STMT: A Spatial-Temporal Mesh Transformer for MoCap-Based Action Recognition

Xiaoyu Zhu\textsuperscript{1}, Po-Yao Huang\textsuperscript{2†}, Junwei Liang\textsuperscript{3†}, Celso M. de Melo\textsuperscript{4}, Alexander Hauptmann\textsuperscript{1}
\textsuperscript{1}Carnegie Mellon University, \textsuperscript{2}FAIR, Meta AI, \textsuperscript{3}HKUST (Guangzhou), \textsuperscript{4}DEVCOM Army Research Laboratory

Abstract

We study the problem of human action recognition using motion capture (MoCap) sequences. Unlike existing techniques that take multiple manual steps to derive standardized skeleton representations as model input, we propose a novel Spatial-Temporal Mesh Transformer (STMT) to directly model the mesh sequences. The model uses a hierarchical transformer with intra-frame off-set attention and inter-frame self-attention. The attention mechanism allows the model to freely attend between any two vertex patches to learn non-local relationships in the spatial-temporal domain. Masked vertex modeling and future frame prediction are used as two self-supervised tasks to fully activate the bi-directional and auto-regressive attention in our hierarchical transformer. The proposed method achieves state-of-the-art performance compared to skeleton-based and point-cloud-based models on common MoCap benchmarks. Code is available at \url{https://github.com/zgzxy001/STMT}.

1. Introduction

Motion Capture (MoCap) is the process of digitally recording the human movement, which enables the fine-grained capture and analysis of human motions in 3D space \cite{40, 50}. MoCap-based human perception serves as key elements for various research fields, such as action recognition \cite{15, 46–48, 50, 57}, tracking \cite{47}, pose estimation \cite{1, 27}, imitation learning \cite{76}, and motion synthesis \cite{47}. Besides, MoCap is one of the fundamental technologies to enhance human-robot interactions in various practical scenarios including hospitals and manufacturing environment \cite{22, 28, 41, 43, 45, 47}. For example, Hayes \cite{22} classified automotive assembly activities using MoCap data of humans and objects. Understanding human behaviors from MoCap data is fundamentally important for robotics perception, planning, and control.

Skeleton representations are commonly used to model MoCap sequences. Some early works \cite{3, 29} directly used body markers and their connectivity relations to form a skeleton graph. However, the marker positions depend on each subject (person), which brings sample variances within each dataset. Moreover, different MoCap datasets usually have different numbers of body markers. For example, ACCAD \cite{48}, BioMotion \cite{64}, Eyes Japan \cite{15}, and KIT \cite{42} have 82, 41, 37, and 50 body markers respectively. This prevents the model to be trained and tested on a unified framework. To use standard skeleton representa-
The contributions of this paper are three-fold: compared to skeleton-based and point-cloud-based models. Our method achieves state-of-the-art performance for the downstream action recognition task. To predict future frames, the model needs to learn prior knowledge about the human body in the spatial dimension. To reconstruct masked vertices of different body parts, the model needs to learn prior knowledge about the human body in the spatial dimension. To overcome these disadvantages, we propose a mesh-based action recognition method to directly model dynamic changes in raw mesh sequences, as illustrated in Figure 1.

Though mesh representations provide fine-grained body information, it is challenging to classify high-dimensional mesh sequences into different actions. First, unlike structured 3D skeletons which have joint correspondence across frames, there is no vertex-level correspondence in meshes (i.e., the vertices are unordered). Therefore, the local connectivity of every single mesh can not be directly aggregated in the temporal dimension. Second, mesh representations encode local connectivity information, while action recognition requires global understanding in the whole spatial-temporal domain.

To overcome the aforementioned challenges, we propose a novel Spatial-Temporal Mesh Transformer (STMT). STMT leverages mesh connectivity information to build patches at the frame level, and uses a hierarchical transformer which can freely attend to any intra- and inter-frame patches to learn spatial-temporal associations. The hierarchical attention mechanism allows the model to learn patch correlation across the entire sequence, and alleviate the requirement of explicit vertex correspondence. We further define two self-supervised learning tasks, namely masked vertex modeling and future frame prediction, to enhance the global interactions among vertex patches. To reconstruct masked vertices of different body parts, the model needs to learn prior knowledge about the human body in the spatial dimension. To predict future frames, the model needs to understand meaningful surface movement in the temporal dimension. To this end, our hierarchical transformer pre-trained with those two objectives can further learn spatial-temporal context across entire frames, which is beneficial for the downstream action recognition task.

We evaluate our model on common MoCap benchmark datasets. Our method achieves state-of-the-art performance compared to skeleton-based and point-cloud-based models. The contributions of this paper are three-fold:

- We introduce a new hierarchical transformer architecture, which jointly encodes intrinsic and extrinsic representations, along with intra- and inter-frame attention, for spatial-temporal mesh modeling.
- We design effective and efficient pretext tasks, namely masked vertex modeling and future frame prediction, to enable the model to learn from the spatial-temporal global context.
- Our model achieves superior performance compared to state-of-the-art point-cloud and skeleton models on common MoCap benchmarks.

2. Related Work

Action Recognition from Depth and Point Cloud. 3D action recognition models have achieved promising performance with depth [34, 55, 56, 68, 71] and point clouds [18,36,52,70]. Depth provides reliable 3D structural and geometric information which characterizes informative human actions. In MVDI [71], dynamic images [4] were extracted through multi-view projections from depth videos for 3D action recognition. 3DFCNN [55] directly exploited a 3D-CNN to model depth videos. Another popular category of 3D human action recognition is based on 3D point clouds. PointNet [51] and PointNet++ [52] are the pioneering works contributing towards permutation invariance of 3D point sets for representing 3D geometric structures. Along this avenue, MeteorNet [36] stacked multi-frame point clouds and aggregates local features for action recognition. 3DV [70] transferred point cloud sequences into regular voxel sets to characterize 3D motion compactly via temporal rank pooling. PSTNet [18] disentangled space and time to alleviate point-wise spatial variance across time. Action recognition has shown promising results with 3D skeletons and point clouds. Meshes, which are commonly used in representing human bodies and creating action sequences, have not been explored for the action recognition task. In this work, we propose the first mesh-based action recognition model.

MoCap-Based Action Recognition. Motion-capture (MoCap) datasets [15, 44, 46–48, 50, 57] serve as key elements for various research fields, such as action recognition [15, 44, 46–48, 50, 57], tracking [47], pose estimation [1, 27], imitation learning [76], and motion synthesis [47]. MoCap-based action recognition was formulated as a skeleton-based action recognition problem [50]. Various architectures have been investigated to incorporate skeleton sequences. In [14, 35, 75], skeleton sequences were treated as time-series inputs to RNNs. [24,69] respectively transformed skeleton sequences into spectral images and trajectory maps and then adopted CNNs for feature learning. In [72], Yan et al. leveraged GCN to model joint dependencies that can be naturally represented with a graph. In this paper, we propose a novel method to directly model the dynamics of raw mesh sequences which can benefit from surface motion and body shape knowledge.
Masked Autoencoder. Masked autoencoder has gained attention in Natural Language Processing and Computer Vision to learn effective representations using auto-encoding. Stacked denoising autoencoders [66] treated masks as a noise type and used denoising autoencoders to denoise corrupted inputs. ViT [13] proposed a self-supervised pre-training task to reconstruct masked tokens. More recently, BEiT [2] proposed to learn visual representations by reconstructing the discrete tokens [53]. MAE [23] proposed a simple yet effective asymmetric framework for masked image modeling. In 3D point cloud analysis, Wang et al. [67] chose to first generate partial point clouds by calculating occlusion from random camera viewpoints, and then complete occluded point clouds using autoencoding. Point-BERT [73] followed the success of BERT [12] to predict the masked tokens learned from points. However, applying self-supervised learning to temporal 3D sequences (i.e., point cloud, 3D skeleton) has not been fully explored. One probable reason is that self-supervised learning on high-dimensional 3D temporal sequences is computationally-expensive. In this work, we propose an effective and efficient self-supervised learning method based on masked vertex modeling and future frame prediction.

3. Method

3.1. Overview

In this section, we describe our model for mesh-based action recognition, which we call STMT. The inputs of our model are temporal mesh sequences: \( M = ((P_1, A_1), (P_2, A_2), \ldots, (P_t, A_t)) \), where \( t \) is the frame number. \( P_i \in \mathbb{R}^{N \times 3} \) represents the vertex positions in Cartesian coordinates, where \( N \) is the number of vertices. \( A_i \in \mathbb{R}^{N \times N} \) represents the adjacency matrix of the mesh. Element \( A_{mn} \) is one when there is an edge from vertex \( V_m \) to vertex \( V_n \), and zero when there is no edge. The mesh representation with vertices and their adjacent matrix is a unified format for various body models such as SMPL [39], SMPL-H [54], and SMPL-X [49]. In this work, we use SMPL-H body models from AMASS [40] to obtain the mesh sequences, but our method can be easily adapted to other body models.

Mesh’s local connectivity provides fine-grained information. Previous methods [21, 59] proved that explicitly using surface (e.g., mesh) connectivity information can achieve higher accuracy in shape classification and segmentation tasks. However, classifying temporal mesh sequences is a more challenging problem, as there is no vertex-level correspondence across frames. This prevents graph-based models from directly aggregating vertices in the temporal dimension. Therefore, we propose to first leverage mesh connectivity information to build patches at the frame level, then use a hierarchical transformer which can freely attend to any intra- and inter-frame patches to learn spatial-temporal associations. In summary, it has the following key components:

- **Surface Field Convolution** to form local vertex patches by considering both intrinsic and extrinsic mesh representations.
- **Hierarchical Spatial-Temporal Transformer** to learn spatial-temporal correlations of vertex patches.
- **Self-Supervised Pre-Training** to learn the global context in terms of appearance and motion.

See Figure 2 for a high-level summary of the model, and the sections below for more details.

3.2. Surface Field Convolution

Because displacements in grid data are regular, traditional convolutions can directly learn a kernel for elements within a region. However, mesh vertices are unordered and irregular. Considering the special mesh representations, we represent each vertex by encoding features from its neighbor vertices inspired by [51, 52]. To fully utilize meshes’ local connectivity information, we consider the mesh properties of extrinsic curvature of submanifolds and intrinsic curvature of the manifold itself. Extrinsic curvature between two vertices is approximated using Euclidean distance. Intrinsic curvature is approximated using Geodesic distance, which is defined as the shortest path between two vertices on mesh surfaces. We propose a light-weighted surface field convolution to build local patches, which can be denoted as:

\[
F_{VG}^{(x,y,z)}(x, y, z) = \sum_{(\delta_x, \delta_y, \delta_z) \in G(x, y, z)} W^{(x, y, z)}(x + \delta_x, y + \delta_y, z + \delta_z)
\]

(1)

\[
F_{VE}^{(x,y,z)}(x, y, z) = \sum_{(\xi_x, \xi_y, \xi_z) \in E(x, y, z)} W^{(x, y, z)}(x + \xi_x, y + \xi_y, z + \xi_z)
\]

(2)

\( G \) and \( E \) is the local region around vertex \((x, y, z)\). In this paper, we use k-nearest-neighbor to sample local vertices. \((\delta_x, \delta_y, \delta_z)\) and \((\xi_x, \xi_y, \xi_z)\) represent the spatial displacement in geodesic and euclidean space, respectively. \(F^{(x,y,z)}\) denotes the feature of the vertex at position \((x, y, z)\).

3.3. Hierarchical Spatial-Temporal Transformer

We propose a hierarchical transformer that consists of intra-frame and inter-frame attention. The basic idea behind our transformer is three-fold: (1) Intra-frame attention can encode connectivity information from the adjacency matrix, while such information can not be directly
features and the input features by element-wise subtraction. It translates the offset (difference) between the self-attention (SA) tex patches. Specifically, the offset-attention layer calculation tasks [20]. We adapt offset-attention to attend to vertex patches. (3) Inter-frame self-attention can learn feature correlations in the spatial-temporal domain.

### 3.3.1 Intra-Frame Offset-Attention

Graph convolution networks [6] show the benefits of using a Laplacian matrix \( L = D - E \) to replace the adjacency matrix \( E \), where \( D \) is the diagonal degree matrix. Inspired by this, offset-attention has been proposed and achieved superior performance in point-cloud classification and segmentation tasks [20]. We adapt offset-attention to attend to vertex patches. Specifically, the offset-attention layer calculates the offset (difference) between the self-attention (SA) features and the input features by element-wise subtraction. Offset-attention is denoted as:

\[
F_{\text{out}} = OA(F_{\text{in}}) = \phi(F_{\text{in}} - F_{sa}) + F_{\text{in}},
\]

where \( \phi \) denotes a non-linear operator. \( F_{\text{in}} - F_{sa} \) is proved to be analogous to discrete Laplacian operator [20], i.e.,

\[
F_{\text{in}} - F_{sa} \approx LF_{\text{in}}.
\]

As Laplacian operators in geodesic and euclidean space are expected to be different, we propose to use separate transformers to model intrinsic patches and extrinsic patches. Specifically, the aggregated feature for vertex \( V \) is denoted as:

\[
F_{V_{sa}}^{(x,y,z)} = OA_G(F_{V_{G}}^{(x,y,z)}) \oplus OA_E(F_{V_{E}}^{(x,y,z)})
\]

Here \( F_{V_{G}}^{(x,y,z)} \in \mathbb{R}^{N \times d_g} \) and \( F_{V_{E}}^{(x,y,z)} \in \mathbb{R}^{N \times d_e} \) are local patches learned using Equ. 1 and Equ. 2. \( F_{V_{G}}^{(x,y,z)} \in \mathbb{R}^{N \times d} \) denotes the local patch for position \((x, y, z)\), where \( d = d_g + d_e \). The weights of \( OA_G \) and \( OA_E \) are not shared.

### 3.3.2 Inter-Frame Self-Attention

Given \( F' \) which encodes local connectivity information, we use self-attention (SA) [65] to learn semantic affinities between different vertex patches across frames. Specifically, let \( Q, K, V \) be the query, key and value, which are generated by applying linear transformations to the input features \( F' \in \mathbb{R}^{N \times d} \) as follows:

\[
(Q, K, V) = (F') \cdot (W_q, W_k, W_v)
\]

\[
Q, K \in \mathbb{R}^{N \times d_a}, \quad V \in \mathbb{R}^{N \times d}
\]

\[
W_q, W_k, W_v \in \mathbb{R}^{d \times d_a}, \quad W_v \in \mathbb{R}^{d \times d}
\]

where \( W_q, W_k \) and \( W_v \) are the shared learnable linear transformation, and \( d_a \) is the dimension of the query and key vectors. Then we can use the query and key matrices to calculate the attention weights via the matrix dot-product:

\[
A = (\tilde{a})_{i,j} = \text{softmax}(\frac{Q \cdot K^T}{\sqrt{d_a}}).
\]

\[
F_{sa} = A \cdot V
\]

The self-attention output features \( F_{sa} \) are the weighted sums of the value vector using the corresponding attention weights. Specifically, for a vertex patch in position \((x, y, z)\),
its aggregated feature after inter-frame self-attention can be computed as: \( P'_x, y, z = \sum A(x, y, z), (x', y', z') \times V(x', y', z') \), where \((x', y', z')\) belongs to the Cartesian coordinates of \( F'_x \).

### 3.4. Self-Supervised Pre-Training

Self-supervised learning has achieved remarkable results on large-scale image datasets [23]. However, self-supervised learning for temporal 3D sequences (i.e. point cloud, 3D skeleton) remains to be challenging and has not been fully explored. There are two possible reasons: (1) self-supervised learning methods rely on large-scale datasets to learn meaningful patterns [10]. However, existing MoCap benchmarks are relatively small compared to 2D datasets like ImageNet [11]. (2) Self-supervised learning for 3D data sequences is computationally expensive in terms of memory and speed. In this work, we first propose a simple and effective method to augment existing MoCap sequences, and then define two effective and efficient self-supervised learning tasks, namely masked vertex modeling and future frame prediction, which enable the model to learn global context. The work that is close to us is OcCO [67], which proposed to use occluded point cloud reconstruction as the pretext task. OcCO has a computationally-expensive process to generate occlusions, including point cloud projection, occluded point calculation, and a mapping step to convert camera frames back to world frames. Different from OcCO, we randomly mask vertex patches or future frames on the fly, which saves the pre-processing step. Moreover, our pre-training method is designed for temporal mesh sequences and considers both bi-directional and autoregressive attention.

#### 3.4.1 Data Augmentation through Joint Shuffle

Considering the flexibility of SMPL-H representations, we propose a simple yet effective approach to augment SMPL-H sequences by shuffling body pose parameters. Specifically, we split SMPL-H pose parameters into five body parts: bone, left/right arm, and left/right leg. We use \( I_{\text{bone}}, I_{\text{left}}, I_{\text{right}}, I_{\text{left}}, I_{\text{right}} \) to denote the SMPL-H pose indexes of the five body parts. Then we synthesize new sequences by randomly selecting body parts from five different sequences. We keep the temporal order for each part such that the merged action sequences have meaningful motion trajectories. Pseudocode for the joint shuffle is provided in Algorithm 1. The input to Joint Shuffle are SMPL-H pose parameters \( \theta \in \mathbb{R}^{b \times t \times n \times 3} \), where \( b \) is the sequence number, \( t \) is the frame number, and \( n \) is the joint number. We randomly select the shape \( \beta \) and dynamic parameters \( \phi \) from one of the five SMPL-H sequences to compose a new SMPL-H body model. Given \( b \) SMPL-H sequences, we can synthesize \( b \times \frac{5^t (5^t - 1)}{2} \) number of new sequences.

We prove that the model can benefit from large-scale pre-training in Section 4.6.

#### Algorithm 1: Pseudocode of STMT Joint Shuffle

```plaintext
Algorithm 1: Pseudocode of STMT Joint Shuffle
1: function STMT_JOINT_SHUFFLE(\( \theta \in \mathbb{R}^{b \times t \times n \times 3} \), \( I_{\text{bone}}, I_{\text{left}}, I_{\text{right}}, I_{\text{left}}, I_{\text{right}} \))
2: \( \theta_s \leftarrow \text{random sample}(\theta, 5) \) \( \Rightarrow \theta_s \in \mathbb{R}^{5 \times t \times n \times 3} \), randomly sample five SMPL-H sequences
3: \( t_{\text{max}} \leftarrow \text{get max length}(\theta_s) \) \( \Rightarrow \text{compute the maximum sequence length in } \theta_s \)
4: \( \theta_{\text{new}} \leftarrow \text{Initialize}(t_{\text{max}}, n, 3) \)
5: \( P \leftarrow \{ I_{\text{bone}}, I_{\text{left}}, I_{\text{right}}, I_{\text{left}}, I_{\text{right}} \} \)
6: for \( i \) in 0, 1, 2, 3, 4 do
7: \( \theta_s \leftarrow \text{repeat}(\theta_s[i], (t_{\text{max}}, n, 3)) \) \( \Rightarrow \text{pad each sequence to the max length using repeating} \)
8: \( \theta_{\text{new}}[P[i]] \leftarrow \theta_s[i][P[i]] \) \( \Rightarrow \text{assign the body-part sequence} \)
9: return \( \theta_{\text{new}} \)
```

#### 3.4.2 Masked Vertex Modeling with Bi-Directional Attention

To fully activate the inter-frame bi-directional attention in the transformer, we design a self-supervised pretext task named Masked Vertex Modeling (MVM). The model can learn human prior information in the spatial dimension by reconstructing masked vertices of different body parts. We randomly mask \( r \) percentages of the input vertex patches, and force the model to reconstruct the full sequences. Moreover, we use bi-directional attention to learn correlations among all remaining local patches. Each patch will attend to all patches in the entire sequence. It models the joint distribution of vertex patches over the whole temporal sequences \( x \) as the following product of conditional distributions, where \( x_i \) is a single vertex patch:

\[
p(x) = \prod_{i=1}^{N} p(x_i | x_1, ..., x_i, ..., x_N). \tag{8}
\]

Where \( N \) is the number of patches in the entire sequence \( x \) after masking. Every patch will attend to all patches in the entire sequence. In this way, bi-directional attention is fully-activated to learn spatial-temporal features that can accurately reconstruct complete mesh sequences.

#### 3.4.3 Future Frame Prediction with Auto-Regressive Attention

The masked vertex modeling task is to reconstruct masked vertices in different body parts. The model can reconstruct completed mesh sequences if it captures the human body prior or can make a movement inference from nearby frames. As action recognition requires the model to understand the global context, we propose the future frame...
prediction (FFP) task. Specifically, we mask out all the future frames and force the transformer to predict the masked frames. Moreover, we propose to use auto-regressive attention for the future frame prediction task, inspired by language generation models like GPT-3 [5]. However, directly using RNN-based models [9] in GPT-3 to predict future frames one by one is inefficient, as 3D mesh sequences are denser compared to language sequences. Therefore, we propose to reconstruct all future frames in a single forward pass. For auto-regressive attention, we model the joint distribution of vertex patches over a mesh sequence as the following product of conditional distributions, where \( x_i \) is a single patch at frame \( t_i \):

\[
p(x) = \prod_{i=1}^{N} p(x_i | x_1, x_2, ..., x_M).
\]  

(9)

Where \( N \) is the number of patches in the entire sequence \( x \) after masking, \( M = (t_i - 1) \times n \), where \( n \) is the number of patches in a single frame. Each vertex patch depends on all patches that are temporally before it. The auto-regressive attention enables the model to predict movement patterns and trajectories, which is beneficial for the downstream action recognition task.

3.5. Training

In the pre-training stage, we use PCN [74] as the decoder to reconstruct masked vertices and predict future frames. The decoder is shared for the two pretext tasks. Since mesh vertices are unordered, the reconstruction loss and future prediction loss should be permutation-invariant. Therefore, we use Chamfer Distance (CD) as the loss function to measure the difference between the model predictions and ground truth mesh sequences.

\[
CD(M_{\text{pred}}, M_{\text{gt}}) = \frac{1}{|M_{\text{pred}}|} \sum_{x \in M_{\text{pred}}} \min_{y \in M_{\text{gt}}} \|x - y\|_2 + \frac{1}{|M_{\text{gt}}|} \sum_{y \in M_{\text{gt}}} \min_{x \in M_{\text{pred}}} \|y - x\|_2
\]

(10)

CD (10) calculates the average closest euclidean distance between the predicted mesh sequences \( M_{\text{pred}} \) and the ground truth sequences \( M_{\text{gt}} \). The overall loss is a weighted sum of masked vertex reconstruction loss and future frame prediction loss:

\[
L = \lambda_1 CD(M_{\text{pred}^{MV}}, M_{\text{gt}}) + \lambda_2 CD(M_{\text{pred}^{FFP}}, M_{\text{gt}})
\]

(11)

In the fine-tuning stage, we replace the PCN decoder with an MLP head. Cross-entropy loss is used for model training.

4. Experiment

4.1. Datasets

Following previous MoCap-based action recognition methods [50, 63], we evaluate our model on the most widely used benchmarks: KIT [42] and BABEL [50]. KIT is one of the largest MoCap datasets. It has 56 classes with 6,570 sequences in total. (2) BABEL is the largest 3D MoCap dataset that unifies 15 different datasets. BABEL has 43 hours of MoCap data performed by over 346 subjects. We use the 60-class subset from BABEL, which contains 21,653 sequences with single-class labels. We randomly split each dataset into training, test, and validation set, with ratios of 70%, 15%, and 15%, respectively. Note that existing action recognition datasets with skeletons only are not suitable for our experiments, as they do not provide full 3D surfaces or SMPL parameters to obtain the mesh representation.

Motion Representation. Both KIT and BABEL’s MoCap sequences are obtained from AMASS dataset in SMPL-H format. A MoCap sequence is an array of pose parameters over time, along with the shape and dynamic parameters. For skeleton-based action recognition, we follow previous work [50] which predicted the 25-joint skeleton from the vertices of the SMPL-H mesh. The movement sequence is represented as \( X = \{x_1, \ldots, x_L\} \), where \( x_i \in \mathbb{R}^{J \times 3} \) represents the position of the \( J \) joints in the skeleton in Cartesian coordinates. For point-cloud-based action recognition, we directly use the vertices of SMPL-H model as the model input. The point-cloud sequence is represented as \( P = \{p_1, \ldots, p_L\} \), where \( p_i \in \mathbb{R}^{V \times 3} \) and \( V \) is the number of vertices. For mesh-based action recognition, we represent the motion as a series of mesh vertices and their adjacent matrix over time, as introduced in Section 3.1. See Sup. Mat. for more details about datasets and pre-processing.

4.2. Baseline Methods

We compare our model with state-of-the-art 3D skeleton-based and point cloud-based action recognition models, as there is no existing literature on mesh-based action recognition. 2s-AGCN [61], CTR-GCN [7], and MS-G3D [37] are used as skeleton-based baselines. Among those methods, 2s-AGCN trained with focal loss and cross-entropy loss are used as benchmark methods in the BABEL dataset [50]. For the comparison with point-cloud baselines, we choose PSTNet [18], SequentialPointNet [30], and P4Transformer [16]. Those methods achieved top performance on common point-cloud-based action recognition benchmarks.

4.3. Implementation Details

For skeleton-based baselines, we use the official implementations of 2s-ACGN, CTR-GCN, and MS-G3D from
<table>
<thead>
<tr>
<th>Method</th>
<th>Input</th>
<th>KIT (Top-1) (%)</th>
<th>BABEL-60 (Top-5) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2s-AGCN-FL [61] (CVPR’19)</td>
<td>3D Skeleton</td>
<td>42.44</td>
<td>49.62</td>
</tr>
<tr>
<td>2s-AGCN-CE [61] (CVPR’19)</td>
<td>3D Skeleton</td>
<td>57.46</td>
<td>63.57</td>
</tr>
<tr>
<td>CTR-GCN [7] (ICCV’21)</td>
<td>3D Skeleton</td>
<td>64.65</td>
<td>67.30</td>
</tr>
<tr>
<td>MS-G3D [37] (CVPR’20)</td>
<td>3D Skeleton</td>
<td>65.38</td>
<td>67.43</td>
</tr>
<tr>
<td>PSTNet [18] (ICLR’21)</td>
<td>Point Cloud</td>
<td>56.93</td>
<td>61.94</td>
</tr>
<tr>
<td>P4Transformer [16] (CVPR’21)</td>
<td>Point Cloud</td>
<td>62.15</td>
<td>63.54</td>
</tr>
<tr>
<td>STMT (Ours)</td>
<td>Mesh</td>
<td>65.59</td>
<td>67.65</td>
</tr>
</tbody>
</table>

Table 1. Experimental Results on KIT and BABEL Dataset.

<table>
<thead>
<tr>
<th>Intrinsic</th>
<th>Extrinsic</th>
<th>MVM</th>
<th>FFP</th>
<th>Top-1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>63.40</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>64.03</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>64.96</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>64.13</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>65.59</td>
</tr>
</tbody>
</table>

Table 2. Performance of ablated versions. Intrinsic and Extrinsic stand for the intrinsic (geodesic) and extrinsic (euclidean) features in surface field convolution. MVM stands for Masked Vertex Modelling. FFP stands for Future Frame Prediction.

4.4. Main Results

Comparison with State-of-the-Art Methods. As indicated in Table 1, STMT outperforms all other state-of-the-art models. Our model can outperform point-cloud-based models by 3.44% and 4.11% on KIT and BABEL datasets in terms of top-1 accuracy. Moreover, compared to skeleton-based methods which involve manual efforts to convert mesh vertices to skeleton representations, our model achieves better performance by directly modeling the dynamics of raw mesh sequences.

We visualize the inter-frame attention weights of our hierarchical transformer in Figure 3. We observe that the model can pay attention to key regions across frames. This supports the intuition that our hierarchical transformer can take the place of explicit vertex tracking by learning spatial-temporal correlations.

4.5. Ablation Study

Ablation Study of STMT. We test various ablations of our model on the KIT dataset to substantiate our design decisions. We report the results in Table 2. Note that Joint Shuffle is used in all of the self-supervised learning experiments (last three rows). We observe that each component of our model gains consistent improvements. The comparison of the first two rows proves the effectiveness of encoding both intrinsic and extrinsic features in vertex patches. Comparing the last three rows with the second row, we observe a consistent improvement using self-supervised pre-training. Moreover, the downstream task can achieve better performance with MVM compared to FFP. One probable reason is that the single task for future frame prediction is more challenging than masked vertex modeling, as the model can only see the person movement in the past. The model can achieve the best performance with both MVM and FFP, which demonstrates that the two self-supervised tasks are supplementary to each other.

4.6. Analysis

Different Pre-Training Strategies. We pre-train our model with different datasets and summarize the results in Table 3. The first row shows the case without pre-training. The second shows the result for the model pre-trained on the KIT dataset (without Joint Shuffle augmentation). The third shows the result for the model pre-trained on KIT dataset (with Joint Shuffle). We observe our model can achieve better performance with Joint Shuffle, as it can synthesize large-scale mesh sequences.
### Table 3. Comparison of Different Pre-Training Strategies. JS stands for Joint Shuffle.

<table>
<thead>
<tr>
<th>Method</th>
<th>Top-1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o pre-training</td>
<td>64.03</td>
</tr>
<tr>
<td>pre-training w/o JS</td>
<td>64.13</td>
</tr>
<tr>
<td>pre-training w/ JS</td>
<td><strong>65.59</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>r</th>
<th>Pre-Train Loss ($\times 10^4$)</th>
<th>Fine-Tune Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.39</td>
<td>64.44</td>
</tr>
<tr>
<td>0.3</td>
<td>0.41</td>
<td>64.55</td>
</tr>
<tr>
<td><strong>0.5</strong></td>
<td><strong>0.40</strong></td>
<td><strong>65.59</strong></td>
</tr>
<tr>
<td>0.7</td>
<td>0.43</td>
<td>64.19</td>
</tr>
<tr>
<td>0.9</td>
<td>0.48</td>
<td>65.07</td>
</tr>
<tr>
<td>Rand</td>
<td>0.43</td>
<td>64.75</td>
</tr>
</tbody>
</table>

Table 4. Effect of Different Masking Ratios.

**Different Masking Ratios.** We investigate the impact of different masking ratios. We report the converged pre-training loss and the fine-tuning top-1 classification accuracy on the test set in Table 4. We also experiment with the random masking ratio in the last row. For each forward pass, we randomly select one masking ratio from 0.1 to 0.9 with step 0.1 to mimic flexible masked token length. The model with a random masking ratio does not outperform the best model that is pre-trained using a single ratio (i.e. 0.5). We observe that as the masking ratio increases, the pre-training loss mostly increases as the task becomes more challenging. However, a challenging self-supervised learning task does not necessarily lead to better performance. The model with a masking ratio of 0.7 and 0.9 have a high pre-train loss, while the fine-tune accuracy is not higher than the model with a 0.5 masking ratio. The conclusion is similar to the comparison of MVM and FFP training objectives, where a more challenging self-supervised learning task may not be optimal.

**Different Number of Mesh Sequences for Pre-Training.**

We test the effect of different numbers of mesh sequences used in pre-training. We report the fine-tuning top-1 classification accuracy in Figure 4. We observe that a large number of pre-training data can bring substantial performance improvement. The proposed Joint Shuffle method can greatly enlarge the dataset size without any manual cost, and has the potential to further improve model performance.

**Experimental Results on Noisy Body Pose Estimations.**

Body pose estimation has been a popular research field [25, 27, 32], but how to leverage the estimated 3D meshes for downstream perception tasks has not been fully explored. We apply the state-of-the-art body pose estimation model VIBE [27] on videos of NTU RGB+D dataset to obtain 3D mesh sequences. Skeleton and point cloud representations are derived from the estimated meshes to train the baseline models (see Sup. Mat.). We report the results in Table 5. We observe that STMT can outperform the best skeleton-based and point cloud-based action recognition model by 1.79% and 3.44% respectively. This shows that STMT with meshes as input, is more robust to input noise compared to other state-of-the-art methods with 3D skeletons or point clouds as input.

### Table 5. Experimental results on body poses estimated by VIBE [27] on NTU RGB+D dataset. The skeleton, point cloud, and mesh representations are derived from the same noisy body estimations.

<table>
<thead>
<tr>
<th>Method</th>
<th>Input</th>
<th>Top-1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2s-AGCN-FL [61]</td>
<td>3D Skeleton</td>
<td>58.67</td>
</tr>
<tr>
<td>2s-AGCN-CE [61]</td>
<td>3D Skeleton</td>
<td>57.49</td>
</tr>
<tr>
<td>CTR-GCN [7]</td>
<td>3D Skeleton</td>
<td>62.25</td>
</tr>
<tr>
<td>MS-G3D [37]</td>
<td>3D Skeleton</td>
<td>60.01</td>
</tr>
<tr>
<td>PSTNet [18]</td>
<td>Point Cloud</td>
<td>51.48</td>
</tr>
<tr>
<td>SequentialPointNet [30]</td>
<td>Point Cloud</td>
<td>60.60</td>
</tr>
<tr>
<td>P4Transformer [16]</td>
<td>Point Cloud</td>
<td>57.84</td>
</tr>
<tr>
<td>STMT(Ours)</td>
<td>Mesh</td>
<td><strong>64.04</strong></td>
</tr>
</tbody>
</table>

5. Conclusion

In this work, we propose a novel approach for MoCap-based action recognition. Unlike existing methods that rely on skeleton representation, our proposed model directly models the raw mesh sequences. Our method encodes both intrinsic and extrinsic features in vertex patches, and uses a hierarchical transformer to freely attend to any two vertex patches in the spatial and temporal domain. Moreover, two self-supervised learning tasks, namely Masked Vertex Modeling and Future Frame Prediction are proposed to enforce the model to learn global context. Our experiments show that STMT can outperform state-of-the-art skeleton-based and point-cloud-based models.

Acknowledgment

This work was supported by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-17-5-0003. This work used Bridges-2 GPU resources provided by PSC through allocation CIS220012 from the ACCESS program.
References


[23] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021. 3, 5


[30] Xing Li, Qian Huang, Zhiyian Wang, Zhenjie Hou, and Tianjin Yang. Sequentialpointnet: A strong frame-level parallel point cloud sequence network for 3d action recognition, 2021. 6, 7, 8


SFU. SFU Motion Capture Database. http://mocap.cs.sfu.ca/. 1

Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. NTU RGB+D: A large scale dataset for 3D human activity analysis. In CVPR, 2016. 2


Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matthew J. Kusner. Unsupervised point cloud pre-training via occlusion completion. In International Conference on Computer Vision, ICCV, 2021. 3, 5


