No Pain, Big Gain: Classify Dynamic Point Cloud Sequences with Static Models by Fitting Feature-level Space-time Surfaces

Jia-Xing Zhong, Kaichen Zhou, Qingyong Hu✉, Bing Wang, Niki Trigoni, Andrew Markham

Deparment of Computer Science, University of Oxford

{jiaxing.zhong, rui.zhou, qingyong.hu, bing.wang, niki.trigoni, andrew.markham}@cs.ox.ac.uk

Abstract

*Scene flow is a powerful tool for capturing the motion field of 3D point clouds. However, it is difficult to directly apply flow-based models to dynamic point cloud classification since the unstructured points make it hard or even impossible to efficiently and effectively trace point-wise correspondences. To capture 3D motions without explicitly tracking correspondences, we propose a **kine**matics-inspired neural **net**work (Kinet) by generalizing the kinematic concept of ST-surfaces to the feature space. By unrolling the normal solver of ST-surfaces in the feature space, Kinet implicitly encodes feature-level dynamics and gains advantages from the use of mature backbones for static point cloud processing. With only minor changes in network structures and low computing overhead, it is painless to jointly train and deploy our framework with a given static model. Experiments on NvGesture, SHREC'17, MSRAction-3D, and NTU-RGBD demonstrate its efficacy in performance, efficiency in both the number of parameters and computational complexity, as well as its versatility to various static backbones. Noticeably, Kinet achieves the accuracy of 93.27% on MSRAction-3D with only 3.20M parameters and 10.35G FLOPS. The code is available at https://github.com/jx-zhong-for-academic-purpose/Kinet.*

1. Introduction

Due to continued miniaturization and mass production, 3D sensors are becoming less esoteric and increasingly prevalent in geometric perception tasks. These sensors typically represent scene geometry through a point cloud, which is an unordered and irregular data structure consisting of distinct spatial 3D coordinates. As a fundamental problem in point cloud understanding, classification of static scenes [11, 30, 72] or objects [7, 58, 59] has witnessed rapid advances over the past few years. Whilst impressive, these techniques do not directly account for the fact that the real 3D world is also changing, through egocentric and/or al-
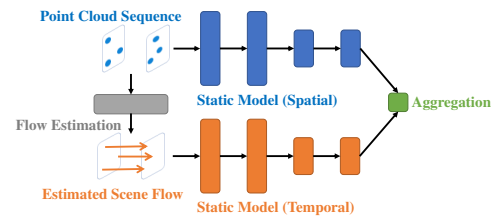
(a) Vanilla two-stream framework based on physical scene flow.
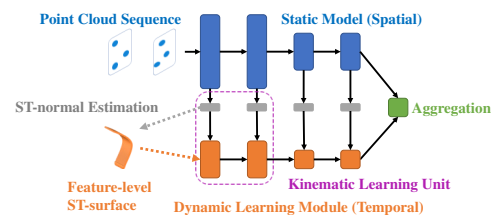
(b) Kinematic two-stream framework guided by feature-level ST-surfaces.

Figure 1. *Comparison between the flow-based framework and ours.* With neither explicit point-wise correspondence estimation nor the stand-alone temporal branch, our framework is lightweight and efficient.

locentric motion. To better understand our time-varying world, a handful of recent works [15–17, 44, 47, 48, 80] have been applied to dynamic point cloud classification, a task in which the model is required to output a video-level category for a given sequence of 3D point clouds.

As a natural extension of 2D optical flow, 3D scene flow captures the motion field of point clouds. Based on optical flow, two-stream networks [6, 19, 68, 76, 89] have already proven to be successful in image-based video classification. Hence, it should be a natural choice to classify dynamic point clouds with the help of scene flow. However, to the best of our knowledge, scene flow has not been utilized in point cloud sequences despite the prevalence of mature scene flow estimators [3, 22, 24, 43, 50, 56, 73, 82].

What then hampers us from applying scene flow to dynamic point cloud classification? Although scene flow is a powerful tool, it is difficult to estimate it efficiently and effectively from sequential point clouds - the computation of 3D scene flow inevitability has higher time expenditure,

larger memory consumption, and lower accuracy than that of 2D optical flow. These challenges are mainly caused by the irregular and unordered nature of dynamic point clouds. This unstructured nature makes it difficult to track the point-wise correspondences of the moving point sets across different frames.

Why not extract dynamic information without explicitly finding the point-wise correspondences? If this were possible, researchers could gain advantages from the decoupled motion representations but not suffer from the painful computational process of scene flow. Similar to the gains seen in two-stream networks in image-based models, we would be able to *preserve the benefit of mature static solutions* in inference and training, such as well-benchmarked network architectures, transferable pre-trained weights, and ready-to-use source code. At the same time, the pain of scene flow estimation would be significantly relieved, only with *minor network modification and low computational overhead*.

For this purpose, we get inspiration from kinematics and propose a neural network (**Kinet**) to bypass direct scene flow estimation by generalizing the kinematic concept of space-time surfaces [55] (ST-surfaces) from the physical domain of point clouds to the feature space. In this way, normal vectors w.r.t. these ST-surfaces (ST-normals) establish the representation field of dynamic information as shown in Figure 1b. Thus, *motions are implicitly represented by feature-level ST-surfaces without explicitly computing point-wise correspondences*. Inspired by iterative normal refinement [49], we unroll the solver for ST-normals and make it jointly trainable alongside the static model in an end-to-end manner. Inheriting intermediate features from static network layers, Kinet is lightweight in parameters and efficient in computational complexity, compared with the vanilla flow-based framework depicted in Figure 1a which requires extra scene flow estimation and the independent temporal branch.

Experiments are conducted on four datasets (*NvGesture* [52], *SHREC'17* [14], *MSRAction-3D* [40] and *NTU-RGBD* [66]) for two tasks (gesture recognition and action classification) with three typical static backbones (MLP-based *PointNet++* [60], graph-based *DGCNN* [78] and convolution-based *SpiderCNN* [83]). Noticeably, 1) in gesture recognition, our framework outperforms humans for the first time with the accuracy of 89.1% on *NvGesture*; 2) in action classification, it achieves a new record of 93.27% on 24-frame *MSRAction-3D* with only $3.20M$ parameters and $10.35G$ FLOPS.

In summary, our main contribution is as follows:

- By introducing Kinet, we decouple temporal information from spatial features, thereby easily extending static backbones to dynamic recognition and entirely preserving the merits of these mature backbones.
- Without the pain of tracking point-wise correspon-

dences, we encode point cloud dynamics by unrolling the ST-normal solver in the feature space. This method is jointly trainable alongside the static model, with minor structural changes and low computing overhead.

- Extensive experiments on various datasets, tasks, and static backbones show its efficacy in performance, efficiency in parameters and computational complexity, as well as versatility to different static backbones. The code is available at https://github.com/jx-zhong-for-academic-purpose/Kinet.

2. Related Work

Deep Learning on Static Point Clouds Recently, deep learning on 3D point clouds has attracted increased attention [25], with substantial progress achieved in several fields including shape classification [7, 41, 51, 58], object detection [5, 38, 57, 67] and scene segmentation [4, 11, 28, 30, 85]. This can be mainly attributed to the availability of various high-quality datasets [4, 11, 29] and sophisticated neural architectures [27, 41, 58, 60]. From the perspective of scene representations, existing works can be roughly divided into 1) Voxel-based methods [10, 23, 46, 65, 90], 2) Projection-based methods [8, 69], 3) Point-based methods [31, 41, 58, 60, 78, 83], and 4) Hybrid methods [12, 45, 59]. Based on the well-developed static classification models, we attempt to apply them to dynamic point cloud recognition with minor structural surgery and low computational overhead.

Deep Learning on Dynamic Point Clouds A handful of recent works have explored dynamic problems on point clouds, such as recognition [16,17,44,48,80], detection [32, 61, 87], tracking [20, 62], prediction [34, 53, 63, 64, 81] and scene flow estimation [3, 22, 24, 43, 50, 56, 73, 82]. Existing works on sequence classification are based on convolutional [44, 44, 47], recurrent [48], self-attentional [16, 80], or multi-stream neural networks [79]. As a convolutional framework, MeteorNet [44] modeled point cloud dynamics via spatio-temporal neighbor aggregations [44]. Likewise, PSTNet [17] applied point spatio-temporal convolutions to capture information along the time dimension and the space domain. Derived from recurrent networks, PointLSTM [48] updated the hidden states with the combination of past and current features. Fan *et al.* [16] and Wei *et al.* [80] adopted self-attentional structures along with the popularity of video transformers [21]. By extracting the offline dynamic voxel, 3DV [79] encoded motions and appearances through multiple streams. Our Kinet shares the same idea of decoupling spatial and temporal information as 3DV, but the presented framework requires neither the offline motion extraction nor the extra stand-alone temporal stream.

Flow-guided Classifiers for Image-based Videos The presented work obtains inspiration from the similar idea of encoding optical flow information into deep representations for image-based video classification [33, 37, 39, 42,
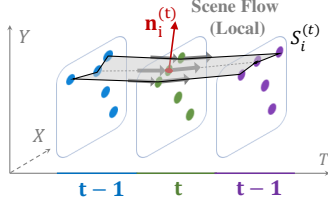
Figure 2. *2D local ST-surface and its normal.* Scene flow (darkgray arrows) lies on ST-surface $S_i^{(t)}$ and it is orthogonal to the normal $\mathbf{n}_i^{(t)}$.
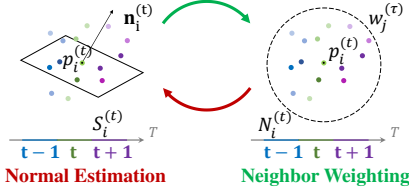

Figure 3. *Solver of iterative normal refinement.* The darkness of colors represents weights of neighbor points. The ST-surface is estimated upon the point-wise weights, and vice versa.
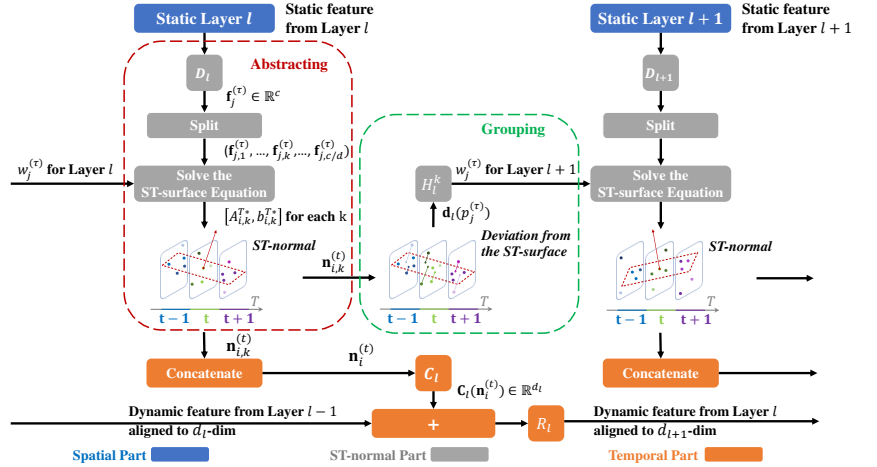

Figure 4. *Kinematic learning unit.* As shown in the violet dotted frame of Figure 1b, a stack of these units comprises the temporal branch of our framework. The alternate abstracting (red dotted frame) and grouping (green dotted frame) unroll the ST-normal solver (Figure 3) in the feature space. Only the **squares** are parametric operations whereas the **rectangles** introduce no parameters - kinematic learning just requires a small number of learnable parameters.

70, 88]. By subtracting feature maps along the temporal axis, OFF [70], STM [33] and PAN [88] robustly imitated optical-flow calculations. Similarly, Piergiovanni & Ryoo [54] and Fan *et al.* [18] mimicked TV-L1 optical flow iterations [71] inside network layers. Heeseung *et al.* [37] introduced correlations to continuous feature maps. For the purpose of acceleration, temporal shift modules [42] or spatial shift filters [39] were utilized to model multi-frame interactions. The above methods rely on feature maps retaining spatial correspondences of regular pixels in images, whereas features of irregular point clouds usually cannot manifest point-wise correspondences across frames. Consequently, all of the aforementioned feature-level operations (subtraction, shift, correlation, *etc.*) are ineffective for point cloud models. To achieve the similar goal of encoding dynamics from static features, we propose a distinct approach from them.

3. Methodology

Denote an input point cloud sequence with T frames as $P = (P_1, P_2, ..., P_{T-1}, P_T)$. The tth frame $P_t = \{p_i^{(t)}|i = 1, 2, ..., m_t - 1, m_t\}$ is a set of m_t points, in which the position of the ith point $p_i^{(t)}$ is specified by its spatial coordinates $\mathbf{x}_i^{(t)} = (x_i^{(t)}, y_i^{(t)}, z_i^{(t)}) \in \mathbb{R}^3$. The goal of dynamic point cloud classification is to output the sequence-level category label y for a particular input P.

3.1. Background: Kinematic ST-Surface

Our method extends the kinematic concept of ST-surfaces [55] and adopts the solver of iterative normal re-

finement [49] from the 3D physical world to deep representation learning.

Intuitively, the local **ST-surface** $S_i^{(t)} \subset \mathbb{R}^4$ centered at the point $p_i^{(t)}$ is *a surface that fits as many as possible space-time neighbor points* of $p_i^{(t)}$. Figure 2 illustrates ST-surface of 2D dynamic point clouds within 3 frames, which can be easily generalized to the 3D case and more frames. According to spatial kinematics [55], instantaneous velocity vectors always lie on the ST-surface. Equivalently, for a point cloud sequence as depicted in Figure 2, Mitra *et al.* [49] point out that *local scene flow lies on the same ST-surface* due to the vicinal (*i.e.*, local neighbourhood) consistency of movements. As a result, the ST-normal $\mathbf{n}_i^{(t)}$ is orthogonal to local scene flow[1] and the field of those normals describes the motions of sequential point clouds.

Mathematically, the space-time neighbors of $p_i^{(t)}$ are a point set: $N_{\Delta r}^{\Delta t}(p_i^{(t)}) = \{p_j^{(\tau)}|\ |t-\tau| \le \Delta t, ||\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(\tau)}|| \le \Delta r\}$, hereinafter referred to as $N_i^{(t)}$ for simplicity. $S_i^{(t)}$ is specified by its tangent plane with the surface equation $A\mathbf{x} + b = t$, of which the coefficient $A \in \mathbb{R}^{1 \times 3}$ and b satisfy:

$$A\mathbf{x}_j^{(\tau)} + b = \tau, \quad \forall p_j^{(\tau)} \in N_i^{(t)}. \tag{1}$$

In practice, Equation (1) may be an over-determined linear system since the local neighbor area may be too large to reflect the instantaneous velocity. In this case, space-time neighbors $N_i^{(t)}$ cannot be completely represented by

---

[1]Strictly speaking, the ST-normal $\mathbf{n}_i^{(t)}$ is orthogonal to the local tangent plane of scene flow.

coefficients of the tangent plane and there exists no exact solution. Therefore, a least-squared approximation is introduced to seek the optimal coefficients A^* and b^*:

$$A^*, b^* = \underset{A,b}{arg\,min} \sum_{p_j^{(\tau)} \in N_i^{(t)}} ||A\mathbf{x}_j^{(\tau)} + b - \tau||^2. \quad (2)$$

To alleviate the influence of noisy point clouds, a commonly-used objective is to obtain the coefficients based on weighted neighbor points:

$$A^*, b^* = \underset{A,b}{arg\,min} \sum_{p_j^{(\tau)} \in N_i^{(t)}} w_j^{(\tau)} ||A\mathbf{x}_j^{(\tau)} + b - \tau||^2, \quad (3)$$

where $w_j^{(\tau)}$ is the point-wise weight of these neighbors. As a solver of Equation (3), **iterative normal refinement** [49] robustly encodes dynamics via the normal field of ST-surfaces. This was earliest used in Dynamic Geometry Registration [49], a traditional method to register large-scale moving and deforming point clouds. As shown in Figure 3, the basic idea is to alternately re-compute the ST-surface $S_i^{(t)}$ and its normal $\mathbf{n}_i^{(t)}$ based on the neighbors' weights $w_j^{(\tau)}$ and re-weight the space-time neighbors $N_i^{(t)}$ based on the estimated ST-surface $S_i^{(t)}$ until convergence.

3.2. Kinematic Representation Learning

3.2.1 Framework

The vanilla flow-based framework (Figure 1a) explicitly extracts scene flow (or dynamic voxels as in the case of 3DV [79]), while Kinet implicitly encodes motions with feature-level ST-surfaces. As shown in Figure 1b, Kinet contains three parts: 1) a spatial branch (marked in blue) identical to common static models, 2) a temporal model comprised of stacked kinematic learning units (the violet dotted frame), followed by 3) the final aggregation (marked in green) of spatial and temporal results.

3.2.2 Kinematic Learning Unit

Typically, a learning unit for point clouds has two crucial operations, *i.e.*, **grouping** and **abstracting**. The former selects the neighbors around centroids (*e.g.* ball query in PointNet++), while the latter encodes the local feature from these neighbors (*e.g.* PointNet layers in PointNet++). Figure 3 demonstrates that iterative normal refinement alternates between two similar operations: **neighbor weighting** corresponding to grouping and **normal estimation** analogous to abstracting. The original integrative normal refinement works in the non-differentiable physical space of three-dimensional point sets - we unroll its solver in a fully *differentiable* fashion and generalize it to the *high-dimensional* feature space for joint optimization in neural networks as depicted in Figure 4.

Assume that we obtain a series of features $F_l(P) = F_l(P_1), F_l(P_2), ..., F_l(P_{T-1}), F_l(P_T)$ from the lth layer (marked in blue in Figure 4) of a certain static model F. Based on sequential static features $F_l(P)$, our learning unit aims to obtain dynamic representations by fitting feature-level ST-surfaces.

Abstracting with Normal Estimation (Red Dotted Frame in Figure 4) To decrease the computational complexity, we first utilize a 1×1 convolution D_l to reduce its dimension to c, where c is proportional to the dimension of a static feature. For a given point $p_i^{(t)}$, the corresponding $c-$dimensional feature vector is denoted as $\mathbf{f}_i^{(t)} \in \mathbb{R}^c$. Similar to the physical space, the tangent hyper-plane of ST-surfaces in the feature space is specified by its surface equation $A\mathbf{f} + b = t$, where the coefficients A and b satisfy:

$$A\mathbf{f}_j^{(\tau)} + b = \tau, \ \ \forall p_j^{(\tau)} \in N_i^{(t)}. \quad (4)$$

Likewise, *time-varying changes of those static features lie on the corresponding ST-(hyper)surface in the representation space*. The vector field of normals w.r.t. such feature-level ST-surfaces orthogonally describes the dynamic information based on static representations.

The equation of ST-surfaces in the 3D physical space is usually over-determined ($|N_i^{(t)}| > 3$), whereas it is not the case in the $c-$dimensional feature space ($|N_i^{(t)}| < c$). To ensure exact coefficient solutions in Equation 4, we split the point-wise feature $\mathbf{f}_j^{(\tau)}$ into several $d-$dimensional groups $\mathbf{f}_j^{(\tau)} = (\mathbf{f}_{j,1}^{(\tau)}, ..., \mathbf{f}_{j,k}^{(\tau)}, ..., \mathbf{f}_{j,c/d}^{(\tau)})$ where $\mathbf{f}_{j,k}^{(\tau)} \in \mathbb{R}^d$ specifies the kth group. For each group $\mathbf{f}_{j,k}^{(\tau)}$, the number of neighbors $N_i^{(t)}$ is sufficiently large to solve the following weighted least-squared approximation:

$$A_{i,k}^*, b_{i,k}^* = \underset{A,b}{arg\,min} \sum_{p_j^{(\tau)} \in N_i^{(t)}} w_{j,k}^{(\tau)} ||A\mathbf{f}_{j,k}^{(\tau)} + b - \tau||^2, \quad (5)$$

where $w_{j,k}^{(\tau)}$ is the point-wise weight of each neighbor. The vanilla iterative normal refinement leverages weighted-PCA [36] to solve the normals, which is unfriendly to back-propagation [77]. To this end, we attempt to directly fit this equation via its closed-form least-squared solution:

$$[A_{i,k}^{T*}, b_{i,k}^*] = (F_{i,k}^{(t)T} W_{i,k}^{(t)} F_{i,k}^{(t)})^{-1} F_{i,k}^{(t)T} W_{i,k}^{(t)} \boldsymbol{\tau}_{i,k}^{(t)}, \quad (6)$$

where the weight matrix $W_{i,k}^{(t)} = diag(w_{1,k}^{(\tau)}, ..., w_{|N_i^{(t)}|,k}^{(\tau)}) \in \mathbb{R}^{|N_i^{(t)}| \times |N_i^{(t)}|}$, the feature matrix $F_{i,k}^{(t)} = [(\mathbf{f}_{1,k}, 1), .., (\mathbf{f}_{|N_i^{(t)}|,k}, 1)] \in \mathbb{R}^{|N_i^{(t)}| \times (d+1)}$ and the time vector $\boldsymbol{\tau}_{i,k}^{(t)} \in \mathbb{R}^{|N_i^{(t)}|}$. The normal vector is as follows:

$$\mathbf{n}_{i,k}^{(t)} = \frac{(A_k^{T*}, -1)}{||(A_k^{T*}, -1)||}, \quad (7)$$

where $|| \cdot ||$ is the $\ell2$-norm. The concatenated normals $\mathbf{n}_i^{(t)} = concat(\mathbf{n}_{i,1}^{(t)}, .., \mathbf{n}_{j,\frac{c}{d}}^{(t)})$ are fed into a 1×1 convolution C_l to obtain the d_l−dimensional abstracted dynamic feature $C_l(\mathbf{n}_i^{(t)})$. After the dimension alignment with another 1×1 convolution R_l, the feature is forwarded to the next layer via a residual connection.

Grouping with Weighted Neighbors (Green Dotted Frame in Figure 4) For each neighbor $p_j^{(\tau)}$, we compute its weight $w_{j,k}^{(\tau)}$ for better point-wise representations in the lth layer. Inspired by iterative normal refinement, we feed the channel-wise fitting deviation (a.k.a. residual) from the $(l-1)^{th}$ layer into a 1×1 convolution H_{l-1}^k activated by $sigmoid$ to obtain the neighbor weights within $[0,1]$:

$$w_{j,k}^{(t)} = H_{l-1}^k(\mathbf{d}_{l-1}(p_j^{(\tau)})), \qquad (8)$$

where $\mathbf{d}_{l-1}(p_j^{(\tau)})$ is the vector of fitting deviations in the $(l-1)^{th}$ layer. In the first layer, $w_{j,k}^{(t)}$ is set to 1 for all of the neighbors.

3.2.3 Aggregation

In the last layer, we leverage the dynamic features to obtain $softmax$ classification scores of a point cloud sequence. The spatial and temporal stream is respectively optimized through a cross-entropy loss in the training stage. During the testing phase, category predictions from the static model and the temporal stream are averaged as the final outputs. Since all the operations of kinematic representation learning are differentiable, it can be seamlessly plugged into a wide range of static neural architectures with minor structural surgery.

4. Experiments

To evaluate the performance of Kinet, we conduct experiments on three datasets (NvGesture [52], SHREC'17 [14], MSRAction-3D [40] and NTU-RGBD [66]) for gesture recognition or action classification. The proposed framework is implemented with TensorFlow [1]. All experiments are conducted on the NVIDIA DGX-1 stations with Tesla V100 GPUs. In most experiments, PointNet++ [60] is adopted as the static backbone in Kinet. For a fair comparison, we follow the identical settings of network layers to [47]. The other backbones are mainly chosen to evaluate the versatility. If not specified explicitly, we keep hyperparameters of the original backbone (see **Appendix** for more details). For training stability, we first train the static backbone (spatial stream) until convergence and then freeze its weights to individually optimize the dynamic branch

(temporal stream). Following [17,44,48], classification accuracy is the main evaluation metric and we use default data splits for fair comparisons.

4.1. NvGesture: Hyper-settings & Ablations

NvGesture consists of 1532 (1050 for training and 482 for testing) videos composed of 25 classes. Following [48], we uniformly sample 32 frames from a video and generate 512 points for each frame. To provide intuition behind the operation of our framework, we first investigate the impact of various network settings and components before we turn to further studies.

Influence of Hyper-settings We explore the effectiveness of crucial hyper-parameters with the 10-fold validation protocol [47]. To exclude the interference of the static branch, we calculate the accuracy only using the classification results of the dynamic stream without ensembles. Box-whisker plots are utilized to analyze the performance and the stability of hyper-settings as shown in Figure 5.

Feature dimensions c controls the output size of convolution D_l to reduce computational overheads, as depicted in Figure 4, which is proportional to the dimension of static features. We vary the ratio from 12.5% to 100% to find a value that makes the representations as compact as possible but informative enough. As shown in Figure 5a, 50% is adequate to encode the dynamic information from a static layer. In comparison with the standalone flow-based branch, this saves at least half of the network parameters.

Group size d controls the dimensions in each group. Ideally, d is expected to be sufficiently large to describe motions. However, excessively large matrices cause inefficiency in the ST-normal solvers. By changing d from 2 to 32, we empirically find that $d = 4$ properly balances the accuracy and computational overheads as shown in Figure 5b.

Temporal radius Δt controls the receptive field of our dynamic branch along the temporal axis. It should be sufficiently small to retain details but large enough to model long-term interactions. It is observed that multi-frame information ($\Delta t \leq 1$) indeed improves the performance over a single frame ($\Delta t = 0$). In the searching range of $[0, 5]$, the best temporal radius Δt is 1 as depicted in Figure 5c.

Spatial radius Δr controls the receptive field in the spatial dimension. Similar to Δt, Δr should have a moderate optimum. From the comparison of $\Delta r \in [0.125, 2.0]$ in Figure 5d, $\Delta r = 0.5$ consistently has decent performance.

In the remaining experiments, we set the ratio of feature reduction as 50%, group-wise dimensions $d = 4$, temporal radius $\Delta t = 1$, and spatial radius $\Delta r = 0.5$, respectively.

Ablation Studies We conduct ablation studies on different components on the test set of *NvGesture*.

Is it beneficial to pretrain on static datasets? For two-stream models of image-based videos, it is well-known that pretraining on images significantly improves the per-
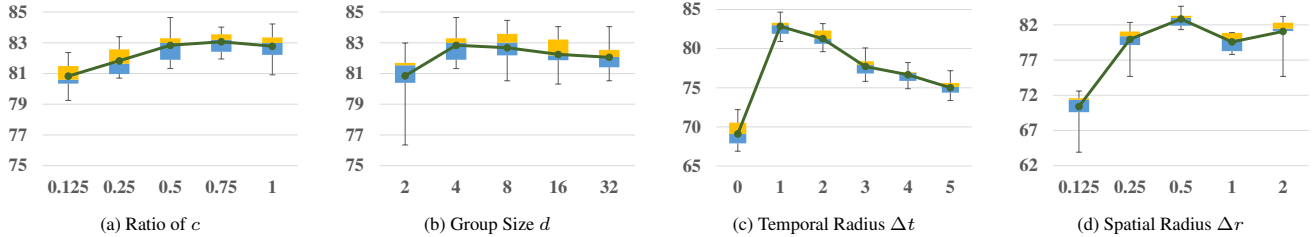
Figure 5. *Box-whisker plots of temporal-stream performance on validation set of* NvGesture *under different hyper-settings.* The x-axis is the value of hyper-parameters, while y-axis is the validation accuracy (%).

| Settings | Spatial (Static) | Temporal (Dynamic) | | | Accuracy |
No.	Pretrain	Pretrain	ST-normal	Weighted	(%)
i.	✗	–	–	–	82.6
ii.	✓	–	–	–	84.5
iii.	–	✗	✓	✓	80.9
iv.	–	✓	✓	✓	82.4
v.	✓	✓	✗	✗	85.3
vi.	✓	✓	✓	✗	87.9
vii.	✓	✓	✓	✓	89.1

Table 1. *Ablation studies on* NvGesture. ✓/✗ *means that the operation is applied/not applied to the framework, while* – *means that the predictions of the corresponding spatial/temporal stream are excluded from the evaluation. The results of the upper/lower part is obtained from a single stream/two streams, respectively.*

formance on videos [6, 68]. However, this fact has not been verified for point cloud sequences. By pretraining the static PointNet++ on ModelNet40 [7], we analyze the individual performance change for each branch. Table 1 *i.* & *ii.* demonstrate that the performance of spatial stream on multi-frame predictions increases by 1.9% (from 82.6% to 84.5%), while the temporal branch using the pretrained static features also boosts the accuracy by 1.5% as shown in Table 1 *iii.* & *iv.*. Obviously, both the static branch and the dynamic stream benefit from pretraining. Based on the pretrained backbone, the fusion of two-stream results achieves the accuracy of 89.1% since the two branches are complementary to each other.

Is it useful to abstract with normal estimation? To evaluate the efficacy of our abstracting operation, we remove the process of normal calculation in Figure 4. Instead, the reduced feature is directly fed into the convolution C_l and aggregated with max pooling. In this manner, all the trainable convolutions are unchanged but the accuracy significantly drops from 89.1% to 85.3% as shown in Table 1 *v.* & *vii.*. By comparing *ii.* with *v.* in Table 1, we find that purely introducing additional convolution parameters over the static backbone contributes only 0.8% to the performance gain. Evidently, normal estimation is a vital component in the abstracting operation.

Is it helpful to group with weighted neighbors? By replacing all the learnable weights of neighbor points with

the fixed value of 1.0, we train the model to analyze the effect of weighted grouping. Table 1 *vi.* shows that the performance decreases to 87.9%, which means the grouping operation with weighted neighbors is capable of further improving the performance of our feature abstraction.

Methods	Modalities	Accuracy (%)
R3DCNN [52]	Infrared Image	63.5
R3DCNN [52]	Optical Flow	77.8
R3DCNN [52]	Depth Map	80.3
PreRNN [86]	Depth Map	84.4
MTUT [2]	Depth Map	84.9
R3DCNN [52]	RGB Frame	74.1
PreRNN [86]	RGB Frame	76.5
MTUT [2]	RGB Frame	81.3
PointNet++ [60]	Point Cloud	63.9
FlickerNet [47]	Point Cloud	86.3
PointLSTM-base [48]	Point Cloud	85.9
PointLSTM-early [48]	Point Cloud	87.9
PointLSTM-PSS [48]	Point Cloud	87.3
PointLSTM-middle [48]	Point Cloud	86.9
PointLSTM-late [48]	Point Cloud	87.5
Human [52]	RGB Frame	88.4
Kinet	Point Cloud	**89.1**

Table 2. *Quantatitve results achieved on* NvGesture.

Comparisons *NvGesture* is a multi-modality dataset and allows us to compare our method to state-of-the-art techniques under the standard data split [52]. As shown in Table 2, our approach with an accuracy of 89.1% not only outperforms all of the existing point cloud methods, but also achieves higher performance than those models using other modalities. It is worth mentioning that Kinet is even superior to the human recognition on RGB videos (88.4%) *for the first time*.

4.2. SHREC'17: Robustness to Noisy Backgrounds

SHREC'17 is comprised of 2800 videos in 28 classes for gesture recognition, of which 70% (2960 videos) are training data and the other 30% (840 videos) are the test set. It has two types of supervisory signals, *i.e.*, video-level classification labels and bounding boxes (BBox) of hand skeletons. Unlike most of the prior works only focusing on the background-free cases, we adopt two input settings to verify whether Kinet can capture useful movements and ignore meaningless ones: 1) w/ BBox (Figure 6a) - used by a ma-
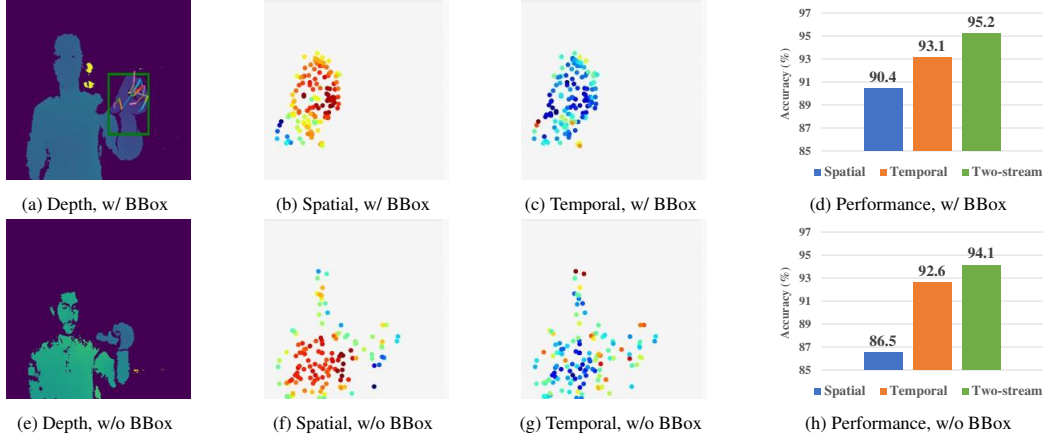
Figure 6. *Raw depth inputs, PACs and stream-wise accuracy on* SHREC'17. In PACs, the points in red have the highest activation values, while the blue ones are the lowest activating points. Best viewed in Adobe Reader where (a)-(c) & (e)-(g) should play as videos.

jority of existing methods for high accuracy, based on the area inside the bounding boxes of hand skeletons without background interference; 2) w/o BBox (Figure 6e) - raw videos with noisy backgrounds (the performer's body).

Qualitative Analysis on Robustness We visualize the learned point activation clouds (PACs) [17, 47] in Figure 6. With bounding boxes removing noisy backgrounds, the two streams work complementary - the static branch (Figure 6b) highlights the main parts (the palms) of spatial appearances, whereas the temporal representations (Figure 6c) capture key motions, such as the movement of fingers and wrists. In the case with redundant backgrounds (without bounding boxes), the static stream (Figure 6f) excessively focuses on the large yet useless background portions (the performer's body), while the temporal stream (Figure 6g) captures the moving parts (arms and fingers). Inevitably, the temporal stream also highlights several redundant points of the performer's shaking head by mistake.

Quantitative Analysis on Robustness From Figure 6d & 6h, it is observed that the dynamic branch shows strong robustness to motion-irrelevant backgrounds, where the accuracy slightly drops from 93.1% to 92.6%, compared with that of the spatial stream which plunges by nearly 4%.

Comparison As shown in Table 3, we compare the performance of Kinet to existing models. With the same bounding box supervision signal, our two-stream PointNet++ outperforms others with the accuracy of 95.2%. Noticeably, even for the highly challenging inputs without bounding boxes, our framework boosts the accuracy of static PointNet++ from only 86.5% to 94.1%, which is comparable to state-of-the-art models with bounding boxes.

4.3. MSRAction-3D: Different Tasks & Backbones

MSRAction-3D has 567 videos of 20 action categories. Following Fan *et al*. [16], we adopt the standard data split-

Methods	Modalities	BBox	Accuracy (%)
Key frames [14]	Depth Map	✗	71.9
SoCJ+HoHD+HoWR [13]	Skeleton	✓	81.9
Res-TCN [26]	Skeleton	✓	87.3
STA-Res-TCN [26]	Skeleton	✓	90.7
ST-GCN [84]	Skeleton	✓	87.7
DG-STA [9]	Skeleton	✓	90.7
PointLSTM-base [48]	Point Cloud	✓	87.6
PointLSTM-early [48]	Point Cloud	✓	93.5
PointLSTM-PSS [48]	Point Cloud	✓	93.1
PointLSTM-middle [48]	Point Cloud	✓	94.7
PointLSTM-late [48]	Point Cloud	✓	93.5
Kinet	Point Cloud	✓	**95.2**
Kinet	Point Cloud	✗	94.1

Table 3. *Quantitative results achieved on* SHREC'17.

ting protocol [44, 75] and report the average accuracy over 10 runs.

Versatility across Static Backbones Under the taxonomy defined in [25], three types of backbones dominate the classification models of static point clouds: MLP-based, convolution-based, and graph-based methods. In this paper, we choose one typical static architecture for each paradigm respectively, *i.e.*, MLP-based Point-Net++ [60], convolution-based SpiderCNN [83] and graph-based DGCNN [78] in order to demonstrate the ease of extending these to dynamic point cloud tasks. By grouping a video into 16-frame clips as the input unit, we run 3 sets of experiments for each static model: 1) Directly feed videos into the static model; 2) Fuse the static spatial model and the temporal streams; 3) Ensemble classification scores from two static models, one is trained on the raw point clouds, while the other flow-based model is trained on scene flow estimated by Justgo [50], a self-supervised scene flow estimation tool. Apart from classification performance, we also take memory consumption and computational complexity into consideration. They are measured with the accuracy,

Methods	Modalities	# of Frames	Accuracy (%)
Vieira *et al.* [74]	Depth Map	20	78.20
Kläser *et al.* [35]	Depth Map	18	81.43
Actionlet [75]	GroundTruth Skeleton	Full	88.21
PointNet++ [60]	Point Cloud	1	61.61
MeteorNet [44]	Point Cloud	4 / 8 / 12 / 16 / 24	78.11 / 81.14 / 86.53 / 88.21 / 88.50
P4Transformer [16]	Point Cloud	4 / 8 / 12 / 16 / 24	80.13 / 83.17 / 87.54 / 89.56 / 90.94
PSTNet [17]	Point Cloud	4 / 8 / 12 / 16 / 24	**81.14** / 83.50 / 87.88 / 89.90 / 91.20
Kinet	Point Cloud	4 / 8 / 12 / 16 / 24	79.80 / **83.84** / **88.53** / **91.92** / **93.27**

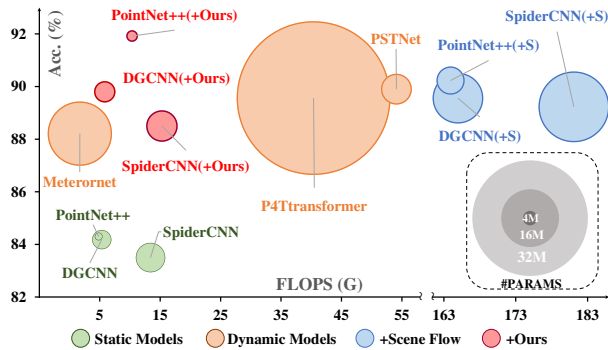Table 4. *Quantitative results achieved on* MSRAction-3D.

Figure 7. *Comparison of FLOPS, parameter number, and accuracy on 16-frame* MSRAction-3D. Quantitative details can be found in **Appendix**.

the number of parameters, and floating-point operations per second (FLOPS), respectively.

As illustrated in Figure 7, the extra input modality of estimated scene flow in setting 3) (+*Scene Flow* ●) considerably improves the accuracy of the three static models (●) to a level comparable to the state-of-the-art. However, the scene flow estimator and another flow-based classifier almost triple the number of parameters. Even worse, the estimation of scene flow introduces more than 150G FLOPS of extra calculations since the point-wise dense predictions are required between every two consecutive frames. For setting 2) (+*Ours* ●), it is observed that our kinematic representations consistently increase the accuracy of the static predictions by 5.99%~9.04% relative gains. By utilizing the kinematic representations, the FLOPS only increases to 5.83G~15.29G, and the number of parameters increases by 0.59M~1.08M. These computing overheads are negligible and make the fused model extremely lightweight. Compared with state-of-the-art dynamic networks (●), they achieve comparable or superior performance with fewer model parameters and lower computational complexity. Intriguingly, Kinet has higher performance gains in higher-performance static models, possibly indicating its *limitation* - the informativeness of kinematic representations is constrained by the static features. A poor static backbone would benefit from the proposed approach, but it cannot improve the underlying static representation.

Comparison We compare the PointNet++-based Kinet with existing classification models on dynamic point clouds. Following prior research [16, 17, 44], we set the length as $\{4, 8, 12, 16, 24\}$ frames (2048 points/frame) and report the mean accuracy of 10 runs. As shown in Table 4, our framework shows the superiority in long videos (length$\geq$ 8) and achieves the 24-frame accuracy of 93.27%, exceeding others by a large margin.

4.4. NTU-RGBD: Effectiveness on Large-scale Data

After we add the author list and the acknowledgement section to our camera-ready version, this part has to be moved to **Appendix** because of limited space. Please see **Appendix** for more details.

5. Conclusion

In this paper, we propose Kinet to bridge the gap between static point cloud models and dynamic sequences. By extending kinematic ST-surfaces to the high-dimensional feature space and unrolling the ST-normal solver differentiably, the presented framework gains advantages from mature static models. Without the pain of modeling point-wise correspondences, it can be seamlessly integrated into arbitrary static point cloud learning backbones, with only minor structural surgery and low computing overhead. Experiments on four datasets, two tasks, and three static networks demonstrate the efficacy of our framework in dynamic classification, the efficiency in parameters and FLOPS, and the versatility to various static backbones. An obvious limitation is that Kinet is constrained by the performance of the static model - a poor static baseline will not be rectified by adding our dynamic extension. Therefore, we will explore a method to self-improve static representations in future works.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 5

[2] Mahdi Abavisani, Hamid Reza Vaezi Joze, and Vishal M Patel. Improving the performance of unimodal dynamic hand-gesture recognition with multimodal training. In *CVPR*, 2019. 6

[3] Aseem Behl, Despoina Paschalidou, Simon Donné, and Andreas Geiger. Pointflownet: Learning representations for rigid motion estimation from point clouds. In *CVPR*, 2019. 1, 2

[4] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences. In *ICCV*, 2019. 2

[5] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 2

[6] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 1, 6

[7] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015. 1, 2, 6

[8] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3D object detection network for autonomous driving. In *ICCV*, 2017. 2

[9] Yuxiao Chen, Long Zhao, Xi Peng, Jianbo Yuan, and Dimitris N Metaxas. Construct dynamic graphs for hand gesture recognition via spatial-temporal attention. In *BMVC*, 2019. 7

[10] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *ICCV*, 2019. 2

[11] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *ICCV*, 2017. 1, 2

[12] Angela Dai and Matthias Nießner. 3DMV: Joint 3D-multi-view prediction for 3D semantic scene segmentation. In *ECCV*, 2018. 2

[13] Quentin De Smedt, Hazem Wannous, and Jean-Philippe Vandeborre. Skeleton-based dynamic hand gesture recognition. In *CVPRW*, 2016. 7

[14] Quentin De Smedt, Hazem Wannous, Jean-Philippe Vandeborre, Joris Guerry, Bertrand Le Saux, and David Filliat. Shrec'17 track: 3D hand gesture recognition using a depth and skeletal dataset. In *3DOR*, 2017. 2, 5, 7

[15] Hehe Fan and Yi Yang. PointRNN: Point recurrent neural network for moving point cloud processing. *arXiv preprint arXiv:1910.08287*, 2019. 1

[16] Hehe Fan, Yi Yang, and Mohan Kankanhalli. Point 4D transformer networks for spatio-temporal modeling in point cloud videos. In *CVPR*, 2021. 1, 2, 7, 8

[17] Hehe Fan, Xin Yu, Yuhang Ding, Yi Yang, and Mohan Kankanhalli. PSTNet: Point spatio-temporal convolution on point cloud sequences. In *ICLR*, 2021. 1, 2, 5, 7, 8

[18] Lijie Fan, Wenbing Huang, Chuang Gan, Stefano Ermon, Boqing Gong, and Junzhou Huang. End-to-end learning of motion representation for video understanding. In *CVPR*, 2018. 3

[19] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. 1

[20] Silvio Giancola, Jesus Zarzar, and Bernard Ghanem. Leveraging shape completion for 3d siamese tracking. In *CVPR*, 2019. 2

[21] Rohit Girdhar, João Carreira, Carl Doersch, and Andrew Zisserman. Video Action Transformer Network. In *CVPR*, 2019. 2

[22] Zan Gojcic, Or Litany, Andreas Wieser, Leonidas J Guibas, and Tolga Birdal. Weakly supervised learning of rigid 3d scene flow. In *CVPR*, 2021. 1, 2

[23] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3D semantic segmentation with submanifold sparse convolutional networks. In *ICCV*, 2018. 2

[24] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. HPLFlowNet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *CVPR*, 2019. 1, 2

[25] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3D point clouds: A survey. *IEEE TPAMI*, 2020. 2, 7

[26] Jingxuan Hou, Guijin Wang, Xinghao Chen, Jing-Hao Xue, Rui Zhu, and Huazhong Yang. Spatial-temporal attention res-tcn for skeleton-based dynamic hand gesture recognition. In *ECCVW*, 2018. 7

[27] Qingyong Hu, Bo Yang, Guangchi Fang, Yulan Guo, Ales Leonardis, Niki Trigoni, and Andrew Markham. Sqn: Weakly-supervised semantic segmentation of large-scale 3d point clouds with 1000x fewer labels. *arXiv preprint arXiv:2104.04891*, 2021. 2

[28] Qingyong Hu, Bo Yang, Sheikh Khalid, Wen Xiao, Niki Trigoni, and Andrew Markham. Towards semantic segmentation of urban-scale 3D point clouds: A dataset, benchmarks and challenges. In *CVPR*, 2021. 2

[29] Qingyong Hu, Bo Yang, Sheikh Khalid, Wen Xiao, Niki Trigoni, and Andrew Markham. Sensaturban: Learning se-

mantics from urban-scale photogrammetric point clouds. *International Journal of Computer Vision*, pages 1–28, 2022. 2

[30] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. RandLA-Net: Efficient semantic segmentation of large-scale point clouds. In *CVPR*, 2020. 1, 2

[31] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Learning semantic segmentation of large-scale point clouds with random sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2

[32] Rui Huang, Wanyue Zhang, Abhijit Kundu, Caroline Pantofaru, David A. Ross, Thomas Funkhouser, and Alireza Fathi. An lstm approach to temporal 3d object detection in lidar point clouds. In *ECCV*, 2020. 2

[33] Boyuan Jiang, MengMeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. Stm: Spatiotemporal and motion encoding for action recognition. In *ICCV*, 2019. 2, 3

[34] Boyan Jiang, Yinda Zhang, Xingkui Wei, Xiangyang Xue, and Yanwei Fu. Learning compositional representation for 4d captures with neural ode. In *CVPR*, 2021. 2

[35] Alexander Klaser, Marcin Marszałek, and Cordelia Schmid. A spatio-temporal descriptor based on 3D-gradients. In *BMVC*, 2008. 8

[36] W.P. Krijnen and H.A.L. Kiers. An efficient algorithm for weighted pca. *Computational Statistics*, 10(3):299–306, 1995. 4

[37] Heeseung Kwon, Manjin Kim, Suha Kwak, and Minsu Cho. Motionsqueeze: Neural motion feature learning for video understanding. In *ECCV*, 2020. 2, 3

[38] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 2

[39] Myunggi Lee, Seungeui Lee, Sungjoon Son, Gyutae Park, and Nojun Kwak. Motion feature network: Fixed motion filter for action recognition. In *ECCV*, 2018. 2, 3

[40] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. Action recognition based on a bag of 3D points. In *CVPRW*, 2010. 2, 5

[41] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution on X-transformed points. In *NeurIPS*, 2018. 2

[42] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, 2019. 2, 3

[43] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. FlowNet3D: Learning scene flow in 3D point clouds. In *CVPR*, 2019. 1, 2

[44] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. In *ICCV*, 2019. 1, 2, 5, 7, 8

[45] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3D deep learning. In *NeurIPS*, 2019. 2

[46] Daniel Maturana and Sebastian Scherer. Voxnet: A 3D convolutional neural network for real-time object recognition. In *IROS*, 2015. 2

[47] Yuecong Min, Xiujuan Chai, Lei Zhao, and Xilin Chen. Flickernet: Adaptive 3D gesture recognition from sparse point clouds. In *BMVC*, 2019. 1, 2, 5, 6, 7

[48] Yuecong Min, Yanxiao Zhang, Xiujuan Chai, and Xilin Chen. An efficient PointLSTM for point clouds based gesture recognition. In *CVPR*, 2020. 1, 2, 5, 6, 7

[49] Niloy J Mitra, Simon Flöry, Maks Ovsjanikov, Natasha Gelfand, Leonidas J Guibas, and Helmut Pottmann. Dynamic geometry registration. In *Symposium on geometry processing*, 2007. 2, 3, 4

[50] Himangi Mittal, Brian Okorn, and David Held. Just go with the flow: Self-supervised scene flow estimation. In *CVPR*, 2020. 1, 2, 7

[51] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *CVPR*, 2019. 2

[52] Pavlo Molchanov, Xiaodong Yang, Shalini Gupta, Kihwan Kim, Stephen Tyree, and Jan Kautz. Online detection and classification of dynamic hand gestures with recurrent 3D convolutional neural network. In *CVPR*, 2016. 2, 5, 6

[53] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *ICCV*, 2019. 2

[54] AJ Piergiovanni and Michael S Ryoo. Representation flow for action recognition. In *CVPR*, 2019. 3

[55] Helmut Pottmann and Johannes Wallner. *Computational Line Geometry*. Springer Science & Business Media, 2001. 2, 3

[56] Gilles Puy, Alexandre Boulch, and Renaud Marlet. FLOT: Scene Flow on Point Clouds Guided by Optimal Transport. In *ECCV*, 2020. 1, 2

[57] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3D object detection from RGB-D data. In *CVPR*, 2018. 2

[58] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017. 1, 2

[59] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3D data. In *CVPR*, 2016. 1, 2

[60] 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*, 2017. 2, 5, 6, 7, 8

[61] Charles R. Qi, Yin Zhou, Mahyar Najibi, Pei Sun, Khoa Vo, Boyang Deng, and Dragomir Anguelov. Offboard 3d object detection from point cloud sequences. In *CVPR*, 2021. 2

[62] Haozhe Qi, Chen Feng, Zhiguo Cao, Feng Zhao, and Yang Xiao. P2b: Point-to-box network for 3d object tracking in point clouds. In *CVPR*, 2020. 2

[63] Davis Rempe, Tolga Birdal, Yongheng Zhao, Zan Gojcic, Srinath Sridhar, and Leonidas J. Guibas. Caspr: Learning canonical spatiotemporal point cloud representations. In *NeurIPS*, 2020. 2

[64] Davis Rempe, Srinath Sridhar, He Wang, and Leonidas J. Guibas. Predicting the physical dynamics of unseen 3D objects. In *WACV*, 2020. 2

[65] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3D representations at high resolutions. In *CVPR*, 2017. 2

[66] 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, 5

[67] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3D object proposal generation and detection from point cloud. In *CVPR*, 2019. 2

[68] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NeurIPS*, 2014. 1, 6

[69] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *ICCV*, 2015. 2

[70] Shuyang Sun, Zhanghui Kuang, Lu Sheng, Wanli Ouyang, and Wei Zhang. Optical flow guided feature: A fast and robust motion representation for video action recognition. In *CVPR*, 2018. 2, 3

[71] Javier Sánchez Pérez, Enric Meinhardt-Llopis, and Gabriele Facciolo. TV-L1 Optical Flow Estimation. *Image Processing On Line*, 3:137–150, 2013. https://doi.org/10.5201/ipol.2013.26. 3

[72] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. KPConv: Flexible and deformable convolution for point clouds. In *ICCV*, 2019. 1

[73] Ivan Tishchenko, Sandro Lombardi, Martin R Oswald, and Marc Pollefeys. Self-supervised learning of non-rigid residual flow and ego-motion. In *3DV*, 2020. 1, 2

[74] Antonio W Vieira, Erickson R Nascimento, Gabriel L Oliveira, Zicheng Liu, and Mario FM Campos. Stop: Spacetime occupancy patterns for 3D action recognition from depth map sequences. In *Iberoamerican congress on pattern recognition*, 2012. 8

[75] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *CVPR*, 2012. 7, 8

[76] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Val Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016. 1

[77] Wei Wang, Zheng Dang, Yinlin Hu, Pascal Fua, and Mathieu Salzmann. Backpropagation-friendly eigendecomposition. In *NeurIPS*, 2019. 4

[78] 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 TOG*, 2019. 2, 7

[79] 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 *CVPR*, 2020. 2, 4

[80] Yimin Wei, Hao Liu, Tingting Xie, Qiuhong Ke, and Yulan Guo. Spatial-temporal transformer for 3d point cloud sequences. In *WACV*, 2021. 1, 2

[81] Xinshuo Weng, Jianren Wang, Sergey Levine, Kris Kitani, and Nick Rhinehart. Inverting the Pose Forecasting Pipeline with SPF2: Sequential Pointcloud Forecasting for Sequential Pose Forecasting. In *CoRL*, 2020. 2

[82] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation. In *ECCV*, 2020. 1, 2

[83] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *ECCV*, 2018. 2, 7

[84] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI*, 2018. 7

[85] Bo Yang, Jianan Wang, Ronald Clark, Qingyong Hu, Sen Wang, Andrew Markham, and Niki Trigoni. Learning object bounding boxes for 3D instance segmentation on point clouds. In *NeurIPS*, 2019. 2

[86] Xiaodong Yang, Pavlo Molchanov, and Jan Kautz. Making convolutional networks recurrent for visual sequence learning. In *CVPR*, 2018. 6

[87] Junbo Yin, Jianbing Shen, Chenye Guan, Dingfu Zhou, and Ruigang Yang. Lidar-based online 3d video object detection with graph-based message passing and spatiotemporal transformer attention. In *CVPR*, 2020. 2

[88] Can Zhang, Yuexian Zou, Guang Chen, and Lei Gan. Pan: Persistent appearance network with an efficient motion cue for fast action recognition. In *ACM MM*, 2019. 2, 3

[89] Shiwen Zhang, Sheng Guo, Weilin Huang, Matthew R Scott, and Limin Wang. V4D: 4D convolutional neural networks for video-level representation learning. In *ICLR*, 2020. 1

[90] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3D object detection. In *CVPR*, 2018. 2