DETECTS: DEEP CLUSTERING OF TEMPORAL SKELETONS FOR GRAPH-BASED SEGMENTATION

Vipul Baghel *Bikash Kumar Badatya* *Ravi Hegde*
Indian Institute of Technology Gandhinagar, Gujarat, India.

1. METHODOLOGY

1.1. Overview

Recent advances in spatio-temporal graph learning and clustering provide conceptual foundations for our approach. The *Deep Temporal Graph Clustering (TGC)* framework introduces deep clustering tailored to dynamic interaction graphs by integrating temporal batch-processing with assignment distribution and reconstruction modules, highlighting the potential of clustering in temporal graph domains [1]. In the context of skeleton-based modeling, *DG-STGCN* proposes dynamic affinity learning and group-wise temporal convolution to model changing joint dependencies and action dynamics flexibly [2]. Similarly, the *Multi-Scale Spatial Temporal GCN (MST-GCN)* employs hierarchical spatial and temporal graph convolutions to capture long-range dependencies, enriching feature representations for skeleton-based action recognition [3].

These works collectively suggest that (i) temporal graph clustering is feasible within deep networks, (ii) dynamic graph structure enhances adaptability to complex motion, and (iii) multi-scale design supports modeling of both fine and coarse motion patterns. Motivated by these insights, we frame temporal action localization as a **spatial-temporal graph separation problem** via **deep clustering** of pose-graph embeddings, enabling label-free discovery of action boundaries in 3D skeleton sequences.

1.2. Input Representation

Given an input 3D pose sequence $\mathbf{X} \in \mathbb{R}^{T\times J\times 3}$, where T represents the total number of frames, J the number of joints, and 3 the spatial dimensions (x, y, z), the sequence is partitioned into N non-overlapping sub-pose sequences of size $(W, J, 3)$. Here, W is a fixed window size, and N is the total number of sub-sequences, given by:

$$N = \frac{T}{W} \tag{1}$$

Each sub-pose sequence $\mathbf{X}_i \in \mathbb{R}^{W\times J\times 3}$ is fed into the **ASTGCN** model to compute embeddings of each sub-pose.

1.3. Feature Extraction

The ASTGCN model processes each sub-pose sequence $\mathbf{X}_i \in \mathbb{R}^{W\times J\times 3}$ through a series of graph convolutions. The graph is constructed where nodes correspond to joints, and edges encode spatial and temporal relationships between adjacent joints and frames. The spatial-temporal graph convolution operation is defined as:

$$\mathbf{H}^{(l)} = \mathbf{A} \cdot \mathbf{H}^{(l-1)} \cdot \mathbf{W}^{(l)} \tag{2}$$

where $\mathbf{H}^{(l)}$ is the feature matrix at layer l, $\mathbf{A}$ is the adjacency matrix capturing joint relationships, and $\mathbf{W}^{(l)}$ is the learnable weight matrix at layer l. The ASTGCN outputs a feature embedding $\mathbf{H} \in \mathbb{R}^{W\times J\times D}$, where D is the embedding dimension.

1.4. Denoising - Sub-Pose Sequence Reconstruction

After obtaining the embeddings, we apply a **denoising stage** to refine the learned representations. This stage uses a fully connected layer to reconstruct the original input sequence from the embeddings. The reconstruction loss is computed as:

$$\mathcal{L}_{\text{rec}} = \|\hat{\mathbf{X}}_i - \mathbf{X}_i\|_2^2 \tag{3}$$

where $\hat{\mathbf{X}}_i$ is the reconstructed sub-pose sequence and $\mathbf{X}_i$ is the original sub-pose sequence.

1.5. Temporal Graph Clustering

In the clustering stage, we aim to separate the action segments by clustering the frame embeddings. To do this, we first apply **adaptive spatial [CLS] pooling** using a transformer encoder:

$$\mathbf{z}_i = \text{Transformer}_{\text{encoder}}(\mathbf{H}_i) \tag{4}$$

where $\mathbf{z}_i$ is the pooled embedding for the i-th sub-pose sequence. The embeddings $\mathbf{z}_i \in \mathbb{R}^{W\times D}$ are then clustered using **DBSCAN**, a density-based clustering method that groups together spatially dense points and separates sparse regions. DBSCAN requires two key parameters: the **minimum number of points** (minPts) in a cluster and the **minimum radius** (ϵ) to consider points as neighbors.

1.5.1. DBSCAN and MBC-B Algorithm

DBSCAN operates by iteratively exploring the neighborhood of each point in the embedding space. If a point has enough neighboring points within a given radius ϵ, it forms a core point and a cluster is assigned. Otherwise, the point is labeled as noise. The main challenge in applying DBSCAN is choosing the correct ϵ value. We use the *Minimum Ball Radius Covering B Points (MBC-B)* Algorithm [4] to calculate the optimal ϵ by examining the **k-distance** graph. The MBC-B algorithm computes ϵ as the distance to the k-th nearest neighbor for each point:

$$\epsilon = \text{MBC-B}(k) = \text{distance to the k-th nearest neighbor} \quad (5)$$

where k is typically set as $2D + 1$, ensuring that the radius is adaptively determined based on the data distribution.

1.6. Clustering Loss Functions

To guide the clustering process, we use two loss functions to improve the quality of the clusters: 1. **Silhouette Score Loss** ($\mathcal{L}_{\text{SC}}$) measures how similar a sample is to its own cluster compared to other clusters:

$$\mathcal{L}_{\text{SC}} = \frac{1}{N} \sum_{i=1}^{N} \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \quad (6)$$

where $a(i)$ is the average distance of the i-th point to all other points in the same cluster, and $b(i)$ is the minimum average distance to points in a different cluster.

2. **Sum of Squared Error Loss** ($\mathcal{L}_{\text{SSE}}$) minimizes the variance within clusters:

$$\mathcal{L}_{\text{SSE}} = \sum_{i=1}^{N} \|\mathbf{z}_i - \mathbf{c}_{y_i}\|_2^2 \quad (7)$$

where $\mathbf{c}_{y_i}$ is the centroid of the cluster to which the i-th point belongs, and y_i is the cluster assignment.

The total loss function is:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{rec}} + \lambda_1 \mathcal{L}_{\text{SC}} + \lambda_2 \mathcal{L}_{\text{SSE}} \quad (8)$$

where λ_1 and λ_2 are hyperparameters controlling the importance of each loss term.

1.7. End-to-End Training

The entire model is trained in an end-to-end manner using a self-supervised approach. No temporal labels or annotations are required during training. The model simultaneously learns to generate embeddings for the pose sequences, perform denoising, and perform clustering on the temporal embeddings to identify action boundaries.

1.8. Inference

During inference, a pose sequence $\mathbf{X} \in \mathbb{R}^{T \times J \times 3}$ is partitioned into N sub-pose sequences of size $(W, J, 3)$. Each sub-pose sequence is passed through the pre-trained ASTGCN model to obtain embeddings. The clustering stage (Stage 2) is applied to the complete sequence of embeddings, and DBSCAN is used to identify action segments. Temporal segmentation is obtained by finding cluster boundaries. The mean Average Precision (mAP) is then computed for evaluation of the temporal localization performance.

1.9. Theoretical Justification: Temporal Graph Segmentation via Deep Clustering

We now provide a mathematical and graph-theoretic interpretation of our self-supervised temporal segmentation framework. Our method can be understood as an approximate solution to a **temporal graph partitioning** problem in a learned latent space, where motion transitions correspond to weakly connected regions in a temporal graph formed over pose embeddings.

1.9.1. Pose Sequence as Spatio-Temporal Graph

Let a 3D skeleton sequence be denoted as $\mathcal{P} = \{P_t\}_{t=1}^{T}$, where $P_t \in \mathbb{R}^{J \times 3}$ represents the J-joint skeletal pose at time t. We divide $\mathcal{P}$ into N non-overlapping temporal blocks $\mathcal{S}_i \in \mathbb{R}^{W \times J \times 3}$ of window size $W \ll T$. Each sub-sequence $\mathcal{S}_i$ is treated as a **spatio-temporal graph** $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$, where: - Each node $v_{tj} \in \mathcal{V}_i$ corresponds to joint j at time t in block i. - Edges $\mathcal{E}_i$ are formed spatially (between joints) and temporally (across consecutive frames).

Using ASTGCN, we learn an embedding tensor $\mathbf{X}_i \in \mathbb{R}^{W \times J \times D}$, where each node embedding $\mathbf{x}_{tj} \in \mathbb{R}^D$ captures local spatial and temporal dynamics of joint j at time t in block i.

1.9.2. Spatial Pooling via Transformer

To aggregate joint-level features into frame-level embeddings, we apply a transformer encoder with a learned `[CLS]` token across the J joints for each time t. The attention mechanism adaptively weighs spatial contributions of joints and yields a temporally contextualized spatially pooled representation:

$$\mathbf{z}_t = \text{CLS-Pool}(\{\mathbf{x}_{tj}\}_{j=1}^{J}) \in \mathbb{R}^D.$$

Over all $t \in \{1, \ldots, T\}$, we obtain the embedding sequence $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_T\}$ representing the temporally smoothed dynamics.

1.9.3. Latent Temporal Graph Construction

We define a **latent temporal graph** $\mathcal{G}_T = (\mathcal{V}_T, \mathcal{E}_T)$ in the embedding space where: - Each node $v_t \in \mathcal{V}_T$ corresponds to

frame-level embedding $\mathbf{z}_t$. - Edge weights encode temporal similarity:

$$w_{tt'} = \exp\left(-\frac{\|\mathbf{z}_t - \mathbf{z}_{t'}\|^2}{\sigma^2}\right), \quad |t - t'| \leq k,$$

for some temporal neighborhood k and temperature σ. This yields a sparse, weighted graph encoding temporal continuity.

1.9.4. DBSCAN as Graph Cut Approximation

DBSCAN operates by identifying high-density regions in $\mathbb{R}^D$ and separating them by low-density gaps. Formally, it partitions $\mathcal{Z}$ into disjoint sets $\{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$ satisfying: - For each $\mathbf{z}_t \in \mathcal{C}_k$, there exist at least points in ϵ-neighborhood:

$$\mathcal{N}_\epsilon(\mathbf{z}_t) = \{\mathbf{z}_{t'} \mid \|\mathbf{z}_t - \mathbf{z}_{t'}\| \leq \epsilon\}.$$

- Clusters are maximal w.r.t. density-connectivity.

From a graph-theoretic view, this is equivalent to identifying connected components in a ϵ-radius **ϵ-graph** over $\mathcal{G}_T$, where:

$$(v_t, v_{t'}) \in \mathcal{E}_T^\epsilon \iff \|\mathbf{z}_t - \mathbf{z}_{t'}\| \leq \epsilon.$$

Thus, DBSCAN implicitly performs **graph cutting** by severing edges across low-similarity (sparse) regions—these cuts align with **motion transitions** where pose dynamics change.

1.9.5. Relation to Motion Boundaries

Let $\mathbf{z}_t$ and $\mathbf{z}_{t+1}$ belong to different DBSCAN clusters. Then:

$$\|\mathbf{z}_t - \mathbf{z}_{t+1}\| > \epsilon \quad \Rightarrow \quad w_{t,t+1} < \tau,$$

where $\tau = \exp(-\epsilon^2/\sigma^2)$ is the induced edge weight threshold. Such weak edges imply a high likelihood of discontinuity in the underlying motion manifold.

This aligns with perceptual motion boundaries where limb trajectories, joint angles, or activity type shifts abruptly. Since $\mathbf{z}_t$ encodes ASTGCN-derived kinematic and topological context, discontinuities in $\mathbf{z}_t$ reflect semantic action boundaries.

Our method offers a principled interpretation of unsupervised temporal segmentation as a *latent-space spatio-temporal graph cut* problem. This provides strong theoretical motivation for the observed empirical alignment between DBSCAN clusters and actual motion transitions.

2. REFERENCES

[1] Meng Liu, Yue Liu, Ke Liang, Wenxuan Tu, Siwei Wang, Sihang Zhou, and Xinwang Liu, "Deep temporal graph clustering," *arXiv preprint arXiv:2305.10738*, 2023.

[2] Haodong Duan, Jiaqi Wang, Kai Chen, and Dahua Lin, "Revisiting skeleton-based action recognition: Modeling intra-body relation with dynamic group-based spatial-temporal gcn," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 6500–6509.

[3] Zhan Chen, Sicheng Li, Bing Yang, Qinghan Li, and Hong Liu, "Multi-scale graph convolutional network for skeleton-based action recognition," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

[4] Weicheng Xu, Tianwei Xu, Leonid Sigal, and Alexander Mandt, "Contrastive learning under temporal segmentation," in *Advances in Neural Information Processing Systems*, 2023.