Deep Hierarchical Representation of Point Cloud Videos via Spatio-Temporal Decomposition

Hehe Fan, Xin Yu, Yi Yang, Senior Member, IEEE, and Mohan Kankanhalli, Fellow, IEEE

Abstract—In point cloud videos, point coordinates are irregular and unordered but point timestamps exhibit regularities and order. Grid-based networks for conventional video processing cannot be directly used to model raw point cloud videos. Therefore, in this work, we propose a point-based network that directly handles raw point cloud videos. First, to preserve the spatio-temporal local structure of point cloud videos, we design a point tube covering a local range along spatial and temporal dimensions. By progressively subsampling frames and points and enlarging the spatial radius as the point features are fed into higher-level layers, the point tube can capture video structure in a spatio-temporally hierarchical manner. Second, to reduce the impact of the spatial irregularity on temporal modeling, we decompose space and time when extracting point tube representations. Specifically, a spatial operation is employed to encode the local structure of each spatial region in a tube and a temporal operation is used to encode the dynamics of the spatial regions along the tube. Empirically, the proposed network shows strong performance on 3D action recognition, 4D semantic segmentation and scene flow estimation. Theoretically, we analyse the necessity to decompose space and time in point cloud video modeling and why the network outperforms existing methods.

Index Terms—Point cloud, spatio-temporal modeling, video analysis, action recognition, semantic segmentation, scene flow estimation

1 INTRODUCTION

Point cloud videos provide more flexibility for action recognition and decision making in poor visibility environments, and covers more precise geometry dynamics than the conventional videos. A point cloud video can be represented by a sequence of 3D point coordinate sets. Moreover, when RGB images are available, they are often used as additional features associated with 3D points to enhance the discriminativeness of point clouds. However, unlike conventional grid-based videos, point cloud videos are irregular and unordered in the spatial dimension and points do not emerge consistently over time. Therefore, as shown in Fig. 1, existing grid-based methods on conventional videos [1], [2], [3] are not suitable for directly modeling raw point cloud videos.

To capture the dynamics in point cloud videos, one solution is converting a point cloud video to a sequence of 3D voxels, and then applying grid-based 4D convolutions to the voxel sequence. However, directly performing convolutions on voxel sequences requires a large amount of computation and special engineering efforts, e.g., sparse convolution [4]. Furthermore, quantization errors are inevitable during voxelization, which may restrict applications that require precise measurement of scene geometry. Another solution is appending 1D temporal dimension to 3D points and treating point cloud videos as unordered 4D point sets [5], in which point tracking is usually employed to capture the dynamics in a region. Because points may flow in and out across frames, computing an accurate point trajectory is extremely difficult, especially for long videos. Moreover, simply concatenating coordinates and timestamps together neglects the temporal order and regularity, which may not properly exploit the temporal information and lead to inferior performance.

In this paper, we propose a novel point-based deep neural network, named PSTNet++, to directly encode raw point cloud videos without resorting to unstable point tracking. Inspired by 3D convolutional neural networks [1], [2], [3] on conventional video modeling, our PSTNet++ encodes point cloud videos in a spatio-temporally hierarchical manner. Instead of tracing points along an entire video, the proposed method models a video by capturing spatio-temporal local structure changes and thus alleviates the requirement for point tracking. Specifically, we introduce a point tube, which is formed by selecting an anchor point and propagating the anchor and its surrounding spatial region to a few nearest frames. By subsampling frames and points and enlarging spatial regions as the point features are fed into higher-level layers, our PSTNet++ can construct spatio-temporal hierarchy for point cloud video modeling.

To reduce the impact of the spatial irregularity of points on temporal modeling, we propose a point spatio-temporal operation (PSTOp) that decouples the spatio-temporal structure encoding. Specifically, PSTOp consists of (i) a spatial operation that encodes the spatial structure of 3D points
BenEFITING FROM OUR POINT TUBES, WE BUILD A SPATIO-

WE PROPOSE A POINT SPATIO-TEMPORAL OPERATION THAT

EXTENSIVE EXPERIMENTS ON SEVEN DATASETS INDICATE

TO DECODE POINT CLOUD VIDEOS FOR POINT-LEVEL PREDI-

IMPRESSIVE

INPUT TO AN OUTPUT OF SIZE

WHERE $C$, $L$, $H$ AND $W$ DENOTE THE FEATURE DIMENSION, THE NUMBER OF

FRAME, HEIGHT AND WIDTH, RESPECTIVELY. A 3D CONVOLUTION ENCODES AN

COORDINATE TENSOR ($C \times L \times N$) AND A FEATURE TENSOR ($C' \times L' \times N'$). USUALLY,

L' $\leq$ L AND N' $\leq$ N SO THAT NETWORKS CAN MODEL POINT CLOUD VIDEOS IN A

SPATIO-TEMPORALLY HIERARCHICAL MANNER. NOTE THAT POINTS IN DIFFERENT

FRAMES ARE NOT CONSISTENT, AND THIS IS CHALLENGING TO PRESERVE THE SPA-

TIO-TEMPORAL STRUCTURE.


TO CONDUCT POINT-WISE SEGMENTATION TASKS, WE EXTEND OUR

PSTOp TO A TRANSPOSED VERSION THAT INTERPOLATES TEMPORAL DYNAMICS AND SPATIAL FEATURES. EXPERIMENTS ON 3D ACTION RECOGNITION, 4D SEMANTIC SEGMENTATION AND SCENE FLOW ESTIMATION DEMONSTRATE THE EFFECTIVENESS OF PSTNet++ ON POINT CLOUD VIDEO MODELING. THE CONTRIBUTIONS OF THIS PAPER ARE FOURFOLD:

- BENEFITING FROM OUR POINT TUBES, WE BUILD A SPATIO-

TEMPORALLY HIERARCHICAL NETWORK TO PRESERVE THE LOCAL STRUCTURE OF RAW POINT CLOUD VIDEOS.

- WE PROPOSE A POINT SPATIO-TEMPORAL OPERATION THAT DECOMPOSES SPATIAL AND TEMPORAL INFORMATION IN ENCODING THE LOCAL STRUCTURE OF RAW POINT CLOUD VIDEO.

- TO DECODE POINT CLOUD VIDEOS FOR POINT-LEVEL PREDIC-

TION TASKS, WE PROPOSE A POINT SPATIO-TEMPORAL TRANS-

POSED OPERATION THAT IS ABLE TO INTERPOLATE THE TEMPORAL DYNAMICS AND SPATIAL FEATURES.

- EXTENSIVE EXPERIMENTS ON SEVEN DATASETS INDICATE THAT OUR METHOD REMARKABLY IMPROVES THE ACCURACY OF 3D ACTION RECOGNITION, 4D SEMANTIC SEGMENTATION AND SCENE FLOW ESTIMATION.

THIS PAPER IS AN EXTENSION OF OUR PREVIOUS WORK PSTNet [6]. FOR PSTNet, WE DID NOT PROVIDE A THEOREM TO GUARANTEE ITS EFFECTIVENESS. IN PSTNet++, WE IMPROVE PSTNet AND THEORETICALLY PROVE THE EFFECTIVENESS OF PSTNet++ ON RAW POINT CLOUD VIDEO MODELING. EXPERIMENTS ALSO DEMONSTRATE THAT PSTNet++ IS SUPERIOR TO PSTNet. MOREOVER, COMPARED TO OUR PREVIOUS WORK PSTNet, WE INCLUDE ANOTHER TWO DATASETS FOR 3D ACTION RECOGNITION, I.E., N-UCLA [7] AND UWA3DII [8].

2 RELATED WORK

LEARNING REPRESENTATIONS ON GRID-BASED VIDEOS. IMPRESSIVE PROGRESS HAS BEEN MADE ON GENERATING COMPACT AND DISCRIMINATIVE REPRESENTATIONS FOR RGB/RGBD VIDEOS DUE TO THE SUCCESS OF DEEP NEURAL NETWORKS. FOR EXAMPLE, TWO-STREAM CONVOLUTIONAL NEURAL NETWORKS [9], [10] EMPLOY A SPATIAL STREAM AND AN OPTICAL FLOW STREAM FOR VIDEO MODELING. TO CAPTURE THE TEMPORAL DEPENDENCIES OF VIDEOS, RECURR-

RENT NEURAL NETWORKS [11], [12] AND POOLING TECHNIQUES [13] ARE EMPLOYED. IN ADDITION, BY STACKING MULTIPLE 2D FRAMES INTO A 3D TENSOR, 3D CONVOLUTIONAL NEURAL NETWORKS [1], [2], [3], [14] ARE WIDELY USED TO LEARN SPATIO-TEMPORAL REPRESENTATIONS FOR VIDEOS, AND ACHIEVE PROMISING PERFORMANCE. BE-SIDES, INTERPRETABLE VIDEO OR ACTION REASONING METHODS [15], [16] ARE PROPOSED BY EXPLICITLY PARSING CHANGES IN VIDEOS.

STATIC POINT CLOUD PROCESSING. STATIC POINT CLOUD ANALYSIS HAS BEEN WIDELY INVESTIGATED IN MANY PROBLEMS, SUCH AS CLASSIFICATION, OBJECT PART SEGMENTATION, SCENE SEMANTIC SEGMENTATION [17], [18], [19], [20], [21], [22], [23], SCENE FLOW ESTIMATION [24], [25] AND OBJECT DETECTION [26], [27]. MOST RECENT WORKS AIM TO DIRECTLY MANIPULATE POINT SETS WITHOUT TRANSFORMING COORDINATES INTO REGULAR VOXEL GRIDS. SINCE A POINT CLOUD IS ESSENTIALLY A SET OF UNORDERED POINTS AND INVARIANT TO PERMUTATIONS OF ITS POINTS, STATIC POINT CLOUD PROCESSING METHODS MAINLY FOCUS ON DESIGNING EFFECTIVE POINT-BASED SPATIAL CORRELATION OPERATIONS THAT DO NOT REly ON POINT ORDERS. HOWEVER, THOSE METHODS DO NOT TAKE THE TEMPORAL DYNAMIC INFORMATION INTO ACCOUNT AND MAY PRODUCE SUBOPTIMAL RESULTS WHEN PROCESSING POINT CLOUD VIDEOS.

DYNAMIC POINT CLOUD MODELING. COMPARED WITH STATIC POINT CLOUD PROCESSING, DYNAMIC POINT CLOUD MODELING IS A FAIRLY NEW TASK BUT VERY IMPORTANT FOR INTELLIGENT SYSTEMS TO UNDERSTAND THE DYNAMIC WORLD. FAST AND FURIOUS (FAP) [28] CONVERTS A POINT CLOUD FRAME INTO A BIRD’S VIEW VOXEL AND THEN EXTRACTS FEATURES VIA 3D CONVOLUTIONS. MINKOWSKI NET [4] USES 4D SPATIO-TEMPORAL CONVNETS ON A 4D OCCUPANCY GRID. FAN AND YANG [29] PROPOSED A SERIES OF POINT RECURRENT NEURAL NETWORKS (PointRNNs) FOR MOVING POINT CLOUD PREDICTION. METEORNET [5] EXTENDS 3D POINTS TO 4D POINTS BY APPENDING A TEMPORAL DIMENSION AND THEN EMPLOYS THE FRAMEWORK OF POINTNet++ TO PROCESS THOSE 4D POINTS. 3DV [30] FIRST INTEGRATES 3D MOTION INFORMATION INTO A REGULAR COMPACT VOXEL SET AND THEN APPLIES PointNet++ TO EXTRACT REPRESENTATIONS FROM THE SET FOR 3D ACTION RECOGNITION VIA TEMPORAL RANK POOLING. P4Transformer [31] EMPLOYS TRANSFORMER TO AVOID POINT TRACKING FOR RAW POINT CLOUD VIDEO MODELING. NIEMEYER ET AL. [32] LEARNED A TEMPORAL AND SPATIAL VECTOR FIELD IN WHICH EVERY POINT IS ASSIGNED WITH A MOTION VECTOR OF SPACE AND TIME FOR 4D RECONSTRUCTION. PRANTL ET AL. [33] LEARNED STABLE AND TEMPORAL COHERENT FEATURE SPACES FOR POINT-BASED SUPER-RESOLUTION. CASPR [34] LEARNS TO ENCODE SPATIO-TEMPORAL CHANGES OF OBJECT SHAPES FROM DYNAMIC POINT CLOUDS FOR RECONSTRUCTION AND CAMERA POSE ESTIMATION. DIFFERENT FROM PREVIOUS WORKS, WE PROPOSE A POINT-BASED NETWORK TO MODEL SPATIO-TEMPORAL INFORMATION OF RAW POINT CLOUD VIDEOS VIA DECOMPOSING SPACE AND TIME HIERARCHICALLY. besides, scene flow estimation [35], [36] can be seen as computing the motion between two point cloud frames. DIFFERENT FROM THE SCENE FLOW ESTIMATION METHODS, OUR PSTNet++ AIMS TO CAPTURE THE LONG-TERM DEPENDENCY OF MULTIPLE FRAMES.
To construct the temporal hierarchy, our method employs a frame selection mechanism, which involves a temporal kernel size, a temporal striding size and a temporal padding size.

3 PROPOSED METHOD

In this section, we first present point tubes. Second, we introduce how the proposed point spatio-temporal operation (PSTOp) extracts features from point tubes. In order to address dense point prediction tasks, i.e., semantic segmentation, we develop a point spatio-temporal transposed operation (PSTTransOp). Finally, we incorporate our operations into deep spatio-temporally hierarchical networks, i.e., PSTNet++, to address different dynamic point cloud tasks.

Let \( P_t \in \mathbb{R}^{3 \times N} \) and \( F_t \in \mathbb{R}^{C \times N} \) denote the point coordinates and features of the \( t \)th frame in a point cloud video, where \( N \) and \( C \) denote the number of points and feature channels, respectively. Given a point cloud video \([P_1; F_1], [P_2; F_2], \ldots, [P_L; F_L]\) of \( L \) frames, the proposed PSTOp will encode the video to embedded frames \([P'_1; F'_1], [P'_2; F'_2], \ldots, [P'_L; F'_L]\), where \( L \) and \( L' \) indicate the number of frames and \( P'_t \in \mathbb{R}^{3 \times N'} \) and \( F'_t \in \mathbb{R}^{C' \times N'} \) represent the encoded coordinates and features (usually \( N' < N \), \( L' < L \) and \( C' > C \)).

3.1 Point Tube

The power of 3D convolutional neural networks (CNNs) in conventional video modeling comes from spatio-temporally hierarchical architectures. This motivates us to build spatio-temporal hierarchy in raw point cloud video modeling. Grid-based 3D convolutions can be easily performed on regular conventional videos by shifting along the height, length and width dimensions to capture the spatio-temporal local structure. However, it is challenging to preserve spatio-temporal local structure for point cloud video encoding, because point cloud videos are irregular and unordered in 3D space and points emerge inconsistently across different frames. To address this problem, we introduce a point tube. In contrast to pixel cubes, i.e., the receptive field of 3D convolutions, in which pixels are distributed regularly, point tubes are dynamically generated according to input videos so that dense areas have more tubes than sparse ones. Specifically, the point tube is constructed as follows:

Temporal Anchor Frame Selection. For an entire point cloud video, we need to select some anchor frames to generate our tubes. Temporal anchor frames in a point cloud video are automatically selected based on temporal radius \((r_t)\), temporal stride \((s_t)\) and temporal padding \((p)\), as shown in the solid frames in Fig. 2. Moreover, we set \( r_t \geq p \) to avoid selecting a padding frame as an anchor frame. Specifically, the frames with timestamp \( \{t_1 + 1 - p, t_1 + 1 - p + s_t, t_1 + 1 - p + 2s_t, \ldots, L + p - r_t\} \) are selected as temporal anchor frames, and \( L + 2p - 2r_t - 1 \) can be divided by \( s_t \) so that point tubes are correctly constructed with the same temporal size. The length \( L' \) of the encoded video is then expressed as \( L' = \frac{L + 2p - 2r_t}{s_t} + 1 \).

Spatial Anchor Point Sampling. Once a temporal anchor frame is selected, we need to choose spatial anchor points that can represent the distribution of all the points in the frame. Given a spatial subsampling rate \( s_s \), this operation aims to subsample \( N \) points to \( N' = \lfloor \frac{N}{s_s} \rfloor \) points. We employ the farthest point sampling (FPS) [18] to sample points in each anchor frame. Point tubes are generated based on the sampled anchor points.

Transferring Spatial Anchor Points. 3D convolutions can effectively capture local changes within a cube rather than tracking a specific pixel. Inspired by this idea, we propagate the positions of sampled anchor points to the neighboring frames without tracking, and they are regarded as the anchor points in those frames. Specifically, each anchor point is transferred to the \( r_t \) nearest frames. In general, the temporal radius \( r_t \) is much smaller, e.g., 1 or 2, than the input length, to effectively model local temporal dynamics. The original and transferred anchor points form the central axis of a tube.

Spatial Neighbor. This step aims to find the spatial neighbors of every anchor point in each frame for performing the spatial operation. A spatial radius \( r_s \) is used to search neighbors within the tube slice, where spatial local structure of points is depicted. Note that, padding is usually used in grid-based operation to align feature maps. However, point-based spatial operation is not conducted on grids and thus spatial padding is not employed.

By capturing the spatio-temporal structure in point tubes, our network is able to capture the dynamic changes in local areas. The temporal radius \( r_t \) and spatial radius \( r_s \) allow our point tube to capture spatio-temporal local structure, and progressively subsampling frames (according to \( s_t \) and points (according to \( s_s \)) and enlarging spatial radius \( r_s \) as the layer goes deeper, our point tube can capture longer dependencies of temporal and spatial information of point cloud videos, as shown in Fig. 2.

3.2 Point Spatio-Temporal Operation (PSTOp)

3.2.1 Decomposing Space and Time in Point Cloud Video Modeling

To capture the structure in point tubes, we propose a point spatio-temporal operation (PSTOp). Because point cloud videos are spatially irregular and unordered but temporally ordered, our PSTOp decouples these two dimensions in order to reduce the impact of the spatial irregularity of points on temporal modeling. Moreover, the scales of spatial displacements and temporal differences in point cloud videos may not be compatible. Treating them uniformly is not conducive for network optimization. Besides, because space and time are orthogonal and independent of each other, this also motivates us to decompose spatial and temporal modeling. In this way, we not only make the spatio-temporal modeling easier but also significantly improve the ability to capture the temporal information.

Our PSTOp consists of a spatial operation and a temporal operation. The spatial operation is employed to capture the local structure of points within each spatial region of a point tube. The temporal operation is used to model the dynamics of the spatial regions along the point tube. In this fashion, we are able to capture the spatial layout and temporal dynamics of points in a point tube. Note that, the decomposition can also be expressed as applying temporal operation first to encode point trajectories and then spatial operation to trajectory features. However, doing so requires point tracking to achieve point trajectories. Because it is difficult to computing accurate point trajectories and tracking points...
usually relies on point colors [5] and may fail to handle colorless point clouds, we opt to model the spatial structure of irregular points first, and then capture the temporal information from the spatial regions.

Benefiting from the decomposition of spatio-temporal modeling, we can employ existing methods [18], [19], [20], [21], [23] in static point cloud processing as the spatial operation. However, in order to encode long videos, the spatial operation should be both computation-efficient and memory-efficient. In this paper, we employ the simple yet effective PointNet++ [18] as the spatial operation. Moreover, as shown in Section 3.2.2, the use of PointNet++ can theoretically guarantee the effectiveness of our PSTOp. For the temporal operation, since point cloud videos are temporally ordered and regular, we propose to use a series of time-related multilayer perceptrons (MLPs) to distinguish different moments, instead of directly encoding timestamps like MeteorNet [5]. Our PSTOp is formulated as follows,

\[
M^t = \max_{d_\delta} S - \text{MLP}\left(\mathbf{F}^{(\delta_\delta, \delta_\delta)}; \delta_x, \delta_y, \delta_z\right),
\]

where \((x, y, z) \in P_t\) and \((\delta_x, \delta_y, \delta_z)\) represents displacement. The \(\mathbf{F}^{(\delta, \delta)} \in \mathbb{R}^\mathcal{C}\) is the feature of point \((x, y, z)\) at time \(t\). The \(r_s\) and \(r_t\) are the spatial radius and temporal radius, respectively. The \(S - \text{MLP}\) is the shared MLP of the spatial operation. The \(\mathcal{T} - \text{MLP}\) is the time-related MLPs of the temporal operation. We illustrate our PSTOp in Fig. 2.

3.2.2 Theoretical Analysis

For simplicity, we do not consider spatial or temporal radius in this section. Let \(S = [P, F] \in \mathcal{D} = 3 + C\) and \(\mathcal{X} = \{\mathcal{S} | \mathcal{S} \subseteq [0, 1]^3, |\mathcal{S}| = N, N \in \mathbb{Z}^+\}\) is the set of \(D\)-dimensional point clouds inside a \(D\)-dimensional unit cube. Suppose \(\mathcal{X}_t\) is the set of point clouds at time \(t \in \mathcal{Z}\) and \(\mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_t\) is the set of point cloud videos of length \(L\). Suppose \(\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_L) \in \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_L\) is a specific video sampled from the set of point cloud videos.

1. Static Point Cloud (Individual Point Cloud Frame)

A point cloud is essentially an unordered point set. Therefore, the Hausdorff metric can be used to measure the distance of two individual point cloud frames [5], [17], [18]. Specifically, the Hausdorff distance returns the greatest of all the distances from a point in one cloud to the closest point in the other cloud, to measure how far two point cloud frames are from each other. Given two point clouds \(\mathcal{S}\) and \(\mathcal{S}'\), the Hausdorff distance \(d_H(\cdot, \cdot)\) is defined as follows,

\[
d_H(\mathcal{S}, \mathcal{S}') = \max\left\{\sup_{s \in \mathcal{S}} \inf_{s' \in \mathcal{S}'} d(s, s'), \sup_{s' \in \mathcal{S}'} \inf_{s \in \mathcal{S}} d(s, s')\right\},
\]

where \(d(\cdot, \cdot)\) is a distance metric for two points. Based on the Hausdorff distance, PointNet [17] theoretically proves its ability to model static point clouds (shown in Table 1).

2. Point Cloud Video

A point cloud video is a sequence of point clouds. To measure the distance between two point cloud videos, MeteorNet [5] defines the video-level distance as the maximum per-frame Hausdorff distance among all respective frame pairs,

\[
d_H(\mathcal{S}, \mathcal{S}') = \max\left\{\sup_{s \in \mathcal{S}} \inf_{s' \in \mathcal{S}'} d(s, s'), \sup_{s' \in \mathcal{S}'} \inf_{s \in \mathcal{S}} d(s, s')\right\}.
\]
The voxel-based MinkowskiNet employs a 4D convolution to capture the spatio-temporal structure of point cloud videos. The \( d_g(S, S') \) operation to continuous functions on point cloud videos. We prove the universal approximation ability of PSTOp by showing the universal approximation ability of PSTOp. In this way, it cannot properly represent the global difference between two point cloud videos and may overestimate the difference at a certain timestep. To better model the distance of two point cloud videos, we propose to use the sum of per-frame Hausdorff distance as the distance metric.

Based on our point cloud video distance metric Eq. (5), we provide the theoretical foundation for our PSTNet++ by showing the universal approximation ability of PSTOp to continuous functions on point cloud videos.

**Theorem 1.** Suppose \( f : X_1 \times X_2 \times \cdots \times X_L \to \mathbb{R} \) is a continuous function w.r.t the \( d_g(\cdot, \cdot) \) in Eq. (5). \( \forall \epsilon > 0 \), \( \exists \) a continuous function \( h \), a set of continuous functions \( \eta_1, \ldots, \eta_L \) and a continuous function \( \gamma \) such that for any \( S = (S_1, S_2, \ldots, S_L) \) \((S_i)_{t=0}^{T} \in X_1 \times X_2 \times \cdots \times X_L, f(S) - \gamma \left( \sum_{t=0}^{T} \eta_t (\max(h(S_i^t), \ldots, h(S_N^t))) \right) < \epsilon, \)

where \( S_1^t, S_2^t, \ldots, S_N^t \) are the elements of \( S_i \) extracted in a certain order and \( \epsilon \) is a vector max operator that takes \( N \) vectors as input and returns a new vector of the element-wise maximum.

The proof of this theorem is in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2021.3135117.

The metric Eq. (5) when \( h \) is a vector max operator that takes \( N \) vectors as input and returns a new vector of the element-wise maximum.
We use four distances \( d = \frac{T}{C_0} \), \( d = \frac{S}{C_0} \), \( \frac{h}{2} \), \( \frac{d^2}{C_0} \) and \( \frac{h}{C_0} \). Therefore, PointNet++ becomes a single function. Therefore, PointNet++ can model the spatial and temporal information simultaneously. However, as the max operation may not properly represent global temporal distance and may lead to inaccuracy, we employ the sum operation to capture the temporal difference. Due to the inconsistency of basic operations between the spatial and temporal dimensions, we decompose space and time in modeling.

2) PointNet++ can be viewed as a special case of PSTNet++. When there is only one frame in a video, Theorem 1 becomes \( \frac{T}{C_0} \left( y \circ \eta_1 \right) \left( \max \left( h_1, \ldots, h_7 \right) \right) < \epsilon \), which is equal to the theorem of PointNet++ (shown in Table 1) by considering \( y \circ \eta_1 \) as a single function. Therefore, PointNet++ can be seen as a special case of our PSTNet++.

3) Comparison with MinkowskiNet [4]. MinkowskiNet is a voxel-based method, which jointly captures the spatial and temporal structure of 4D tesseract via a 4D convolution kernel. In contrast, our PSTNet++ is a point-based method, which separately models the spatial and temporal structure of point tubes via a spatial MLP and a set of temporal MLPs. Both MinkowskiNet and PSTNet++ aim to capture the structure of a spatio-temporal local area, which is the 4D tesseract for MinkowskiNet and point tube for PSTNet++. The different is that, as a voxel-based method, MinkowskiNet can learn a 4D convolution kernel from different discrete directions or displacements. In contrast, as point-based, PSTNet++ captures the spatial structure by directly encoding continuous directions or displacements. Moreover, MinkowskiNet treats space and time equally and models the spatial and temporal structure together, while our PSTNet++ decouples spatio-temporal modeling, as space and time are orthogonal and independent and the scales of space and time are not compatible.

3.3 Point Spatio-Temporal Transposed Operation (PSTTransOp)

After PSTOp, the original point cloud video is both spatially and temporally subsampled. However, for point-level prediction tasks, we need to provide point features for all the original points. Thus, we develop a point spatio-temporal transposed operation (PSTTransOp). Suppose \( \left[ P_1, P_2, \ldots, P_L \right] \) is the encoded video of the original one \( \left[ P_1, P_2, \ldots, P_L \right] \). PSTTransOp propagates features \( \left( F_1, F_2, \ldots, F_L \right) \) to the original point coordinates \( \left( P_1, P_2, \ldots, P_L \right) \), thus outputting new features \( \left( F_1', F_2', \ldots, F_L' \right) \), where \( F_1' \in \mathbb{R}^{C_t \times N} \) and \( C_t \) denotes the new feature dimension. To this end, PSTTransOp first recovers the temporal length by a temporal transposed operation, and then increases the number of points and assigns temporal features to original points via feature propagation [18],

\[
M_{t+h}^{F(x,y,z)} = T-\text{MLP}_{t}^{F(x,y,z)} \left( \sum_{\left( \left( x_1, y_1, z_1 \right), \ldots, \left( x_r, y_r, z_r \right) \right)} w_{\left( \left( x_1, y_1, z_1 \right), \ldots, \left( x_r, y_r, z_r \right) \right)} \right) \text{MLP}_{t}^{F(x+h,y+h,z+h)}
\]

where \( k \in [-r, r] \) and \( w(x, y, z) = \frac{1}{\|\left( x-r, y-r, z-r \right)\|^2} \). An illustration of PSTTransOp operation is shown in Appendix B, available in the online supplemental material.
The temporal padding \( p \) is split as \([p_1, p_2]\), where \( p_1 \) and \( p_2 \) denote the beginning padding and the ending padding, respectively. As 3D action recognition usually uses many frames, to improve computational efficiency and reduce memory usage, we only use only layer in each MLP of our 3D action recognition architecture.

In addition, batch normalization and ReLU activation are inserted between layers. The specs of our PSTNet++ architectures for 3D action recognition and 4D semantic segmentation are described in Table 2.

### 4 Experiments

In this section, we conduct a video-level classification, i.e., 3D action recognition, and a point-level classification, i.e., 4D semantic segmentation, to demonstrate the effectiveness and superiority of PSTNet++. We also apply our PSTNet++ to scene flow estimation to explicitly show its ability for motion modeling.

To build a unified network and train the network with mini-batch, the number of spatial neighbors needs to be fixed. We follow existing point-based works [6], [18], [19], [20] to randomly sample a fixed number of spatial neighbors. For 3D action recognition, the number is set to 9. For 4D semantic segmentation, we follow [5], [6] to set the number to 32. If the actual number of neighbors of a point is less than the set number, we randomly repeat some neighbors.

We train our models for 35 epochs with the SGD optimizer. Learning rate is set to 0.01, and decays with a rate of 0.1 at the 10th epoch and the 20th epoch, respectively.

#### 4.1 3D Action Recognition

To show the effectiveness in video-level classification, we apply our PSTNet++ to 3D action recognition. Following [5], [6], [30], we sample 2,048 points for each frame. Point cloud videos are split into multiple clips (with a fixed number of frames) as inputs. For training, video-level labels are used as clip-level labels. For evaluation, the mean of the clip-level predicted probabilities is used as the video-level prediction. Point colors are not used.

#### 4.1.1 MSR-Action3D

The MSR-Action3D [40] dataset consists of 567 Kinect v1 depth videos, with 20 action categories and 23K frames in total. We use the same training/test split as previous works [5], [6], [39]. Following [5], [6], batch size is set to 16. We set the initial spatial radius \( r_o \) to 0.7. We conduct experiments with 10 times and report the mean.

**Accuracy comparison with the state-of-the-art.** We compare our PSTNet++ with skeleton-based, depth-based and point-based 3D action recognition methods on this dataset. The accuracy comparisons are reported in Table 3. The proposed PSTNet++ significantly outperforms all the state-of-the-art methods, demonstrating the superiority of our PSTOp on feature extraction. For example, when using 24 frames, our PSTNet++ outperforms MeteorNet by 4.18%.

**Comparison with skeleton-based and depth-based methods.** Skeleton-based methods ignore the object and scene information during action recognition. In contrast, our point-based method is able to take the advantage of all the information in a video and thus enhance action reasoning. Depth-based methods project width and height but treat depth as pixel value. As width, height and depth are treated in different ways, depth-based methods may increase the challenge for neural networks to understand 3D data. Moreover, it is difficult to capture precise 3D motion from 2D depth frames. In contrast, our point-based method treats width, height, and depth equally, and thus properly models 3D data and captures 3D motion.

**Memory usage and computational efficiency.** To investigate the running time, we conduct experiments using 1 Nvidia Quadro RTX 6000 GPU. As shown in Table 4, our PSTNet++ uses fewer parameters and is more efficient than MeteorNet. This is because, on one hand, MeteorNet employs three-layer MLPs while the MLPs in our PSTNet++ only contain one layer. One the other hand, when applying to 3D action recognition, to better capture temporal structure, besides each spatio-temporal neighbor’s feature and 4D displacement, MeteorNet also feeds the anchor’s feature into the input of its MLPs. By contrast, benefiting from decomposing space and time, the \( S-\)MLPs in our PSTNet++ only
TABLE 3
Action Recognition Accuracy (%) on the MSR-Action3D [40] Dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Input</th>
<th># Frames</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vieira et al. [37]</td>
<td>depth</td>
<td>20</td>
<td>78.20</td>
</tr>
<tr>
<td>Kläser et al. [38]</td>
<td>depth</td>
<td>18</td>
<td>81.43</td>
</tr>
<tr>
<td>ActionNet [39]</td>
<td>skeleton</td>
<td>all</td>
<td>88.21</td>
</tr>
<tr>
<td>PointNet++ [18]</td>
<td>point</td>
<td>1</td>
<td>61.61</td>
</tr>
<tr>
<td>MeteorNet [5]</td>
<td>point</td>
<td>4</td>
<td>78.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>81.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td>86.53</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16</td>
<td>88.21</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24</td>
<td>88.50</td>
</tr>
<tr>
<td>PSTNet [6]</td>
<td>point</td>
<td>4</td>
<td>81.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>83.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td>87.88</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16</td>
<td>89.90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24</td>
<td>91.20</td>
</tr>
<tr>
<td>PSTNet++ (ours)</td>
<td>point</td>
<td>4</td>
<td>81.53</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>83.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td>88.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16</td>
<td>90.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24</td>
<td>92.68</td>
</tr>
</tbody>
</table>

TABLE 4
Comparison of Memory Usage and Computational Efficiency on 3D Action Recognition

<table>
<thead>
<tr>
<th>Method</th>
<th># Parameters (M)</th>
<th>Running time (ms)</th>
<th>16 frame</th>
<th>24 frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>MeteorNet [5]</td>
<td>17.6</td>
<td>54.46</td>
<td>80.11</td>
<td></td>
</tr>
<tr>
<td>PSTNet [6]</td>
<td>8.44</td>
<td>31.92</td>
<td>43.88</td>
<td></td>
</tr>
<tr>
<td>PSTNet++ (ours)</td>
<td>8.44</td>
<td>31.76</td>
<td>43.19</td>
<td></td>
</tr>
</tbody>
</table>

need to encode each spatial neighbor’s feature and 3D displacement, and thus save parameters and computation.

When using one-layer MPLs for 3D action recognition, our temporal operation is identical to the temporal convolution of PSTNet. To highlight the difference in the spatial modeling, we reformulate their spatial operations as follows,

\[
PSTNet: \sum_{t} S_{l}^{d_{x}, d_{y}, d_{z}} \cdot \left( F_{l}^{t(x+d_{x}, y+d_{y}, z+d_{z})} \right) = \sum \left( W_{1} \cdot \left[ \left( \delta_{x}, \delta_{y}, \delta_{z} \right)^{T} \right] \cdot W_{2} \right) \cdot F_{l}^{t(x+d_{x}, y+d_{y}, z+d_{z})} = \sum \left( W_{1} \cdot \left[ \left( \delta_{x}, \delta_{y}, \delta_{z} \right)^{T} \right] \circ \left( W_{2} \cdot F_{l}^{t(x+d_{x}, y+d_{y}, z+d_{z})} \right) \right)
\]

\[
PSTNet++: \text{MAX} S-MLP \left( F_{l}^{t(x+d_{x}, y+d_{y}, z+d_{z})}, \delta_{x}, \delta_{y}, \delta_{z} \right) = \text{MAX} W \cdot \left( F_{l}^{t(x+d_{x}, y+d_{y}, z+d_{z})}, \delta_{x}, \delta_{y}, \delta_{z} \right) = \text{MAX} \left( W_{1} \cdot \left[ \left( \delta_{x}, \delta_{y}, \delta_{z} \right)^{T} \right] + \left( W_{2} \cdot F_{l}^{t(x+d_{x}, y+d_{y}, z+d_{z})} \right) \right)
\]

where \( W_{1} \in \mathbb{R}^{C_{m} \times 3} \), \( W_{2} \in \mathbb{R}^{C_{m} \times C} \), \( W = [W_{1}, W_{2}] \), \( 1 = (1, \ldots, 1) \in \mathbb{R}^{1 \times C} \) is for broadcasting, \( \cdot \) is element-wise product, and \( C_{m} \) is the dimension of the intermediate feature. As shown in the above equations, the spatial operation of our PSTNet++ uses the same number of parameters as the spatial convolution of PSTNet. The difference is that PSTNet is based on sum pooling and element-wise product while PSTNet++ is based max pooling and addition. Because addition requires less computation than element-wise product, the proposed PSTNet++ is slightly faster than PSTNet.

**What does PSTOp learn?** To investigate what PSTOp learns, we visualize the output of each layer of our PSTNet++ for 3D action recognition in Fig. 5. Because we use ReLU as the activation function, all outputs are greater than zero and large outputs indicate high activation. To visualize the outputs, we squeeze point feature vectors to scalars via \( l_{i} \) norm. As expected, PSTOp outputs higher activation on moving areas. This demonstrates that our PSTOp captures the most informative clues in action reasoning.

![Fig. 4. Influence of temporal radius and initial spatial radius on MSR-Action3D with 24 frames.](image)

4.1.2 N-UCLA and UWA3DII

The N-UCLA [7] dataset contains 1475 action videos, with 10 action categories. These videos are captured using Microsoft Kinect v1 from 3 different viewpoints. The cross-view evaluation setting is used for test. The UWA3DII [8] dataset contains 1075 videos, with 30 categories. The video samples are captured using Microsoft Kinect v1. Batch size is set to 16. The initial spatial radius \( r_{0} \) to 0.5.

The experimental results are listed in Table 6. The proposed PSTNet++ outperforms all the state-of-the-art methods, demonstrating the effectiveness of our method. For example, with the UWA3DII dataset, our PSTNet++ outperforms 3DV-PointNet++ by 3.0%.

4.1.3 NTU RGB+D 60 and NTU RGB+D 120

The NTU RGB+D 60 [44] is the second largest dataset for 3D action recognition. It consists of 56K videos, with 60 action categories and 4M frames in total. The videos are captured using Kinect v2, with 3 cameras and 40 subjects (performers).
The dataset defines two types of evaluation, i.e., cross-subject and cross-view. The cross-subject evaluation splits the 40 performers into training and test groups. Each group consists of 20 performers. The cross-view evaluation uses all the samples from camera 1 for testing and samples from cameras 2 and 3 for training. The NTU RGB+D 120 [45] dataset, the largest dataset for 3D action recognition, is an extension of NTU RGB+D 60. It consists of 114K videos, with 120 action categories and 8M frames in total. The videos are captured with 106 performers and 32 collection setups (locations and backgrounds). Besides cross-subject evaluation, the dataset defines a new evaluation setting, i.e., cross-setup, where 16 setups are used for training, and the others are used for testing.

Batch size is set to 16. We set the frame sampling rate and the initial spatial radius $r_0$ to 2 and 0.1, respectively.

As indicated in Table 8, PSTNet++ outperforms other approaches. Particularly, as indicated by the cross-setup evaluation on NTU RGB+D 120, PSTNet++ outperforms the second best 3DV-PointNet++ [30] by 6.2%. Moreover, compared to PointNet++ [18], which only exploit appearance information, our PSTNet++ effectively captures temporal correlation and thus significantly improves recognition accuracy. For example, the cross-subject evaluation results on NTU RGB+D 60 shows PSTNet++ outperforms PointNet++ by a large margin of 11.3%.

4.2 4D Semantic Segmentation
To demonstrate that our method can be generalized to point-level dense prediction tasks, we apply PSTNet++ to 4D semantic segmentation. Following the works [4], [5], [6], we conduct experiments on point cloud videos with length of 3 frames.

Synthia 4D [4] uses the Synthia dataset [50] to create 3D videos. The Synthia 4D dataset includes 6 sequences of driving scenarios, where both objects and cameras are moving. Each sequence consists of 4 stereo RGB-D images taken from the top of a moving car. Following [5], we reconstruct 3D point cloud sequences from RGB and depth videos, and use the same training/validation/test split, with 19,888/815/1,886 frames, respectively. Following [5], batch size is set to 12. We set the initial spatial radius $r_0$ to 0.9.

We compare PSTNet++ with the state-of-the-art voxel-based and point-based methods on this dataset. Note that, although semantic segmentation can be achieved from a single frame, exploring temporal consistency would facilitate exploring the structure of scenes, thus improving segmentation accuracy and robustness to noise. As seen in Table 10, our PSTNet++ achieves four best accuracies among them. Moreover, our method

![Fig. 5. Visualization of the output of each PSTOp layer in PSTNet++. Top: input point cloud video, where color encodes depth. Bottom: output of PSTOp, where brighter color indicates higher activation. Input videos consist of 24 frames. Due to the spatial subsampling $s_s$ and the temporal stride $s_t$, points and frames progressively decrease along the network layers. Interestingly, PSTOp outputs high activation to salient motion, which supports our intuition that PSTOp effectively captures dynamics of point cloud videos. Note that, PSTOp1 does not capture temporal correlation because its temporal radius $r_t = 0$. In this case, PSTOp1 focuses on the appearance and therefore outputs high activation to the performer contour. Best viewed in color.](image-url)
TABLE 8
Action Recognition Accuracy (%) on the NTU RGB+D 60 [44] and NTU RGB+D 120 [45] Datasets

<table>
<thead>
<tr>
<th>Method</th>
<th>Input</th>
<th>NTU RGB+D 60 Subject</th>
<th>NTU RGB+D 60 View</th>
<th>NTU RGB+D 120 Subject</th>
<th>NTU RGB+D 120 Setup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body Pose Evolution Map [46]</td>
<td>skeleton</td>
<td>88.5</td>
<td>95.1</td>
<td>64.6</td>
<td>66.9</td>
</tr>
<tr>
<td>2s-AGCN [47]</td>
<td>skeleton</td>
<td>89.9</td>
<td>96.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DGNN [48]</td>
<td>skeleton</td>
<td>89.1</td>
<td>96.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Wang et al. [49]</td>
<td>depth</td>
<td>87.1</td>
<td>84.2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MV3I [43]</td>
<td>depth</td>
<td>84.6</td>
<td>87.3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>NTU RGB+D 120 Baseline [45]</td>
<td>depth</td>
<td>-</td>
<td>-</td>
<td>48.7</td>
<td>40.1</td>
</tr>
<tr>
<td>PointNet++ (appearance) [18]</td>
<td>point</td>
<td>80.1</td>
<td>85.1</td>
<td>72.1</td>
<td>79.4</td>
</tr>
<tr>
<td>3DV (motion) [30]</td>
<td>voxel</td>
<td>84.5</td>
<td>95.4</td>
<td>76.9</td>
<td>92.5</td>
</tr>
<tr>
<td>3DV-PointNet++ [30]</td>
<td>voxel + point</td>
<td>88.8</td>
<td>96.3</td>
<td>82.4</td>
<td>93.5</td>
</tr>
<tr>
<td>PSTNet [6]</td>
<td>point</td>
<td>90.5</td>
<td>96.5</td>
<td>87.0</td>
<td>93.8</td>
</tr>
<tr>
<td>PSTNet++ (ours)</td>
<td>point</td>
<td>91.4</td>
<td>96.7</td>
<td>88.6</td>
<td>93.8</td>
</tr>
</tbody>
</table>

saves 0.11M, relative 6% of parameters compared to MeteorNet [5]. We visualize a few segmentation examples in Fig. 6.

4.3 Scene Flow Estimation
Following the previous PSTNet [6] work, we also apply our PSTNet++ to scene flow estimation. We follow the setting proposed by MeteorNet [5], i.e., given a point cloud sequence, estimating a flow vector for every point in the last frame. However, because our point tube is constructed according to the middle frame, which is not applicable to the last-frame scene flow estimation, we follow [6] to adapt temporal anchor frame selection and spatial anchor point transferring. Specifically, we select the last frame in each tube as the anchor frame. Then, after spatial sampling, each anchor point is transferred to its previous nearest \( l \) frames. Following [5], [6], we first train our model on a FlyingThings3D dataset according to the synthetic method in [35], and then fine-tune the model on a KITTI scene flow dataset [5]. Point tracking is not used. As shown in Table 9, our PSTNet++ achieves the highest accuracy on scene flow estimation.

4.4 Ablation Study

4.4.1 Spatio-Temporally Hierarchical Modeling
Because points are not consistent and even flow in and out of a spatial region, embedding points in a spatially local area along an entire video handcrafts capturing accurate local dynamics of point clouds. To address this problem, we construct temporal hierarchy in point cloud modeling. As shown in Table 5, even though without the decomposition of spatio-temporal encoding, our PSTNet++ still outperforms MeteorNet. Moreover, without temporally hierarchical modeling, MeteorNet only achieves a slight improvement of 0.29% from 16 frames to 24 frames. By contrast, PSTNet++ achieves an improvement of 1.39%, showing the effectiveness of temporally hierarchical modeling when encoding long videos.

4.4.2 Decomposition of Spatio-Temporal Encoding
To investigate our PSTNet++ without decomposing space and time, we treat a 3D point cloud video as a 4D point set so that our method can model pint coordinates and timestamps together. This makes our method similar to MeteorNet, which is based on the max-based video distance metric Eq. (3). As shown in Table 5, even though without temporal hierarchy, decomposing space and time helps our PSTNet++ to outperform MeteorNet. Moreover, from 16 frames to 24 frames, our PSTNet++ achieves an improvement of 1.05%, while MeteorNet only achieves a slight improvement of 0.29%. The supports our motivation that, by decomposing space and time, we can leverage our sum-based video distance metric Eq. (5), which is able to reflect the global difference of two point cloud videos. However, the max-based video distance metric used in MeteorNet tends to focus on a specific moment and miss the information in other frames and thus cannot adequately capture the temporal structure of long videos.

4.4.3 Temporal Modeling
In this paper, we propose to use the sum of per-frame Hausdorff distance as the point cloud video distance metric, as shown in Eq. (4). In this way, the difference at each moment in a video is involved and considered. Moreover, we further add the temporal encoding, i.e., \( \{ \lambda_t \} \), into Eq. (4) to capture the temporal structure, resulting in Eq. (5). In this section, we investigate the influence of the temporal encoding. As shown in Table 7, exploiting the temporal encoding effectively improves action recognition.

4.4.4 Clip Length
Usually, information is not equally distributed in videos along time. Short point cloud clips may miss key frames and thus confuse models as noise. Therefore, as shown in Table 3, increasing clip length (i.e., the number of frames) benefits models for action recognition.

4.4.5 Temporal Radius
The temporal radius \( r_t \) controls the temporal dynamics modeling of point cloud videos. Fig. 4 shows the accuracy on MSR-Action3D with different \( r_t \).

TABLE 9
Scene Flow Estimation Accuracy on the KITTI Scene Flow Dataset [5]

<table>
<thead>
<tr>
<th>Method</th>
<th>Input</th>
<th># Frames</th>
<th>End-Point-Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlowNet3D [35]</td>
<td>points</td>
<td>2</td>
<td>0.287</td>
</tr>
<tr>
<td>MeteorNet [5]</td>
<td>points</td>
<td>3</td>
<td>0.282</td>
</tr>
<tr>
<td>PSTNet [6]</td>
<td>points</td>
<td>3</td>
<td>0.278</td>
</tr>
<tr>
<td>PSTNet++ (ours)</td>
<td>points</td>
<td>3</td>
<td>0.276</td>
</tr>
</tbody>
</table>

Authorized licensed use limited to: University of Technology Sydney. Downloaded on November 10,2022 at 04:23:57 UTC from IEEE Xplore. Restrictions apply.
When \( r_t \) is set to 0, temporal correlation is not captured. However, PSTNet++ can still observe 24 frames and the pooling operation allows PSTNet++ to capture the pose information of an entire clip. Moreover, some actions (e.g., “golf swing”) can be easily recognized by a certain pose, and thus PSTNet++ with \( r_0 = 1 \) can still achieve satisfactory accuracy.

When \( r_t \) is greater than 1, PSTNet++ models temporal dynamics and therefore improves accuracy on actions that rely on motion or trajectory reasoning (e.g., “draw x”, “draw tick” and “draw circle”).

When \( r_t \) is greater than 3, the accuracy decreases. This mainly depends on motion in sequences. Because most actions in MSR-Action3D are fast (e.g., “high arm wave”), using smaller temporal kernel size facilitates capturing fast motion, and long-range temporal dependencies will be captured in high-level layers. Since we aim to present generic point-based operations, we do not tune the kernel size for each action but use the same size.

4.4.6 Spatial Radius

The spatial radius controls the range of the spatial structure to be modeled, and thus is important for extracting the spatially-local correlation. As shown in Fig. 4, using too small spatial radius cannot capture sufficient structure information while using large spatial radius will decrease the discriminativeness of local structure for temporal modeling.

5 CONCLUSION

In this paper, we propose an effective 3D point spatio-temporal operation (PSTOp) to learn informative representations from point cloud videos. Considering 3D points are spatially irregular and temporally ordered, we decouple the feature extraction in two successive steps, i.e., a spatial operation and a temporal operation, with the help of our newly introduced point tubes. Then, we develop a point spatio-temporal transposed operation (PSTTransOp) for point-level dense prediction tasks. We also demonstrate that by incorporating the proposed operations into deep network, dubbed PSTNet++, our method is competent to address various point-based tasks. Extensive experiments demonstrate that our PSTNet++ significantly improves the recognition and segmentation performance by effectively modeling point cloud videos. Moreover, we theoretically prove that our PSTNet++ is able to effectively model point cloud videos.

**REFERENCES**


Hehe Fan received the PhD degree in faculty of engineering and information technology from the University of Technology Sydney, Sydney, NSW, Australia, in 2020. He is currently a research fellow with the School of Computing, National University of Singapore, Singapore. His research interests include deep learning, computer vision and multimedia, with an emphasis on video analysis, dynamic point cloud, unsupervised learning, and vision-language applications.

Xin Yu received the BS degree in electronic engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2009 and the dual PhD degree from the Department of Electronic Engineering, Tsinghua University, Beijing, China, in 2015 and from the College of Engineering and Computer Science, Australian National University, Canberra, ACT, Australia, in 2019. He is currently a lecturer with the University of Technology Sydney. His research interests include computer vision and image processing.

Yi Yang (Senior Member, IEEE) received the PhD degree in computer science from Zhejiang University, Hangzhou, China, in 2010. He was a postdoctoral researcher with the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA. His research interests include machine learning and its applications to multimedia content analysis and computer vision, including multimedia indexing and retrieval, surveillance video analysis, and video content understanding.

Mohan Kankanhalli (Fellow, IEEE) received the BTech degree from IIT Kharagpur and the MS and PhD degrees from the Rensselaer Polytechnic Institute. He is currently the Provost’s chair professor with the Department of Computer Science, National University of Singapore (NUS). He is the director of N-CRiPT and also the dean with the School of Computing, NUS. His research interests include multimedia computing, multimedia security and privacy, image or video processing, and social media analysis. He is active with the Multimedia Research Community and on the editorial boards of several journals.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.