論文 / 著書情報
Article / Book Information

Title	Cross-View Human Action Recognition from Depth Maps Using Spectral Graph Sequences
Authors	Tommi Kerola, Nakamasa Inoue, Koichi Shinoda
Citation	Elsevier Journal of Computer Vision and Image Understanding (CVIU), vol. 154, pp. 108-126
Pub. date	2017, 1
DOI	http://dx.doi.org/10.1016/j.cviu.2016.10.004
Creative Commons	See next page.
Note	This file is author (final) version.

License

Cross-view Human Action Recognition from Depth Maps Using Spectral Graph Sequences

Tommi Kerola[a,*], Nakamasa Inoue[a], Koichi Shinoda[a]

[a]*Department of Computer Science, Tokyo Institute of Technology, Tokyo 152-8552, Japan*

Abstract

We present a method for view-invariant action recognition from depth cameras based on graph signal processing techniques. Our framework leverages a novel graph representation of an action as a temporal sequence of graphs, onto which we apply a spectral graph wavelet transform for creating our feature descriptor. We evaluate two view-invariant graph types: skeleton-based and keypoint-based. The skeleton-based descriptor captures the spatial pose of the subject, whereas the keypoint-based is able to capture complementary information about human-object interaction and the shape of the point cloud. We investigate the effectiveness of our method by experiments on five publicly available datasets. By the graph structure, our method captures the temporal interaction between depth map interest points and achieves a 19.8% increase in performance compared to state-of-the-art results for cross-view action recognition, and competing results for frontal-view action recognition and human-object interaction. Namely, our method results in 90.8% accuracy on the cross-view N-UCLA Multiview Action3D dataset and 91.4% accuracy on the challenging MSRAction3D dataset in the cross-subject setting. For human-object interaction, our method achieves 72.3% accuracy on the Online RGBD Action dataset. We also achieve 96.0% and 98.8% accuracy on the MSRActionPairs3D and UCF-Kinect datasets, respectively.

Keywords: Human action recognition, depth cameras, spectral graph theory, graph signal processing, graph wavelets, wavelet transform

1. Introduction

We live in a world where machines are able to either aid or completely replace humans in a large variety of tasks. Most such tasks are quite trivial and monotonic, but thanks to the advent of machine learning, we are at the verge of being able to demand satisfying performance even for more complex tasks. One such task is action recognition. If machines could robustly recognize and interpret human actions and gestures, the benefits would be vast for a number of areas, including games, health care and the security industry.

Classic approaches to action recognition based on simple color images face numerous difficulties due to intra-class variations of actions, background clutter and illumination variations. However, thanks to the emergence of cheap and affordable depth maps with devices such as the Microsoft Kinect, there has been a recent increase in research using 3D features [1]. Leveraging 3D cameras solves the problem of separating the action subject from the video background, and also eliminates irrelevant information such as illumination variance. Recently, due to the work of Shotton *et al.* [2], we have access to low-dimensional skeletons mapped to the human body. Out of the box, these skeletons are much more discriminative than the raw high-dimensional RGB-D data and allow the development of efficient methods for action recognition. However, while the 3D skeletons provide means of alleviating the action recognition task, they also provide new challenges due to unstable joint positions resulting from tracking errors in the noisy depth maps.

A recurring question in machine learning is the one of how to best represent objects for handling the pattern learning task. Generally, the approaches to this problem can be divided into two: statistical and structural [3]. While statistical methods have received a great deal of attention in the past years, we ask ourselves if objects are not better represented by an explicit structure suitable to the task at hand. Actions are typically defined by a sequence of interactions between several interest points [4]. E.g. "draw circle":

1. Move hand towards left side of waist.
2. Move hand up.
3. Move hand down towards right side of waist.
4. Move hand down towards feet.

Naturally, a good descriptor for action recognition needs to capture interactions between different parts of the body, all of which also vary temporally during the duration of

*Corresponding author

Email addresses: kerola@ks.cs.titech.ac.jp (Tommi Kerola), inoue@ks.cs.titech.ac.jp (Nakamasa Inoue), shinoda@cs.titech.ac.jp (Koichi Shinoda)

the action. While most existing action recognition methods from depth maps capture such interactions [5–9], most of them are inherently view-dependent. That is, their performance depend on the camera angle from which the action was recorded. Cross-view action recognition is the task of recognizing an action independent of the camera angle used for recording the video. For RGB videos, this has previously been explored to some extent [10–20]. For depth maps, however, the number of methods that apply to cross-view action recognition from pure 3D data are much fewer [21–23]. This despite the added advantage of being able to perform action recognition without compromising the identity of the user, which is essential for health care applications.

In this work, we consider to use graphs to represent actions due to the following reasons. First, a graph provides a natural structure for representing interactions between interest points. Furthermore, since graphs naturally capture pair-wise information, a graph-based representation is inherently view-invariant provided that this holds for the signal defined on the vertices. This is our motivation for exploring the usage of graphs for action recognition.

In real life problems, graphs can be found everywhere. They occur in forms of *e.g.* social- and transportation networks, finite state machines, and also in domains such as brain fMRI and computer graphics [24]. Recent approaches for using graphs in machine learning include graph kernels [25–29], generalizations of signal processing frameworks to the graph domain [24, 30], and also graph wavelets [31–35].

Graph signal processing allows signal propagation that follows the natural structure of objects, and applications include edge-aware image processing [36], depth video coding [37], image compression [38], anomaly detection in wireless sensor networks [39], bridge structure health monitoring [40], brain functional connectivity analysis [41] and mobility pattern prediction [42].

Our interest in using graph signal processing for human action recognition lies in the graph frequency information it is able to provide. As our results will show in this paper, using generalizations of wavelet transforms to graphs allows us to capture multi-scale information about the interactions between depth map interest points along with their temporal propagation, leading to an efficient method for classifying a wide range of actions.

In this paper, we propose a system for view-invariant depth map action recognition based on graph signal processing techniques. Our framework leverages a novel graph representation of an action as a temporal sequence of graphs. Specifically, our method takes depth map interest points and embeds these on an augmented graph describing said points' temporal progression. Extending a preliminary study on this subject [43], we investigate two types of interest points:

- Tracked skeleton joints, which capture subject pose and provides a semantic labeling of body parts.
- Spatio-temporal keypoints, which capture human-object interaction and other fine intrinsic detail.

We define view-invariant graph signals based on the above interest points, and we represent them using a novel graph representation that is shown to out-perform more classic representations, such as bag-of-words (BoW) [44] combined with a support vector machine (SVM) [45]. Particularly, we leverage the spectral graph wavelet transform (SGWT) framework of Hammond *et al.* [31] for creating a multi-scale representation of the interest points. Graph wavelets capture information about a signal at different scales, in several dimensions on the augmented temporal graph; both between interest points and along time. Further, spectral graph wavelets offer more flexibility than classical wavelets due to the freedom of graph design. To capture the sequential behavior of actions, we utilize a temporal pyramid pooling scheme [6, 8, 46] on the wavelet coefficients. This improves over approaches that consider only global information [47, 48], since it allows us to capture differently segmented levels of temporal dependencies. Classification is finally performed using an off-the-shelf SVM.

Our proposed method has the following advantages:

- The underlying graph has an explicit block sparsity structure, which we exploit to create a memory-efficient algorithm for calculating the SGWT (see Sec. 4.3.1).
- The feature's underlying spectral basis is mathematically well defined [31], enabling analysis about each part of the descriptor. On the contrary, methods based on *e.g.* sparse coding [6] or deep learning [16] produce bases that are not easily analyzable (see Sec. 4.9).
- For skeleton-based graphs, the number of interest points N is small, making the method efficiently computable in $\mathcal{O}(TN)$ time, where T is the number of frames, making it more computationally efficient than approaches that rely on solving heavy optimization problems [6, 7] (see Sec. 4.7).
- For keypoint-based graphs, the descriptor is shown to capture more information than a baseline BoW-representation, which makes our method perform better using our spectral representation (see Sec 5).

While this paper focuses on recognition of actions, the framework can in general be applied to any time series of graphs.

The paper is organized as follows. Section 2 reviews related research in action recognition and graph signal processing. Section 3 discusses how to represent actions as graphs. Our proposed method is then shown in Sec. 4, with related experiments in Sec. 5. Section 6 finally concludes the paper.

1.1. Notation

We use lower-case bold letters $\mathbf{a} = [a(1), \ldots, a(n)]^T$ to denote vectors, and $a(i)$ denotes the i-th element of a vector. We use upper-case bold letters $\mathbf{A}, \mathbf{B}, \mathbf{C}$ to denote matrices, with $\mathrm{A}(i, j)$ referring to the element at the i-th row and j-th column of $\mathbf{A}$. Let $\mathbf{a}_n$ denote the n-th vector in a set of vectors. We use $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ to denote an undirected graph with vertex set $\mathcal{V} = \{v_i\}$ and edge set $\mathcal{E} = \{e_k : e_k = (v_i, v_j) \Leftrightarrow v_i \sim v_j;\ v_i, v_j \in \mathcal{V}\}$ and $v_i \sim v_j$ denotes that vertices i, j are connected by an edge. The weight matrix $\mathbf{W}$ stores the weight of an edge (v_i, v_j) in entry $\mathrm{W}(i, j)$.

2. Related Work

2.1. 3D Action Recognition

The advent of cheap 3D cameras such as the Kinect has enabled a great performance increase for action recognition tasks [47]. The availability of RGB-D data has considerably eased the task of segmenting an actor from its background; something that is normally quite challenging when using only RGB data. Related research in this field can be roughly divided into three categories: depth map-based, skeleton-based, and methods that utilize both.

Methods that make use of the raw depth map voxel data include Li *et al.* [47], who present a method where a bag of 3D points is sampled from 2D projections of salient depth map poses. Their results show that 3D action recognition clearly outperforms 2D approaches while additionally providing robustness against occlusions. Viera *et al.* [49] introduced space-time occupancy patterns (STOP), where the 3D points of the depth map are represented by a modified 4D histogram. Oreifej and Liu [5] learn a non-uniformly quantized 4D space, in which histograms of oriented 4D normals (HON4D) of the depth map are used for classification. Yang *et al.* [48] create DMM-HOG, which stacks orthogonally projected depth maps that are then applied to histograms of oriented gradients. Rahmani *et al.* [21] develop histograms of oriented principal components (HOPC), which capture a quantized spatio-temporal shape of the point cloud using a 20-dimensional regular polytype. Their work also introduces spatio-temporal keypoints (STKP), which represent view-invariant spatio-temporal locations that can be used as base features for cross-view action recognition. In another work, the same authors develop a transfer-learning system based on deep learning for engineering a bottleneck feature that can be used for cross-view action recognition (NKTM) [16]. Although their bottleneck feature is effective in the cross-view case, their system requires generating synthetic poses from a large auxiliary motion capture dataset for learning the neural network. While depth map-based methods are able to capture information about shapes in great detail, they do however suffer from not knowing the correspondence between regions in the RGB-D data and the human body.

Other approaches rely only on the provided 3D skeletons. This includes DL-GSGC by Luo *et al.* [6], which uses sparse coding with constraints for group sparsity and feature geometry to increase the discriminative power. Together with max pooling and a temporal pyramid pooling scheme, their method also achieves an enhanced sequential representation structure. Zhao *et al.* [9] create SSS, which employs sparse coding and dictionary template learning to learn gestures based on distances between pairwise joints. Another method includes HOJ3D by Xia *et al.* [22], which applies linear discriminant analysis to create a time series of visual words (postures) that are then used as features in a hidden Markov model. Other methods use nearest-neighbor classifiers for classifying derivatives [50] (MP), or dimensionality-reduced relative measurements [51] (Eigenjoints) of 3D joint positions. Gowayyed *et al.* [46] create histograms of oriented displacements (HOD), where quantized angles of skeleton joints are applied to a temporal pyramid for handling temporal dependencies of actions. Ellis *et al.* [52] create a low latency scheme for classifying actions by finding canonical poses using multiple instance learning. While their method is efficiently computable, it is unsuitable for actions that have a strict temporal structure rather than a characteristic pose, such as the action "*drawing an x*". Wang *et al.* [53] uses the tracked skeleton information to learn an AND-OR graph (AOG) of base features (which is actually a tree structure) for cross-view action recognition that is able to capture the compositional structure of base features among different views. While their method is able to recognize actions using only RGB data once trained, adding new action classes requires expensive re-tuning of the parameters of the AOG.

Finally, some works utilize both depth data and 3D skeletons simultaneously. Wang *et al.* [8] create an algorithm for selecting discriminative relative joint pairs that reduce ambiguity between action classes (AE). They also utilize a temporal pyramid, and classification is done using multiple kernel learning. Wang and Wu [7] develop MMTW for tackling temporal misalignment of actions by leveraging a discriminatively learned warping matrix for aligning action sequences before the classification step. Warping templates are learned one per class and classification is done using a latent structural SVM.

While the availability of depth maps has resulted in a recent boost in performance on benchmark datasets [8, 47], most approaches to human action recognition are however inherently view-dependent [5–7]. That is, they depend on the camera angle from which the action was recorded. Natural actions can not however be said to be defined by the angle from which they are seen, but rather from what interactions occur between different body parts. Cross-view action recognition has been explored to some extent for RGB-based action recognition, which includes approaches based on geometric transformations [10, 11], view-invariant features [12–15] and knowledge transfer between different views [16–20]. The number of approaches using only depth maps for cross-view action

recognition are, however, much fever [21–23, 54], despite the added privacy advantage of being able to perform action recognition without compromising the identity of the user. Preserving privacy is essential for *e.g.* health care applications. The approach proposed in this paper falls in the first category of geometric transformations, with the added benefit of being computable using only depth data.

2.2. Signal Processing on Graphs

Recently, several techniques for generalizing classical signals processing (CSP) techniques to arbitrary graphs have been proposed [24]. Graph signal processing (GSP) provides graph analogs to classical Fourier transform tools, such as filtering, translation, convolution, *etc.*. CSP is restricted to signals in regular grids, but most natural signals do not follow this structure (*e.g.* sensor networks and anthropometric meshes). On the other hand, GSP allows processing signals on graphs that are directly adapted to the signal domain itself. By the increased freedom of graph design, we are able to extend CSP approaches to include additional information along *e.g.* extra added graph edges, ultimately increasing the descriptive power of the signal itself.

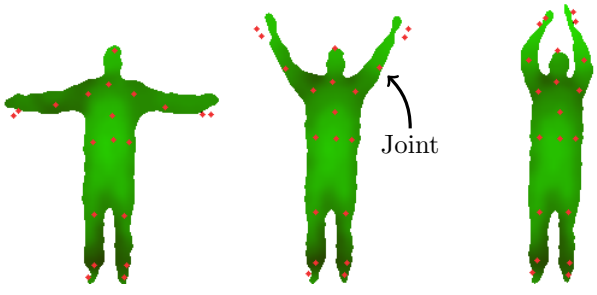
Several works have created wavelets on graphs using GSP [31–35]. One of the earliest works on graph wavelets include a method by Crovella and Kolaczyk [33] for analyzing computer traffic data on unweighted graphs. Hammond *et al.* [31] develop a spectral graph wavelet transform (SGWT), which allows analysis of localized signals on the graph Fourier spectrum of an undirected graph. We note that spectral graph wavelets can been seen related to sparse coding [55, 56]. The spectral graph wavelets are however more efficiently computable, since they are based on a fixed mathematical structure (see Appendix B).

In addition to these frameworks, applications of graph signal processing include edge-aware image processing [36], depth video coding [37], image compression [38], anomaly detection in wireless sensor networks [39], bridge structure health monitoring [40], brain functional connectivity analysis [41] and mobility pattern prediction [42]. To the best of our knowledge, at time of publication, our conference paper [43] was the first application of GSP to human action recognition. Since then, some related work has emerged in this direction [57].

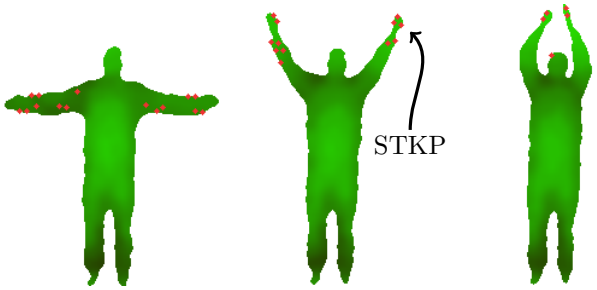
3. Representing Actions as Graphs

3.1. Outline

In this section, we will discuss about how to represent actions as graphs, and consider two different view-invariant candidates for explicit graph construction. The first candidate is based on tracked skeleton joints [2], while the second variant is based on spatio-temporal keypoints [21] (see Fig. 1).



(a) Skeleton joints



(b) Spatio-temporal keypoints

Figure 1: Interest point types used as basis for graph construction. The detected interest points (red) for three frames are here shown for the point cloud of the "*two hand wave*" action. The skeleton joints have the benefit of their spatial locations having a semantic meaning, while the keypoints are mostly detected on locations describing fine intrinsic detail about the spatio-temporal shape of the point cloud.

Graphs based on skeleton joints capture the spatial pose of the human body, which is suitable for representing actions that are defined by larger general limb movements, where the semantic knowledge of body part positions is vital for recognition.

Spatio-temporal keypoints, on the other hand, capture complementary detailed information directly from the point cloud. Each keypoint describes the spatio-temporal shape of a point cloud, and is thus able to capture fine intrinsic detail, while also being robust against noisy skeleton estimates, which can be caused by complex poses.

The graph constructions in this section focus on the part-wise interactions within a single frame; how these graphs will be used to create a feature for action recognition will be discussed in Sec. 4.

3.2. Motivation

Actions can be defined as a sequences of interactions between parts. Indeed, an early study by Johansson [4] investigated human motion perception from limited information. In his series of experiments, a number of lights were attached to the subject's body, and the 3D human motion perception is evaluated depending on the number of activated lights. [1] The study showed that capturing the interactions between several parts of the human body is

[1] Online at *e.g.* `https://www.youtube.com/watch?v=1F5ICP9SYLU`

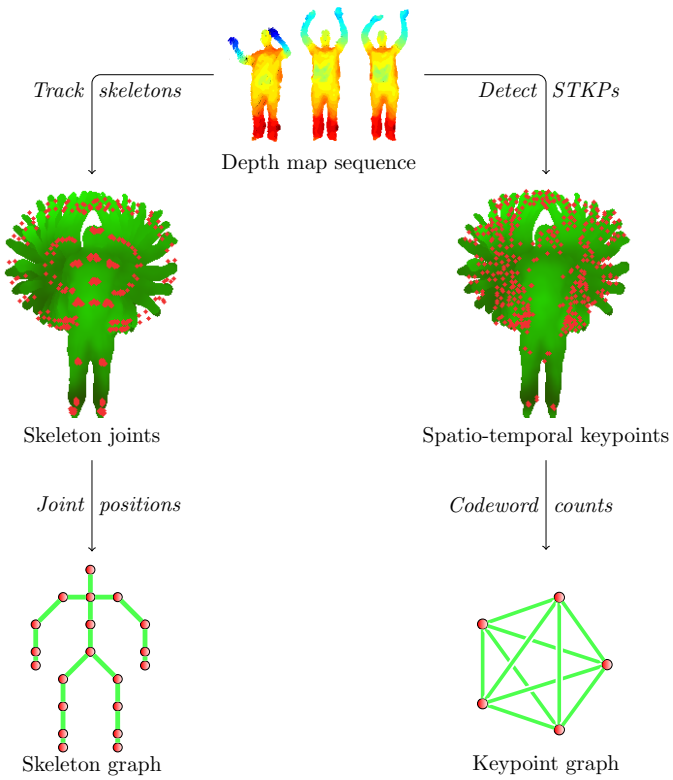


Figure 2: Procedure for graph construction. Given a depth map sequence, we detect interest points, such tracked skeleton joints or spatio-temporal keypoints (STKP). The skeleton-based graph construction uses the relative position of the joints, which yields a graph that follows the human skeleton structure. Keypoint-based graphs, on the other hand, are constructed from the occurrence counts of STKP codewords, and the graph structure is defined by pair-wise similarities of the codewords based on χ^2-distance. Note that in practice, the keypoint graph is not necessarily complete; dissimilar codeword pairs will get edge weights close to zero.

helpful for decreasing the ambiguity between several motion categories. Therefore, we consider using a graph for representing the interactions in order to jointly capture information about the different parts.

Each interactive part, or interest point, can be thought of as a vertex v_i in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$. An edge $e = (v_i, v_j) \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ then captures some relationship between the points using a weight $\mathrm{W}(i,j) \in \mathbb{R}^+$. Further, we assume we have additional information about each point using a D-dimensional feature vector mapped to each vertex.

In the next two sections, we consider two candidate graph representations for actions based on interest points.

3.3. Skeleton-based Graphs

As shown in the study by Johansson [4], the human skeleton joints can be used for discriminating between action categories given that we make use of enough joint positions (the study concluded 10–12 joints to be adequate). Recently, due to the nominal work of Shotton *et al.* [2], we have ready access to tracked skeletons of the human body gotten directly from depth images. Their skeleton tracking algorithm results in $N = 20$ tracked joint positions. In previous action recognition research, these joint positions have been shown to provide useful as a good base feature for building a discriminative feature [6, 8, 50].

The i-th joint at frame t has a 3D position $\mathbf{p}_{t,i} = [\mathrm{x}_i(t), \mathrm{y}_i(t), \mathrm{z}_i(t)]^T$. As body size differs between different human subjects, we use the limb normalization procedure of Zanfir *et al.* [50] for normalizing skeleton limb length, while still keeping limb angles and positions intact. As noted in previous research [6, 8], the relative inter-joint positions give quite discriminative features. As the center hip joint of the tracked 3D skeleton is deemed quite stationary throughout actions, we create a relative position vector

$$\hat{\mathbf{p}}_{t,i} = \mathbf{p}_{t,i} - \mathbf{p}_{t,\text{center hip}} \tag{1}$$

for describing the position of joint i.

3.3.1. Rotation Cancellation

Since depth cameras conform to a Cartesian coordinate system, the relative joint positions are not view-invariant by nature. Therefore, we propose a simple approach to rotate the skeletons in order to bring them into a canonical coordinate system which is independent of the camera angle.

View-invariant action recognition with 3D skeletons has been described previously by *e.g.* Xia *et al.* [22], where spherical histograms are rotated to a canonical view for each frame. Our approach differs in that we achieve rotation normalization using information from the whole action sequence, which increases robustness against tracking errors and non-straight poses. This approach is also taken by Wang *et al.* [23], where a plane is fit using the RANSAC procedure to estimate a rotation matrix. We choose a simpler approach that does not rely on fitting any parameters from data. To the best of our knowledge, we are not aware of any previous work achieving view-invariance in our proposed manner.

Our approach is as follows. We first find a vector pointing upwards (perpendicular to the floor). Note that since the camera view angle cannot be assumed to be planar to the floor (*e.g.* slanted top-down view), the up vector does not necessarily point along the positive y-coordinate of the Cartesian coordinate system. Consequently, we turn to the tracked skeleton information. Since the skeleton joints have a semantic meaning, we can define the up vector candidate for frame t as $\mathbf{v}_{t,\text{up}} = \mathbf{p}_{t,\text{head}} - \mathbf{p}_{t,\text{center hip}}$. Assuming the camera to be static, we then vote for an up vector $\mathbf{v}_{\text{up}}$ representing the whole action sequence by taking the marginal median

$$\mathrm{v}_{\text{up}}(d) = \underset{t \in \{1,\ldots,T\}}{\text{median}} \left\{ \mathrm{v}_{t,\text{up}}(d) \right\}, \ \forall d \ , \tag{2}$$

which is gotten by taking the median for each axis independently. The reason for using the median instead of taking

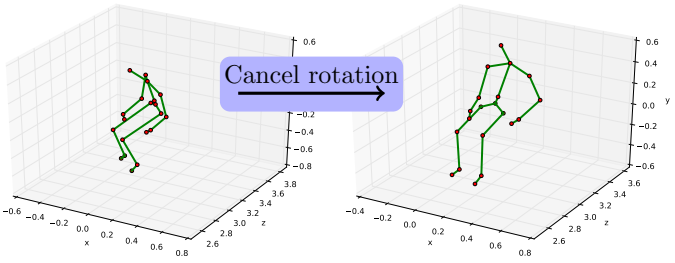


Figure 3: Rotation cancellation using the Gram-Schmidt process for a frame of the "*stand up*" action on the left. The right skeleton shows the resulting canonical view facing the camera at $(x, y, z) = (0, 0, 0)$. See text for details.

the mean is because some candidate up vectors can be regarded as noise due to some frames containing poses where the head-hip vector is not pointing straight up, *e.g.* when bending down. We assume, however, that the majority of the frames in the action sequence feature the subject standing straight up, which should allow the above procedure to yield an up vector estimate close to the ground truth.

Next, we define the vector pointing to the right as $\mathbf{v}_{t,\text{right}} = \mathbf{p}_{t,\text{right hip}} - \mathbf{p}_{t,\text{left hip}}$. Our goal is to find a rotation matrix so that we can put the skeleton pose into a canonical view. Since $\mathbf{v}_{\text{up}}$ and $\mathbf{v}_{t,\text{right}}$ are not orthonormal, they cannot directly be used for rotation. To remedy this, we employ the Gram-Schmidt orthonormalization process [58] in order to create a rotation matrix $\mathbf{R} = [\mathbf{r}_x, \mathbf{r}_y, \mathbf{r}_z]$, where

$$\mathbf{u}_y = \mathbf{v}_{\text{up}} \ , \tag{3}$$

$$\mathbf{r}_y = \mathbf{u}_y / \|\mathbf{u}_y\| \ , \tag{4}$$

$$\mathbf{u}_x = \mathbf{v}_{t,\text{right}} - \frac{\mathbf{v}_{t,\text{right}}^T \mathbf{u}_y}{\langle \mathbf{u}_y, \mathbf{u}_y \rangle} \mathbf{u}_y \ , \tag{5}$$

$$\mathbf{r}_x = \mathbf{u}_x / \|\mathbf{u}_x\| \ , \tag{6}$$

$$\mathbf{r}_z = \mathbf{r}_x \times \mathbf{r}_y \ . \tag{7}$$

The orthogonal matrix $\mathbf{R}$ satisfies $\mathbf{R}^{-1} = \mathbf{R}^T$ and thus we can cancel the camera angle and create a view-invariant relative position vector $\hat{\mathbf{p}}_{t,i} = \mathbf{R}^T(\mathbf{p}_{t,i} - \mathbf{p}_{t,\text{center hip}})$ for describing the position of joint i independent of the camera angle. An example of rotation cancellation can be seen in Fig. 3. As we will see in our experiments, this rotation is crucial for achieving good performance in recognizing actions across different views. Naturally, for non-cross-view recognition tasks, we can choose to not apply the rotation $\mathbf{R}^{-1}$.

3.3.2. Graph Construction

Given a set of N tracked joints, we seek to construct a graph and an associated signal that describes the subject pose at a time t. Note that each tracked skeleton itself can be viewed as a graph $\mathcal{G}_{\text{skel}} = (\mathcal{V}_{\text{skel}}, \mathcal{E}_{\text{skel}})$ in 3D space (see Fig. 2). We thus proceed and create a graph with N vertices, where each vertex corresponds to a skeleton joint.

We assume that a signal along an edge provides relevant information inversely proportional to the distance between a pair of joints. Edge weights are therefore set by a radial basis function

$$\mathrm{W}(i, j) = \exp\left(-\frac{\|\hat{\mathbf{p}}_{t,i} - \hat{\mathbf{p}}_{t,j}\|_2^2}{2\sigma^2}\right) \tag{8}$$

for neighboring joints $(v_i, v_j) \in \mathcal{E}_{\text{skel}}$, which gives spatially closer joints a higher weight. We assume that σ is not equal for all connected joint pairs in the skeleton, and therefore define a pair-specific $\sigma = \{\frac{1}{3}\sum_a \sigma_{i,j}(a)\}$, where $\boldsymbol{\sigma}_{i,j} \in \mathbb{R}^3$ is a vector describing the axis-wise standard deviation between joints i and j.

Note that we only connect edges corresponding to neighbors in the natural human body structure. This is because due to natural physical constraints of human limbs, any signal defined on *e.g.* the subject's hand will be correlated with the signal on the elbow. Using a fully connected graph is not appropriate, as this would imply that there is strong correlation between your hand and foot, which does not usually hold for human motions, and would only introduce noise into any subsequent feature extracted from the graph.

The feature vector associated with each vertex v_i is set to be the relative position vector $\hat{\mathbf{p}}_{t,i}$, with optional rotation cancellation.

3.4. Keypoint-based Graphs

Although graphs obtained through skeleton tracking are easily constructible, skeleton joint positions have several flaws:

- Abundant noise due to complex poses.
- Inability to capture fine intrinsic details, such as human-object interaction and hand shapes.

Therefore, we consider an alternative graph construction gotten directly from the 3D point cloud in the depth image sequence. Care must however be taken when considering the size of the graph to be created. Actions of interactive nature are typically a few seconds of length, which corresponds to a spatio-temporal point cloud containing the order of 10^6 points. Clearly, using each point as a vertex will create graphs of intractable size. Furthermore, not each point is relevant for action recognition. We therefore propose to detect a set of keypoints in locations that are of interest for describing actions.

3.4.1. Spatio-temporal Keypoints

We use the recently proposed spatio-temporal keypoint (STKP) detector by Rahmani *et al.* [21]. These keypoints have several desirable properties, including detection repeatability, which means that the keypoints can be detected in different samples of the same action sequence despite noise. The keypoints also have a unique coordinate

basis, which allows them to create a view-invariant description of the point cloud. Finally, the keypoints are localized spatio-temporally, which means that they are mainly detected at spatio-temporal locations where the actual action is being performed (see Fig. 1).

For detecting keypoints, we proceed in line with previous work and calculate histograms of oriented principal components (HOPC) [21] for each point $\mathbf{p} \in \mathbb{R}^3$ at time t in the depth map sequence. Each HOPC describes the principal axis distribution of the variance of all points within the spatio-temporal support volume $\Omega(\mathbf{p})$, which is a set containing all points within radius r and time interval $[t-\tau, t+\tau]$ of the point $\mathbf{p}$, where τ is a parameter. The smallest principal axis is the least squares estimate for the surface normal, which renders HOPC more robust against noise than surface-normal methods based on depth gradients [5].

Subsequently, two HOPC descriptors are calculated for each point: $\mathbf{h}_s$ using the spatial support volume (*i.e.* $\tau = 0$), and $\mathbf{h}_{st}$ using the spatio-temporal support volume. Points are then pruned based on low eigenratios in order to discard candidates whose support volume is symmetrical along any pair of axes. The candidate keypoints are then sorted according to their χ^2 distance between $\mathbf{h}_s$ and $\mathbf{h}_{st}$ in descending order and a set of L keypoints are selected. Non-maximum suppression is also performed, by discarding points that are within a radius $\sigma_r r$ and time interval $\sigma_\tau \tau$ of a keypoint, where σ_r, σ_τ are parameters. The resulting STKPs have non-ambiguous eigensystems, so any surrounding point cloud can be aligned with the STKP axes. Consequently, any subsequent feature computed from the rotated points will be view-invariant.

3.4.2. Graph Construction

Given a set of detected keypoints, we seek to construct a graph and an associated signal that describes the spatio-temporal shape of the point cloud at a time t. We proceed to construct our graph as follows (see also Fig. 2). A codebook with N codewords is created using K-means clustering of the detected STKPs. Each STKP is then assigned to its closest codeword and a BoW representation is used for representing the STKPs in each frame. We then create a graph with N vertices, where each vertex corresponds to a codebook vector. Edge weights are set using a χ^2 kernel on a pair of vertices, assuming the signal to be correlated amongst keypoints of approximately similar shapes:

$$\mathrm{W}(i,j) = \exp\left(-\sum_d \frac{(\mathrm{h}_i(d) - \mathrm{h}_j(d))^2}{\mathrm{h}_i(d) + \mathrm{h}_j(d)}\right) , \qquad (9)$$

where $\mathbf{h}_i$ is the spatio-temporal HOPC descriptor of the keypoint represented by vertex i.

Note that unlike the skeleton-based graph, we can assume correlation between STKPs of close χ^2 distance, as they will describe a similar spatio-temporal shape. This means that we can use a (weighted) fully connected graph for describing the relationships between the keypoints.

The feature vector associated with each vertex v_i is set to be the codeword occurrence count for the codeword represented by v_i in the current video frame, which does not affect the view-invariant property of the STKPs.

4. Spectral Graph Sequences (SGS)

4.1. Outline

In this section, we present our feature descriptor (SGS) for temporal sequences of graphs based on the spectral graph wavelet transform (SGWT) [31]. The overview of our method can be seen in Fig. 4. We assume that we have sequence of graphs created from interest points in a depth video (gotten using *e.g.* the Kinect).

Our system consists of five parts. First, we design an augmented graph by connecting together a sequence of graphs using temporal edges. Each graph in the sequence describes the point cloud in a single frame using either skeleton-based or keypoint-based graphs, as previously discussed in Sec. 3. Second, spectral graph wavelet coefficients are calculated using the SGWT. The coefficients capture second order gradient information about the graph signal along both temporal and local edge directions. Third, in order to cope with varying action sequence length, we leverage a temporal pyramid pooling scheme. The pooling operator aggregates information about the wavelet coefficients, while the pyramid structure allows us to capture the temporal order of the graph signal propagation. Fourth, we reduce the dimensionality of the feature vector using PCA and apply a standard SVM for classification. Finally, using late fusion of SVM decision functions, we can also combine the complementary effects of several graph types.

In addition, the end of this section presents some analysis of the interpretation and effects of the proposed feature descriptor.

4.2. Graph Design

We consider a temporal sequence of T graph signals $\mathbf{f}_1, \ldots, \mathbf{f}_T$, all of which are embedded on a common graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ with $|\mathcal{V}| = N$. Our goal is to create a descriptor of the temporal propagation of the signals. Each vertex v is associated with a D-dimensional feature vector. For comparing graphs, while previous work has employed graph kernels for creating an implicit vector mapping into a reproducing kernel Hilbert space [25–29], we instead look at the propagation of signals defined on the vertices of the graph, as will be explained in the following. As graph signals are scalars by definition [24], we subsequently process each axis of the D-dimensional space separately, by the graph signal $f : \mathcal{V} \to \mathbb{R}$. We proceed to create an augmented graph $\mathcal{G}_{\text{aug}} = (\mathcal{V}_{\text{aug}}, \mathcal{E}_{\text{aug}}, \mathbf{W}_{\text{aug}})$ by stacking T copies of $\mathcal{G}$ to create a sequence of graphs $\mathcal{G}_1, \ldots, \mathcal{G}_T$. The choice of the graph $\mathcal{G}$ can be *e.g.* one of the two we previously discussed in Sec. 3. Since we assume the signals to be embedded on a common graph $\mathcal{G}$, we let

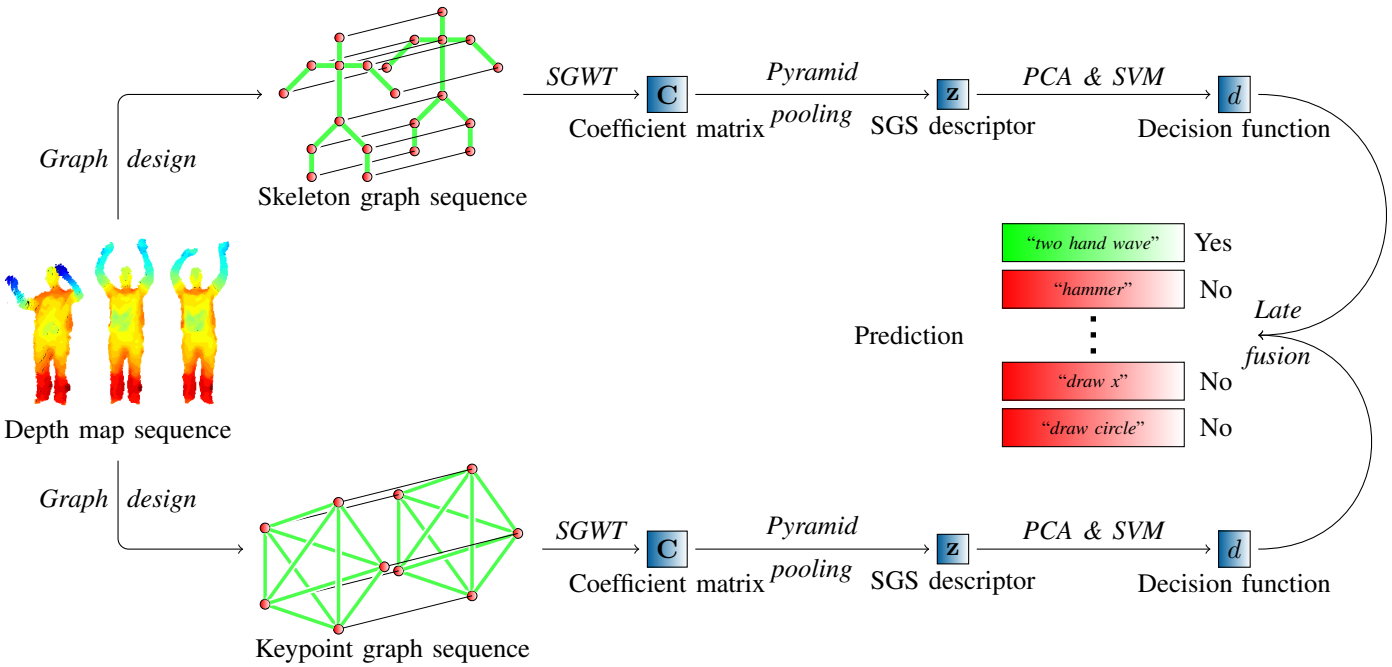


Figure 4: Overview of the proposed action recognition system. Given an input depth map sequence, SGS descriptors based on graphs from both skeletons and keypoints are calculated. An SVM is trained for each descriptor and their decision functions are finally combined using late fusion. Note that the skeleton graph in this figure is simplified for the purpose of illustration, and thus has fewer than the 20 joints given by Shotton *et al.* [2].

v_{i_t} denote the i-th vertex in the t-th copy of the graph. The signal associated with vertex v_{i_t} is therefore given by $\mathrm{f}_t(i)$. We then connect each vertex v_{i_t} in frame t with its temporally equivalent vertices $v_{i_{t-1}}$, $v_{i_{t+1}}$ corresponding to the same vertex in the previous and next frame, respectively, creating temporal edges. Each graph in the sequence already has pre-defined local edge weights, set by some distance kernel $\exp(-\mathrm{dist}(v_{i_t}, v_{j_t}))$ (see Sec. 3). We assume strong signal correlation across the temporal direction ($\exp(-\mathrm{dist}(v_{i_t}, v_{i_{t+1}})) \approx 1$), so we set temporal edge weights to unity. The augmented weight matrix $\mathbf{W}_{\mathrm{aug}}$ therefore has the following fixed sparse block structure:

$$\mathbf{W}_{\mathrm{aug}} = \begin{pmatrix} \mathbf{W} & \mathbf{I} & & & & \\ \mathbf{I} & \mathbf{W} & \mathbf{I} & & & \\ & \ddots & \ddots & \ddots & & \\ & & \mathbf{I} & \mathbf{W} & \mathbf{I} & \\ & & & \mathbf{I} & \mathbf{W} \end{pmatrix}, \tag{10}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{W}$ is the local edge weight matrix, which is similar for all graphs in the sequence.

We have two edge categories:

Temporal
: These edges capture the propagation of the graph signal between consequent graphs.

Local
: These edges connect vertices within one single graph and capture pair-wise interactions within a frame.

The structure of $\mathcal{G}_{\mathrm{aug}}$ now allows us to analyze the temporal propagation of the signal.

4.3. Spectral Graph Wavelet Transform

We seek a multi-scale decomposition of the graph signal in order to capture information about the signal propagation with respect to the graph structure. For this, we turn to the SGWT framework of Hammond *et al.* [31], which is a generalization of classical wavelet transforms onto arbitrary graphs. Given a kernel $g : \mathbb{R}^+ \to \mathbb{R}^+$ acting as a band-pass filter, a spectral graph wavelet $\boldsymbol{\psi}_{t,n} \in \mathbb{R}^N$ at scale t localized around vertex n can be written explicitly as a vector [31]

$$\psi_{t,n}(m) = \sum_{\ell=0}^{N-1} g(t\lambda_\ell)\mathrm{u}_\ell(n)\mathrm{u}_\ell(m) , \tag{11}$$

where $\{\lambda_\ell, \mathbf{u}_\ell\}_{\ell=0,\ldots,N-1}$ denote the eigenvalue and eigenvector pairs of the graph Laplacian, which is used for representing a graph as a matrix. Details about the graph Laplacian and its eigenvectors are deferred to Appendix A. Given a graph signal $\mathbf{f}$, SGWT coefficients are extracted by the matrix-vector multiplication $\boldsymbol{\Psi}_{t_j}^T \mathbf{f}$, where $\boldsymbol{\Psi}_{t_j} = [\boldsymbol{\psi}_{t_j,1}, \ldots, \boldsymbol{\psi}_{t_j,|\mathcal{V}|}]$. Details are deferred to Appendix B.

We proceed to create the normalized graph Laplacian matrix $\boldsymbol{\mathcal{L}}_{\mathrm{aug}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}_{\mathrm{aug}}\mathbf{D}^{-1/2}$, where $\mathbf{D} = \mathrm{diag}(\mathbf{W}_{\mathrm{aug}}\mathbf{1})$ and $\mathbf{1}$ is the vector of all ones. The matrix $\boldsymbol{\mathcal{L}}_{\mathrm{aug}}$ has an eigenbasis that conforms to harmonic analysis of graph signals [24], and has a maximum

eigenvalue $\lambda_{\max} = 2$ [59]. Hammond *et al.* [31] presented a fast approximation of the SGWT based on truncated Chebyshev polynomial series [60], which avoids expensive eigendecomposition of $\boldsymbol{\mathcal{L}}_{\text{aug}}$ (see Appendix C). Using this approximation, we extract wavelet and scaling coefficients from $\mathcal{G}_{\text{aug}}$ at each vertex n and scale t_j. The wavelet scales t_j are chosen to be a set of J values logarithmically equispaced in the range $[0, \lambda_{\max}]$. We acquire the transform coefficients by calculating the approximate SGWT [31]

$$\boldsymbol{\Psi}_{t_j}^T \mathbf{f}_{\text{aug}} \approx \frac{1}{2} c_{j,0} \mathbf{f}_{\text{aug}} + \sum_{k=1}^{M_j} c_{j,k} \overline{\mathbf{T}}_k(\boldsymbol{\mathcal{L}}_{\text{aug}}) \mathbf{f}_{\text{aug}} \ , \qquad (12)$$

where $\mathbf{f}_{\text{aug}} = [\mathbf{f}_1; \ldots; \mathbf{f}_T] \in \mathbb{R}^{NT}$ is the vector of stacked graph signals representing the human action sequence, and $\overline{\mathbf{T}}_k(\boldsymbol{\mathcal{L}}_{\text{aug}})$, $c_{j,k}$ are the Chebyshev polynomials and coefficients, respectively. The scaling vector coefficients $\boldsymbol{\Phi}^T \mathbf{f}_{\text{aug}}$, which capture low-frequency information of the signal, are calculated in a similar way. Consequently, each vertex will result in $J + 1$ coefficients per axis, one for each wavelet scale (including the scaling kernel). The coefficients capture information about localized frequencies of the graph signal that follow the graph structure. We store the coefficients in a matrix $\mathbf{C} \in \mathbb{R}^{T \times DN(J+1)}$, shaped so that the coefficient $\boldsymbol{\psi}_{t_j,n}^T \mathbf{f}_{\text{aug}}$ is stored on the t-th row.

Note that in order for the coefficients to capture similar information across graphs created from different action sequences of the same class, we require the graph Laplacian basis to be constant (*i.e.* temporal edge weights are fixed). We assume that the only varying quantity is the graph signal. Variable temporal edge weights are not supported by the proposed formulation of the method, and investigation of this matter is left as an open problem.

4.3.1. Efficient algorithm

The fast approximate SGWT accesses the graph Laplacian matrix $\boldsymbol{\mathcal{L}}$ only through matrix-vector multiplications, which takes $\mathcal{O}(|\mathcal{E}| + J|\mathcal{V}|)$ time for J scales and is fast if $\boldsymbol{\mathcal{L}}$ is sparse [31]. Since our graph has a special sparsity structure (10), we can further optimize this step. Fig. 5 presents the algorithm for calculating the transform. Calculating the matrix-vector multiplication $(\boldsymbol{\mathcal{L}} - \mathbf{I})\overline{\boldsymbol{\tau}}_{k-1} = (\boldsymbol{\mathcal{L}} - \mathbf{I})\overline{\mathbf{T}}_{k-1}(\boldsymbol{\mathcal{L}})\mathbf{f}_{\text{aug}}$ is by far the most expensive operation in the SGWT approximation, which is done $\max_j M_j$ times during the course of the algorithm.

We propose to modify two steps of the SGWT approximation algorithm. While the standard fast approximate SGWT requires explicit construction of $\boldsymbol{\mathcal{L}}$ to calculate $(\boldsymbol{\mathcal{L}} - \mathbf{I})\overline{\boldsymbol{\tau}}_{k-1}$ in step 3 and 9, we propose to exploit the explicit sparsity structure of our graph to avoid unnecessary memory usage by using the algorithm in Fig. 6. The algorithm can be derived by explicitly expanding said matrix multiplication and noting that for each $\mathbf{f}_t$ in $\mathbf{f}_{\text{aug}}$, the transform always depends at most on $\mathbf{f}_{t-1}$ and $\mathbf{f}_{t+1}$ in the neighboring frames. Our algorithm avoids explicit construction of $\boldsymbol{\mathcal{L}}$ by carrying out the calculations implicitly using only the weight matrix $\mathbf{W}$.

Require: $\mathbf{f}$: Graph signal; $\mathbf{f} = [\mathbf{f}_1; \ldots; \mathbf{f}_T]$
Ensure: $\mathbf{C}$: Approximated wavelet coefficients
1: **function** FastSGWT($\mathbf{f}$)
2: $\quad\overline{\boldsymbol{\tau}}_0 \leftarrow \mathbf{f}$
3: $\quad$Calc. $\mathbf{v} \leftarrow (\boldsymbol{\mathcal{L}} - \mathbf{I})\mathbf{f}$ using alg. in Fig. 6.
4: $\quad\overline{\boldsymbol{\tau}}_1 \leftarrow \mathbf{v}$
5: $\quad$**for** $j = 0, \ldots, J$ **do**
6: $\quad\quad\mathbf{r}_j \leftarrow \frac{1}{2} c_{j,0} \overline{\boldsymbol{\tau}}_0 + c_{j,1} \overline{\boldsymbol{\tau}}_1$
7: $\quad$**end for**
8: $\quad$**for** $k = 2, \ldots, \max_j M_j$ **do**
9: $\quad\quad$Calc. $\mathbf{v} \leftarrow (\boldsymbol{\mathcal{L}} - \mathbf{I})\overline{\boldsymbol{\tau}}_{k-1}$ using alg. in Fig. 6.
10: $\quad\quad\overline{\boldsymbol{\tau}}_k \leftarrow 2\mathbf{v} - \overline{\boldsymbol{\tau}}_{k-2}$
11: $\quad\quad$**for** $j = 0, \ldots, J$ **do**
12: $\quad\quad\quad$**if** $M_j \geq k$ **then**
13: $\quad\quad\quad\quad\mathbf{r}_j \leftarrow \mathbf{r}_j + c_{j,k} \overline{\boldsymbol{\tau}}_k$
14: $\quad\quad\quad$**end if**
15: $\quad\quad$**end for**
16: $\quad$**end for**
17: $\quad\mathbf{R} \leftarrow [\mathbf{r}_0, \ldots, \mathbf{r}_J]$
18: $\quad\mathbf{C} \leftarrow \mathbf{R}$, reshaped to store $\boldsymbol{\psi}_{t_j,n}^T \mathbf{f}$ on row t.
19: $\quad$**return C**
20: **end function**

Figure 5: Fast SGWT approximation with incorporated efficient matrix-vector multiplication steps on lines 3 and 9.

While the naïve algorithm requires $\mathcal{O}(N^2 T)$ memory, our efficient matrix-vector algorithm ensures that we need only $\mathcal{O}(N^2)$ memory for calculating the transform, which can result in a memory usage difference crucial for being able to carry out the transform for long action sequences. Furthermore, due to reduced requirements on memory bandwidth, we have found the algorithm to also be computationally faster than the conventional SGWT approximation algorithm in practice.

4.4. Pyramid Pooling

Since wavelets have zero mean, taking the average of the coefficients does not yield any information [61]. However, by applying a non-linearity (taking the absolute value) and then taking the mean, we can retain some useful information about the signal. In order to cope with varying action sequence length, we leverage a vector-valued pooling function $p : \mathbb{R}^{t \times DN(J+1)} \to \mathbb{R}^{DN(J+1)}$ to create a feature vector $\mathbf{z} = p(\mathbf{C})$, where t is equal to the input matrix row count. The pooling function can for example be chosen as to do either absolute max or mean pooling along the temporal axis as

$$p_{\max}(\mathbf{C}) = \left[\max_t |\mathrm{C}(t,i)| \right]_{i=1,\ldots,DN(J+1)} , \qquad (13)$$

$$p_{\text{mean}}(\mathbf{C}) = \left[\frac{1}{T} \sum_{t=1}^{T} |\mathrm{C}(t,i)| \right]_{i=1,\ldots,DN(J+1)} . \qquad (14)$$

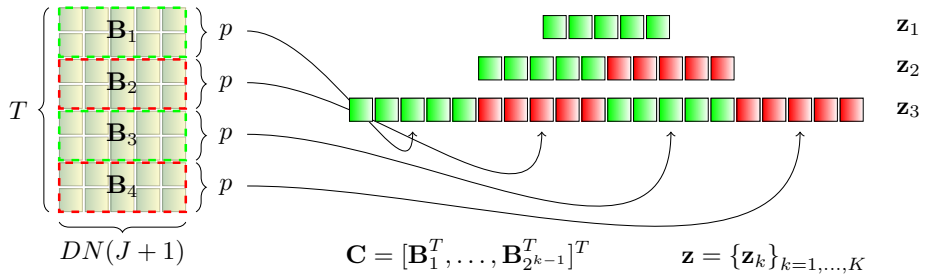


Figure 7: Temporal pyramid pooling. Coefficient matrix $\mathbf{C}$ from a T frames long action sequence is pooled by a function $p : \mathbb{R}^{t \times DN(J+1)} \rightarrow \mathbb{R}^{DN(J+1)}$ into $K = 3$ pyramid levels. The arrows illustrate the creation of the level 3 pyramid level vector $\mathbf{z}_3$. The final feature vector $\mathbf{z}$ is given by concatenation of the pyramid level vectors $\{\mathbf{z}_k\}_{k=1,\ldots,K}$.

Require: $\mathbf{f}$: Vector to multiply with; $\mathbf{f} = [\mathbf{f}_1; \ldots; \mathbf{f}_T]$
Require: $\mathbf{W}$: Weight matrix for local edges
Ensure: $\mathbf{r} = (\boldsymbol{\mathcal{L}} - \mathbf{I})\mathbf{f}$

function EFFICIENTMATVEC($\mathbf{f}, \mathbf{W}$)
  Define $\mathbf{x}_k : x_k(i) = \sqrt{\sum_j \mathrm{W}(i,j) + k}$
  Define $\mathbf{X}_k : \mathbf{W} \oslash (\mathbf{x}_k \mathbf{x}_k^T)$
  Define $\mathbf{y}_{ab} : (\mathbf{x}_a \odot \mathbf{x}_b)$
  if $T = 1$ **then**
    $\mathbf{r}_1 \leftarrow -\mathbf{X}_0 \mathbf{f}_1$
  else if T=2 **then**
    $\mathbf{r}_1 \leftarrow -\mathbf{X}_1 \mathbf{f}_1 - \mathbf{f}_2 \oslash \mathbf{y}_{11}$
    $\mathbf{r}_T \leftarrow -\mathbf{X}_1 \mathbf{f}_T - \mathbf{f}_{T-1} \oslash \mathbf{y}_{11}$
  else
    $\mathbf{r}_1 \leftarrow -\mathbf{X}_1 \mathbf{f}_1 - \mathbf{f}_2 \oslash \mathbf{y}_{12}$
    $\mathbf{r}_T \leftarrow -\mathbf{X}_1 \mathbf{f}_T - \mathbf{f}_{T-1} \oslash \mathbf{y}_{12}$
    if T=3 **then**
      $\mathbf{r}_2 \leftarrow -\mathbf{X}_2 \mathbf{f}_2 - (\mathbf{f}_1 + \mathbf{f}_T) \oslash \mathbf{y}_{12}$
    else
      $\mathbf{r}_2 \leftarrow -\mathbf{X}_2 \mathbf{f}_2 - \mathbf{f}_1 \oslash \mathbf{y}_{12} - \mathbf{f}_{T-1} \oslash \mathbf{y}_{22}$
      $\mathbf{r}_{T-1} \leftarrow -\mathbf{X}_2 \mathbf{f}_{T-1} - \mathbf{f}_{T-2} \oslash \mathbf{y}_{22} - \mathbf{f}_T \oslash \mathbf{y}_{12}$
    end if
  end if
  for $t = 3, \ldots, T-2$ **do**
    $\mathbf{r}_t \leftarrow -\mathbf{X}_2 \mathbf{f}_t - (\mathbf{f}_{t-1} + \mathbf{f}_{t+1}) \oslash \mathbf{y}_{22}$
  end for
  $\mathbf{r} \leftarrow [\mathbf{r}_1; \ldots; \mathbf{r}_T]$
  return $\mathbf{r}$
end function

Figure 6: Efficient matrix-vector multiplication algorithm for our graph. The operators $\odot$ and $\oslash$ denote element-wise multiplication and division, respectively.

In the case of absolute mean pooling, the resulting feature will encode the average second order gradient of the graph signal for each dimension and vertex, windowed by SGWT kernels.

Similar to previous research [6, 8, 46], we create a temporal pyramid of coefficients for capturing the temporal order of actions. Let K denote the maximum pyramid level. Then, the pooled feature vector at pyramid level $k \leq K$ is defined as $\mathbf{z}_k = [p(\mathbf{B}_1)^T, \ldots, p(\mathbf{B}_{2^{k-1}})^T]^T$, where $\{\mathbf{B}_i\}$ is a set of non-intersecting block matrices dividing $\mathbf{C}$ uniformly so that $\mathbf{C} = [\mathbf{B}_1^T, \ldots, \mathbf{B}_{2^{k-1}}^T]^T$. The final feature vector $\mathbf{z}$ is then a concatenation of the pyramid level vectors $\{\mathbf{z}_k\}_{k=1,\ldots,K}$. A visual explanation of the temporal pyramid pooling scheme applied to $\mathbf{C}$ can be seen in Fig. 7.

4.5. PCA & SVM

Most natural signals are sparse [62]. If our graph signal between each time step varies only sparsely (*i.e.* only a subset of the vertices have changed signal value), then most elements of $\mathbf{z}$ will become close to zero. We therefore reduce the $(2^K - 1)N(J+1)$-dimensional $\mathbf{z}$ using PCA. After applying PCA to $\mathbf{z}$, we ℓ^2-normalize and finally classify each action using a standard SVM.

4.6. Late Fusion

We can combine the descriptors from skeleton-based and keypoint-based graphs. Here, we consider late fusion by averaging the output from the SVM decision functions d for an input depth map $\mathbf{x}$ as

$$f(\mathbf{x}) = \frac{1}{2} d_{\text{skeleton}}(\mathbf{x}) + \frac{1}{2} d_{\text{keypoint}}(\mathbf{x}) \ . \qquad (15)$$

We take care to normalize each decision function so that it has unit variance, in order to give the contributions from each feature equal weight. While a convex combination of the decision functions could be explored [63], we have found simple averaging to yield good enough results.

4.7. Feature Vector

The resulting feature descriptor encodes the spectral content of a sequence of graphs, so we name it *spectral graph sequences* (SGS). As computing the SGWT approximation [31] in $\mathcal{O}(|\mathcal{E}| + J|\mathcal{V}|)$ time is the most costly part of the descriptor creation process, we have that for one action sequence, the descriptor is computable in $\mathcal{O}(TN)$ time, treating parameters K, J constant. Therefore, when N is small, such as for skeleton-based graphs, our descriptor becomes essentially linear in the action sequence length, and more computationally efficient than approaches that rely on solving heavy optimization problems [6, 7].

4.8. Ring Structure

We note that if we apply our method to a task where the start and end position of the sequence tends to be the same, then we can additionally connect the first graph in the sequence together with the first one, in order to create a ring structure. Indeed, it has been shown that the graph Fourier transform on the graph Laplacian of a 1D ring graph produces an eigenbasis equal to the basis of the discrete Fourier transform on the real line [64]. In our experiments, we have found ring structure to help on applicable datasets. In such conditions, the ring structure will not hinder signal smoothness, and may actually provide additional information to the graph. Consider the case where the start and end poses are identical. The graph signal on such a graph with a ring structure will then be blind to where the action sequence starts and ends [2]. Thus, the ring structure is able to help with localization of the action in the graph, as the action descriptor will no longer be biased on the action starting on any specific frame in the time sequence. Note that if a ring structure is used, then the algorithm in Fig. 6 needs to be modified accordingly to ensure that the exactness of the calculation of $(\boldsymbol{\mathcal{L}}-\mathbf{I})\overline{\boldsymbol{\tau}}_{k-1}$ still holds true.

4.9. Interpretation and Effect of Coefficients and Edges

Unlike methods based on *e.g.* sparse coding [6] or deep learning [16], our method allows for some direct interpretation of the effects of the descriptor. In this subsection, we present some insights about the information captured by the SGWT coefficients and the edges of our graph structure.

The graph Laplacian is a difference operator that satisfies [24]

$$(\boldsymbol{\mathcal{L}}\mathbf{f})(i) = \frac{1}{\sqrt{d_i}} \sum_{(v_j,v_i)\in\mathcal{E}} \mathrm{W}(i,j) \left(\frac{\mathrm{f}(i)}{\sqrt{d_i}} - \frac{\mathrm{f}(j)}{\sqrt{d_j}} \right) . \quad (16)$$

For our graph structure (10), this becomes

$$(\boldsymbol{\mathcal{L}}\mathbf{f})(i_t) = \frac{1}{\sqrt{d_{i_t}}} \left(2\frac{\mathrm{f}(i_t)}{\sqrt{d_{i_t}}} - \frac{\mathrm{f}(i_{t-1})}{\sqrt{d_{i_{t-1}}}} - \frac{\mathrm{f}(i_{t+1})}{\sqrt{d_{i_{t+1}}}} \right) + \frac{1}{\sqrt{d_{i_t}}} \sum_{(v_{j_t},v_{i_t})\in\mathcal{E}} \exp(-\mathrm{dist}(v_{i_t},v_{j_t})) \left(\frac{\mathrm{f}(i_t)}{\sqrt{d_{i_t}}} - \frac{\mathrm{f}(j_t)}{\sqrt{d_{j_t}}} \right) , \quad (17)$$

where d_{i_t} denotes the degree of vertex v_{i_t}. This is the temporal second order gradient of the signal plus the second order gradient between neighboring interest points within the same frame, which we can denote as

$$(\boldsymbol{\mathcal{L}}\mathbf{f})(i_t) = \Delta^2_{\mathrm{temporal}}(i_t) + \Delta^2_{\mathrm{local}}(i_t) . \quad (18)$$

[2] We never assume any explicit order on the vertices of a graph.

Therefore, since the SGWT is essentially a frequency-modulated graph Laplacian matrix acting as an operator on a signal [31], the SGWT coefficients capture second order information about the propagation of the signal. Another property of the SGWT coefficients can be seen by noting that due to (11), the coefficient $\boldsymbol{\psi}^T_{t_j,i}\mathbf{f}$ at scale t_j and vertex i satisfies

$$\boldsymbol{\psi}^T_{t_j,i}\mathbf{f} = \sum_{\ell=0}^{N-1} g(t_j\lambda_\ell) \sum_j \mathrm{u}_\ell(i)\mathrm{u}_\ell(j)\mathrm{f}(j) \quad (19)$$

$$= \sum_j \mathrm{f}(j) \sum_{\ell=0}^{N-1} g(t_j\lambda_\ell)\mathrm{u}_\ell(i)\mathrm{u}_\ell(j) . \quad (20)$$

Following Shuman *et al.* [24], if we assume that g is an order K polynomial $g(\lambda) = \sum_{k=0}^K c_k\lambda^k$, we can simplify (20) to

$$\sum_j \mathrm{f}(j) \sum_{k=0}^K c_k(\boldsymbol{\mathcal{L}}^k)(i,j) . \quad (21)$$

Inserting (18) then leads to

$$c_0\mathrm{f}(i) + \sum_{k=1}^K c_k\boldsymbol{\mathcal{L}}^{k-1}(\boldsymbol{\Delta}^2_{\mathrm{temporal}} + \boldsymbol{\Delta}^2_{\mathrm{local}})(i) = \hat{\Delta}^2_{\mathrm{temporal}}(i) + \hat{\Delta}^2_{\mathrm{local}}(i) , \quad (22)$$

where $\hat{\Delta}^2(i) = \frac{1}{2}c_0\mathrm{f}(i) + \sum_{k=1}^K c_k(\boldsymbol{\mathcal{L}}^{k-1}\boldsymbol{\Delta}^2)(i)$ denotes the second order gradient modulated by the kernel $g(t_j\cdot)$, That is, the SGWT coefficients depend on vertices j in a weighted K-hop neighborhood of vertex i, since $(\boldsymbol{\mathcal{L}}^k)(i,j) = 0$ if i,j are more than K hops apart [31]. If $(\boldsymbol{\mathcal{L}}^k)(i,j) \neq 0$, then there exists an s-length path $v_i, v_{p_1}, v_{p_2}, \ldots, v_{p_{s-1}}, v_j$ between the vertices, weighted by the edges along the path, which can be seen by explicit expansion of the matrix power [31]. Using the approximate SGWT, the wavelet kernel g becomes exactly an order K polynomial and the coefficients c_k are gotten from the K-degree Chebyshev approximation (see Appendix C).

Consequently, doing absolute mean pooling on the SGWT coefficients captures the quantity

$$\frac{1}{T}\sum_t |\hat{\Delta}^2_{\mathrm{temporal}}(i_t) + \hat{\Delta}^2_{\mathrm{local}}(i_t)| \leq \frac{1}{T}\sum_t \left(|\hat{\Delta}^2_{\mathrm{temporal}}(i_t)| + |\hat{\Delta}^2_{\mathrm{local}}(i_t)| \right) . \quad (23)$$

This shows that our method captures the average second-order information of the graph signal along both temporal and local edges.

Another way of thinking about the interaction between local and temporal edges is to consider a numerical example with the scenario in Fig. 8. In the figure, we have graphs corresponding to a time series of $T = 3$ graph signals of two interest points. First, let us have a look

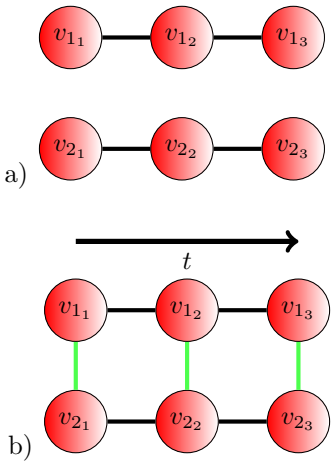


Figure 8: Graphs corresponding to a time sequence of $T = 3$ graph signals. (a) Without local edges. (b) With local edges. See text for details.

at graph (a). Suppose the signal on v_1 increases over time, similar to v_2. An example of such a graph signal is $\mathbf{f}_1 = [1, 2, 4, 1, 2, 4]$, where the three first components belong to the first interest point and the rest belong to the second one. Applying the graph Laplacian matrix $\mathbf{Ł}_{(a)}$ of graph (a) as an operator to this signal reveals the second-order gradient information $\mathbf{Ł}_{(a)}\mathbf{f}_1 = [-1, -1, 2, -1, -1, 2]$. Contrast this with a graph signal where v_2 is constant: $\mathbf{f}_2 = [1, 2, 4, 1, 1, 1]$. Applying the graph Laplacian matrix as an operator to this signal reveals $\mathbf{Ł}_{(a)}\mathbf{f}_2 = [-1, -1, 2, 0, 0, 0]$. Indeed, the information about the acceleration of v_1 is kept intact, despite v_2 not changing.

Consider instead that we would want to know how v_1 and v_2 are changing together in a joint fashion. In graph (b), we have local edges connecting v_1 and v_2. Applying the graph Laplacian to $\mathbf{f}_1$ gives the same result as before, but applying it to $\mathbf{f}_2$ results in $\mathbf{Ł}_{(b)}\mathbf{f}_2 = [-1, 0, 5, 0, -1, -3]$, which shows that the change in v_2 has affected v_1 as well. Therefore, using both local and temporal edges captures more information than just using temporal edges. Of course, using the normalized graph Laplacian $\boldsymbol{\mathcal{L}}$ is possible as well, although the example becomes slightly less pedagogical due to the degree normalization [24].

5. Experiments

In this section, the proposed method is evaluated on five publicly available datasets: MSRAction3D [47], MSRActionPairs3D [5], UCF-Kinect [52], N-UCLA Multiview Action3D [53] and Online RGBD Action [65]. Accuracies of previous work are obtained from literature. In addition to the experiments, the end of the section provides some analysis of the parameters of the method.

MSRAction3D is a standard benchmark dataset for 3D action recognition, which has remained a challenging dataset due to high inter-class similarities between actions. For testing the ability of our method to capture temporal directionality of actions, we turn to the MSRActionPairs3D dataset, which consists of pairs of actions that differ only in the direction that the action is performed. UCF-Kinect is a dataset that contains actions suitable for interactive movements used in games. The N-UCLA Multiview Action3D dataset is quite different from the previous three, as it was captured with three different camera angles, which drastically changes the appearance of the actions, requiring the usage of features that are invariant across different views. Finally, the Online RGBD Action dataset aims to evaluate human-object interaction, and contains several action types that differ in the type of object interacted with.

Experiments on these datasets using both skeleton-based and keypoint-based graphs show the efficiency of our method. In particular, skeleton-based graphs work well for interactive actions due to the semantic labeling of the skeleton joints to body parts, whereas keypoint-based graphs show their strength in capturing complementary information to the skeleton joints, as it provides a feature that captures the spatio-temporal shape of the depth map point cloud. The details of our experiments are described in the sections that follow.

5.1. Experimental Settings

The PCA dimension is set so that 98% of the variance explained by the principal components is retained. For the SVM, we use a linear or radial basis function kernel. Both absolute max (Eq. (13)) and mean (Eq. (14)) pooling are tried. The choice of kernel, pyramid level K, and the number of spectral graph wavelet scales J is decided by cross-validation on the training set of each dataset.

Due to the lack of a publicly available implementation of STKP, we carefully implemented the method in C++. For simplicity, we use a fixed radius r of 10cm for the spatio-temporal support volume $\Omega(\mathbf{p})$, which is large enough to capture *e.g.* hand shapes but still small enough to capture fine detail.

As the parameter settings of the STKP detector are not disclosed, we here propose a set of parameter settings suitable for our purpose. For the non-maximum suppression step in STKP detection, two points are to be pruned if their volume intersection ratio ρ is larger than 0.5. This value is inspired by common intersection ratios used in non-maximum suppression for bounding boxes [66]. This gives us $\sigma_r = 0.7$ using the relation $4\cos[(2\pi - \arccos(\rho - 1))/3]$. By a similar argument, we want temporal overlap to be 0.5, so $\sigma_\tau = 0.5$. We detect $L = 400$ keypoints per action sequence, which we found empirically results in a good balance between the number of keypoints detected, and a low number of noisy keypoints. For keypoint-based graphs, we use $N = 1500$ codewords, which was selected by cross-validation.

Table 2: Recognition performance on the MSRAction3D dataset for the three different subject configurations on the three action sets as in Li *et al.*. [47] Each cell shows accuracy (%). Test 1 uses the first 1/3 samples for training and the rest for testing. Test 2 uses the first 2/3 samples for training and the rest for testing. The cross-subject test follows the same setup as in Table 1.

Method	Test 1				Test 2				Cross-subject test			
	AS1	AS2	AS3	Avg.	AS1	AS2	AS3	Avg.	AS1	AS2	AS3	Avg.
DL-GSGC [6]	100	98.7	100	99.6	100	98.7	100	99.6	97.2	95.5	99.1	97.3
SGS($p_{\max}$, skeleton-view-dep.)	**94.5**	**94.8**	**96.6**	**95.3**	**94.6**	**98.7**	**97.3**	**96.9**	**89.3**	**95.0**	**100**	**94.8**
SGS(p_{mean}, skeleton-view-dep.)	**96.6**	**90.8**	**98.0**	**95.1**	**98.6**	**96.0**	**98.6**	**97.7**	**88.4**	**91.6**	**100**	**93.3**
DMM-HOG [48]	97.3	92.2	98.0	95.8	98.7	94.7	98.7	97.4	96.2	84.1	94.6	91.6
STOP [49]	98.2	94.8	97.4	96.8	99.1	97.0	98.7	98.3	84.7	81.3	88.4	84.8
Eigenjoints [51]	94.7	95.4	97.3	95.8	97.3	98.7	97.3	97.8	74.5	76.1	96.4	82.3
HOJ3D [22]	98.5	96.6	93.5	96.2	98.6	97.9	94.9	97.2	88.0	85.5	63.5	79.0
Bag of 3D points [47]	89.5	89.0	96.3	91.6	93.4	92.9	96.3	94.2	72.9	71.9	79.2	74.7

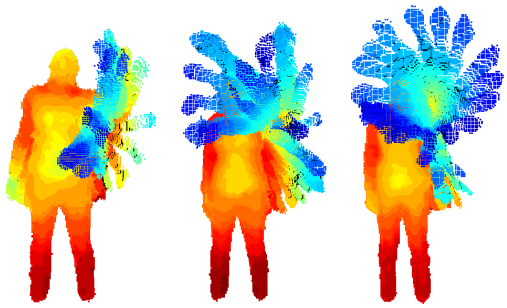


Figure 9: Frontal view examples of the actions "*hammer*" (left), "*draw x*" (middle) and "*draw circle*" (right) in the MSRAction3D dataset.

Table 1: Recognition performance on the MSRAction3D dataset.

Method	Accuracy (%)
DL-GSGC [6]	96.7
MMTW [7]	92.7
MP [50]	91.7
SGS(p_{mean}, skeleton-view-dep.)	**91.4**
HOD [46]	90.2
HON4D [5]	88.9
AE [8]	88.2
SGS($p_{\max}$, skeleton-view-dep.)	**86.3**
SGS(p_{mean}, skeleton-view-inv.)	**83.5**
SSS [9]	81.7
SGS($p_{\max}$, skeleton-view-inv.)	**79.4**
SGS(p_{mean}, keypoint)	**73.9**
Canonical poses [52]	65.7
HMM [67]	63.0
Motion Templates + DTW [68]	54.0

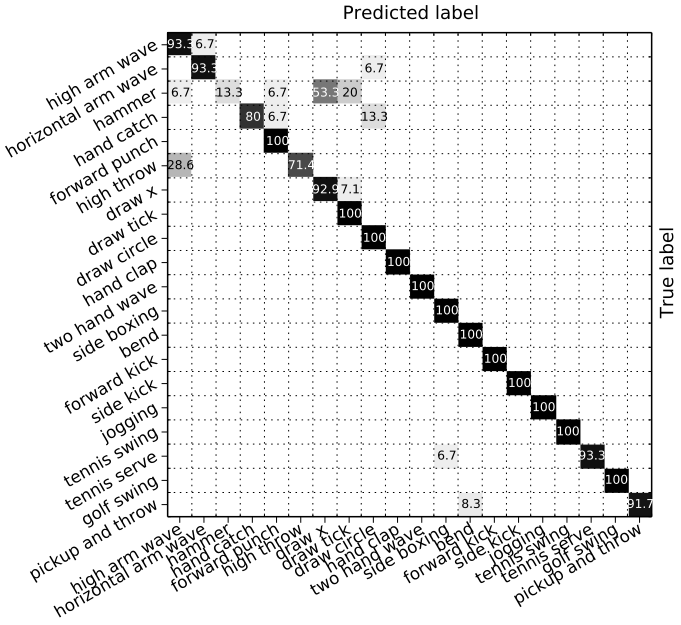


Figure 10: Confusion matrix for using our method on the MSRAction3D dataset. Each cell shows classification accuracy (%) from white (0) to black (100) in the cross-subject setting. The average accuracy is 91.4%.

5.2. Datasets and Results

5.2.1. MSRAction3D

The MSRAction3D dataset [47] contains 10 subjects performing 20 different actions, of out which some are quite similar, such as "*draw x*" and "*draw circle*". Each subject performs each action up to three times; not necessarily in the same manner each time. Due to a large body of related research, this dataset has become quite a representative benchmark for 3D action recognition. Despite the availability of discriminative depth maps, this dataset remains quite challenging due to an abundance of visually similar actions as well as noisy joint positions. For fair comparison with previous research, we run our experiments in the cross-subject setting, where samples from half of the subjects (*i.e.* subjects $1, 2, 3, 4, 5$) are used for train-

ing, and the rest for testing. This dataset contains some frames where the skeleton tracking fails, resulting in the joints to be erroneously located at the origin of the 3D coordinate system. We judge values to be missing only when the coordinates $(x, y, z) = (0, 0, 0)$, which Kinect outputs when the object is closer than 40 cm, or when no depth value could be found. For such missing values, the invalid joint positions are repaired using standard inter-frame linear interpolation.

The best parameters were $K = 4$ and $J = 50$. PCA reduced the feature dimension from 45900 to 152. The best results was gotten using *view-dependent* skeleton-based graphs and can be seen in Table 1. The confusion matrix is shown in Fig. 10. This dataset contains actions all captured from the frontal view, so view-invariance is seen to actually harm the performance, as the rotation cancellation can remove some information vital for classification (consider "*side boxing*" vs. "*forward punch*"). We see that mean pooling works better than max pooling, although both seem to be quite effective. Our SGS descriptor worked best with $K = 4$, but we note that even with $K = 1$ (no temporal pyramid), we got 83.5% recognition accuracy. Note that $K > 4$ could not be tested due to insufficiently long sequences in the dataset. Our method is able to fully distinguish between visually similar actions such as "*draw x/circle*" (see Fig. 9) and achieves perfect accuracy for most actions. On the other hand, the method repeatedly mistakes the action "*hammer*" for "*draw x*". These two classes are both characterized by similar highly accelerating movements along all axes of the 3D space. While SGS is able to capture different ranges of acceleration, it has trouble capturing the small temporal order of how these accelerations occur.

Although our method gains comparable results to most previous researches, it is unable to achieve results comparable to the sparse coding approach DL-GSGC [6]. Note however that our method has the advantage of being computable in time linear in the sequence length, while DL-GSGC requires solving a computationally heavy optimization problem. A difference between our method and sparse coding-based methods is that while sparse coding learns the relations between the interest points from data, our approach uses the explicit graph structure for describing this relationship. An interesting future direction would be to close this gap by learning the graph structure from data. Indeed, there has recently been some initial work in this direction [69].

Our method falls just short of MMTW [7], but it should be noted that while MMTW discriminatively learns a non-uniform warping of the time axis, our method works with a mere uniform division of the action sequence due to our temporal pyramid pooling scheme. Augmenting our temporal pyramid with non-uniform division is a probable point of future work.

Using SGS based on spatio-temporal keypoints ($J = 1$) did not yield a competitive result on this dataset since classification of actions in this dataset does not require the knowledge of human-object interaction. Indeed, we found that doing late fusion of the skeleton-based and keypoints-based graphs did not improve performance. Additionally, since MSRAction3D is a frontal-view dataset, the view-invariant property of the STKPs can actually hurt performance, which was shown to happen with view-invariant skeleton graphs. We conclude that for this dataset, the information about the spatial locations of semantically labeled body parts provides more discriminative information than the spatio-temporal shape of the point cloud. Note that while methods such as HON4D [5] perform well using only spatio-temporal point cloud data, they represent their feature using a spatial histogram, which implicitly encodes the spatial locations.

Table 3: Ablative analysis of performance on the MSRAction3D dataset.

Method	Accuracy (%)
SGS(p_{mean}, skeleton-view-dep.)	91.4
SGS(p_{mean}, skeleton-view-dep.), no PCA	88.3
SGS(p_{mean}, skeleton-view-dep.), no local edges	88.0
SGS(p_{mean}, skeleton-view-dep.), no ring graph	87.6
3D joints + DTW	77.7
3D joints + SGWT + DTW	74.9
SGS(p_{mean}, skeleton-view-dep.), no SGWT	74.2

Earlier work has also reported results on three separate action sets of MSRAction3D. The three action sets are defined to group visually similar action classes together [47], in order to test performance on small sets of similar actions. Our experiments follow this setup and results are shown in Table 2. Contrary to the previous experiment, max pooling is here seen slightly superior to mean pooling, indicating that the choice of max or mean pooling might depend on datasets. We can see that in this scenario with fewer action classes, our method achieves performance closer to DL-GSGC while being more efficiently computable.

Ablative Analysis. In order to illustrate the effect of each part of the proposed method, we perform an ablative analysis. Basically, our pipeline consists of the following parts: relative 3D joint positions, SGWT, temporal pyramid pooling, PCA, and SVM. We illustrate the significance of each part in Table 3, where several parts of the pipeline have been disabled. First, we disable PCA, and train an SVM directly on the high-dimensional pooled SGWT coefficients. This causes a slight decrease in performance, which supports our argument in Sec. 4.5 for doing dimensionality reduction. We also note that an added bonus of PCA is that the SVM training time is significantly reduced. Second, if we set the local edges to zero, we can see that we get inferior performance, illustrating the effect

of the spatio-temporal graph structure. Third, the table also shows that connecting the last skeleton with the first, creating a "ring graph", provides a slight improvement in performance, as argued in Sec. 4.8. Fourth, we can see that if we disable the SGWT, by applying temporal pyramid pooling directly to the raw 3D coordinates, we get a large decrease in performance. This is because the SGWT captures second-order information important for several action classes consisting of accelerating movement, as analyzed in Sec. 4.9.

Finally, we conduct an experiment where we apply dynamic time warping (DTW) [70] to the raw relative skeleton joint 3D coordinates and perform classification by finding the nearest neighbor to each test sample in terms of DTW measure. DTW has been popular with previous work in this field, and is a standard benchmark [8, 68]. Wang *et al.* [8] reported that while DTW is sensitive to noise, temporal pooling is more robust against noise and also temporal misalignment. DTW therefore makes an interesting baseline method for seeing the effects of temporal pooling on the SGWT coefficients. We note here that while stochastic variants of DTW have been proposed [71], estimation of the parameters requires a large number of available frames [72], and we here only evaluate the deterministic version.

The frame-to-frame distance used for comparing the sequence of graphs with DTW is the sum of the Euclidean distance between the corresponding skeleton joint 3D positions (Eq. (1)), and the optimal warping function between the sequences is found by dynamic programming [70]. By the results in the table, we can conclude that while this DTW-based approach is able to capture some characteristics for action classification, it inherently becomes a pose-based approach that will ignore higher-order characteristics such as velocity and acceleration. Still, we can see from the table that DTW is able to perform better than the temporal pyramid on the raw 3D coordinates. This is is due to DTW being able to handle non-linear expansions and contractions of the action (*i.e.* speed variation). The temporal pyramid is unable to do this, and will only be able to capture linear speed variations, as the temporal bins used for pooling are stretched uniformly in a linear manner. The proposed pipeline does however handle speed variations in a different manner: the SGWT creates a multi-scale decomposition of the graph signal, so different speeds will correspond to different scales of the decomposed signal.

It is also interesting to note that the positive effect that the SGWT coefficients have on the temporal pyramid pooling step does not generalize to DTW, as can be seen in the table by doing DTW on the SGWT coefficients with $J = 50$ scales. This is probably because DTW uses Euclidean distance, which does not work well in the high-dimensional space created by the SGWT [73].

Note also that the proposed pipeline has another advantage over DTW. Since DTW can only stretch the signal, it is unable to distinguish periodic actions when the number of periods differ [74]. One such action is waving your hand. It does not matter how many times you wave the hand; the action remains unchanged. Since the SGWT basis is wavelike with respect to the graph structure [31], the SGWT coefficients will be invariant to the exact number of repeats of the waving motion, while the DTW measure will differ in this case.

In summary, we can conclude by the ablative analysis that each step in the pipeline is effective and important for achieving the full performance of the SGS descriptor, where the SGWT accounts for most of the improvement.

Table 4: Recognition performance on the MSRActionPairs3D dataset.

Method	Accuracy (%)
HON4D [5]	96.7
SGS($p_{\mathbf{mean}}$, skeleton-view-dep.)	**96.0**
SGS($p_{\mathbf{max}}$, skeleton-view-dep.)	**93.1**
AE [8]	82.2
DMM-HOG [48]	66.1

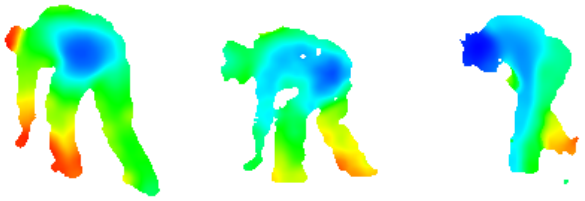
Table 5: Recognition performance on the UCF-Kinect dataset.

Method	Accuracy (%)
SGS($p_{\mathbf{mean}}$, skeleton-view-dep.)	**98.8**
SGS($p_{\mathbf{max}}$, skeleton-view-dep.)	**98.8**
MP [50]	98.5
Canonical poses [52]	95.9

5.2.2. MSRActionPairs3D [5]

This dataset was created to test performance for recognizing action pairs that are similar in motion, and differ in motion directionality only. An example of such an action pair is "*pick up box*" and "*put down box*". The dataset contains six action pairs performed by ten subjects, each subject performed each action three times. We run our experiments in the cross-subject setting, where the first five actors are used for training, and the rest for testing. As this is a frontal-view dataset, we only evaluate SGS based on view-dependent skeleton graphs.

The best parameters were $K = 5$ and $J = 1$ (decided by 5-fold cross-validation). PCA reduced the feature dimension from 3720 to 80. Results on MSRActionPairs3D can be seen in Table 4. Our method achieves comparable performance to HON4D [5], despite using only skeleton information. Additionally, HON4D discriminatively learns a non-uniform quantization of the 4D space, while our method works with only a simple uniform quantization along time using the temporal pyramid. We note that our method gets accuracy 56.6% with $K = 1$ and 86.3% with $K = 2$, confirming the importance of the temporal pyramid pooling scheme for recognizing motion directionality.



View 1 View 2 View 3

Figure 11: Cross-view examples of the action "*pick up with one hand*" from three different views in the N-UCLA Multiview Action3D dataset.

Table 6: Recognition performance on the N-UCLA Multiview Action3D dataset. The plus sign indicates late fusion.

Method	Accuracy (%)
SGS($p_{\max}$, skel.-view-inv.+keypoint)	**90.8**
SGS($p_{\max}$, skel.-view-inv.)	**87.4**
SGS(p_{mean}, skel.-view-inv.)	**84.4**
SGS(p_{mean}, keypoint)	**77.7**
SGS($p_{\max}$, keypoint)	**77.3**
NKTM [16]	75.8
SGS($p_{\max}$, skel.-view-dep.)	**74.7**
AOG [53]	73.3
nCTE [75]	68.6
SGS(p_{mean}, skel.-view-dep.)	**64.9**
CVP [19]	60.6
DVV [18]	58.5
Hankelets [12]	45.2

5.2.3. UCF-Kinect

The UCF-Kinect dataset [52] contains pre-segmented actions suitable for games, *e.g.* "*climb ladder*", "*leap*" and "*twist left*", with 1280 action sequences in total. 16 actions are performed by 16 subjects, with each subject performing each action five times each. Note that in this dataset the provided skeletons only have 15 joints. As the center hip joint is missing, we approximate it by the average of the left and right hip joint positions. We run our experiments in the same setting as Ellis *et al.* [52], reporting the average accuracy of 4-fold cross-validation. As this is a frontal-view dataset, we only evaluate SGS based on view-dependent skeleton graphs.

The best parameters were $K = 3$ and $J = 43$. PCA reduced the feature dimension from 18480 to 127. Results on UCF-Kinect can be seen in Table 5. We can see that our method achieves superior performance compared to the original canonical pose approach [52], while performing slightly better than MP [50]. This shows that our proposed framework is suitable for recognition of game-related actions that make use of all tracked parts of the body.

Table 7: Ablative analysis of performance on the N-UCLA Multiview Action3D dataset.

Method	Accuracy (%)
SGS(p_{mean}, keypoint)	77.7
SGS(p_{mean}, keypoint), no local edges	75.4
SGS(p_{mean}, keypoint), no SGWT	73.6
STKP + BoW vector	68.5

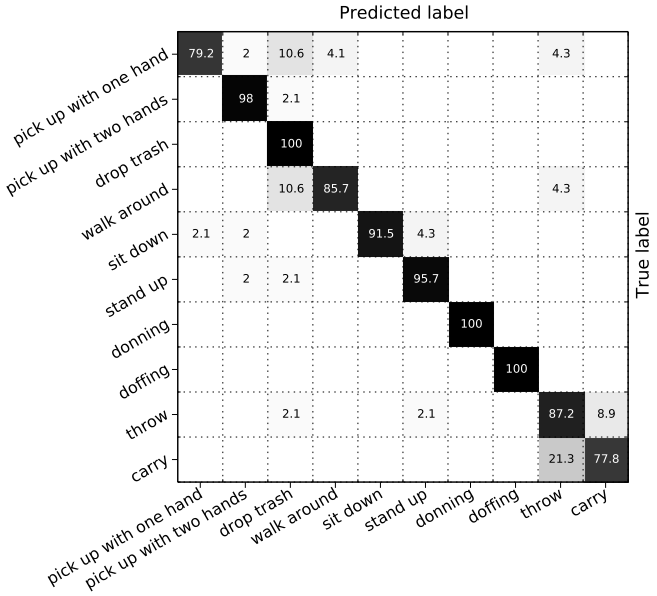


Figure 12: Confusion matrix for using our method on the N-UCLA Multiview Action3D dataset. Each cell shows classification accuracy (%) from white (0) to black (100) in the cross-view setting. The average accuracy is 90.8%.

5.2.4. N-UCLA Multiview Action3D

The N-UCLA Multiview Action3D dataset [53] aims to capture daily actions performed by humans from multiple camera angles, such as "*throw trash*", "*walk around*", "*stand up*" or "*carry*". Ten subjects were instructed to perform 10 actions. The dataset was captured simultaneously by three Kinect cameras and contains 1493 action sequences in total (see also Fig. 11). Further, several actions include interaction with objects, such as "*drop trash*" and "*carry*". This dataset is very challenging not only due to each action being captured from different views; most actions also include walking and some actions are very similar, such as "*pick up with one hand*" and "*pick up with two hands*". Another challenging action is "*drop trash*", which includes some sequences with extremely subtle motion that could be easily mistaken for "*walk around*".

For the skeleton-based graph, we apply our proposed rotation scheme to make the feature view-invariant. The keypoint-based graph is also well-suited for cross-view action recognition as STKPs are view-invariant, captures interaction with objects, and the BoW graph signal is largely unaffected by viewpoint changes (up to occlusions). Our

experiments are run in the cross-view setting, with the first two views used for training, and the third one for testing.

Results can be seen in Table 6. Accuracies of previous work are due to results by Rahmani *et al.* [16]. The confusion matrix is shown in Fig. 12. The best parameters were $K = 4$ and $J = 1$ for keypoint-based graphs and $K = 4$ and $J = 11$ for skeleton-based graphs. PCA reduced the feature dimension for keypoint-based graphs from 45000 to 863 and from 10800 to 513 for skeleton-based graphs. We achieve good results despite the dataset containing some labeling noise (mislabeled samples) and noisy subject segmentations. Our method performs better than NKTM [16], which requires a large labeled auxiliary motion-capture dataset with about 26000 samples for training a deep neural network. Our method, on the other hand, is able to represent view-invariant actions using only the raw point cloud training data.

On this dataset, even keypoint-based graphs achieve state-of-the-art results, but we achieve ever better performance with view-independent skeleton graphs. Finally, by combining the two graph types through late fusion, we achieve a large increase in performance, advancing the state-of-the-art results on this dataset by 19.8% compared to NKTM. We believe this increase in accuracy is due to the view-invariant graphs used by our method being solely depth map-based, in contrast to AOG and NKTM, which make use of skeleton information during training, but apply their method only on RGB images during testing. Indeed, depth-map methods do in general outperform RGB-based methods, since they alleviate the problems caused by illumination variations and background clutter [1].

To illustrate the power of our proposed framework, we perform ablative analysis of the keypoint-based graphs, showing the performance of each part of our system in Table 7. In the table, BoW vector refers to training an SVM with a histogram intersection kernel on the BoW vector representation of the detected STKPs (this is the same setup as in Rahmani *et al.* [21], but they do not test on a public dataset, so a direct comparison is not possible). As can be seen in the table, each part of the system gives a significant improvement over the baseline of using just a BoW vector representation of the STKPs. The difference in performance between the BoW vector representation and our keypoint-based SGS representation is statistically significant ($p < 3 \cdot 10^{-5}$ using McNemar's test [76]). We can also see a drop in accuracy when removing the SGWT coefficients from the pipeline, and just applying a pyramid pooling directly to the graph signal of the keypoint graph. Furthermore, the performance deteriorates to 75.4% when setting all local edge weights to zero, which shows that local edges capture additional information relevant for discriminating between action classes and, in this case, achieving a better result than NKTM.

5.2.5. Online RGBD Action

The Online RGBD Action dataset [65] aims to capture various daily life actions, with focus on human-object interaction. It contains several actions that are similar in motion, but only differ in object appearance, such as "*reading phone (sending SMS)*" and "*reading book*". In total, the dataset contains seven action classes, and is divided into several subsets. Subset S1 contains 8 subjects performing each of the seven actions twice, yielding 112 action sequences. Subset S2 is constructed in a similar manner, but contains 8 new subjects. Finally, subset S3 follows the same setup, with 8 new subjects, but was recorded in a different environment from S1 and S2, which means that it can be used for a cross-environment evaluation. This dataset is therefore useful for evaluating the ability of an action recognition method to distinguish between different types of human-object interaction.

We follow the two experimental configurations used by Yu *et al.* [65]: *SameEnv* and *CrossEnv*. SameEnv reports the two-fold cross-validation accuracy of S1 and S2, while CrossEnv trains on S1 ∪ S2 and tests on S3. For simplicity, we evaluate only max pooling and view-dependent skeleton graphs for this experiment. The results are shown in Tables 8 and 9.

Our proposed method achieves state-of-the-art results for human-object interaction for the SameEnv setting, and slightly worse results for the CrossEnv setting. This is despite that our method is designed for actions defined largely by movement, which is not the case for actions such

Table 8: Recognition performance on the Online RGBD Action dataset in the SameEnv setting. The plus sign indicates late fusion.

Method	Accuracy (%)
SGS($p_{\max}$, skel.-view-dep.+keypoint)	**72.3**
Orderlet mining [65]	71.4
AE [8]	66.0
SGS($p_{\max}$, keypoint)	**64.7**
DSTIP+DCSF [77]	61.7
SGS($p_{\max}$, skel.-view-dep.)	**59.4**
HOSM [78]	49.5
Eigenjoints [51]	49.1
MP [50]	38.4

Table 9: Recognition performance on the Online RGBD Action dataset in the CrossEnv setting. The plus sign indicates late fusion.

Method	Accuracy (%)
Orderlet mining [65]	66.1
AE [8]	59.8
SGS($p_{\max}$, skel.-view-dep.+keypoint)	**57.1**
SGS($p_{\max}$, skel.-view-dep.)	**46.4**
HOSM [78]	50.9
SGS($p_{\max}$, keypoint)	**42.0**
Eigenjoints [51]	35.7
MP [50]	28.5
DSTIP+DCSF [77]	21.5

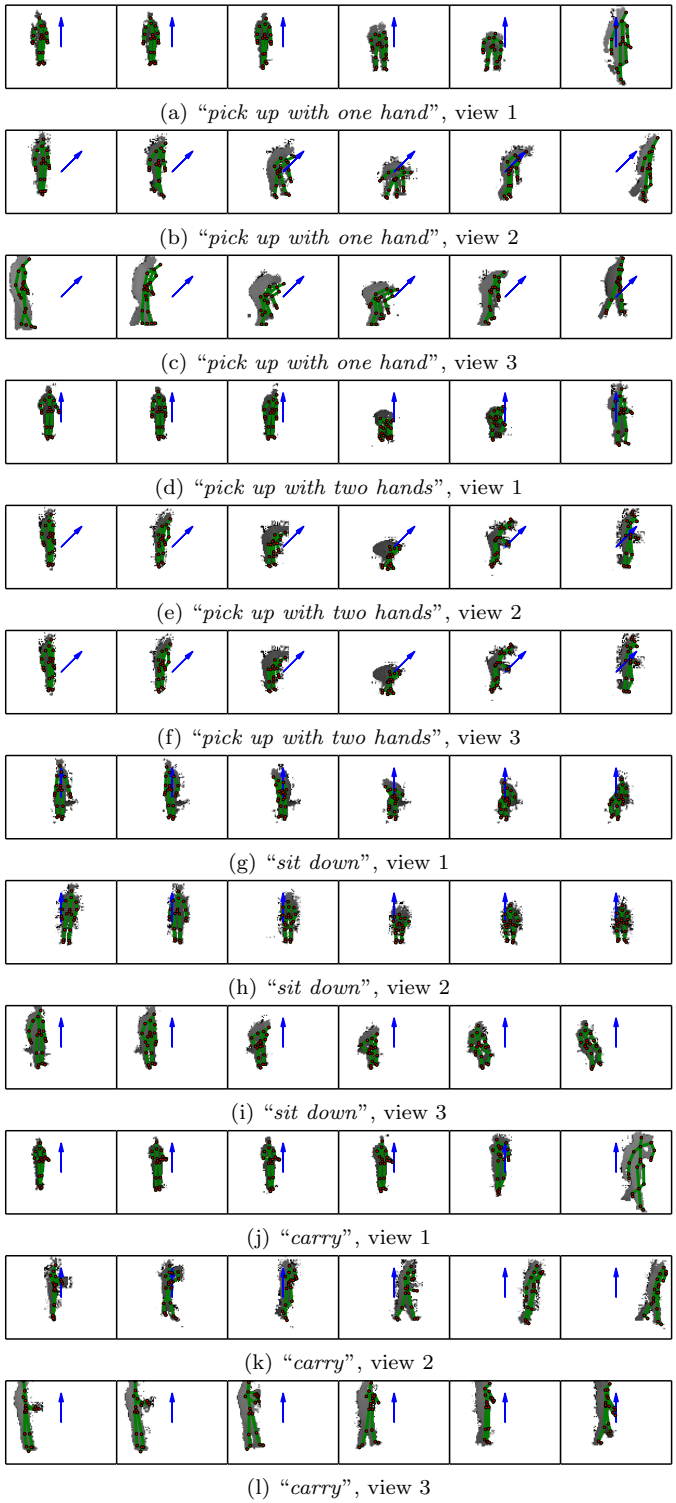
as "*reading phone*", where there is very little movement close to the object of interest and will not be captured by the STKPs in our keypoint-based graph. We can also see that the skeletons and keypoints capture complementary information, where we get the best results by late fusion. While the orderlet mining method [65] generalizes slightly better across environments when doing the CrossEnv evaluation, we can see that our method is also able to retain reasonable performance. This is not true for methods such as DSTIP+DCSF [77], which works well in the SameEnv setting, but fails to generalize to new environments, as can be seen in Table 9.

We further note that Yu *et al.* [65], similar to us, achieve their best performance of 71.4% accuracy when using both object and skeleton features. They report that when using only their object feature, Local Occupancy Pattern (LOP), they get 46.4% accuracy, while our proposed keypoint-based graphs achieves 64.7%. This is probably due to STKPs capturing the spatio-temporal shape of the point cloud, while LOPs only capture spatial 3D shape information [8].

5.3. Quality of the Estimated Up Vector

For achieving cross-view action recognition for skeleton-based graphs, our method relies on the rotation cancellation scheme presented in Sec. 3.3.1. One concern is that the rotation cancellation is largely determined by the estimated up vector $\mathbf{v}_{\text{up}}$ in (2). In this section, we briefly investigate the robustness of the up vector estimates for different action classes in the N-UCLA Multiview Action3D dataset. Essentially, the up vector creation process using the marginal median is based on the assumption that the subject will stand up-right in most of the frames in the action sequence. For some action classes, however, this might not hold true. Such examples are "*pick up with one hand*" and "*pick up with two hands*". Visualizations of estimated up vectors are shown in Fig. 13. We can see that while the estimated up vector turns out quite sensible for most actions, there is indeed some trouble with action classes where the subject is bending down for an extended amount of time. By looking at the example frames in the figure, one might suggest that the up vector estimate should be taken from the beginning or end of the action sequence, as there the subject always seems to be standing up right. We argue, however, that such an approach is not general enough, and is merely an exercise of overfitting on this particular dataset. This approach would fail in a real-world scenario, where the subject might be taking any possible pose configuration at the boundaries of the segmented sequence.

Nevertheless, despite the sometimes faulty up vector estimates for the action "*pick up with two hands*", we can see that our method is able to still recognize this class with 98% accuracy, as was shown in Fig. 12, and the up vector is estimated in a sensible way for most other action classes.



(a) "*pick up with one hand*", view 1
(b) "*pick up with one hand*", view 2
(c) "*pick up with one hand*", view 3
(d) "*pick up with two hands*", view 1
(e) "*pick up with two hands*", view 2
(f) "*pick up with two hands*", view 3
(g) "*sit down*", view 1
(h) "*sit down*", view 2
(i) "*sit down*", view 3
(j) "*carry*", view 1
(k) "*carry*", view 2
(l) "*carry*", view 3

Figure 13: Visualization of estimated up vectors (blue arrow) on example frames from the N-UCLA Multiview Action3D dataset. Best viewed in color.

Essentially, we can conclude that the up-vector estimation works reasonably well given the assumption that most of the frames contain the subject standing straight up. Note that this also works when the subject is sitting straight up on a chair, as seen in *e.g.* Fig. 13i. For ac-

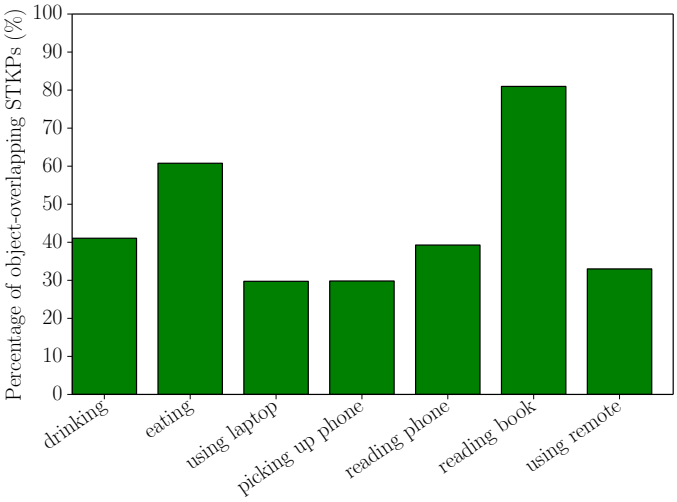
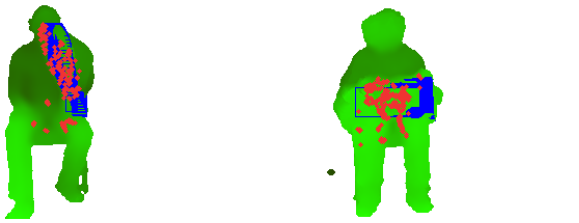


Figure 14: The percentage of detected STKPs in the subset S1 ∪ S2 of the Online RGBD Action dataset, that overlap with ground-truth bounding boxes for each action class. The grand mean is 44.95%.



(a) "*Drinking*". 43.97% of detected STKPs are overlapping with the water bottle.

(b) "*Reading book*". 89.00% of detected STKPs are overlapping with the book.

Figure 15: Visualization of detected STKP locations (red) and ground truth object bounding boxes (blue) on aggregated point clouds from the Online RGBD Action dataset. Best viewed in color.

tion sequences where this assumption does not hold, such as in Fig. 13b and 13c, the estimated vector becomes of lower quality, which is an inherent weakness of the current method, and should be investigated in future work.

5.4. Keypoint Locations

For detection of human-object interaction, it is crucial that the STKPs in our keypoint-based graph are detected in close proximity to objects. Recall that each STKP captures information about the spatio-temporal shape of the surrounding point cloud by using a sphere with radius r that is centered at the STKP. In this section, we investigate the number of STKPs that are overlapping with the ground-truth rectangular object bounding boxes in the Online RGBD Action dataset. The proportion of overlapping STKPs can be seen in Fig. 14. Qualitative examples on aggregated point clouds are shown in Fig. 15.

We can see that "*drinking*" has decent overlap since the motion in the video often involves moving a water bottle between a table and the subject's mouth, which generates high quality STKPs close to the object (the water bottle). Similarly, "*reading book*" has high overlap since most of the motion in the video happens when turning book pages. On the contrary, "*reading phone*" has slightly lower overlap since most of the motion in the video is really fine motion, used for navigating the cellphone with the subject's fingers. This small motion causes the STKP candidate's quality score to become small, which causes them to be pruned as noise.

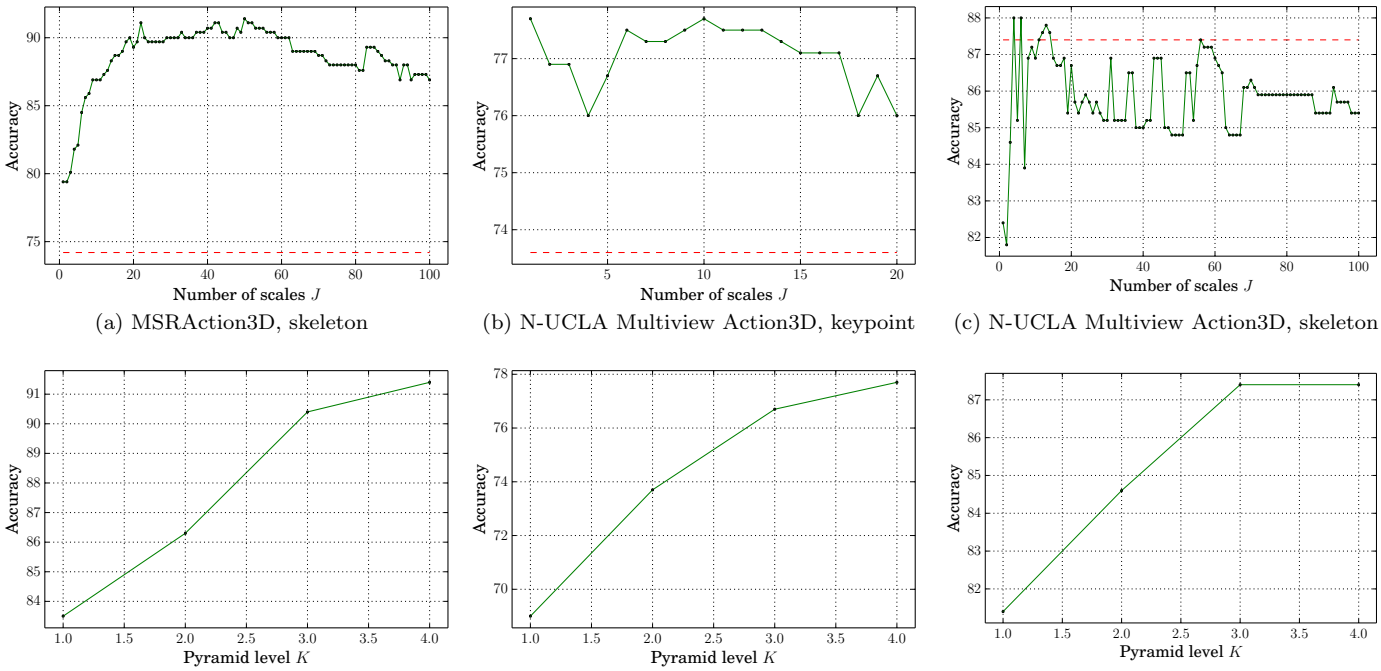
Nevertheless, we can conclude that a suitably large number of STKPs are detected at locations relevant for human-object detection, which was also verified by the experimental results in Table 8.

5.5. Parameter Analysis

This section analyses the effect of the parameters of our method. The number of wavelet scales J for the MSRAction3D dataset can be seen in Fig. 16a, and for the N-UCLA Multiview Action3D dataset in Fig. 16b and 16c. Note that due to the increased graph size for keypoint-based graphs, we cannot test as many wavelet scales as for skeleton-based graphs.

Perhaps surprisingly, we can see in Fig. 16c that on the N-UCLA Multiview Action3D dataset, skipping the SGWT and doing max pooling of the view-invariant relative joint positions works equally well as utilizing the SGWT. We believe this is due to max pooling capturing a set of key poses using the temporal pyramid that provide enough information for classifying the actions in this dataset. This does not hold for absolute mean pooling, which gets 82.4% without the SGWT. Note that the rotation cancellation process is very important for the max pooling performance on this dataset, as we get 66.2% accuracy if we use view-dependent skeleton graphs without the SGWT, in which case using the SGWT gives a better result. Fine-grained knowledge of acceleration is in this dataset not as important as for the skeletons in MSRAction3D, which can be seen by an increasing number of wavelet scales J causing decreasing performance. We can also see that for small J, we get degraded performance, which indicates that low-frequency information alone cannot capture enough information, as small J attenuates high frequencies [31]. For MSRAction3D, however, tuning J is important for good performance, as the dataset contains many spatially similar actions.

For keypoint-based graphs, we can see that a small J is better. A low number of wavelet scales focuses the descriptor on low-frequency information. Since the accuracy does not change much with increasing number of scales on N-UCLA Multiview Action3D, this indicates that high-frequency information is not important for recognizing the actions in the dataset. Indeed, if the dataset does not contain actions that have large intra-class similarity, then the finer distinction of second order information that higher frequencies provide does not help classification. Rather, the difficulty of this dataset is the cross-view camera angles, while the actions themselves are defined by larger global motions. On the contrary, MSRAction3D contains



(a) MSRAction3D, skeleton (b) N-UCLA Multiview Action3D, keypoint (c) N-UCLA Multiview Action3D, skeleton

(d) MSRAction3D, skeleton (e) N-UCLA Multiview Action3D, keypoint (f) N-UCLA Multiview Action3D, skeleton

Figure 16: Test set accuracy as a function of the number of wavelet scales J (top row) and pyramid levels K (bottom row) on the MSRAction3D and N-UCLA Multiview Action3D datasets. The dashed line shows accuracy without using the SGWT.

actions that are very similar spatially, such as "*draw x*" and "*draw circle*". Recognition of these two actions requires knowledge of fine-grained second-order information (in the case of the skeleton-based graph, acceleration), which explains why a larger number of wavelet scales helps improving the recognition performance. Indeed, for the UCF-Kinect dataset, knowledge of acceleration is more important for actions differing by smaller movements such as "*climb ladder*" and "*climb up*". On the other hand, for MSRActionPairs3D, the knowledge of temporal directionality is more important, which explains the choice of scales in these cases as well.

The number of pyramid levels K for the MSRAction3D dataset can be seen in Fig. 16d, and for N-UCLA Multiview Action3D in Fig. 16e and 16f. Accuracy is increasing steadily with increasing number of pyramid levels, which can be explained by that a larger number of pyramid levels better capture the temporal order of actions, although diminishing returns can be seen for larger values of K.

In its basic form ($K = 1$), our descriptor is not able to capture the order in which the information from the coefficients occur, something which is important for actions bound by motion directionality, such as the ones in MSRActionPairs3D. While using the temporal pyramid ($K > 1$) effectively helps capturing such temporal order, we believe that a non-uniform partition of the time-axis might be required to fully capture action classes that exhibit a very locally dependent temporal order. Increasing the value of K does however double the size of the feature descriptor. So in practice, a decision of compromise between accuracy and computational speed might have to be made.

5.6. Descriptor Properties

In this section, we discuss some properties of our proposed descriptor. Since the graph Laplacian matrix $\boldsymbol{\mathcal{L}}$ acts as a graph-analog to the classical Laplace operator [24], SGS is able to capture, per each vertex and axis, the existence of ranges of differential second order information. This range is determined by the SGWT kernel g. The window created by the SGWT kernel h in turn captures aggregated low-frequency information, such as the average position of the action in 3D space. We believe that for skeleton-based graphs, SGS is able to distinguish between actions that can be characterized by different acceleration at each joint. On the other hand, this means that SGS potentially has trouble separating sets of actions that have the same such characteristics. This became evident in MSRAction3D, where "*hammer/draw x/draw tick*" exhibit a set of actions that when looked at along each axis, display similar ranges of acceleration around the same spatial location.

For keypoint-based graphs, our descriptor captures the frequency with which the keypoints are detected on the point cloud and more specifically second order information about their occurrence both temporally and locally.

5.7. Computational Time

For skeleton-based graphs, the whole feature can be calculated in about 0.3 seconds using a Python imple-

mentation on a machine with an Intel i7-3770K 3.5GHz CPU and 32GB RAM given a pre-segmented action sequence. For a 2 second long action sequence recorded at 33 frames per second, this corresponds to a frame rate of 220 FPS. For the keypoint-based graphs, STKP detection takes about 1.6 seconds per frame in C++. The subsequent SGWT calculation then takes about 2.6 seconds per action sequence in Python due to the larger graph size.

6. Conclusion

We have presented a method for view-invariant action recognition from depth cameras based on graph signal processing techniques. Our framework leverages a novel graph representation of an action as a temporal sequence of graphs, onto which we apply the SGWT framework [31] for creating an overcomplete representation of an action that captures both local and temporal variations of the signal. The graph wavelet coefficients are applied to a temporal pyramid pooling scheme, which creates a descriptor of an action sequence. For a T frames long action sequence with N keypoints in each frame, the SGS descriptor is computable in $\mathcal{O}(TN)$ time. We also presented an efficient algorithm that exploits the explicit sparsity structure of our graph for calculating the fast approximate SGWT. The power of our method was demonstrated by experiments on five publicly available datasets, resulting in superior performance for cross-view action recognition, or results comparable to state-of-the-art action recognition approaches for the frontal-view case and for human-object interaction.

In the future, we will investigate other interest point types, as well as explore applying the proposed framework to other temporal classification tasks. For rotation cancellation, alternative methods for up-vector estimation robust to a larger variety of action classes should also be explored. This paper has focused on action recognition, but the proposed framework is in general applicable to any time series of graphs.

Appendices

A. The Graph Laplacian Matrix

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ denote a graph with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$ with $N = |\mathcal{V}|$ vertices. We let $\mathbf{W} \in \mathbb{R}^{N\times N}$ denote the weight matrix associated with $\mathcal{G}$, where $\mathrm{W}(i,j) \in \mathbb{R}^+$ is the weight of the edge between vertices v_i and v_j, or 0 if there is no edge. Then $\mathbf{Ł} = \mathbf{D} - \mathbf{W}$ is the graph Laplacian matrix, where $\mathbf{D} = \mathrm{diag}\{\mathbf{W1}\}$ is the diagonal degree matrix and $\mathbf{1}$ is the vector of all ones. We let $\{\lambda_\ell, \mathbf{u}_\ell\}_{\ell=0,\ldots,N-1}$ denote the eigenvalue and eigenvector pairs of $\mathbf{Ł}$. The spectrum of $\mathbf{Ł}$ carries a frequency interpretation [64], making it applicable for harmonic analysis on graphs. We consider only undirected simple graphs, which makes all eigenvalues real and non-negative, since $\mathbf{Ł}$ is a real positive-semidefinite matrix [59]. For the *normalized* graph Laplacian matrix $\boldsymbol{\mathcal{L}} = \mathbf{D}^{-1/2}\mathbf{Ł}\mathbf{D}^{-1/2}$, there is a trivial upper bound $\lambda_{\max} = 2$ for the maximum eigenvalue, which is tight when the graph is bipartite [59].

A graph signal is a function $f : \mathcal{V} \to \mathbb{R}$ that assigns a value to each vertex. Such a signal can be represented as a vector $\mathbf{f} \in \mathbb{R}^N$ lying on a graph $\mathcal{G}$. By writing the eigendecomposition $\mathbf{Ł} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$, frequency analysis of $\mathbf{f}$ can be performed by taking the graph Fourier transform $\widehat{\mathbf{f}} = \mathbf{U}^T\mathbf{f}$, which gives a graph Fourier mode decomposition of the signal [24].

B. Spectral Graph Wavelets

Hammond *et al.* [31] define a spectral graph wavelet transform (SGWT) for graph signals on the eigenspectrum of $\mathbf{Ł}$.[3] Each spectral graph wavelet is realized by taking a kernel function $g : \mathbb{R}^+ \to \mathbb{R}^+$, scaling its domain by a scalar t, and finally localizing the result by convolving it with an impulse $\boldsymbol{\delta}_n \in \mathbb{R}^N$, which has value 1 at vertex n, and 0 everywhere else. A spectral graph wavelet $\boldsymbol{\psi}_{t,n} \in \mathbb{R}^N$ at scale t localized around vertex n can be written explicitly as a vector

$$\psi_{t,n}(m) = \sum_{\ell=0}^{N-1} g(t\lambda_\ell)\mathrm{u}_\ell(n)\mathrm{u}_\ell(m) \ . \tag{B.1}$$

Given a graph signal $\mathbf{f}$, an SGWT coefficient is extracted by the inner product $\langle \boldsymbol{\psi}_{t,n}, \mathbf{f}\rangle$. The kernel g is chosen to act as the following band-pass filter [31]

$$g(x) = \begin{cases} x_1^{-\alpha}x^\alpha & \text{for } x < x_1 \\ s(x) & \text{for } x_1 \le x \le x_2 \\ x_2^{\beta}x^{-\beta} & \text{for } x > x_2 \end{cases} , \tag{B.2}$$

where $\alpha = \beta = 2$, $x_1 = 1$, $x_2 = 2$ and $s(x)$ is a unique cubic spline that respects the curvature of g. Then, coefficients for smaller scales (small t) will localize high-frequency information around a vertex, while larger scales (large t) capture low-frequency information. The transform also includes a scaling kernel $h : \mathbb{R}^+ \to \mathbb{R}$, $h(x) = \gamma\exp(-(x/(0.6\epsilon))^4)$, for creating a scaling function $\boldsymbol{\phi}_n$ for stably representing low-frequency content in the graph [31]. Here, γ is set so that $h(0)$ equals the maximum value of g, and the design parameter $\epsilon = \lambda_{\max}/20$, where $\lambda_{\max}$ is an upper bound of the maximum eigenvalue of the graph Laplacian. The scaling vector $\boldsymbol{\phi}_n$ is defined similarly to Eq. (B.1), with g replaced by h and setting the parameter $t = 1$.

Let J denote an integer such that the set of wavelet scales is $\{t_j\}_{j=1,\ldots,J}$. Then, the SGWT provides a transform with $J + 1$ scales; J wavelets and one scaling function. By gathering the wavelet and

[3] Online source code available at `http://wiki.epfl.ch/sgwt` .

scaling function vectors in a transformation matrix $\mathbf{T} = [\mathbf{\Psi}_{t_1}, \ldots, \mathbf{\Psi}_{t_J}, \mathbf{\Phi}] = [\boldsymbol{\psi}_{t_1,1}, \ldots, \boldsymbol{\psi}_{t_J,N}, \boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_N]$, the transform coefficients can be expressed as a $(J+1)N$-dimensional vector $\mathbf{c} = \mathbf{T}^T\mathbf{f}$.

We also note that the SGWT is an overcomplete transform, as it contains more wavelet coefficients than vertices in the graph. If a signal is representable using only a few wavelet scales, then the SGWT can be viewed as quite similar to sparse coding [55], and each wavelet as an atom in a sparse dictionary [56]. However, since spectral graph wavelets are based on a fixed mathematical structure, they can be computed more efficiently, while sparse coding requires solving a heavy optimization problem [62]. It should be noted that while attempts to embed graph structure into the learned dictionary exists, this does not guarantee an efficient implementation [56]. Another advantage of spectral graph wavelets is that the explicit mathematical structure enables formal analysis of the effects of each wavelet basis.

C. Fast Approximate Wavelet Transform

In order to avoid explicit computation of the eigenspectrum of $\boldsymbol{\mathcal{L}}$, which takes $\mathcal{O}(|\mathcal{V}|^3)$ time (and is thus only feasible for graphs up to about 1000 vertices), the authors of the SGWT introduced a method based on truncated Chebyshev polynomials for approximating the transform in $\mathcal{O}(|\mathcal{E}| + J|\mathcal{V}|)$ time [31]. They approximate the kernels g and h using low-dimensional Chebyshev polynomials

$$g(t_j\lambda) \approx \frac{1}{2}c_{j,0} + \sum_{k=1}^{M_j} c_{j,k}\overline{T}_k(\lambda) \ , \tag{C.1}$$

where M_j is the degree of the approximation, typically $M_j = 50$. The expression $\overline{T}_k(\lambda) = T_k(\lambda - 1)$ is the shifted Chebyshev polynomial of order k, which satisfies the recurrence relation $T_k(\lambda) = 2\lambda T_{k-1}(\lambda) - T_{k-2}(\lambda)$. Further, $c_{j,k}$ denote the Chebyshev coefficients, which can be estimated given a spectrum upper bound $\lambda_{\max}$ [60].

The approximated transforms are given by

$$\mathbf{\Psi}_{t_j}^T\mathbf{f} \approx \frac{1}{2}c_{j,0}\mathbf{f} + \sum_{k=1}^{M_j} c_{j,k}\overline{\mathbf{T}}_k(\boldsymbol{\mathcal{L}})\mathbf{f} \ , \tag{C.2}$$

$$\mathbf{\Phi}^T\mathbf{f} \approx \frac{1}{2}c_{0,0}\mathbf{f} + \sum_{k=1}^{M_0} c_{j,k}\overline{\mathbf{T}}_k(\boldsymbol{\mathcal{L}})\mathbf{f} \ , \tag{C.3}$$

with $\overline{\mathbf{T}}_0(\boldsymbol{\mathcal{L}}) = \mathbf{I}$ and $\overline{\mathbf{T}}_1(\boldsymbol{\mathcal{L}}) = \boldsymbol{\mathcal{L}} - \mathbf{I}$. The approximation accesses $\boldsymbol{\mathcal{L}}$ only through matrix-vector multiplications and is fast for sparse graphs.

Acknowledgments

We thank the anonymous reviewers for their insightful comments, which helped improve the content of the paper. The first author acknowledges the Japanese Government (Monbukagakusho:MEXT) scholarship support for carrying out this research. This work was supported by JSPS KAKENHI Grant Number 15K12061.

References

[1] J. Han, L. Shao, D. Xu, J. Shotton, Enhanced computer vision with Microsoft Kinect sensor: A review, IEEE Transactions on Cybernetics 43 (5) (2013) 1318–1334.

[2] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, R. Moore, Real-time human pose recognition in parts from single depth images, Communications of the ACM 56 (1) (2013) 116–124.

[3] H. Bunke, K. Riesen, Towards the unification of structural and statistical pattern recognition, Pattern Recognition Letters 33 (7) (2012) 811–825.

[4] G. Johansson, Visual perception of biological motion and a model for its analysis, Perception & psychophysics 14 (2) (1973) 201–211.

[5] O. Oreifej, Z. Liu, W. Redmond, Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences, in: CVPR, 2013.

[6] J. Luo, W. Wang, H. Qi, Group sparsity and geometry constrained dictionary learning for action recognition from depth maps, in: ICCV, 2013.

[7] J. Wang, Y. Wu, Learning maximum margin temporal warping for action recognition, in: ICCV, 2013.

[8] J. Wang, Z. Liu, Y. Wu, J. Yuan, Mining actionlet ensemble for action recognition with depth cameras, in: CVPR, 2012.

[9] X. Zhao, X. Li, C. Pang, X. Zhu, Q. Z. Sheng, Online human gesture recognition from motion data streams, in: ACM MM, 2013.

[10] D. Weinland, E. Boyer, R. Ronfard, Action recognition from arbitrary views using 3d exemplars, in: ICCV, 2007.

[11] A. Yilmaz, M. Shah, Actions sketch: A novel action representation, in: CVPR, 2005.

[12] B. Li, O. I. Camps, M. Sznaier, Cross-view activity recognition using hankelets, in: CVPR, 2012.

[13] V. Parameswaran, R. Chellappa, View invariance for human action recognition, International Journal of Computer Vision 66 (1) (2006) 83–101.

[14] C. Rao, A. Yilmaz, M. Shah, View-invariant representation and recognition of actions, International Journal of Computer Vision 50 (2) (2002) 203–226.

[15] D. Weinland, R. Ronfard, E. Boyer, Free viewpoint action recognition using motion history volumes, Computer Vision and Image Understanding 104 (2) (2006) 249–257.

[16] H. Rahmani, A. Mian, Learning a non-linear knowledge transfer model for cross-view action recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 2458–2466.

[17] A. Farhadi, M. K. Tabrizi, Learning to recognize activities from the wrong view point, in: ECCV, 2008.

[18] R. Li, T. Zickler, Discriminative virtual views for cross-view action recognition, in: CVPR, 2012.

[19] Z. Zhang, C. Wang, B. Xiao, W. Zhou, S. Liu, C. Shi, Cross-view action recognition via a continuous virtual path, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 2690–2697.

[20] J. Zheng, Z. Jiang, Learning view-invariant sparse representations for cross-view action recognition, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 3176–3183.

[21] H. Rahmani, A. Mahmood, D. Q. Huynh, A. Mian, HOPC: Histogram of oriented principal components of 3D pointclouds for action recognition, in: ECCV, 2014.

[22] L. Xia, C.-C. Chen, J. Aggarwal, View invariant human action recognition using histograms of 3d joints, in: CVPR Workshops, 2012.

[23] J. Wang, Z. Liu, Y. Wu, J. Yuan, Learning actionlet ensemble for 3d human action recognition, Pattern Analysis and Machine Intelligence, IEEE Transactions on 36 (5) (2014) 914–927.
[24] D. I Shuman, S. K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains, IEEE Signal Processing Magazine 30 (3) (2013) 83–98.
[25] H. Bunke, K. Riesen, Recent advances in graph-based pattern recognition with applications in document analysis, Pattern Recognition 44 (5) (2011) 1057–1067.
[26] X. Zhu, J. Kandola, J. Lafferty, Z. Ghahramani, Graph kernels by spectral transforms, in: O. Chapelle, B. Schoelkopf, A. Zien (Eds.), Semi-Supervised Learning, MIT Press, 2006, pp. 277–291.
[27] L. Hermansson, T. Kerola, F. Johansson, V. Jethava, D. Dubhashi, Entity disambiguation in anonymized graphs using graph kernels, in: CIKM, ACM, 2013.
[28] E. Stumm, C. Mei, S. Lacroix, J. Nieto, M. Hutter, R. Siegwart, Robust visual place recognition with graph kernels, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
[29] N. Shervashidze, P. Schweitzer, E. J. v. Leeuwen, K. Mehlhorn, K. M. Borgwardt, Weisfeiler-lehman graph kernels, Journal of Machine Learning Research 12 (Sep) (2011) 2539–2561.
[30] A. Sandryhaila, J. M. F. Moura, Discrete signal processing on graphs, IEEE Transactions on Signal Processing 61 (7) (2013) 1644–1656.
[31] D. K. Hammond, P. Vandergheynst, R. Gribonval, Wavelets on graphs via spectral graph theory, Applied and Computational Harmonic Analysis 30 (2) (2011) 129–150.
[32] R. R. Coifman, M. Maggioni, Diffusion wavelets, Applied and Computational Harmonic Analysis 21 (1) (2006) 53–94.
[33] M. Crovella, E. Kolaczyk, Graph wavelets for spatial traffic analysis, in: INFOCOM, 2003.
[34] I. Ram, M. Elad, I. Cohen, Generalized tree-based wavelet transform, IEEE Transactions on Signal Processing 59 (9) (2011) 4199–4209.
[35] S. K. Narang, A. Ortega, Perfect reconstruction two-channel wavelet filter banks for graph structured data, IEEE Transactions on Signal Processing 60 (6) (2012) 2786–2799.
[36] S. K. Narang, Y. H. Chao, A. Ortega, Graph-wavelet filterbanks for edge-aware image processing, in: Statistical Signal Processing Workshop (SSP), IEEE, 2012.
[37] W.-S. Kim, S. K. Narang, A. Ortega, Graph based transforms for depth video coding, in: ICASSP, 2012.
[38] A. Sandryhaila, J. M. F. Moura, Nearest-neighbor image model, in: ICIP, 2012.
[39] H. E. Egilmez, A. Ortega, Spectral anomaly detection using graph-based filtering for wireless sensor networks, in: ICASSP, 2014.
[40] S. Chen, F. Cerda, P. Rizzo, J. Bielak, J. Garrett, J. Kovacevic, Semi-supervised multiresolution classification using adaptive graph filtering with application to indirect bridge structural health monitoring, IEEE Transactions on Signal Processing 62 (11) (2014) 2879–2893.
[41] N. Leonardi, D. Van De Ville, Wavelet frames on graphs defined by fmri functional connectivity, in: ISBI, 2011.
[42] X. Dong, A. Ortega, P. Frossard, P. Vandergheynst, Inference of mobility patterns via spectral graph wavelets, in: ICASSP, 2013.
[43] T. Kerola, N. Inoue, K. Shinoda, Spectral graph skeletons for 3D action recognition, in: Asian Conference of Computer Vision (ACCV), Singapore, Nov 1-5, 2014, pp. 417–432.
[44] G. Salton, M. J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, Inc., New York, NY, USA, 1986.
[45] C.-C. Chang, C.-J. Lin, Libsvm: a library for support vector machines, ACM Transactions on Intelligent Systems and Technology (TIST) 2 (3) (2011) 27.
[46] M. A. Gowayyed, M. Torki, M. E. Hussein, M. El-Saban, Histogram of oriented displacements (hod): describing trajectories of human joints for action recognition, in: IJCAI, 2013.
[47] W. Li, Z. Zhang, Z. Liu, Action recognition based on a bag of 3d points, in: CVPR Workshops, 2010.
[48] X. Yang, C. Zhang, Y. Tian, Recognizing actions using depth motion maps-based histograms of oriented gradients, in: ACM MM, 2012.
[49] A. W. Vieira, E. R. Nascimento, G. L. Oliveira, Z. Liu, M. F. Campos, Stop: Space-time occupancy patterns for 3d action recognition from depth map sequences, in: CIARP, 2012.
[50] M. Zanfir, M. Leordeanu, C. Sminchisescu, The moving pose: An efficient 3d kinematics descriptor for low-latency action recognition and detection, in: ICCV, 2013.
[51] X. Yang, Y. Tian, Eigenjoints-based action recognition using naive-bayes-nearest-neighbor, in: CVPR Workshops, 2012.
[52] C. Ellis, S. Z. Masood, M. F. Tappen, J. J. Laviola Jr, R. Sukthankar, Exploring the trade-off between accuracy and observational latency in action recognition, International Journal of Computer Vision 101 (3) (2013) 420–436.
[53] J. Wang, X. Nie, Y. Xia, Y. Wu, S.-C. Zhu, Cross-view action modeling, learning and recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2649–2656.
[54] H. Rahmani, A. Mian, 3d action recognition from novel viewpoints, in: CVPR, June, 2016.
[55] J. Yang, K. Yu, Y. Gong, T. Huang, Linear spatial pyramid matching using sparse coding for image classification, in: CVPR, 2009.
[56] D. Thanou, D. I. Shuman, P. Frossard, Parametric dictionary learning for graph signals, in: IEEE GlobalSIP, 2013.
[57] T. Batabyal, A. Vaccari, S. T. Acton, Ugrasp: A unified framework for activity recognition and person identification using graph signal processing, in: ICIP, 2015.
[58] G. Strang, Introduction to linear algebra, Wellesley-Cambridge Press, Wellesley (Mass.), 2009.
[59] F. R. Chung, Spectral graph theory, Vol. 92 of CBMS Regional Conference Series in Mathematics, American Mathematical Soc., 1997.
[60] G. M. Phillips, Interpolation and approximation by polynomials, Vol. 14, Springer Science & Business Media, 2003.
[61] N. Tremblay, P. Borgnat, Graph wavelets for multiscale community mining, Signal Processing, IEEE Transactions on 62 (20) (2014) 5227–5239.
[62] M. Elad, Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing, 1st Edition, Springer Publishing Company, Incorporated, 2010.
[63] N. Inoue, K. Shinoda, A fast and accurate video semantic-indexing system using fast map adaptation and gmm supervectors, Multimedia, IEEE Transactions on 14 (4) (2012) 1196–1205.
[64] X. Zhu, M. Rabbat, Approximating signals supported on graphs., in: ICASSP, 2012.
[65] G. Yu, Z. Liu, J. Yuan, Discriminative orderlet mining for real-time recognition of human-object interaction, in: Asian Conference on Computer Vision, Springer, 2014, pp. 50–65.
[66] J. R. Uijlings, K. E. van de Sande, T. Gevers, A. W. Smeulders, Selective search for object recognition, International journal of computer vision 104 (2) (2013) 154–171.
[67] F. Lv, R. Nevatia, Recognition and segmentation of 3-d human action using hmm and multi-class adaboost, in: Computer Vision–ECCV 2006, Springer, 2006, pp. 359–372.
[68] M. Müller, T. Röder, Motion templates for automatic classification and retrieval of motion capture data, in: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation, Eurographics Association, 2006, pp. 137–146.
[69] X. Dong, D. Thanou, P. Frossard, P. Vandergheynst, Laplacian matrix learning for smooth graph signal representation, in: Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on, IEEE, 2015, pp. 3736–3740.
[70] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE transactions on acoustics, speech, and signal processing 26 (1) (1978) 43–49.

[71] S. Nakagawa, H. Nakanishi, Speaker-independent english consonant and japanese word recognition by a stochastic dynamic time warping method, IETE Journal of Research 34 (1) (1988) 87–95.

[72] K. Kulkarni, G. Evangelidis, J. Cech, R. Horaud, Continuous action recognition based on sequence alignment, International Journal of Computer Vision 112 (1) (2015) 90–114.

[73] K. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, When is "nearest neighbor" meaningful?, in: International conference on database theory, Springer, 1999, pp. 217–235.

[74] L. Li, B. A. Prakash, Time series clustering: Complex is simpler!, in: Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011, pp. 185–192.

[75] A. Gupta, J. Martinez, J. Little, R. Woodham, 3d pose from motion for cross-view action recognition via non-linear circulant temporal encoding, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2601–2608.

[76] Q. McNemar, Note on the sampling error of the difference between correlated proportions or percentages, Psychometrika 12 (2) (1947) 153–157.

[77] L. Xia, J. Aggarwal, Spatio-temporal depth cuboid similarity feature for activity recognition using depth camera, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 2834–2841.

[78] W. Ding, K. Liu, F. Cheng, J. Zhang, Learning hierarchical spatio-temporal pattern for human activity prediction, Journal of Visual Communication and Image Representation 35 (2016) 103–111.