, R., & Faugeras, O. (2007). Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *International Journal of Computer Vision, 72*(2), 179–193.
Pontes, J. K., Hays, J., & Lucey, S. (2020). Scene flow from point clouds with or without learning. arXiv preprint. arXiv:2011.00320.
Prantl, L., Chentanez, N., et al. (2019). Tranquil clouds: Neural networks for learning temporally coherent features in point clouds. arXiv preprint. arXiv:1907.05279.
Puy, G., Boulch, A., & Marlet, R. (2020). FLOT: Scene flow on point clouds guided by optimal transport. In *European conference on computer vision*.
Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017a). PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 652–660).
Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017b). PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems* (pp. 5099–5108).
Qi, C. R., Liu, W., Wu, C., Su, H., & Guibas, L. J. (2018). Frustum PointNets for 3D object detection from RGB-D data. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 918–927).
Qi, C. R., Litany, O., He, K., & Guibas, L. J. (2019). Deep Hough voting for 3D object detection in point clouds. In *Proceedings of the IEEE international conference on computer vision* (pp. 9277–9286).
Qi, C. R., Chen, X., Litany, O., & Guibas, L. J. (2020). ImVoteNet: Boosting 3D object detection in point clouds with image votes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 4404–4413).
Ranjan, A., & Black, M. J. (2017). Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4161–4170).
Rempe, D., Birdal, T., Zhao, Y., Gojcic, Z., Sridhar, S., & Guibas, L. J. (2020). CaSPR: Learning canonical spatiotemporal point cloud representations. In *Advances in neural information processing systems (NeurIPS)*.
Revaud, J., Weinzaepfel, P., Harchaoui, Z., & Schmid, C. (2015). EpicFlow: Edge-preserving interpolation of correspondences for optical flow. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1164–1172).
Riegler, G., Osman Ulusoy, A., & Geiger, A. (2017). OctNet: Learning deep 3D representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3577–3586).
Rubner, Y., Tomasi, C., & Guibas, L. J. (2000). The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision, 40*(2), 99–121.
Shahroudy, A., Liu, J., Ng, T. T., & Wang, G. (2016). NTU RGB+D: A large scale dataset for 3D human activity analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1010–1019).
Shen, Y., Feng, C., Yang, Y., & Tian, D. (2018). Mining point cloud local structures by kernel correlation and graph pooling. In *Proceedings*

of the IEEE conference on computer vision and pattern recognition (pp. 4548–4557).
Shi, S., Wang, X., & Li, H. (2019). PointRCNN: 3D object proposal generation and detection from point cloud. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.
Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., & Li, H. (2020). PV-RCNN: Point-voxel feature set abstraction for 3D object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 10529–10538).
Sinkhorn, R. (1964). A relationship between arbitrary positive matrices and doubly stochastic matrices. *The Annals of Mathematical Statistics, 35*(2), 876–879.
Srivastava, N., Mansimov, E., & Salakhudinov, R. (2015). Unsupervised learning of video representations using LSTMs. In *International conference on machine learning* (pp. 843–852).
Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3D shape recognition. In *Proceedings of the IEEE international conference on computer vision* (pp. 945–953).
Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M. H., & Kautz, J. (2018). SPLATNet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2530–2539).
Sun, D., Yang, X., Liu, M. Y., & Kautz, J. (2018). PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 8934–8943).
Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., & Han, S. (2020). Searching efficient 3D architectures with sparse point-voxel convolution. In *European conference on computer vision*.
Teed, Z., Deng, J. (2020). RAFT: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision* (pp. 402–419). Springer.
Thomas, H., Qi, C. R., Deschaud, J. E., Marcotegui, B., Goulette, F., & Guibas, L. J. (2019). KPConv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE international conference on computer vision* (pp. 6411–6420).
Tombari, F., Salti, S., & Di Stefano, L. (2010). Unique signatures of histograms for local surface description. In *European conference on computer vision* (pp. 356–369). Springer.
Vedula, S., Baker, S., Rander, P., Collins, R., & Kanade, T. (1999). Three-dimensional scene flow. In *Proceedings of the IEEE international conference on computer vision* (pp. 722–729).
Wang, S., Suo, S., Ma, W. C., Pokrovsky, A., & Urtasun, R. (2018a). Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2589–2597).
Wang, W., Yu, R., Huang, Q., & Neumann, U. (2018b). SGPN: Similarity group proposal network for 3D point cloud instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2569–2578).
Wang, X., Yeshwanth, C., & Nießner, M. (2020a). SceneFormer: Indoor scene generation with transformers. arXiv preprint. arXiv:2012.09793.
Wang, Y., Long, M., Wang, J., Gao, Z., & Philip, S. Y. (2017). PredRNN: Recurrent neural networks for predictive learning using spatiotemporal LSTMs. In *Advances in neural information processing systems* (pp. 879–888).
Wang, Y., Gao, Z., Long, M., Wang, J., & Philip, S. Y. (2018c). PredRNN++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning. In *Proceedings of international conference on machine learning* (pp. 5110–5119).
Wang, Y., Jiang, L., Yang, M. H., Li, L. J., Long, M., & Fei-Fei, L. (2019a). Eidetic 3D LSTM: A model for video prediction and beyond. In *Proceedings of international conference on learning representations*.
Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., & Solomon, J. M. (2019b). Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics, 38*, 1–12.
Wang, Y., Xiao, Y., Xiong, F., Jiang, W., Cao, Z., Zhou, J. T., & Yuan, J. (2020b). 3DV: 3D dynamic voxel for action recognition in depth video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 511–520).
Wang, Z., Li, S., Howard-Jenkins, H., Prisacariu, V., & Chen, M. (2020c). FlowNet3D++: Geometric losses for deep scene flow estimation. In *The IEEE winter conference on applications of computer vision* (pp. 91–98).
Weng, X., Wang, J., Levine, S., Kitani, K., & Rhinehart, N. (2020). Inverting the pose forecasting pipeline with SPF2: Sequential pointcloud forecasting for sequential pose forecasting. In *Conference on robot learning*.
Wu, B., Wan, A., Yue, X., & Keutzer, K. (2018). SqueezeSeg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3D LiDAR point cloud. In *IEEE international conference on robotics and automation* (pp. 1887–1893). IEEE.
Wu, P., Chen, S., & Metaxas, D. N. (2020a). MotionNet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*.
Wu, W., Qi, Z., & Fuxin, L. (2019). PointConv: Deep convolutional networks on 3D point clouds. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 9621–9630).
Wu, W., Wang, Z. Y., Li, Z., Liu, W., & Fuxin, L. (2020b). PointPWC-Net: Cost volume on point clouds for (self-) supervised scene flow estimation. In *European conference on computer vision* (pp. 88–107). Springer.
Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015). 3D ShapeNets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1912–1920).
Xingjian, S., Chen, Z., Wang, H., Yeung, D. Y., Wong, W. K., Woo, W. C. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Proceedings of advances in neural information processing systems* (pp. 802–810).
Yang, B., Wang, J., Clark, R., Hu, Q., Wang, S., Markham, A., & Trigoni, N. (2019). Learning object bounding boxes for 3D instance segmentation on point clouds. In *Advances in neural information processing systems* (pp. 6737–6746).
Yang, Y., Feng, C., Shen, Y., & Tian, D. (2018). FoldingNet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.
Yao, S., Hsu, T. M. H., Zhu, J. Y., Wu, J., Torralba, A., Freeman, W. T., & Tenenbaum, J. B. (2018). 3D-aware scene manipulation via inverse graphics. arXiv preprint. arXiv:1808.09351.
Yew, Z. J., & Lee, G. H. (2020). RPM-Net: Robust point matching using learned features. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11824–11833).
Yi, L., Zhao, W., Wang, H., Sung, M., & Guibas, L. J. (2019). GSPN: Generative shape proposal network for 3D instance segmentation in point cloud. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3947–3956).
Yin, T., Zhou, X., & Krahenbuhl, P. (2021). Center-based 3D object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11784–11793).
Yin, Z., & Shi, J. (2018). GeoNet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1983–1992).
Yu, L., Li, X., Fu, C. W., Cohen-Or, D., & Heng, P. A. (2018). PU-Net: Point cloud upsampling network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2790–2799).

Zhang, C., Fiore, M., Murray, I., & Patras, P. (2019). CloudLSTM: A recurrent neural model for spatiotemporal point-cloud stream forecasting. arXiv preprint. arXiv:1907.12410.

Zhao, L., & Tao, W. (2020). JSNet: Joint instance and semantic segmentation of 3D point clouds. *In AAAI conference on artificial* intelligence.

Zou, Y., Luo, Z., & Huang, J. B. (2018). DF-Net: Unsupervised joint learning of depth and flow using cross-task consistency. In *Proceedings of the European conference on computer vision* (pp. 36–53).

Zuanazzi, V., van Vugt, J., Booij, O., & Mettes, P. (2020). Adversarial self-supervised scene flow estimation. In *International conference on 3D vision (3DV)* (pp. 1049–1058). IEEE.