

# 3DInAction: Understanding Human Actions in 3D Point Clouds

Yizhak Ben-Shabat<sup>1,2</sup>

Oren Shrouf<sup>2</sup>

Stephen Gould<sup>1</sup>

<sup>1</sup>Australian National University

<sup>2</sup>Technion, Israel Institute of Technology

sitzikbs@technion.ac.il, shrouf.oren@campus.technion.ac.il, stephen.gould@anu.edu.au

<https://github.com/sitzikbs/3dincaction>

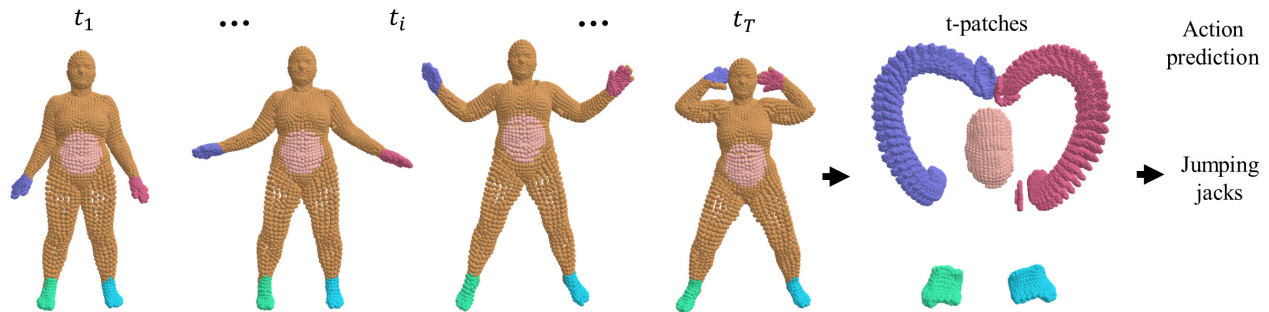


Figure 1. **t-patches for action recognition.** We propose a new representation for dynamic 3D point clouds. Termed *t-patches*, these are locally evolving point cloud sets aggregated over time. Learning features over *t-patches* provides an improved temporal point cloud representation for action understanding.

## Abstract

We propose a novel method for 3D point cloud action recognition. Understanding human actions in RGB videos has been widely studied in recent years, however, its 3D point cloud counterpart remains under-explored despite the clear value that 3D information may bring. This is mostly due to the inherent limitation of the point cloud data modality—lack of structure, permutation invariance, and varying number of points—which makes it difficult to learn a spatio-temporal representation. To address this limitation, we propose the 3DInAction pipeline that first estimates patches moving in time (*t-patches*) as a key building block, alongside a hierarchical architecture that learns an informative spatio-temporal representation. We show that our method achieves improved performance on existing datasets, including DFAUST and IKEA ASM. Code is publicly available at <https://github.com/sitzikbs/3dincaction>.

## 1. Introduction

In this paper, we address the task of action recognition from 3D point cloud sequences. We propose a novel pipeline wherein points are grouped into temporally evol-

ving patches that capture discriminative action dynamics. Our work is motivated by the massive growth of online media, mobile and surveillance cameras that have enabled the computer vision community to develop many data-driven action-recognition methods [5, 12, 26, 31], most of which rely on RGB video data. Recently, commodity 3D sensors are gaining increased momentum, however, the 3D point cloud modality for action recognition has yet been under-exploited due to the scarcity of 3D action-labeled data.

In many cases, a pure RGB video-based inference may not be enough and incorporating other modalities like geometry is required. This is especially necessary for safety critical applications such as autonomous systems, where redundancy is crucial, or in scenarios where the video is heavily degraded (e.g., due to poor lighting). Some approaches incorporate geometrical information implicitly, e.g., through intermediate pose estimation [7]. This often entails extra steps that require more time and resources and is still limited to video input. Therefore a more explicit approach is desirable.

3D sensors provide an alternative modality in the form of point clouds sampled on the environment. Despite the vast research on 3D vision and learning, even static 3D point cloud datasets are significantly smaller than their RGB im-

age counterparts due to difficulties in collecting and labeling. 3D point cloud sequence databases are even smaller,

making it more difficult to learn a meaningful 3D action representation. Furthermore, learning a point cloud representation still remains an active research field because point clouds are unstructured, unordered, and may contain a varying number of points. Learning a temporal point cloud representation is even more challenging since, unlike pixels, there is no one-to-one point correspondence through time.

We address these challenges and propose the 3DinAction pipeline for 3D point cloud action recognition. In our pipeline, we first extract local temporal point patches (t-patches) that reflect a point region’s motion in time, see Figure 1. We then learn a t-patch representation using a novel hierarchical architecture that incorporates spatial features in the temporal domain. We finally get an action prediction for each frame in a sequence by aggregating multiple t-patch representations. This pipeline overcomes the need for ground truth point temporal correspondence, grid structure, point order, and a fixed number of points in each frame. Intuitively, patches reflect local surface deformation and are more robust to point correspondence errors.

We conduct extended experiments to evaluate the performance of our approach compared to existing SoTA methods and show that 3DinAction provides significant performance gains of 13% and 7% in accuracy on DFAUST and IKEA ASM, respectively.

The key contributions of our work are as follows:

- A novel representation for dynamically evolving local point cloud sets termed t-patches.
- A hierarchical architecture that produces an informative spatio-temporal representation for sequences of point clouds.

## 2. Related Work

**Learning 3D point cloud representations.** Point clouds pose a challenge for neural networks due to their unstructured and point-wise unordered nature. To address these challenges, several approaches have been proposed. PointNet [23, 24] uses permutation-invariant operators, such as pointwise MLPs and pooling layers, to aggregate features across a point set. Some approaches construct a graph from the point set. DGCNN [34] applies message passing and performs graph convolutions on kNN graphs, KC-Net [29] uses kernel correlation and graph pooling, and Kd-Networks [15] apply multiplicative transformations and share the parameters based on the subdivisions imposed by kd-trees. Alternatively, the structure can be imposed using a grid of voxels [22, 36], or a grid of Gaussians in 3Dm-FVNet [1]. Another alternative avoids the structure by using Transformer’s attention mechanism [17, 37]. For a comprehensive survey of point cloud architectures please see [14].

Recently, various factors that can impact the training of

different architectures have been investigated [13, 25]. This includes exploring data augmentation strategies and loss functions that are not specific to a particular architecture. The results of this study showed that older PointNet-based architectures [23, 24] can perform comparably to newer architectures with minor changes.

All of the above methods deal with static, single-frame, or single-shape point clouds. In this work, the input is a temporal point cloud where a representation for a short sequence is required and point correspondence between frames is unknown. Therefore extending existing approaches is not trivial.

### **Learning temporal 3D point cloud representations.**

Temporal point clouds have not been as extensively studied as their static counterparts, in particular for action recognition. MeteorNet [21] processes 4D points using a PointNet++ architecture where they appended a temporal dimension to the spatial coordinates. PSTNet [10, 11] proposed spatio-temporal convolutions and utilized some of the temporal consistency for action recognition. Similarly, P4Transformer [8] uses 4D convolutions and a transformer for capturing appearance and motion via self-attention. In a follow-up work PST-Transformer [9] employs a video level of self-attention in search for similar points across entire videos and so encodes spatio-temporal structure. Some works attempt to alleviate the full supervision requirement for 3D action recognition. These include self-supervised features learning [32] by predicting temporal order from a large unlabeled dataset and fine-tuning on a smaller annotated datasets and unsupervised skeleton colorization [35]. Additional supervised approaches include MinkowskiNet [6] that uses a 4D spatio-temporal CNN after converting the point clouds to an occupancy grid, 3DV [33] that encodes 3D motion information from depth videos into a compact voxel set, and Kinet [38] that implicitly encoded feature level dynamics in feature space by unrolling the normal solver of ST-surfaces.

The above methods, perform a single classification per clip. In this paper, we focus on a related, and more challenging, task that requires a prediction per-frame. We propose to convert the point cloud representation into t-patches and use an MLP based hierarchical architecture to get the spatio-temporal representation.

**3D action understanding datasets.** One of the major driving forces behind the success of learning-based approaches is the availability of annotated data. For the task of 3D point cloud action recognition, there is currently no designated standard dataset, however, some existing datasets may be extended. The CAD 60 and CAD 120 [16, 30] datasets include 60 and 120 long-term activity videos of 12 and 10 classes respectively (e.g., making cereal, microwave food). These datasets provide raw RGB, skeletons, and depth data however its small scale and long-term focus limit its effec-

tiveness. The NTU RGB+D 60 [28] and NTU RGB+D 120 [20] provide  $\sim 56\text{K}$  and  $\sim 114\text{K}$  clips containing 60 and 120 actions classes respectively, e.g., taking off a jacket, taking a selfie. They provide three different simultaneous RGB views, IR and depth streams as well as 3D skeletons. While these datasets can be considered large-scale, their contrived nature makes recent skeleton-based methods (e.g., [7]) perform well, making a prior-free approach difficult to justify. The MSR-Action3D dataset [19] includes 20 action classes performed by 10 subjects for a total of 567 depth map sequences, collected using a Kinect v1 device (23K frames). The sequences in this dataset are very short and therefore using it to evaluate learning-based approaches provides a limited indication of generalization. The above datasets provide per clip action annotations.

Some datasets inherently provide per-frame annotations. The IKEA ASM dataset [2] provides 371 videos clipped into 31K clips. It contains 33 action classes related to furniture assembly, annotated per frame. This dataset provides several modalities including three RGB views, and Depth. It is an extremely challenging dataset since the human assembler is often occluded and presents very unique assembly poses. It is also very imbalanced since different assembly actions have different duration and may repeat multiple times within the same assembly. Although it was designed for video action recognition, its challenges are the core reasons for choosing to extend it to the point cloud action recognition task. The DFAUST dataset [3] provides high-resolution 4D scans of human subjects in motion. It includes 14 action categories with over 100 dynamic scans of 10 subjects (1:1 male-to-female ratio) with varying body shapes represented as registrations of aligned meshes, therefore an extension to our task is straightforward. One particularly important feature of this dataset is the GT point correspondences throughout the sequence *i.e.* it is possible to follow each point’s movement through time. While this dataset is not as large-scale as others, it provides ground truth information (correspondence) that most other collected datasets do not. Therefore, we extend this dataset to 3D point cloud action recognition and use it as a testbed for many ablation studies (see Section 4.4).

### 3. 3DinAction pipeline

Our 3DinAction pipeline is illustrated in Figure 2. Given a temporal sequence of 3D point clouds we first extract a set of t-patches (Section 3.1). We then feed the t-patches into a hierarchical neural network (Section 3.2) to produce a per-frame high dimensional feature vector representation. Finally, the feature vectors are fed into a classifier to obtain per-frame predictions. The proposed approach is prior-free (no skeleton extraction required) and therefore general and can be used on different action-understanding datasets.

### 3.1. t-patches

Let  $S = \{x_j \in \mathbb{R}^3 \mid j = 1, \dots, N\}$  denote a 3D point cloud with  $N$  points. In the classic (static) setting, a patch  $\Psi_q$  is extracted around some query point  $x_q$ . For example, the patch  $\Psi_q$  may be constructed by finding the  $k$ -nearest neighbors of  $x_q$  in  $S$ .

In our temporal setting we are given a sequence of point clouds  $\mathcal{S} = \{S^0, \dots, S^T\}$  composed of point cloud frames  $S^t = \{x_j^t \mid j = 1, \dots, N^t\}$ . Here the superscript  $t$  is used to denote the index of the point cloud in the sequence. Instead of extracting a patch within a single frame, we allow patches to extend temporally, and denote them as *t-patches*.

**Definition 3.1** A t-patch  $P_q$  is a sequence of point sets indexed by a query point  $x_q^0$  and jointly moving in time defined by a pointwise mapping function between patches in consecutive frames. Mathematically,  $P_q = \langle \Psi_q^t \rangle_{t=0}^T$  where  $\Psi_q^0$  is the initial (static) patch and  $\Psi_q^t = \Phi(\Psi_q^{t-1})$  is the patch at time  $t$  where  $\Phi$  is a pointwise mapping function.

In practice, it is difficult to find a reliable mapping function  $\Phi$ . Therefore we propose a simplified formulation that, for a given query point  $x_q^0$ , first extracts a patch for the first frame  $\Psi_q^0$  and then iteratively extracts corresponding patches for the next frames (iterating over time), by using the closest point in the next frame as the new query point. More formally, we specify  $\vec{\Psi}_q^0 \triangleq \Psi_q^0$ ,  $\vec{\Psi}_q^t = knn(x_q^{t-1}, S^t)$  and  $x_q^t = nn(x_q^{t-1}, S^t)$  for  $t = 1, \dots, T$ . Here  $knn$  is the  $k$  nearest neighbor and  $nn$  is nearest neighbor. Then, the simplified t-patch formulation is given by

$$\vec{P}_q = \langle \vec{\Psi}_q^t \mid t = 0, \dots, T \rangle \quad (1)$$

See Figure 3 left for an illustration of the t-patch extraction process. Note that if ground truth correspondence is available  $knn$  can be swapped back to  $\Phi$ . However, this does not guarantee improved performance.

**Temporal t-patch collapse.** The simplified formulation of extracting t-patches inherently suffers from the problem of two or more t-patches collapsing into having the same points after a certain frame. We call this scenario t-patch *temporal collapse*. Temporal collapse can happen whenever  $x_q^t = x_p^t$  for  $x_q^0 \neq x_p^0$ . The main issue with temporal collapse is the reduction in point coverage as time progresses, *i.e.* the patches covering the last point cloud have significant overlaps and therefore include fewer points than the first frame and so missing vital data. An illustration of the t-patch collapse problem is available in Figure 3 (right). To mitigate this issue, we propose two solutions. First, adding small noise to each iteration’s query points, *i.e.*  $\vec{\Psi}_q^t = knn(x_q^t + \epsilon, S^{t+1})$  where  $\epsilon \sim \mathcal{N}(\mu, \sigma^2)$  is a small Gaussian noise. Second, we propose to construct t-patches from the first to last frame but also in reverse, initializing

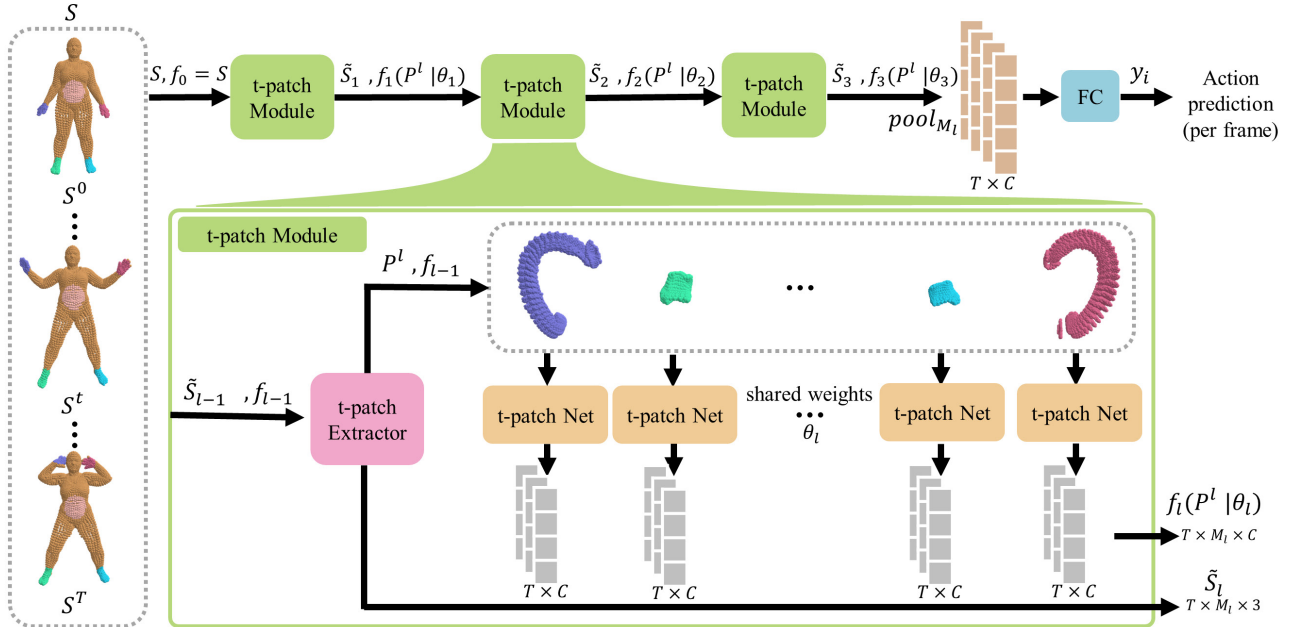


Figure 2. **3DinAction pipeline.** Given a sequence of point clouds, a set of t-patches is extracted. The t-patches are fed into a neural network to output an embedding vector. This is done hierarchically until finally the global t-patch vectors are pooled to get a per-frame point cloud embedding which is then fed into a classifier to output an action prediction per frame.

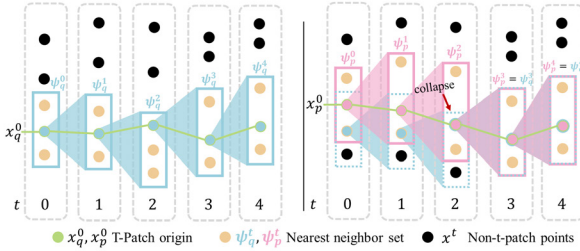


Figure 3. **t-patch construction and collapse.** Illustration of t-patch construction (left) and collapse (right). Starting from an origin point  $x_q^0$  we find the nearest neighbours in the next frame iteratively to construct the t-patch subset (non-black points). A collapse happens when two different origin points,  $x_q^0$  and  $x_p^0$ , have the same nearest neighbour at some time step,  $\Psi_p^3 = \Psi_q^3$  here.

with  $\Psi_q^0$  and  $\Psi_q^T$ , respectively. We name this variation *bidirectional t-patches*. More formally bidirectional t-patches are given by,

$$\overleftrightarrow{P} = \left( \bigcup_q \overrightarrow{P}_q \right) \cup \left( \bigcup_p \overleftarrow{P}_p \right) \quad (2)$$

where  $\overleftarrow{P}_p$  is defined similarly to  $\overrightarrow{P}_q$  but in the reverse direction, i.e.,  $\overleftarrow{\Psi}_p^T \triangleq \Psi_p^T$  and  $\overleftarrow{\Psi}_p^t = knn(x_p^{t+1}, S^t)$  for

$t = T - 1, \dots, 0$ . Here, the final set of t-patches is composed of an equal number of t-patches from both directions.

### 3.2. Hierarchical architecture

The proposed architecture is composed of  $l$  consecutive t-patch modules. Each module receives a point cloud sequence  $S$  as input. The sequence is fed into a t-patch extractor where it undergoes subsampling and t-patch extraction, forming  $\tilde{S}_l$  and  $P^l$  respectively. Then, the t-patches are fed into t-patch Net, a network that computes a high-dimensional feature vector  $f_l$  for each t-patch, parametrized by  $\theta_l$ . The subsampled sequence  $\tilde{S}_l$  and its corresponding t-patch features  $f_l$  are then fed into the next t-patch module. These modules form a hierarchy in the sense that each module receives as input a sparser point cloud with a higher dimensional feature vector representing each point (encoding both spatial and temporal information). Note that both the t-patch points and their features are fed into t-patch Net.

**t-patch extractor.** We first subsample the first frame in the point cloud sequence  $S^0$  using farthest point sampling (FPS) to form a set of  $M$  query points  $\tilde{S}^0 = \{x_j^0 \in FPS(S^0, M)\}$ . The set  $\tilde{S}^0$  is used to form the t-patches. Subsampling is required since computing a t-patch for each point is inefficient and unnecessary due to overlaps. After subsampling, we extract  $M$  t-patches using Equation 2 where  $q \in \tilde{S}^0$ . The extractor operates on both 3D points and their corresponding features (for mod-

ules deeper in the hierarchy).

**Model architecture and t-patch net.** The t-patch network computes a high dimensional representation for each t-patch. The t-patch Net architecture is composed of several MLP layers operating on the non-temporal dimensions (sharing weights across points) followed by a convolutional layer operating on both the temporal and feature dimensions. Note that the network weights are also shared across t-patches. The output of each t-patch Net is a vector for each frame. The final frame representation is obtained by aggregating all of the t-patch features using a max pooling operation *i.e.*  $\text{maxpool}_{M_t}(f_3)$ . This representation is then fed into a classifier consisting of three fully connected layers with temporal smoothing and softmax to output the final action prediction. To train the network we use the same losses of RGB based approaches [2, 5] which include a per-frame prediction cross entropy loss and a per-sequence prediction cross entropy loss (summed and weighted evenly)  $L_{total} = L_{frame} + L_{seq}$ . For full details see supplemental.

## 4. Experiments

We evaluate the performance of our approach on three datasets. The results show that the 3DinAction pipeline outperforms all baselines in DFAUST [3] and IKEA ASM [2] and is comparable in MSR-Action 3D [19]. We then conduct an ablation study for selecting parameters and t-patch extraction method showing that adding jitter and bidirectional t-patches is beneficial. Finally, we report time performance and show the tradeoff between performance and inference time. For more results and experiments, see supplemental material.

**Baselines and evaluation metrics.** For evaluation, we report several standard metrics [4]: the top1 and top3 frame-wise accuracy are the de facto standard for action classification. We compute it by summing the number of correctly classified frames and dividing by the total number of frames in each video and then averaging over all videos in the test set. Additionally, since some of the datasets are imbalanced and may contain different actions for each frame in a clip, we also report the macro-recall by separately computing recall for each category and then averaging (macro). Finally, we report the mean average precision (mAP) since all untrimmed videos contain multiple action labels.

For DFAUST and IKEA ASM we report static methods PointNet [23], PointNet<sup>++</sup> [24], and Set Transformer [18] by applying them on each point cloud frame individually. Additionally, we report temporal methods like PSTNet [10] and also implemented a temporal smoothing version of each static method (PointNet+TS, Pointnet<sup>++</sup>+TS, and Set Transformer+TS respectively) by learning the weights of a convolutional layer over the temporal dimension. Temporal smoothing aims to provide a naive baseline for utilizing temporal information in addition to spatial information.

Note that in all experiments, unless otherwise specified, our method uses the simplified formulation with jitter and bidirectional t-patches.

### 4.1. Experiments on DFAUST dataset

We extend the DFAUST dataset for the task of action recognition and show that the proposed approach outperforms other methods (see Table 1).

**DFAUST dataset [3].** We extended the DFAUST dataset to our task by subdividing it into clips of 64 frames with train and test human subjects. The split was constructed so no subject will appear in both training and test set as well as guarantee that all actions appear in both. The train and test sets contain 76 full-length sequences (395 clips, and ~25K frames) and 53 sequences (313 clips, and ~20K frames) respectively. Each point cloud frame contains 6890 points. These points are mesh vertices and therefore the density varies greatly (e.g., very dense on the face, hands, and feet and sparser on the legs). For all baselines, we sampled a set of 1024 points using the farthest point sampling algorithm to provide a more uniform set of points. For this dataset, all frames in a clip have the same label. Note that not all actions are performed by all subjects. For the full action list and dataset statistics, see the supplemental.

**Results.** The results, reported in Table 1, show that our proposed approach outperforms all baselines by a large margin. It also shows that temporal smoothing boosts performance significantly for all static baselines. Additionally, to explore the influence of our simplified *knn*-based temporal point mapping, we used the GT point correspondence to match the consecutive t-patch origin points and report the results as another baseline (Ours + GT corr). The results show that there is a mAP performance gain with GT correspondence, however, it is limited. Note that in most datasets, this GT correspondence is not available. Finally, we also experimented with a Transformer architecture to process the t-patch learned representations and show that it does not provide additional performance boost. This may be attributed to the dataset size.

**Insight.** We extended the GradCam [27] approach for our 3DinAction pipeline. Using this approach we get a score per point in each t-patch proportional to its influence on classifying the frame to a given target class. The results in Figure 4 show that, as expected, our approach learns meaningful representations since the most prominent regions are the ones with the informative motion. For example, in the Jumping jacks action (top row) the hands are most prominent as they are making a large and distinct motion.

### 4.2. Experiments on IKEA ASM dataset

**IKEA ASM dataset [2].** This dataset consists of 371 videos (3M frames) of people assembling IKEA furniture in different indoor environments. It was collected using a

Method	Frame acc.		
	top 1	top 3	mAP
3DmFVNet [1]	60.86	87.68	0.7171
PointNet [23]	65.67	86.44	0.7161
PointNet <sup>++</sup> [24]	58.51	88.28	0.5842
Set Transformer [18]	52.27	81.98	0.6209
PoinNet [23] + TS	74.10	94.00	0.7863
PointNet <sup>++</sup> [24] + TS	67.88	86.21	0.7563
Set Transformer [18] + TS	62.95	90.33	0.7322
PSTNet [10]	50.70	78.28	0.6490
Ours + GT corr	<u>77.67</u>	<u>95.38</u>	<b>0.8762</b>
Ours + Transformer	77.09	93.7	77.49
Ours	<b>87.26</b>	<b>99.26</b>	<u>0.8616</u>

Table 1. **Action recognition results on DFAUST.** Reporting frame-wise accuracy and mean average precision. Ours outperforms all baselines by a large margin.

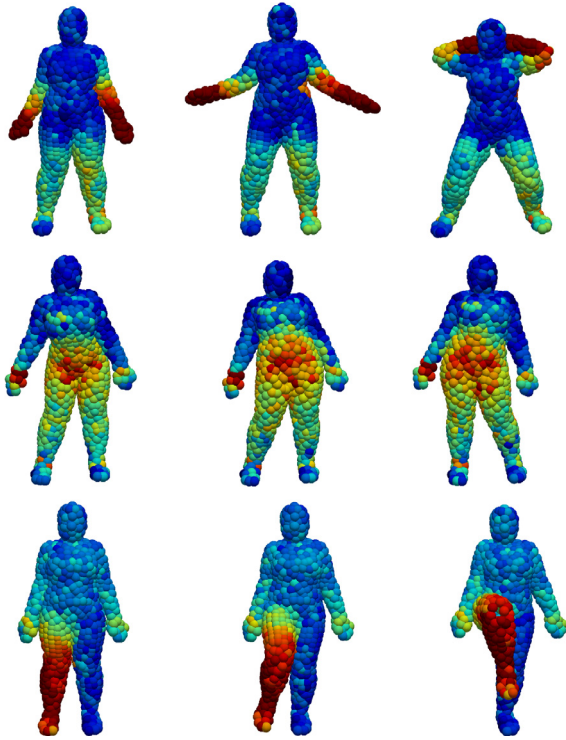


Figure 4. **3DinAction GradCAM scores.** The proposed 3DinAction pipeline learns meaningful representations for prominent regions. The presented actions are jumping jacks (top row), hips (middle row), and knees (bottom row). The columns represent progressing time steps from left to right. Colormap indicates high GradCAM scores in red and low scores in blue.

Kinect V2 camera and provides camera parameters to reconstruct point clouds in camera coordinates. It provides action annotation for each frame (33 classes). It is a highly challenging dataset for two main reasons: (1) It is highly

imbalanced since some actions have a long duration and occur multiple times in each video (e.g., spin leg) and some are shorter and sparser (flip tabletop). (2) The assembly motion includes a lot of self-occlusion as well as subtle movements. The train/test split consists of 254 and 117 full sequences respectively. The split is environment-based (*i.e.* in the test set there is no environment that appeared in the training set). The assembly videos have an average of  $\sim 2735$  frames per video. The point clouds provided in this dataset are aligned to the camera coordinate frame, posing a challenge for methods that are sensitive to rotations since the camera moves between different scans.

**Results.** The results on the IKEA ASM dataset are reported in Table 2. The results show that the proposed 3DinAction pipeline provides a significant performance boost over static approaches and their temporally smooth variants. Additionally, as expected, PointNet and Set Transformer are heavily affected by the variations in coordinate frames. PointNet<sup>++</sup> on the other hand performs better since it uses local coordinate frames for each local region. All methods show an improved mAP when using the temporally smooth variant with degradation in frame-wise accuracy due to the dataset imbalance. For this dataset, the top1 metric is not always indicative of the quality of performance because a high top1 is directly correlated with many frames classified as the most common class. Additionally, we compare to pose-based methods reported in [2] and show that the proposed approach also outperforms these baselines. See supplementary material for confusion matrices.

**t-patch intuition and visualization.** In Figure 5 we visualize the t-patches for the flip table action in the TV Bench assembly. A set of selected t-patches are highlighted in color demonstrating different types of t-patches and their spatio-temporal changes. The **blue** is on the moving TV Bench assembly, it moves rigidly with the assembly. The **maroon** is on the moving person’s arm, it exhibits nonrigid motion and deformations through time. The **teal** is on the static table surface containing some of the TV Bench’s points in the first frame but remains static when it moves since its origin query point is on the table. The **green** is on the static carpet, remaining approximately the same through time. Note that the RGB images are for visualization purposes and are not used in our pipeline.

### 4.3. Experiments on MSR-Action3D dataset

For this dataset, the task is to predict a single class for a sequence of frames (unlike the other datasets where a per-frame prediction is required). To that end, we replace our classifier with a single fully connected layer and max pooled the results over the temporal domain (similar to [10]). The results, reported in Table 3, show that all SoTA methods, including the proposed approach, exhibit very similar performance. This is mainly attributed to the small scale of the

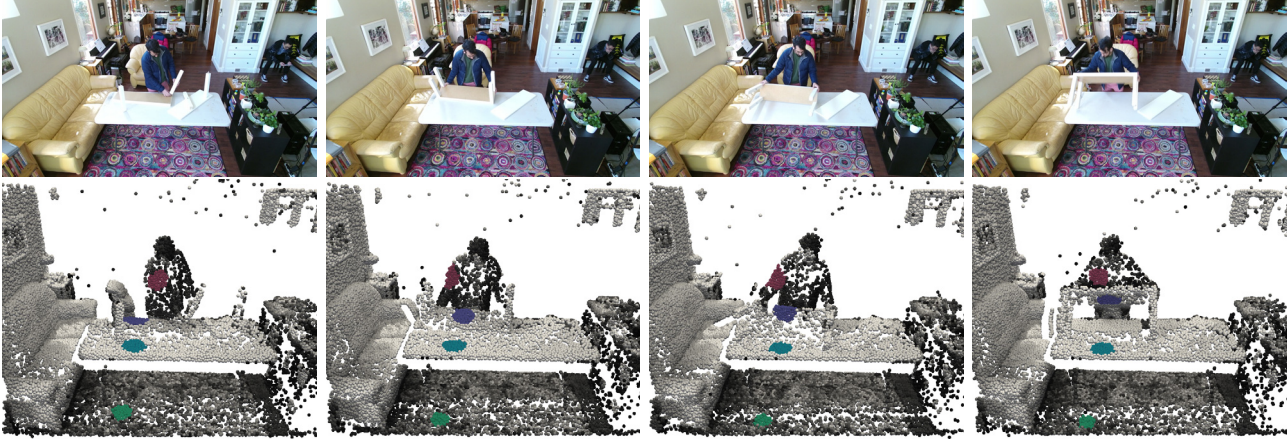


Figure 5. **IKEA ASM example with t-patches.** The flip table action for the TV Bench assembly is visualization including the RGB image (top), and a grayscale 3D point cloud with t-patches (bottom). t-patches are highlighted in color. The **blue** is on the moving TV Bench assembly, **maroon** is on the moving person's arm, **teal** is on the static table surface, and **green** is on the colorful static carpet.

Method	Frame acc.			
	top 1	top 3	macro	mAP
PointNet [23]	4.20	19.86	5.76	0.0346
PointNet <sup>++</sup> [24]	45.97	70.10	29.48	0.1187
Set Transformer [18]	14.96	57.12	13.16	0.0299
PoinNet [23] + TS	6.00	19.48	5.14	0.0804
PointNet <sup>++</sup> [24] +TS	27.84	60.64	27.72	0.2024
Set Transformer [18] + TS	9.54	36.50	10.74	0.1471
PSTNet [10]	17.94	52.24	17.14	0.2016
Human Pose HCN [2]	39.15	65.37	28.18	0.2232
Human Pose ST-GCN [2]	43.4	66.29	26.54	0.1856
Ours without BD	45.16	72.83	35.06	<b>0.2932</b>
Ours	<b>52.91</b>	<b>75.03</b>	<b>38.84</b>	0.2875

Table 2. **Action classification on IKEA ASM.** The proposed approach provides a significant performance boost over other static and dynamic approaches, including the temporal smoothing (TS).

dataset and the lack of diversity in the action classes. Furthermore, we witnessed that the main performance gap is for frames and sequences where the action is indistinguishable (e.g., first few frames of a sequence where no distinguishable action commenced).

#### 4.4. Ablation study

**t-patch extraction.** We studied the t-patch extraction method and its effects on action recognition on a noisy version of the DFAUST dataset. The results reported in Table 4, show the significance of the t-patch collapse problem and the effectiveness of adding small jitter and bidirectional t-patches to overcome it. In the DFAUST dataset, finding the nearest neighbor between frames provides a  $\sim 96\%$  correspondence accuracy (small motion between frames).

Method	# frames				
	4	8	12	16	24
PSTNet [10]	81.14	83.50	87.88	89.90	91.20
P4Transformer [8]	80.13	83.17	87.54	89.56	90.94
PST-Transformer [9]	81.14	83.97	88.15	91.98	93.73
Kinet [38]	79.80	83.84	88.53	91.92	93.27
Ours	80.47	86.20	88.22	90.57	92.23

Table 3. **MSR-Action3D classification results.** Reporting classification accuracy for clips of different lengths. Results show that all methods are comparable since this dataset's scale is limited.

Therefore, in this experiment, we augment the dataset once by adding small Gaussian noise to each point in the dataset ( $\sigma = 0.01$ ), decreasing the correspondence accuracy to  $\sim 62.4\%$  and introducing multiple t-patch collapse instances as well as increasing the classification difficulty.

Several variants of the t-patch extraction were explored. The first variation (GT) incorporates the ground truth correspondence into the t-patch extraction. Using this method, there is no t-patch collapse since there is a one-to-one mapping between frames. We expected this to produce an upper bound on the performance, however, surprisingly the results show that this variation is actually inferior to the proposed t-patch approach. We attribute this to the proposed t-patch extraction inherent augmentation caused by the downsampling and nearest neighbor point jitter. We then continue to explore the proposed approaches for dealing with t-patch collapse which include jitter, *i.e.* adding small noise to each point before finding its nearest neighbor in the next frame, and the bidirectional t-patches that extract patches both from the first to the last frame and from the last to the first frame. The results show that adding jitter is al-

Data	GT	Jitter	BD	Frame acc.		
				top 1	top 3	mAP
clean	✓	✗	✗	77.67	95.38	0.8762
	✗	✗	✗	74.73	92.14	0.8097
	✗	✓	✗	80.49	96.61	<b>0.9023</b>
	✗	✓	✓	<b>87.26</b>	<b>99.26</b>	0.8616
noisy	✓	✗	✗	76.08	95.50	0.9013
	✗	✗	✗	66.74	93.76	0.7626
	✗	✓	✗	<b>81.83</b>	<b>98.97</b>	<b>0.9220</b>
	✗	✓	✓	<u>80.03</u>	<u>97.57</u>	0.8975

Table 4. **t-patch collapse ablation** on DFAUST. Exploring adding (1) GT - ground truth correspondences, (2) jitter - small Gaussian noise in t-patch construction, and (3) BD - bidirectional t-patches.

n	k	Frame acc.		
		top 1	top 3	mAP
256	16	76.96	97.54	0.8430
512	16	<b>80.03</b>	<u>97.57</u>	<b>0.8975</b>
1024	16	77.30	<b>97.88</b>	<u>0.8507</u>
512	8	76.87	96.21	0.7557
512	32	<u>77.91</u>	96.60	0.7453

Table 5. **t-patch parameters ablation**. Results for the number of neighboring points in a patch  $k$  and number of downsampled points  $n$  show that the method is robust.

ways beneficial and provides a boost in performance. The bidirectional t-patches improve accuracy performance significantly when the data is clean and are comparable when the data is noisy. Note that in both dataset variations, the degradation due to temporal t-patch collapse is low compared to Kinect-based scan data, therefore the bidirectional benefits are not fully utilized.

**t-patch parameters.** The core parameters for t-patch extraction are the number of neighbors to extract ( $k$ ) and the number of points to subsample ( $n$ ). Here there is a trade-off between complexity and performance *i.e.* when  $k$  and  $n$  are small, the input to the model is small accordingly but the overall coverage is reduced and therefore performance is lower. We explored their influence on the noisy DFAUST dataset and report the results in Table 5. The results show that the method is fairly robust to the selection of these parameters, producing comparable results for all. The best performance was obtained for  $n = 512, k = 16$ . Surprisingly, the performance slightly degrades when increasing  $k$  and  $n$  beyond these values. This is likely due to the increase in model size, which easily overfits on a dataset of this size.

**Time and parameters.** We report the time performance and the number of parameters of several baselines in Table 6. The results show the tradeoff between performance and time, *i.e.* the temporal approaches exhibit longer pro-

Method	Time [ms]	# parameters
PointNet [23]	64.49	3.5M
PointNet <sup>++</sup> [24]	23.35	1.5M
PSTNet [10]	185.92	8.3M
Ours t-patch extraction	180.65	0
Ours feature computation	12.50	9.8M
Ours classifier	0.36	1.1M
Ours	193.51	10.9M

Table 6. **Time and parameters.** Temporal methods have more parameters and take longer. 3DinAction time is mostly used to extract t-patches.

cessing times and more parameters while performing better. For the proposed approach, we break down the timing of individual components, namely the t-patch extraction, feature computation, and classifier. The results show that the proposed approach is comparable to PSTNet in time while having more parameters. Interestingly, most of the time is used for extracting the t-patches and not for feature extraction or classification. This is attributed to the farthest point sampling and the sequential  $knn$  search, both of which could be further optimized for speed. Note that results are average of 50 runs, each with a batch of 4 and 1024 points per frame.

**Limitations.** Since the simplified formulation of t-patch construction uses  $knn$ , it is sensitive to variations in point densities. A t-patch in a sparse region will occupy a larger volume than a t-patch in a dense region. We use FPS to mitigate this, however, other approaches can be used *e.g.*, using neighbors in a fixed radius. Another limitation is data with a very low frame rate or very fast motion since this breaks the assumption that points in consecutive frames are close to each other, and will cause inconsistent t-patch motion.

## 5. Conclusions

We introduced the 3DinAction pipeline, a novel method for 3D point cloud action recognition. It showed that the creation of temporal patches is beneficial for finding informative spatio-temporal point representations. 3DinAction has demonstrated a performance boost over SoTA methods.

This work opens many interesting future directions of research. These include trying to learn the t-patch construction instead of the  $knn$  selection, imposing stronger temporal structure based on preexisting knowledge and bias (*e.g.*, scene flow or tracking), and exploring using multi-modal inputs with this representation (*e.g.*, RGB or text).

**Acknowledgement.** This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 893465. We also thank the Microsoft for Azure Credits and NVIDIA Academic Hardware Grant Program for providing high-speed A5000 GPU.



## References

- [1] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. 3DMFV: Three-dimensional point cloud classification in real-time using convolutional neural networks. *RAL*, 3:3145–3152, 2018. 2, 6
- [2] Yizhak Ben-Shabat, Xin Yu, Fatemeh Saleh, Dylan Campbell, Cristian Rodriguez-Opazo, Hongdong Li, and Stephen Gould. The ikea asm dataset: Understanding people assembling furniture through actions, objects and pose. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 847–859, 2021. 3, 5, 6, 7
- [3] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: Registering human bodies in motion. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 3, 5
- [4] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Nibbles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–970, 2015. 5
- [5] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 1, 5
- [6] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3075–3084, 2019. 2
- [7] Haodong Duan, Yue Zhao, Kai Chen, Dahua Lin, and Bo Dai. Revisiting skeleton-based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2969–2978, 2022. 1, 3
- [8] Hehe Fan, Yi Yang, and Mohan Kankanhalli. Point 4d transformer networks for spatio-temporal modeling in point cloud videos. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14204–14213, 2021. 2, 7
- [9] Hehe Fan, Yi Yang, and Mohan Kankanhalli. Point spatio-temporal transformer networks for point cloud video modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):2181–2192, 2022. 2, 7
- [10] Hehe Fan, Xin Yu, Yuhang Ding, Yi Yang, and Mohan Kankanhalli. Pstnet: Point spatio-temporal convolution on point cloud sequences. In *International Conference on Learning Representations*, 2021. 2, 5, 6, 7, 8
- [11] Hehe Fan, Xin Yu, Yi Yang, and Mohan Kankanhalli. Deep hierarchical representation of point cloud videos via spatio-temporal decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):9918–9930, 2021. 2
- [12] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211, 2019. 1
- [13] Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting point cloud shape classification with a simple and effective baseline. In *International Conference on Machine Learning*, pages 3809–3820. PMLR, 2021. 2
- [14] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *PAMI*, 2020. 2
- [15] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE international conference on computer vision*, pages 863–872, 2017. 2
- [16] Hema Swetha Koppula, Rudhir Gupta, and Ashutosh Saxena. Learning human activities and object affordances from rgb-d videos. *The International Journal of Robotics Research*, 32(8):951–970, 2013. 2
- [17] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019. 2
- [18] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019. 5, 6, 7
- [19] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. Action recognition based on a bag of 3d points. In *2010 IEEE computer society conference on computer vision and pattern recognition-workshops*, pages 9–14. IEEE, 2010. 3, 5
- [20] Jun Liu, Amir Shahroudy, Mauricio Perez, Gang Wang, Ling-Yu Duan, and Alex C Kot. Ntu rgb+ d 120: A large-scale benchmark for 3d human activity understanding. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2684–2701, 2019. 3
- [21] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Me-teornet: Deep learning on dynamic 3d point cloud sequences. In *Proceedings of the IEEE/CVF Interna-*

- tional Conference on Computer Vision*, pages 9246–9255, 2019. 2
- [22] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 922–928. IEEE, 2015. 2
- [23] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017. 2, 5, 6, 7, 8
- [24] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, volume 30, 2017. 2, 5, 6, 7, 8
- [25] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Abed Al Kader Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *arXiv preprint arXiv:2206.04670*, 2022. 2
- [26] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 1
- [27] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017. 5
- [28] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1010–1019, 2016. 3
- [29] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4548–4557, 2018. 2
- [30] Jaeyong Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. Unstructured human activity detection from rgb-d images. In *2012 IEEE international conference on robotics and automation*, pages 842–849. IEEE, 2012. 2
- [31] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015. 1
- [32] Haiyan Wang, Liang Yang, Xuejian Rong, Jinglun Feng, and Yingli Tian. Self-supervised 4d spatio-temporal feature learning via order prediction of sequential point cloud clips. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3762–3771, 2021. 2
- [33] Yancheng Wang, Yang Xiao, Fu Xiong, Wenxiang Jiang, Zhiguo Cao, Joey Tianyi Zhou, and Junsong Yuan. 3dv: 3d dynamic voxel for action recognition in depth video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 511–520, 2020. 2
- [34] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38:1–12, 2019. 2
- [35] Siyuan Yang, Jun Liu, Shijian Lu, Meng Hwa Er, and Alex C Kot. Skeleton cloud colorization for unsupervised 3d action representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13423–13433, 2021. 2
- [36] Cheng Zhang, Haocheng Wan, Shengqiang Liu, Xinyi Shen, and Zizhao Wu. Pvt: Point-voxel transformer for 3d deep learning. *arXiv preprint arXiv:2108.06076*, 2021. 2
- [37] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H.S. Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16259–16268, October 2021. 2
- [38] Jia-Xing Zhong, Kaichen Zhou, Qingyong Hu, Bing Wang, Niki Trigoni, and Andrew Markham. No pain, big gain: classify dynamic point cloud sequences with static models by fitting feature-level space-time surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8510–8520, 2022. 2, 7