Learning Scene Dynamics from Point Cloud Sequences

Pan He · Patrick Emami · Sanjay Ranka · Anand Rangarajan

Received: date / Accepted: date

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 (Qi et al. 2017a,b; Su et al. 2018; Landrieu and Simonovsky 2018; Liu et al. 2019b; Wang et al. 2019b) and other tasks such as 3D semantic and instance segmentation (Wang et al. 2018b; Hou et al. 2019; Yi et al. 2019; Pham et al. 2019; Zhao and Tao 2020), and 3D object detection (Qi et al. 2018; Gwak et al. 2020; Shi et al. 2020; Qi et al. 2020; Gwak et al. 2020; Yin et al. 2021). However, the community has paid less attention to the
Sequential Scene Flow Estimation

Scene Flow Estimation (Previous work)

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; 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.

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; 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 (Liu et al. 2019b; Gu et al. 2019; Wu et al. 2020b; Puy et al. 2020; Mittal et al. 2020), 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 forecast-
ing (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 (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 SPCM-Net (Sequential Point Cloud Modeling Network). 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 (Qi et al. 2017a; Liu et al. 2019b; Wu et al. 2020b; Fan and Yang 2019; Puy et al. 2020) 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 (Menze and Geiger 2015; Liu et al. 2019c), 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 (Section 2) and then present our proposed method (Section 3). Then, we introduce the proposed benchmark consisting of three new datasets along with appropriate evaluation metrics (Section 4). Experimental results are presented and discussed in Section 5. Related work is discussed in Section 6. We discuss findings, limitations, and future work in Section 7 and draw conclusions in Section 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.
Denoting the problem as high complexity due to the complexity inherent in point cloud sequences, e.g., partial or full occlusion, shape deformation, and scale variations.

When ground truth point-wise motion is 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 Section 3.3 for a formal definition of CD).

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 Section 3.1.2. The MCP module generates multi-scale prediction at each timestep based on features 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 Section 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) (Moenning and Dodgson 2003; Eldar et al. 1997; 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) (Hochreiter and Schmidhuber 1997; Graves 2012). 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 task and SPF. The detailed description of the recurrent cost volume (RCV) layer can be found in Section 3.2.

3.1.3 Multi-scale Coarse-to-fine Prediction

Inspired by scene flow approaches (Revaud et al. 2015; Sun et al. 2018; Gu et al. 2019; 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 Section 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; Moenning and Dodgson 2003; Qi et al. 2017a; Yu et al. 2018; Li et al. 2018b; Qi et al. 2019). 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 \( t \), \( P_i^t \) 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 \( P_i^t \) to obtain the upsampled coarse flow. For each point \( p_i^t \) in the fine level point cloud \( P_i^t \), we find its K nearest neighbors \( N(p_i^t) \) in the coarse level point cloud \( P_i^{l-1} \). Each scene flow \( \hat{SF}_{l,t} \) in \( SF_{l,t} \) for finer level \( l \) is computed via inverse distance weighted interpolation:

\[
\hat{sf}_{l,t}^j = \frac{\sum_{k=1}^{K} \omega(p_i^t, p_j^{l-1,t}) SF_{l-1,t}^k}{\sum_{j=1}^{K} \omega(p_i^t, p_j^{l-1,t})}
\]

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

\[
SF_{l,t} = MLP(\text{PointConv}(\hat{SF}_{l,t}; F_l^t; H_l)).
\]

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 \( P_t = (C_r, X_t) \). Specifically, the encoder consumes the input point cloud sequence frame-by-frame and keeps updating the states of RCV layers till \( P_t \). The obtained states will initialize the states for the decoder. To simplify the problem, we predict the displacements \( \Delta P \) 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.
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-point matching cost (the flow embedding layer) proposed by FlowNet3D (Liu et al. 2019b). SPCM-Net (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 matching cost between two consecutive point clouds following PointPWC-Net (Wu et al. 2020b) in Fig. 3.

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

$$\text{Cost}(p^t_i, p^t_{i-1}) = \phi_{\text{MLP}}(c^t_{i-1} - c^t_i \times x^t_{i-1}, x^t_i).$$  \tag{6}$$

Here, the feature vectors of two points and the directional difference between their spatiotemporal positions are passed to an MLP. Note that $p^t_{i-1}$ comes from a neighboring point set of $p^t_i$ 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 this by aggregating flow votes from neighboring points. Specifically, for a given point $p^t_i$, they find its neighboring points at timestep $t-1$ via ball query. These points are considered as multiple soft correspondence points for $p^t_i$, and are utilized to obtain multiple matching costs defined in Equation 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-point 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 $p^t_i$ is defined as

$$\text{CV}(p^t_i) = \sum_{p^t_{i-1} \in M(p^t_i)} \omega_M(p^t_i, p^t_{i-1}) \times \sum_{p^t_j \in N(p^t_i)} \omega_N(p^t_i, p^t_j) \text{Cost}(p^t_{i-1}, p^t_j)$$  \tag{7}$$

$$\omega_M(p^t_i, p^t_j) = \text{MLP}(c^t_i - c^t_j)$$  \tag{8}$$

$$\omega_N(p^t_i, p^t_j) = \text{MLP}(c^t_i - c^t_j)$$  \tag{9}$$

This requires finding a spatial neighboring point set $M(p^t_i)$ around $p^t_i$ in $P^t_i$. Then for each point $p^t_{i-1} \in M(p^t_i)$, we find a spatiotemporal neighboring point set $N(p^t_{i-1})$ around $p^t_{i-1}$ in $P^t_{i-1}$ (across time). The interaction between these points is modeled by two directional vectors obtained via convolutional operations $\omega_M(p^t_i, p^t_{i-1})$ and $\omega_N(p^t_i, p^t_{i-1})$, and their matching costs. Both the spatial neighboring point set $M(p^t_i)$ and $N(p^t_{i-1})$ 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 $C_t$ and $X_t$ will be fed into the RCV layer as the input.

**Initialization:** Accordingly, the states of the RCV layer are extended to $C_{t-1}$, $H_{t-1}$, and $M_{t-1}$ to track the most recent historic point locations and memory states. For notation clarity, since we already are using $C_t$ to denote the coordinates of points in the point cloud, we will use $M_t$ 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 volume operation to perform updates to the hidden and cell states \((H_{t-1}, M_{t-1})\), making them invariant to any changes to the order of points or to the addition of new points.

**Update:** Let

\[
CV(P_t; P_{t-1}) = CV(C_t, X_t; C_{t-1}, \{H_{t-1}, M_{t-1}\})
\]

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:

\[
I_t = \sigma_f(CV_f(C_t, X_t; C_{t-1}, H_{t-1})),
\]

\[
F_t = \sigma_f(CV_f(C_t, X_t; C_{t-1}, H_{t-1})),
\]

\[
O_t = \sigma_o(CV_o(C_t, X_t; C_{t-1}, H_{t-1})),
\]

\[
M_{t-1} = CV_M(C_t, \text{None}; C_{t-1}, M_{t-1}),
\]

\[
H_t = \tanh(CV_H(C_t, X_t; C_{t-1}, H_{t-1})),
\]

\[
M_t = F_t \odot \hat{M}_{t-1} + I_t \odot \hat{H}_t,
\]

\[
H_t = O_t \odot M_t,
\]

where the operator \(\odot\) denotes the Hadamard product. \(CV_f, CV_H,\) 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 \(M_{t-1}\) represents that we do not use any input features when performing the neighborhood query on the cell state \((x_i^t\) in Eq. 6 is ignored). Then \(\hat{M}_{t-1}\) is modulated via the forget gate \(F_t\) and is further aggregated with the new memory \(H_t\) passed to the input gate \(I_t\) to obtain the latest memory state \(M_t\). The latest hidden state \(H_t\) is obtained by conditionally deciding what to output from \(M_t\) controlled by the output gate \(O_t\). Note that \(C_t\) and \(H_t\) 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 \(SF_{t,l}\) at the pyramid level \(l\) from timestep \(t\) and its ground truth scene flow \(GF_{t,l}\). The objective function is specified as

\[
L_{\text{SSFE}}^{\text{supervised}} = \sum_{t=2}^{T} \sum_{l=1}^{L} \alpha_l ||SF_{t,l} - GF_{t,l}||^2_2.
\]

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 \(\{\alpha_l | l = 1, \ldots, L\}\) 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 Equation 18. We compute the difference between the predicted future frames and the future ground truth frames derived from ground truth scene flow. Denote \(P_{t,l}\) and \(\hat{P}_{t,l}\) as ground truth and predicted frames at the pyramid level \(l\) from timestep \(t\). The objective function is specified as

\[
L_{\text{SPF}}^{\text{supervised}} = \sum_{t=T+1}^{T+K} \sum_{l=1}^{L} \alpha_l ||P_{t,l} - \hat{P}_{t,l}||^2_2.
\]

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

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

(20)

where $P_t$ and $\hat{P}_t$ are ground truth and predicted frames. We apply Equation 20 to all the future frames:

$$L_{SPF}^{self-supervised} = T + K \sum_{t=T+1}^{T+L} \alpha_l D_{CD}(\hat{P}_{t,l}, P_{t,l}).$$

(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 (Wu et al. 2020b; Pontes et al. 2020), 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 (Liu et al. 2019b; Gu et al. 2019; 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 (Liu et al. 2019b; Gu et al. 2019) and only maintain points with a depth of less than 35 meters.

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., 2,048 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.

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 the remaining 40% for testing. We sample 2,048 points for each frame and only consider points with a depth less than 35 meters.

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 seconds. 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 (Section 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., 2,048. 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 $[-32, 32] \times [-8, 8] \times [-\infty, 2]$ meters, 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 5,337 sequences are created. Among them, 4,020 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 3,350 point cloud sequences, where 2,175 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 1,200 test point cloud sequences, where the first five frames are the input and the rest are the ground truth future frames. 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 (Liu et al. 2019b; Gu et al. 2019; Liu et al. 2019c) and extend to sequences. Specifically, the 3D end point error (EPE3D) and accuracy (ACC3D) are used as the metrics. The EPE3D measures the average $\ell_2$ distance between the predicted scene flow vector $\hat{s}_t$ and ground truth scene flow vector $s_t$ for all points in the sequence of length $T - 1$, which is computed as

$$EPE3D = \frac{1}{\sum_{t,i} m_t^i} \sum_{t=2}^{T} \sum_{i=1}^{N} m_t^i ||\hat{s}_t^i - s_t^i||_2$$

(22)

where $m$ is a binary mask and $m_t^i = 0$ denotes an invalid scene flow of the point $p_t^i$. 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 < 0.05m or relative error < 5%.**

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

To measure outlier prediction, we use the Outliers3D to compute the percentage of points whose EPE3D > 0.3m or relative error > 10%. We noticed that Outliers3D is invalid when the ground truth flows are near 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 $EPE3D > 0.3m$ when the ground truth scene flow is small (e.g., its $\ell_2$ 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.

| Dataset | Frame Sampling | X-Y-Z Point Range | Points Per Frame | #Train | #Validation | #Test | SSFE | SPF |
|---------|----------------|-------------------|------------------|--------|-------------|-------|------|-----|
| SFT3D   | n/a            | $X \leq 35$       | 2,048            | 4,020  | 446         | 871   | ✓    | ✓   |
| VKS     | Sample 10 frames per second | $X \leq 35$       | 2,048            | 2,175  | N/A         | 1,175 | ✓    | ✓   |
| SAG     | Sample 10 frames per second | $[-32, 32] \times [-8, 8] \times [-\infty, 2]$ | 2,048 | On-the-fly | N/A | 1,200 | X    |
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 \( \hat{p}_{t+t} \) and ground truth point \( g_{t+t} \) for all points in each prediction step. Considering the occluded points, it is defined as

\[
\text{ADE} = \frac{1}{\sum_{i,t} m_{T+t}} \sum_{t=1}^{K} \sum_{i=1}^{N} m_{T+t}^i \left\| p_{T+t}^i - g_{T+t}^i \right\|_2
\]

where \( m_{T+t} \) 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 \( \hat{p}_{t+t} \) and ground truth point \( g_{t+K} \) 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 Rangarajan 2000, 2003) but instead adopted a simpler nearest neighbor approach (Barrow et al. 1977; Besl and McKay 1992; Li et al. 2018b; Fan and Yang 2019; 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 Section 3, Equation 20. We apply Chamfer distance to all future frames and sum the displacement error (FDE).

To obtain \( \varphi \), we first obtain the optimal soft assignment (called the softassign in an homage to the softmax nonlinearity) \( \varphi_{soft} \) via the Sinkhorn algorithm (Sinkhorn 1964), and then follow (Li et al. 2021) to obtain hard correspondence. Specifically, we use the point coordinates of \( \hat{p}_t \) and \( \hat{p}_t \) to construct an affinity matrix \( M \) where \( M_{\hat{p}_t, \hat{p}_t} \) is defined as

\[
M_{\hat{p}_t, \hat{p}_t} = -\exp\left(\frac{\gamma}{d(\hat{p}_t, \hat{p}_t)} + \epsilon\right).
\]

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

To handle outliers, we follow (Chui and Rangarajan 2003; Yew and Lee 2020; Gojcic et al. 2021) and add an additional row and column of ones (a slack row and column) to the original input of Sinkhorn normalization (the matrix \( M \)) while only performing the alternating row and column normalization on non-slack rows and columns to obtain a resulting matrix \( \hat{A} \). To generate \( \varphi \), we follow (Li et al. 2021) to modify each row of \( \hat{A} \) 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 \( \ell_2 \) distances between all pairs of points except those which are assigned to slack columns. These are denoted as \( p_{t}^{\text{valid}} \subset p_{t} \) respectively and then the Sinkhorn Distance (SD) evaluation metric is defined as

\[
\mathcal{D}_{\text{SD}}(p_t, \hat{p}_t) = \frac{1}{|p_{t}^{\text{valid}}|} \sum_{p \in p_{t}^{\text{valid}}} \left\| p - \varphi(p, \hat{p}_t) \right\|^2.
\]

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 these metrics: the average displacement error (ADE) and final displacement error (FDE).

\[
\text{ADE} = \frac{1}{\sum_{i,t} m_{T+t}} \sum_{t=1}^{K} \sum_{i=1}^{N} m_{T+t}^i \left\| p_{T+t}^i - g_{T+t}^i \right\|_2
\]

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 \( \hat{p}_{t+t} \) and ground truth point \( g_{t+K} \) for all points at end of the prediction period \( K \).
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 (Section 5.2). Then, we evaluate the two taskson the VKS dataset with the same models (Section 5.3). Next, we evaluate SPCM-Net on the self-supervised SPF task with the SAG dataset (Section 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 (Section 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 spatiotemporal 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 visualiz-
Table 2 Multi-step representation achieves improved performance on the SSFE task for point cloud sequences longer than two frames. 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).

| Method | SFT3D Validation Split | SFT3D Test Split | VKS Dataset |
|--------|------------------------|------------------|-------------|
|        | EPE3D↓ | Acc3DS↑ | Acc3DR↑ | Outliers3D↓ | RectOutliers3D↓ | EPE2D↓ | Acc2D↑ |
| FlowNet3D (Liu et al. 2019b) + FT3D | 0.206 | 0.173 | 0.294 | 0.783 | 0.764 | 11.803 | 0.319 |
| FLOT (Puy et al. 2020) + FT3D | 0.175 | 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 | 7.993 | 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 |

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

| 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) | 0.2856 | 0.4655 | 0.1737 | 0.3408 | 0.3078 | 0.1634 | 0.2514 | 0.1484 |
| 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 | 0.3866 | 0.1737 | 0.3408 | 0.3078 | 0.1634 | 0.2514 | 0.1484 |

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 | 0.3575 | 0.2322 | 0.1363 |
| SPCM-Net (Ours) | 0.2768 | 0.4799 | 0.3418 | 1.3455 | 2.2992 | 1.4845 |

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

ing model predictions on the SFT3D dataset, as shown in Fig. 8.
Learning Scene Dynamics from Point Cloud Sequences

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.

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. Additionally, 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.

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 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.
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.

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.

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 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. Fig. 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 con-
Table 5 Flow estimation results on KITTI scene flow dataset. Metrics are the mean and standard deviation of the End-point-error (EPE) of scene flow.

| Method                      | Frames | Mean  | Std  |
|-----------------------------|--------|-------|------|
| 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|

tain 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 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. Fig. 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 ef-
fectively 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. Pranli 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; Shahroudy et al. 2016; Liu et al. 2019a) 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 Section 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 trajectory 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 (Pons et al. 2007; Huguet and Devernay 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 (Klokov and Lempitsky 2017; Qi et al. 2017a,b; Li et al. 2018a; Wang et al. 2019b), 3D object detection (Qi et al. 2018; Shi et al. 2019), 3D semantic labeling (Landrieu and Simeonovsky 2018; Graham et al. 2018; Choy et al. 2019; Su et al. 2018), and 3D instance segmentation (Pham et al. 2019; Yang et al. 2019; Pan et al. 2020). 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 (Qi et al. 2017a; Pham et al. 2019; Qi et al. 2017b; Wang et al. 2019b) by applying MLPs in a point-wise fashion. To further capture local structures, follow-ups (Qi et al. 2017b; Shen et al. 2018; Wang et al. 2019b; 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 (Lee et al. 2009; Doersch et al. 2015; 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 (Yin and Shi 2018; Zou et al. 2018; Lee and Fowlkes 2019). 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 (Wu et al. 2020b; Pontes et al. 2020) 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 Section 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 extension 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.

Declarations

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

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 HG, Tenenbaum JM, Bolles RC, Wolf HC (1977) Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching. In: International Joint Conferences on Artificial Intelligence

Besl PJ, McKay ND (1992) Method for Registration of 3-D Shapes. In: Sensor Fusion IV: Control Paradigms and Data Structures, International Society for Optics and Photonics, vol 1611, pp 586–606

Bobenko AI, Springborn BA (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, Springer, pp 25–36

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 AH, Vora S, Lioong VE, Xu Q, Krishnan A, Pan Y, Baldan G, Beijbom O (2019) nuScenes: A Multimodal Dataset for Autonomous Driving. arXiv preprint arXiv:190311027

Chang AX, 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:151203312

Chang MF, 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. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol 2, pp 50–51

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 AA (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, Hauser 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 YY (1997) The Farthest Point Strategy for Progressive Image Sampling. IEEE Transactions on Image Processing 6(9):1305–1315

Emami P, Pardalos PM, 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:191008287

Fan H, Su H, Guibas LJ (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 AW (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
Learning Scene Dynamics from Point Cloud Sequences

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.

Goejic Z, Litany O, Wieser A, Guibas LJ, 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, Springer, pp 5–13.

Gu X, Wang Y, Wu C, Lee YJ, Wang P (2019) HPLFlowNet: Hierarchical Permutahedral 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:191212033.

Gwak J, Choy CB, 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 BK, Schunck BG (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, Deverney F (2007) A Variational Method for Scene Flow Estimation from Stereo Sequences. In: IEEE International Conference on Computer Vision, IEEE, pp 1–7.

Hui TW, Tang L, Loy CC (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 48(4):325–340.

Karpathy A, Miller S, Fei-Fei L (2013) Object Discovery in 3D scenes via Shape Analysis. In: 2013 IEEE International Conference on Robotics and Automation, IEEE, pp 2088–2095.

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 AY (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, Powlkes CC (2019) CeMNet: Self-Supervised Learning for Accurate Continuous Ego-Motion Estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp 0–0.

Li J, Chen BM, 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 Optical 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: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, IEEE, pp 9–14.

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 LY, Kot AC (2019a) NTU RGB+D 120: A Large-Scale Benchmark for 3D Human Activity Understanding. IEEE transactions on pattern analysis and machine intelligence 42(10):2684–2703.

Liu X, Qi CR, Guibas LJ (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, IEEE, pp 922–928.

Mayer N, Ilg E, Hauser 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.

Moehning C, Dodgson NA (2003) Fast Marching Farthest Point Sampling. Tech. rep., 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.
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, Yeshwant C, Nießner M (2020a) SceneFormer: Indoor Scene Generation with Transformers. arXiv preprint arXiv:201209793

Wang Y, Long M, Wang J, Gao Z, Philip SY (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 SY (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 MH, Li LJ, 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 SE, Bronstein MM, Solomon JM (2019b) Dynamic Graph CNN for Learning on Point Clouds. ACM Transactions on Graphics

Wang Y, Xiao Y, Xiong F, Jiang W, Cao Z, Zhou JT, 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. 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, IEEE, pp 1887–1893

Wu P, Chen S, Metaxas DN (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 ZY, 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, Springer, pp 88–107

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 DY, Wong WK, Woo Wc (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 TMH, Zhu JY, Wu J, Torralba A, Freeman WT, Tenenbaum JB (2018) 3DV: 3D-Aware Scene Manipulation via Inverse Graphics. arXiv preprint arXiv:180809351

Yew ZJ, Lee GH (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 LJ (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 CW, Cohen-Or D, Heng PA (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:190712410

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 JB (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: 2020 International Conference on 3D Vision (3DV), IEEE, pp 1049–1058

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. 15h of driving data (242km traveled at an average of 16km/h) was collected with the dataset containing
Table 6 Evaluation results on VKS for the SSFE task with the goal to test generalization to unseen driving scenarios.

| Method                     | EPE3D↓ | Acc3DS↑ | Acc3DR↑ | Outliers3D↓ | RectOutliers3D↓ | EPE2D↓ | Acc2D↑ |
|----------------------------|--------|---------|---------|-------------|-----------------|--------|--------|
| FlowNet3D (Liu et al. 2019b) | 0.0422 | 0.8790  | 0.9718  | 0.2043      | 0.0855          | 3.968  | 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          | 3.2182 | 0.9306 |
| SPCM-Net (Ours)            | 0.0618 | 0.9212  | 0.9556  | 0.2061      | 0.0877          | 2.4082 | 0.9180 |

Table 7 Evaluation results on nuScenes for the SPF task.

| Method                     | CD↓ | EMD↓ | SD↓ |
|----------------------------|-----|------|-----|
| PointNet++ (Qi et al. 2017b) + LSTM | 0.6176 | 0.8354 | 0.6920 |
| PointRNN (Fan and Yang 2019) | 0.9750 | 0.9878 | 1.0969 |
| SPCM-Net (Ours)              | 0.6393 | 0.7858 | 0.7113 |

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

| Method                     | ADE | FDE | CD | EMD |
|----------------------------|-----|-----|----|-----|
| PointNet++ (Qi et al. 2017b) + LSTM | 0.9914 | 1.5926 | 0.5871 | 1.2671 |
| PointRNN (Fan and Yang 2019) | 0.2203 | 0.3292 | 0.1242 | 0.2856 |
| SPCM-Net (Ours)              | 0.2599 | 0.4327 | 0.1418 | 0.3415 |

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. 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, 2,048 points are randomly sampled from every point cloud frame in the sequence. Following the practice of (Wu et al. 2020b), the point clouds are cropped to extract the region defined by $[-32, 32] \times [-8, 8] \times [-1.3, 2]$ meters, 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.

A.1.2 Additional SSFE results on VKS

In Section 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 Section 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 1,876 train and 1,474 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 Section A.1.2. Our SPCM-Net still achieves a comparable performance compared to PointRNN (Fan and Yang 2019), drawing a similar conclusion as in Section 5.3.