論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	A Graph Spectral Approach to Human Action Recognition from Depth Maps
著者(和文)	KerolaTommi Matias
Author(English)	Tommi Matias Kerola
出典(和文)	学位:博士(学術), 学位授与機関:東京工業大学, 報告番号:甲第10383号, 授与年月日:2016年12月31日, 学位の種別:課程博士, 審査員:篠田 浩一,徳永 健伸,小池 英樹,藤井 敦,下坂 正倫
Citation(English)	Degree:Doctor (Academic), Conferring organization: Tokyo Institute of Technology, Report number:甲第10383号, Conferred date:2016/12/31, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

A Graph Spectral Approach to Human Action Recognition from Depth Maps

Tommi Kerola

Supervised by Professor Koichi Shinoda

Department of Computer Science
Graduate School of Information Science and Engineering
Tokyo Institute of Technology

Dissertation submitted to the Tokyo Institute of Technology
for the degree of Doctor of Philosophy

November 2016

Abstract

In this dissertation, 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. While this study focuses on action recognition, the proposed framework can in general be applied to any time series of graphs.

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

Tommi Kerola November 2, 2016

Acknowledgments

I would like to sincerely thank my academic adviser, Prof. Koichi Shinoda, for introducing me to the field of graph signal processing, which led to a, in my opinion, very interesting thesis topic. His constructive criticism throughout this study has been highly appreciated, as it has led to me becoming a better researcher, and more self-aware of my flaws. I am also grateful for him always having taken the time to thoroughly answer my questions about research, reading paper drafts, and discussing experimental results, regardless of the time of the day. The stay in the research lab has been really great, and was also my introduction to the fascinating world of computer vision and statistical pattern recognition back in 2011. I am grateful that I got the chance be an exchange student in the same lab back then, in order to study the fundamentals of this field a long time ago. It was truly a life-changing moment, as without that experience, I am certain that I would not have been writing this thesis.

I am also grateful for the advice from Assistant Prof. Nakamasa Inoue, whose useful suggestions led to improved results. Moreover, I am thankful for his help with solving and debugging problems, reading paper drafts and this thesis.

A word of thanks goes also to my thesis committee: Prof. Takenobu Tokunaga, Prof. Hideki Koike, Associate Prof. Atsushi Fujii and Associate Prof. Masamichi Shimosaka. Their insightful and useful suggestions helped improve the quality of the thesis.

This study was generously supported by the Japanese Government (Monbukagakusho: MEXT) scholarship. I wish to express great gratitude to MEXT for providing this opportunity to conduct research in Japan. Carrying out this research would have been impossible without their support.

I am thankful towards the anonymous reviewers in ACCV and CVIU for their helpful suggestions that helped improve the presentation of this work.

Moreover, I am glad to have been part of a lab with a great group of lab members. Thank you for all the fun and interesting discussions, lunches and parties. Especially, I would like to thank Dr. Johan Rohdin, Dr. Ryan Price, Dr. Yuan Liang, Dr. Sangeeta Biswas and Dr. Felipe Gómez-Caballero for bearing with my many questions and teaching me much of the basic

knowledge when I was new to this research field.

I would also like to thank all the people I met during my internship at Rakuten Institute of Technology. While the results in this thesis are not related to the work done during the internship, several of the techniques and approaches toward conducting research learnt there have been significant for achieving the results in the latter part of the PhD program. Especially, I would like to thank Dr. Tony Tung, Dr. Jiu Xu, Dr. Björn Stenger, Dr. Mitsuru Nakazawa and Dr. Robin Swezey for all the interesting discussions and enjoyable times.

Thank you, all my friends, for all the adventures and making stressful times less stressful. Especially, I would like to thank Linus and Yasu, for all the onsen trips, which is without doubt one of the best ways to relax after doing research.

I would like to thank my family in Sweden for their never-ending encouragement during this study, despite being located on the other side of the world.

Finally, thank you Yuri, for having stayed by my side during all these years of study in Japan. It would not have been the same without you.

Dedication

This dissertation is dedicated to my parents, Ilkka and Sari.

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 $\mathrm{W}(i, j)$.

Contents

List of Figures

List of Tables

1

Introduction

SINCE the beginning of the industrial revolution in the late 18th century, humans have benefitted from the autonomous help gained by machines for alleviating mundane tasks in daily life. Recently, however, advancements in computer vision and machine learning research has pushed the limits of what machines are capable of to the next level. This level of performance includes autonomous understanding of speech, images and videos. Such multimedia content naturally includes information about human behavior.

Autonomous understanding of human behavior has been, and is still, an ambitious goal for the computer vision research community. The task of human action recognition is to develop algorithms for understanding what action is taking place, given a sequence of video frames. Problems such as subject segmentation, lighting, and inter-subject variations make this task challenging at first glance. Successfully solving all involved problems would result in large benefits for several industries, such as health care, games, surveillance and the security industry.

Traditionally, human action recognition has been attempted on standard RGB videos. Such approaches, however, encounter several hardships due to problems such as background clutter, illumination variations, and action intra-class variations. These problems have been remedied to some extent by the recent availability of affordable depth cameras for a reasonable price. Such devices includes the Microsoft Kinect, which has contributed to a recent surge in research using RGB-D data [38]. The Kinect and similar devices solve the problem of video subject-background separation, while also being robust against varying illumination, due to the usage of infrared sensors for capturing the depth data. Further, the nominal work of Shotton *et al.* [88] has led to ready access of tracked 3D skeleton joints of low dimension that are mapped to points on the human body. Compared to the raw RGB-D data, the tracked skeleton joints offer a much more discriminative description of the human body, which has led to the possibility of efficient, real-time methods for human action recognition. However, the skeleton joint estimates are not perfect and often subject to large amounts of noise, which has created a new set of challenges for utilizing the joints in an optimal way for recognizing actions.

In machine learning, it is often debated how a certain object should be modeled or represented for best suiting the task of pattern recognition. Typically, the debate is divided into two camps: approaches that are statistical, and those that are structural [14]. Statistical approaches are very popular and have previously enjoyed a great amount of attention. In this dissertation, however, we ask ourselves if an explicit structure suitable to the task at hand is better for representing an object, such as a human action. Generally, human actions can be defined as a sequence of interactions between several interest points [44]. One example is “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.

Of course, in order to capture the characteristics of an action, an ideal action descriptor needs to capture the interactions that occur between several different parts of the body. Further, these interactions do typically have significant temporal variation throughout the course of the performed action. Although previously proposed 3D action recognition methods do capture these interactions [72, 60, 100, 101, 118], the majority of them are view-dependent. This means that the viewing angle of the camera towards the subject plays a major role in the resulting action recognition performance. Cross-view action recognition is the task of recognizing an action from any possible viewing angle, independent of the angles used for recording the videos in the training dataset. In the case of RGB videos, this problem has been previously researched to some extent [105, 112, 55, 73, 80, 104, 76, 32, 57, 117, 119]. For 3D action recognition methods, however, the number of approaches that are view-independent using pure depth data are much fewer [78, 108, 102]. In our opinion, this is unfortunate, since the action recognition performed using only depth data has the added advantage of protecting the identity of the user, which is often essential for health care applications.

In this work, we are interested in leveraging graphs for action sequence representation due to the following reasons. Firstly, interactions between interest points can be naturally modeled by a graph, as it captures pair-wise relations between points. Second, due to graphs capturing only pair-wise information, a graph is able to provide a representation that is invariant to viewpoint changes, assuming that this holds for any signal that is defined on the vertices of the graph. This summarizes our motivation for using graphs for the task of 3D human action recognition.

In the real world and its related problems, it is easy to find graph representations that occur in various shapes and appearances. Some examples are finite state machines, and social- and transportation networks. Graphs are also abundant in computer graphics and the brain fMRI domain. Machine learning has also turned to graphs before, where recent examples include graph kernels [13, 121, 39, 45, 20, 50, 92, 87], as well as graph signal processing frameworks [89, 84] and graph wavelets [37, 19, 23, 79, 67].

The field of graph signal processing is quite recent. Although much is based on results from spectral graph theory from the 90’s [18], many exciting results in the field have appeared in roughly the past ten years [89]. Graph signal processing generalizes classical signal processing methods to arbitrary graphs and allows signals to be propagated in a manner that is true to the intrinsic structure of an object. Examples of applications include mobility pattern prediction [27], brain functional connectivity analysis [54], bridge structure health monitoring [16], anomaly detection in wireless sensor networks [29], edge-aware image processing [68], depth video coding [48] and image compression [83].

Our primary interest in leveraging graph signal processing for the task of 3D human action recognition is due to the graph frequency information gained from using such methods. As our findings will show in the latter part of this work, wavelet transform generalizations onto graphs enables us to gather information about the interactions between interest points on a depth map at multiple scales. Together with knowledge of the temporal propagation of the points and their interactions, we will derive an efficient method that is able to classify and recognize a large set of human actions.

In this study, we propose an efficient system for 3D human action recognition from depth maps, based on graph signal processing techniques. We represent an action as a time series using a novel graph structure, which is further robust against viewpoint changes. The main contributions in this dissertation are the following:

- We present a framework for the task of 3D human action recognition based on graph signal processing. Contrary to previous approaches based on *e.g.* only temporal pyramids, the proposed method is capable of capturing interest point interactions throughout time due to the novel graph structure [46].

- We propose a simple, but efficient skeleton rotation cancellation method based on Gram-Schmidt orthonormalization [91].

- We present a graph-based action representation that is view-invariant due to the graph structure. Our approach is shown to significantly advance the performance compared to previous state-of-the-art methods for the task of cross-view action recognition [47].

- We present an efficient algorithm for doing the calculation of the spectral graph wavelet transform. Our algorithm is memory-efficient due to it taking explicit advantage of the block sparsity structure of our graph representation [47].

Specifically, our methods works as follows. Given a set of interest points gotten from a depth map, we embed the points on an augmented graph that is capable of describing the temporal progression of the points. We conduct investigation on two types of interest point candidates:

- Tracked skeleton joints. These capture the subject's pose, and additionally associates a semantic label to each body part.
- Spatio-temporal keypoints. These capture the interaction between humans and objects, as well as other fine intrinsic detail.

Based on the above interest points, we define graph signals that are view-invariant, and represent the signals using a novel graph structure. Our graph structure is shown to outperform several more traditional representations, such as bag-of-words (BoW) [82] combined with a support vector machine (SVM) [15]. In particular, for analyzing the signal on the graph, we turn to the recently proposed spectral graph wavelet transform (SGWT) framework of Hammond *et al.* [37]. Through the SGWT, we create a multi-scale representation of the extracted interest points. The graph wavelets of the SGWT are capable of capturing multi-scale information about a signal. The information is captured in several dimensions on the augmented temporal graph, and the signal propagates throughout both between interest points and time. Compared to classical wavelets, spectral graph wavelets follow the graph design, and provide a more flexible representation directly adapted to the graph structure. Subsequently, we leverage a temporal pyramid pooling scheme [60, 34, 101] on the wavelet coefficients for capturing an action's sequential behavior. In comparison to approaches that use only global information [58, 111], our method gains better performance, as it captures information about temporal order and dependencies along several temporal segmentation levels. After pyramid pooling, classification is done using a standard SVM classifier.

The method proposed in this dissertation has the following advantages:

- Our method uses a graph with an explicit sparse block structure, which we exploit to create a memory-efficient algorithm for doing the SGWT coefficient calculation (see Sec. 5.2.1).
- Our method creates a feature descriptor that has a mathematically well-defined underlying spectral basis [37]. This, in turn, enables us to perform analysis of the effects of the proposed feature descriptor. This is not true for other approaches, such as methods based on *e.g.* sparse

coding [60] or deep learning [76]. Both sparse coding and deep learning tend to produce bases that do not submit easily to analysis (see Sec. 5.8).

- 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. Compared to approaches that are based on solving large optimization problems [60, 100], our method is computable in a more efficient manner (see Sec. 5.6).

- 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 Chapter 6).

Although this study focuses on 3D human action recognition, the proposed framework is general enough to be applied to other problems where the input can be formulated as a time series of graphs.

The remainder of this dissertation is organized as follows. Chapter 2 introduces the task of human action recognition, and reviews previous approaches towards solving the task. Chapter 3 discusses several important concepts in graph signal processing, and reviews related research. Chapter 4 discusses how to represent actions as graphs. Our proposed method is then presented in Chapter 5. Experimental evaluation is presented in Chapter 6. Finally, conclusions and future work are discussed in Chapter 7.

2

3D Human Action Recognition

In this chapter, the task of human action recognition will be introduced, along with fundamental approaches towards solving the task. Additionally, problems to solve for creating a good action recognition system will be discussed, as well as a survey of previous research.

Words such as “action”, “gesture” and “activity” are often used interchangeably for referring to either the same or different visual things humans do in daily life. To clarify the focus of this study, we turn to a taxonomy similar to previous studies [96, 3], and define three levels of abstraction for human behavior:

Gesture

A basic simple movement, such as lifting your arm.

Action

A combination of gestures, such as waving or doing a tennis swing. Possible interaction with single object.

Activity

A combination of actions. This can include complex interactions between multiple persons and objects, such as playing basketball or ballroom dancing.

Naturally, there is no clear definite border that divides these categories from each other; there is a significant gray area between different types of human behavior. For example, consider the gestures of a music conductor, which requires rigorous training. Are these movements to be classified as simple gestures despite the inherent difficulty of the task? Nevertheless, the above definitions provide a weak guidance, where in general actions tend to be around 2-10 seconds long, gestures much shorter, and activities even longer (example actions can be seen in Fig. 2.1).

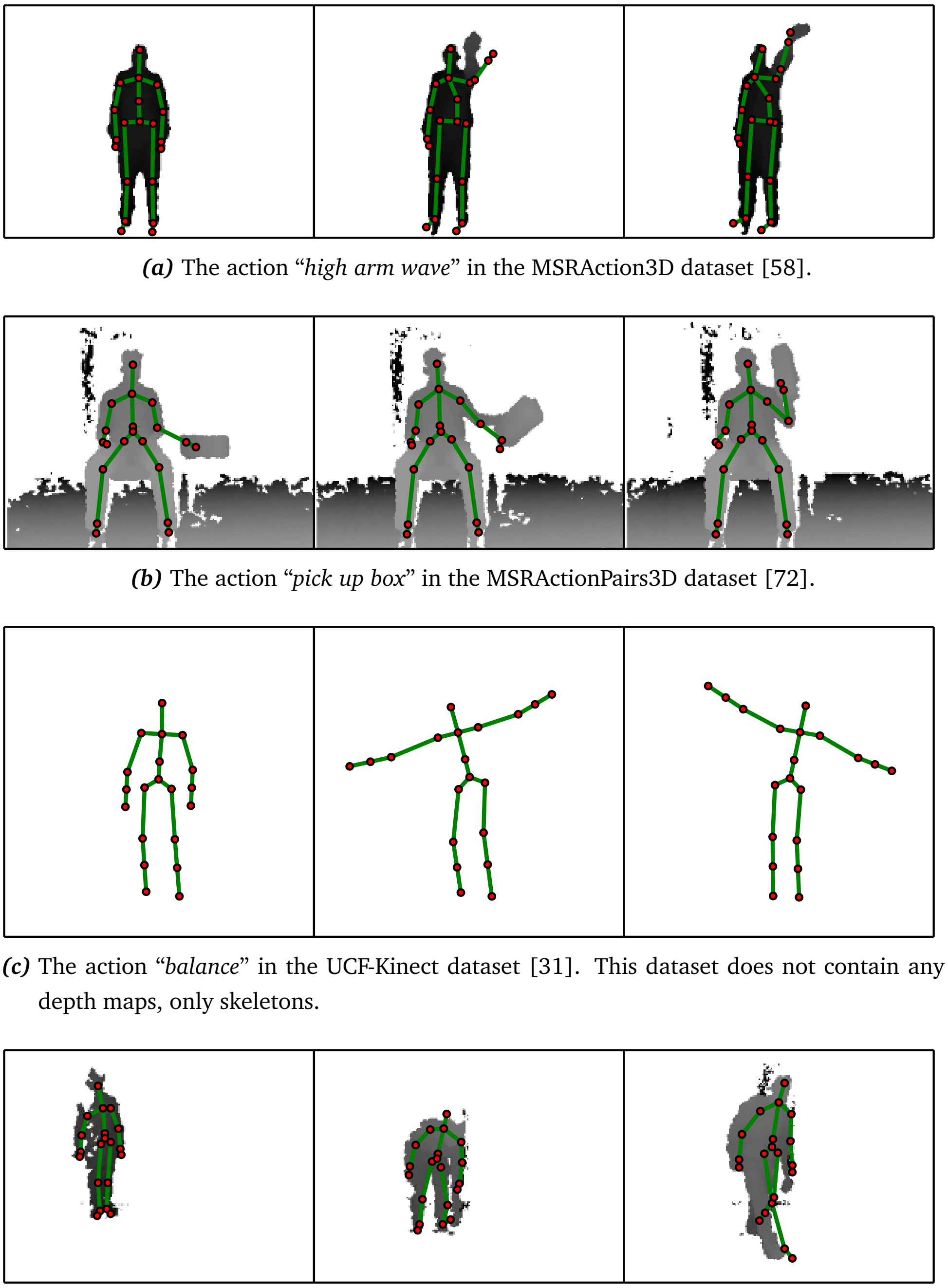
In the remainder of this chapter, we will state the limitations set for this study, present a general framework for action recognition, as well as discuss potential applications, and finally conclude with describing previous approaches towards solving this task.

2.1 Limitations

We limit ourselves in this dissertation to focus on the recognition of actions, and do not perform any *explicit* detection of interaction between objects or multiple people. [1] Therefore, we seek to create a system that is able to implicitly explore the interaction between several gestures or body parts in order to recognize actions. Further, multi-action classification is out of the scope of this study; classifying a single action is already a challenging task. Therefore, we do not perform any action detection or localization [17, 90, 106], and assume that each input video only contains one single action.

Further, we choose to consider only action recognition using depth cameras due to the following reasons. Traditionally, human action recognition has been attempted on standard RGB videos. Such approaches, however, encounter several hardships due to problems such as background clutter, illumination variations, and action intra-class variations. These problems have been remedied to some extent by the recent availability of depth cameras for a reasonable price. Such devices includes the Microsoft Kinect, which has contributed to a recent surge in research using RGB-D data [38]. The

[1] Although we do not target explicit human-object interaction, some of the experiments in Chapter 6 include object interaction, such as "*pick up box*". We emphasize here that our system primarily intends to recognize the movement of picking up *something*, and will not focus on distinguish between picking up a *box* and a *pen*, as this could be solved by an explicit object recognition method. Of course, if the interest points are altered by the shape of objects, then our method will be able to capture *implicit* human-object interaction.



(a) The action "*high arm wave*" in the MSRAction3D dataset [58].

(b) The action "*pick up box*" in the MSRActionPairs3D dataset [72].

(c) The action "*balance*" in the UCF-Kinect dataset [31]. This dataset does not contain any depth maps, only skeletons.

(d) The action "*pick up with one hand*" in the N-UCLA Multiview Action3D dataset [103].

Figure 2.1: Example depth frames and skeletons of actions from datasets explored in this study. Each skeleton consists of joints (red) and limbs (green), connecting the joints together. Note that the tracked skeletons are not always following the underlying raw depth data closely and are subject to noise.

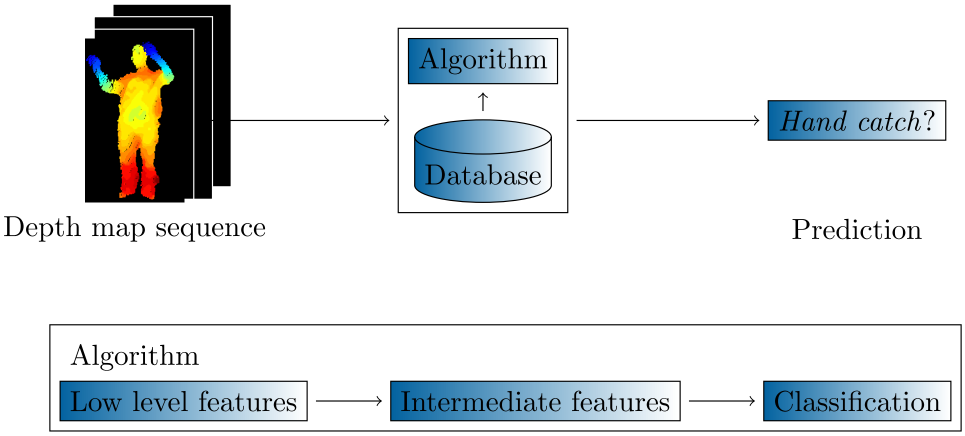


Figure 2.2: General framework for human action recognition. The input to our system is a sequence of depth maps, gotten by a depth camera such as the Microsoft Kinect. These are then fed to a database and an algorithm for doing action recognition. The algorithm consists of three main parts: low-level feature extraction, intermediate feature representation, and a classification result. At the end of the pipeline, we get the output label of the predicted action, such as "*hand catch*".

Kinect and similar devices solve the problem of video subject-background separation, while also being robust against varying illumination, due to the usage of infrared sensors for capturing the depth data. Depth cameras are currently quite affordable, and it is our belief that the future will only provide ever more consumer applications that take advantage of depth cameras. We are therefore able to ignore the problems faced by RGB cameras, and can focus on recognizing actions using only depth maps. We also note that depth cameras have the added advantage of being able to perform action recognition without compromising the identity of the user, which is essential for health care applications.

2.2 General Framework

The essential task solved by a human action recognition system is presented in Figure 2.2. Given an input video with unknown content, the task is to

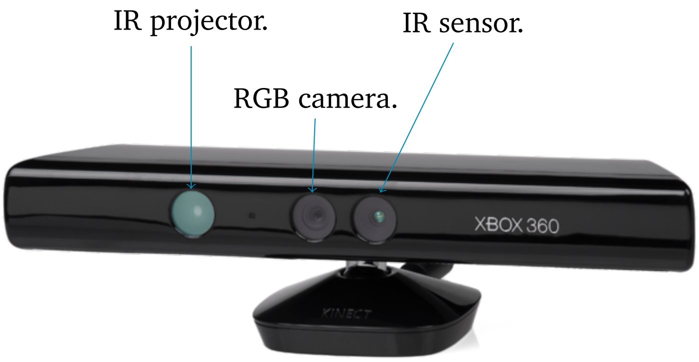


Figure 2.3: Microsoft Kinect depth camera [2]. The three circles are (from the left) infrared (IR) projector, RGB camera, and infrared sensor, respectively.

Table 2.1: Microsoft Kinect camera specifications [1].

View angle (hor. × ver.)	Frame rate	Depth range (near mode)
$57° \times 43°$	30 frames/sec.	0.8-4.0 m (0.4-3.0 m)

recognize any ongoing action that the system is aware of. The task consists of three basic steps. First, low-level features are extracted from the input video data that contains the action. Second, the low-level features are transformed into an intermediate feature representation that describes the whole action sequence. Finally, the intermediate feature representation is used to train a classifier, which learns a decision boundary for distinguishing between actions of different classes. In the following, we well discuss fundamental concepts applicable for each step.

2.2.1 Low-level Feature Extraction

Depth Data

The acquisition of reliable depth data measurements used to be difficult and expensive. For example, the SwissRanger SR-4000 time-of-flight camera [71] had a price tag of about $\$10,000$ in 2010. These problems have been remedied to some extent by the recent availability of affordable depth cameras for a reasonable price. Such devices includes the Microsoft Kinect

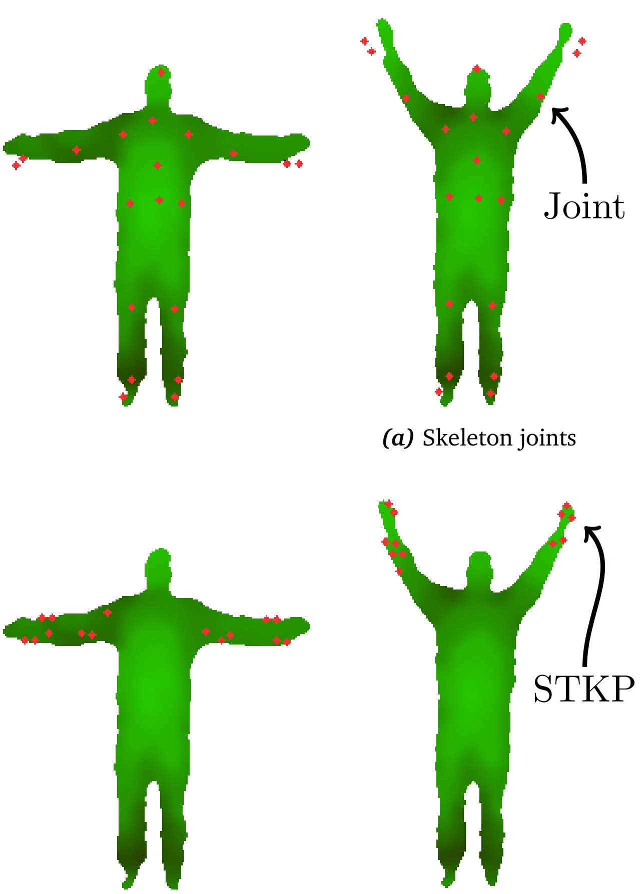
($\sim$ \$100), which has contributed to a recent surge in research using RGB-D data [38]. The Kinect camera uses an infrared laser projector in order to project a regular grid onto the 3D world (see also Fig. 2.3). A sensor is then used for calculating the deformation of the grid, from which the depth of each pixel in a 640×480 resolution image is predicted using 3D triangulation. Camera specifications can be seen in Table 2.1. The Kinect for Windows includes *near mode*, which is improved firmware for allowing depth data to be measured as close as 40 cm from the camera, at the cost of long range measurements.

Skeleton Joint Tracking

Recently, ground-breaking work of Shotton *et al.* [88] has led to ready access to tracked 3D skeleton joints of low dimension that are mapped to points on the human body. Their method predicts 31 intermediate body parts from a single depth image, by formulating the task as a per-pixel classification problem. That is, each pixel in the depth map is mapped to either a specific body part or the background. Local modes of the body parts are then found by using mean shifts, which are used as skeleton joint proposals. Specifically, given a depth image **I** they use a simple depth feature

$$f(\mathbf{I}, \mathbf{p}) = d_{\mathbf{I}}\left(\mathbf{p} + \frac{\mathbf{u}}{d_{\mathbf{I}}(\mathbf{p})}\right) - d_{\mathbf{I}}\left(\mathbf{p} + \frac{\mathbf{v}}{d_{\mathbf{I}}(\mathbf{p})}\right) \ , \tag{2.1}$$

where $d_{\mathbf{I}}(\mathbf{p})$ is the depth at pixel $\mathbf{p}$ and $\boldsymbol{\theta} = (\mathbf{u}, \mathbf{v})$ are offset parameters that are learned in order to distinguishing between different body parts. For predicting the body parts using the above feature, a random forest classifier [12] is used. A random forest is a set of decision trees [75] that are trained on random subsets of the data. Each tree contains split and leaf nodes, where the leaf nodes store a learned probability distribution over body parts. Using a large database of synthesized body poses, a random forest classifier is learned using different parameters $\boldsymbol{\theta}$ at each split node. They train a set of 3 trees of depth 20 using a database of about 1 million images using a computer cluster with 1000 cores. Mean shift clustering [21] is then used to cluster the predicted body part pixels into a fixed set of 20 skeleton joints. Examples of tracked skeleton joints can be seen in Figures 2.1 and 2.4a.



(a) Skeleton joints

(b) Spatio-temporal keypoints

Figure 2.4: Low-level features used as basis for intermediate-level feature construction. The detected feature locations (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.

Spatio-temporal Keypoints

Rahmani *et al.* [78] recently proposed spatio-temporal keypoints (STKP), a descriptor robust to view changes that describes the spatio-temporal shape of a 3D point cloud.

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. 2.4b).

For detecting keypoints, first histograms of oriented principal components (HOPC) [78] are calculated 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. Specifically, we create a scatter matrix $\mathbf{C}$ of all the points $\mathbf{q} \in \Omega(\mathbf{p})$:

$$\mathbf{C} = \frac{1}{|\Omega(\mathbf{p})|} \sum_{\mathbf{q} \in \Omega(\mathbf{p})} (\mathbf{q} - \tilde{\mathbf{q}})(\mathbf{q} - \tilde{\mathbf{q}})^T \,, \tag{2.2}$$

where $\tilde{\mathbf{q}}$ is the mean of the points in $\Omega(\mathbf{p})$. We subsequently take the eigendecomposition of $\mathbf{C}$ to get $\mathbf{C}\mathbf{u}_k = \lambda_k \mathbf{u}_k, \ k = 1, \ldots, 3$. The eigenvalues satisfy $\lambda_1 \geq \lambda_2 \geq \lambda_3$ and describe the magnitude of the principal directions of maximum variance in the point cloud. The smallest principal axis $\mathbf{u}_3$ is the least squares estimate for the surface normal, which renders HOPC more robust against noise than surface-normal methods based on depth gradients [72].

Next, each eigenvector is binned into a set of 20 directions, described by a regular icosahedron (see Figure 2.5). This polytype has 20 vertices

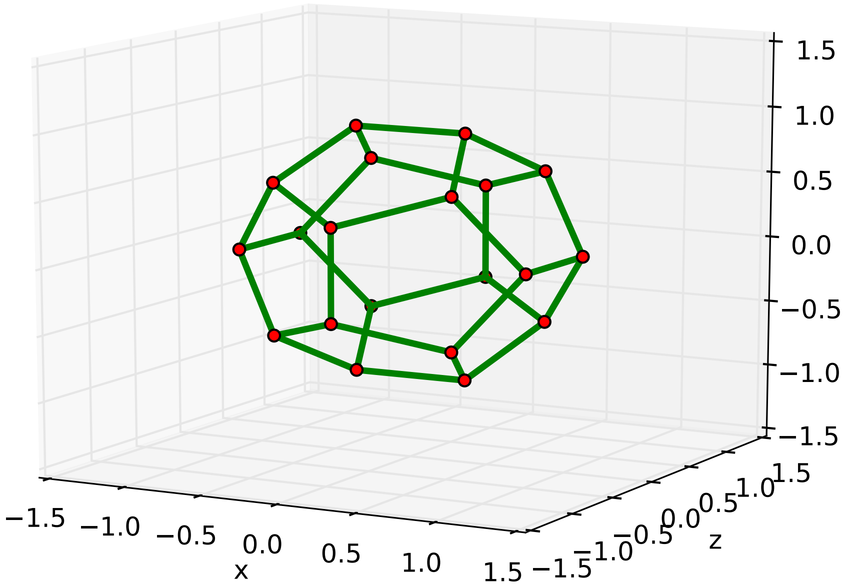


Figure 2.5: An icosahedron is a 3D polytype with 20 vertices. For creating the HOPC descriptor, each eigenvector is quantized into 20 bins represented by each vertex.

$\{\mathbf{v}_i\}_{i=1,\ldots,20}$, each with the coordinates

$$\begin{array}{llllll}
\mathbf{v}_1 = & \left[\frac{-1}{z},\frac{-1}{z},\frac{-1}{z}\right], & \mathbf{v}_2 = & \left[\frac{-1}{z},\frac{-1}{z},\frac{1}{z}\right], & \mathbf{v}_3 = & \left[\frac{-1}{z},\frac{1}{z},\frac{-1}{z}\right] \\
\mathbf{v}_4 = & \left[\frac{-1}{z},\frac{1}{z},\frac{1}{z}\right], & \mathbf{v}_5 = & \left[\frac{1}{z},\frac{-1}{z},\frac{-1}{z}\right], & \mathbf{v}_6 = & \left[\frac{1}{z},\frac{-1}{z},\frac{1}{z}\right] \\
\mathbf{v}_7 = & \left[\frac{1}{z},\frac{1}{z},\frac{-1}{z}\right], & \mathbf{v}_8 = & \left[\frac{1}{z},\frac{1}{z},\frac{1}{z}\right], & \mathbf{v}_9 = & \left[0,\frac{-\phi^{-1}}{z},\frac{-\phi}{z}\right] \\
\mathbf{v}_{10} = & \left[0,\frac{-\phi^{-1}}{z},\frac{\phi}{z}\right], & \mathbf{v}_{11} = & \left[0,\frac{\phi^{-1}}{z},\frac{-\phi}{z}\right], & \mathbf{v}_{12} = & \left[0,\frac{\phi^{-1}}{z},\frac{\phi}{z}\right] \\
\mathbf{v}_{13} = & \left[\frac{-\phi^{-1}}{z},\frac{-\phi}{z},0\right], & \mathbf{v}_{14} = & \left[\frac{-\phi^{-1}}{z},\frac{\phi}{z},0\right], & \mathbf{v}_{15} = & \left[\frac{\phi^{-1}}{z},\frac{-\phi}{z},0\right] \\
\mathbf{v}_{16} = & \left[\frac{\phi^{-1}}{z},\frac{\phi}{z},0\right], & \mathbf{v}_{17} = & \left[\frac{-\phi}{z},0,\frac{-\phi^{-1}}{z}\right], & \mathbf{v}_{18} = & \left[\frac{-\phi}{z},0,\frac{\phi^{-1}}{z}\right] \\
\mathbf{v}_{19} = & \left[\frac{\phi}{z},0,\frac{-\phi^{-1}}{z}\right], & \mathbf{v}_{20} = & \left[\frac{\phi}{z},0,\frac{\phi^{-1}}{z}\right], & &
\end{array} \tag{2.3}$$

where $\phi = (1+\sqrt{5})/2$ denotes the golden ratio, and $z = \sqrt{\phi^2 + \phi^{-2}}$ is factor for making all $\mathbf{v}_i$ of unit length. Since the variance represented by an eigenvector $\mathbf{u}_k$ is symmetric along the principal axis, the sign of each eigenvector is determined by voting as

$$\mathbf{u}_k \leftarrow \mathbf{u}_k \text{sign}\left(\sum_{\mathbf{q} \in \Omega(\mathbf{p})} \text{sign}(r_{q,k}) r_{q,k}^2 \right) , \tag{2.4}$$

where $r_{q,k} = (\mathbf{q} - \mathbf{p})^T \mathbf{u}_k$ and sign : $\mathbb{R} \rightarrow \{-1, 1\}$ returns the sign of the input.

Finally, the binning of the eigenvectors is done using soft thresholding by projecting onto each vertex $\mathbf{v}_j$ as

$$\zeta_{j,k} = \max(\mathbf{u}_k^T \mathbf{v}_j - \psi, 0), \ \forall j \ , \forall k \ , \tag{2.5}$$

where $\psi = (\phi + \phi^{-1})/z^2$ is a threshold value for sparsifying the projection. By gathering the thresholded projections per eigenvector as $\boldsymbol{\zeta}_k = [\zeta_{1,k}, \ldots, \zeta_{20,k}]$, the HOPC descriptor $\mathbf{h} \in \mathbb{R}^{60}$ is calculated by

$$\mathbf{h} = \left[\frac{\lambda_1 \boldsymbol{\zeta}_1}{||\boldsymbol{\zeta}_1||_2}; \frac{\lambda_2 \boldsymbol{\zeta}_2}{||\boldsymbol{\zeta}_2||_2}; \frac{\lambda_3 \boldsymbol{\zeta}_3}{||\boldsymbol{\zeta}_3||_2} \right] , \tag{2.6}$$

which describes the distribution of the principal axes of variance in the point cloud around point $\mathbf{p}$.

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

$$d_{\chi^2}(\mathbf{h}_s, \mathbf{h}_{st}) = \sum_j \frac{(\mathrm{h}_s(j) - \mathrm{h}_{st}(j))^2}{\mathrm{h}_s(j) + \mathrm{h}_{st}(j)} \tag{2.7}$$

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.

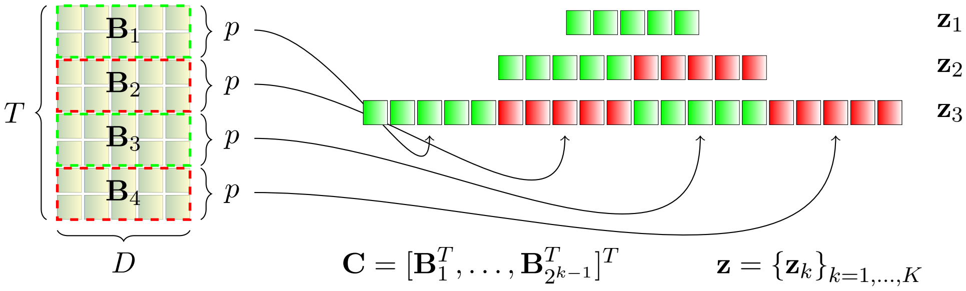


Figure 2.6: Temporal pyramid pooling example. A time series with T frames is represented by a feature matrix $\mathbf{C}$, which is then pooled by a function $p : \mathbb{R}^{t \times D} \rightarrow \mathbb{R}^D$ into a set of $K = 3$ pyramid levels. The creation of the pyramid level vector $\mathbf{z}_3$ at level 3 is shown by the arrows. Finally, by concatenating the pyramid level vectors $\{\mathbf{z}_k\}_{k=1,\ldots,K}$, the final feature vector $\mathbf{z}$ is created.

2.2.2 Intermediate Feature Representation

Temporal Pyramid Pooling

Analysis of time series data bring two problems. First, each input sequence often is of varying length, despite any subsequent classification step usually requiring feature vectors of fixed size. Second, knowledge of the temporal location of certain low-level features is often vital for correct understanding of the time series. For example, for distinguishing between two actions "*stand up*" and "*sit down*", a feature describing the up-right height of the subject might not help if the average height over the whole time sequence is used. But if we know the height at the beginning of the sequence, then distinguishing between the two classes becomes much easier.

In order to solve these two problems, the method of temporal pyramid pooling has been popular [60, 34, 101]. Pyramid pooling methods were first introduced for image classification tasks in order to capture multi-scale aggregate information about local features [35, 52, 109]. Temporal pyramid pooling follows essentially the same idea, but with the aggregation of features being done over time, instead of spatially.

The methods works as follows. Assume we have a matrix of features $\mathbf{C} \in \mathbb{R}^{t \times D}$ describing a set of t D-dimensional feature vectors, gathered from a time series of length t. As time series have varying length, a vector-valued pooling function $p : \mathbb{R}^{t \times D} \to \mathbb{R}^D$ is used for creating the feature vector $\mathbf{z} = p(\mathbf{C})$, where t is equal to the input matrix row count.

Let the maximum pyramid level be denoted by K. 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 $\{\mathbf{z}_k\}_{k=1,\ldots,K}$. The temporal pyramid pooling scheme being applied to $\mathbf{C}$ with $K = 3$ pyramid levels is visually illustrated in Fig. 2.6.

Bag of Words

A popular way of representing a set of low-level features in aggregate form is the bag of words (BoW) approach, which originally was used in the natural language processing community for text classification [43, 94, 59], but has since become a popular technique also in computer vision [24, 52, 5, 78]. Given a codebook $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_K]$ with a set of K codeword vectors, each feature $\mathbf{x}_i$ is assigned to the closest feature in the codebook according to Euclidean distance, creating a K-dimensional vector $\mathbf{c}_i$ defined as

$$\mathrm{c}_i(j) = \begin{cases} 1 & \text{if } j = \arg\min{}_k \|\mathbf{x} - \mathbf{d}_k\|_2^2 \\ 0 & \text{otherwise} \end{cases} . \tag{2.8}$$

We call $\mathbf{c}_i$ the *one-of-K* representation of $\mathbf{x}_i$. The aggregate representation of the set of features is then given by

$$\mathbf{c} = \sum_i \mathbf{c}_i \; . \tag{2.9}$$

K-means Clustering

Since the number of detected low-level features might vary for different depth maps, it is often desirable to find a clustering of the features $\mathbf{X}$ into a fixed-size set of clusters. A classic approach towards this is the K-means clustering algorithm [9]. A codebook $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_K]$ is learned by an iterative expectation-maximization (EM) procedure. First, some initial values for $\mathbf{d}_k$, $\forall k$ are chosen. Second, a one-of-K representation $\mathbf{c}_i$ is computed for each $\mathbf{x}_i$. Third, the codebook is updated by the average of the members in each cluster as

$$\mathbf{d}_k = \frac{\sum_i \mathrm{c}_i(k)\mathbf{x}_i}{\sum_i \mathrm{c}_i(k)}, \ \forall k \ . \tag{2.10}$$

The second and third step are then repeated until convergence; when the distortion measure

$$\sum_i \sum_k \mathrm{c}_i(k) \|\mathbf{x}_i - \mathbf{d}_k\|_2^2 \tag{2.11}$$

does not change more than a certain threshold. The final cluster memberships are then computed for each $\mathbf{x}_i$ using the one-of-K representation.

Principal Component Analysis

Principal component analysis [9] (PCA) is a standard technique for dimensionality reduction of a set of features, while keeping much of the information containing the largest variance.

Given a matrix $\mathbf{X} \in \mathbb{R}^{D \times N}$ describing a set of N D-dimensional features, the idea behind PCA is to project the feature vectors onto a new basis that preserves the maximum amount of variance. Accordingly, the data covariance matrix $\mathbf{C}$ is constructed:

$$\mathbf{C} = (\mathbf{X} - \hat{\mathbf{x}}\mathbf{1}^T)^T(\mathbf{X} - \hat{\mathbf{x}}\mathbf{1}^T) \ , \tag{2.12}$$

where $\hat{\mathbf{x}}$ is the mean feature vector in $\mathbf{X}$ and $\mathbf{1}$ is the vector of all ones of the appropriate size. In order to find a basis that represents the projected variance, an eigendecomposition can be done as

$$\mathbf{C} = \mathbf{V}\mathbf{D}\mathbf{V}^T \ , \tag{2.13}$$

where $\mathbf{V}$ is an orthogonal basis and $\mathbf{D}$ is a diagonal matrix containing the eigenvalues of $\mathbf{C}$. Essentially, the eigenvalues encode the amount of variance explained by each basis vector $\mathbf{v}_i$ in $\mathbf{V}$, so dimensionality reduction can be done by selecting a subset of the basis vectors in order to create a matrix $\hat{\mathbf{V}} \in \mathbb{R}^{D \times R}$ that can be used to project $\mathbf{X}$ into a more compact R-dimensional space. Note that in practice, eigendecomposition of $\mathbf{C}$ is not typically done, but rather $\mathbf{V}$ is gotten by the singular-value decomposition $\mathbf{X} - \hat{\mathbf{x}}\mathbf{1}^T = \mathbf{V}\mathbf{\Sigma}\mathbf{Q}^T$, avoiding explicit creation of the covariance matrix.

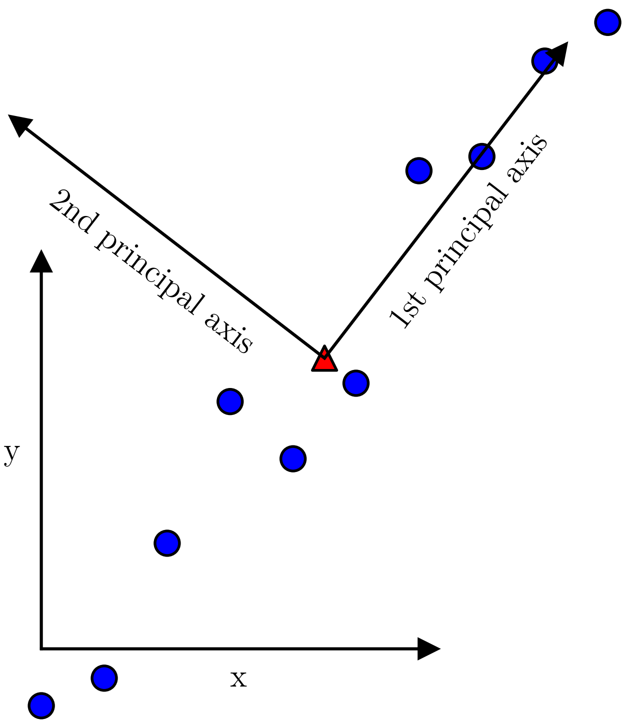


Figure 2.7: Principal component analysis (PCA) on toy data in 2D. The method is applied to the data matrix $\mathbf{X} \in \mathbb{R}^{2\times 20}$ (blue points), whose members clearly have an approximate linear relationship, subject to some noise. PCA reveals the data mean (red triangle) and the two principal component axes, which point along the directions of maximum variance in the data. Note that the second principal axis does not help much in explaining the linear behavior of the data (it explains the noise). We could simply drop the axis in order to get a one-dimensional representation of the data that gives us a better view of the underlying cause of the data using only the first principal axis.

2.2.3 Classification

Classification of the feature vectors into a specific class can be done using several methods. Popular ones include support vector machines, logistic regression and K-nearest neighbor classifiers [9]. The focus of this study is not to compare several classification approaches, so we simply choose to leverage a standard support vector machine.

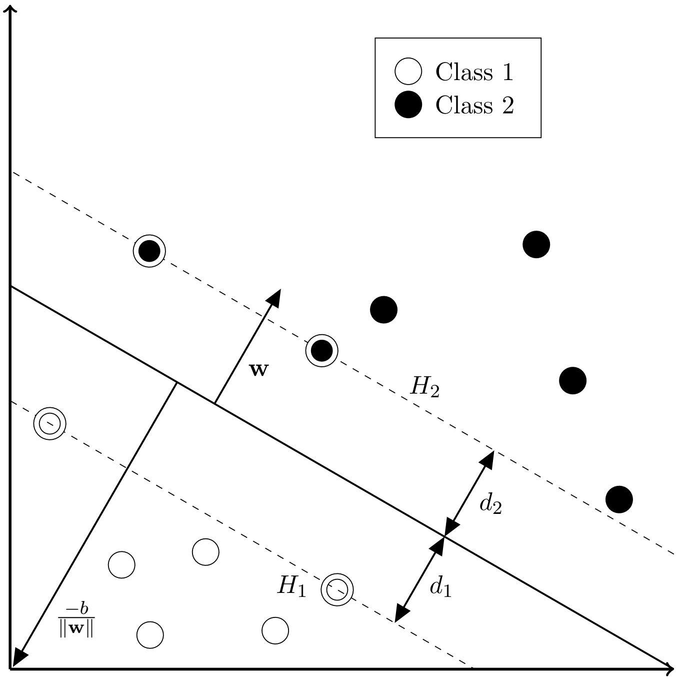


Figure 2.8: Two classes that are linearly separable are separated by a hyperplane. The normal of the hyperplane is denoted by $\mathbf{w}$ and the perpendicular distance from the origin to the hyperplane is given by $\frac{b}{\|\mathbf{w}\|}$. The support vectors lie on the hyperplanes H_1 and H_2. Note that the support vectors lie exactly on the margin boundary. The SVM margin is $d_1 = d_2$.

Support Vector Machine

A support vector machine [9] (SVM) is a linear binary classifier that finds a decision boundary between two classes of input vectors. The SVM is a supervised learning model, whose current form was studied early by Boser et al. [10]. Additional early studies were done by Cortes and Vapnik [22] in 1995, where they showed the SVM to in general have quite low error rate of test datasets (that is, high generalization ability). It is of crucial importance for a learning machine to have good generalization capabilities.

A classifier that simply memorizes the training data cannot be expected to work well on real-world data, as the data is assumed to be subject to noise. Memorizing the training data (called *overfitting* [9]) will then also memorize the noise or detail irrelevant for separating one class from another. An example of overfitting is learning that a *zebra* is something that has exactly 27 stripes on its back. The other extreme, *underfitting*, could be learning that a zebra is something with four legs. It is clear that neither of the two above classification rules are able to generalize and accurately represent of the concept of a zebra. We seek something in between, something that is able to generalize to examples not seen in the training set.

An SVM is trained to discover a hyperplane separating the set of input vectors $\mathcal{X} = \{\mathbf{x}\}$ into two classes using as a large margin as possible. The SVM hyperplane is formulated as

$$\mathbf{w}^T\mathbf{x} + b = 0 \; , \tag{2.14}$$

where $\mathbf{w}$ is the normal vector of the hyperplane and b is a bias term. The distance from the origin to the hyperplane is then $b/\|\mathbf{w}\|_2$. The concept of margin is illustrated in Figure 2.8. Let the training examples having the minimum distance to the hyperplane from each class be called *support vectors*. We define the margin to be the minimum distance from the hyperplane to the support vectors in both of the two classes. A test data vector $\mathbf{x}_{\text{test}}$ is classified into one of the two classes by

$$y_{\text{test}} = \text{sign}(\mathbf{w}^T\mathbf{x}_{\text{test}} + b) \; , \tag{2.15}$$

where sign : $\mathbb{R} \to \{-1, 1\}$ returns the sign of the input.

In order to use an SVM on test data, the SVM must be trained. That is, the hyperplane normal $\mathbf{w}$ must be learned from training data. Given a training data set $\{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,n}$, we solve the SVM primal problem

$$\mathbf{w} = \operatorname{argmin} f(\mathbf{w}) = \frac{\lambda||\mathbf{w}||^2}{2} + \ell_{emp}(\mathbf{w}) \; , \tag{2.16}$$

where λ is a hyperparameter, and $\ell_{emp}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell((\mathbf{x}_i, y_i); \mathbf{w})$. The loss function $\ell(\cdot)$ penalizes misclassified samples. A standard candidate for ℓ is the *hinge-loss*

$$\ell((\mathbf{x}_i, y_i); \mathbf{w}) = \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) \; , \tag{2.17}$$

which has been shown to have good generalization properties for SVMs [64].

Naturally, real-world data is noisy and not always linearly separable. The SVM can handle this case as well using the hyperparameter λ, by finding a hyperplane that does not necessarily separate the training data into two classes, but finds a large margin as possible, while keeping the total number of misclassified samples to a minimum. In the literature, some other works use an alternative hyperparameter C that scales $\ell_{emp}(\cdot)$. These can be related by $\lambda = \frac{1}{nC}$. [64] The parameter λ is *problem-dependent*, and must be set differently for each dataset used. In practice, an analytical solution for λ is usually not known, and it is thus set by cross-validation. [2] Generally, a small λ causes the SVM to severely penalize misclassification of samples. A small λ being optimal might therefore indicate that the problem itself is difficult, and perhaps a better feature vector representation is needed [64].

The SVM primal (2.16) has an alternative dual formulation, which can be solved using quadratic programming (QP) [11]. Using the Lagrangian duality [11], the primal can be explicitly translated into the dual

$$\boldsymbol{\alpha} = \arg\max \; f(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \ , \tag{2.18}$$

$$\text{subject to } 0 \leq \alpha_i \leq \frac{1}{\lambda n} \ , \tag{2.19}$$

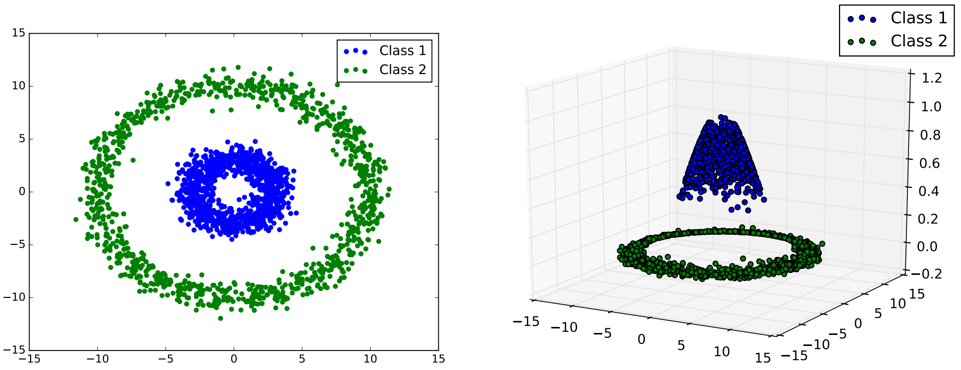
where $\boldsymbol{\alpha}$ is a dual variable to be optimized. Reconstruction of the hyperplane normal $\mathbf{w}$ is gotten by

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i . \tag{2.20}$$

This result is due to the *representer theorem* [49], which states that in some high-dimensional space, the optimal hyperplane is a linear combination of the training examples.

While we have so far discussed the SVM as a linear classifier, it can in fact be made non-linear by using the so-called *kernel trick*. A kernel is a function $K : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$, mapping two features vectors in a feature space $\mathbb{X}$ to a scalar value. Intuitively, the image of K represents how similar the two inputs are. The kernel K is commonly realized as the dot product

[2] K-fold cross-validation is a method for evaluating the generalization performance of a classifier when no explicit validation data is available. One part of the training set is left out and then the classifier is trained on the remaining $k-1$ parts. The performance of the classifier is then tested on the left-out part. By repeating the above procedure using k different left-out parts, we can get an estimate of the generalization capability of the classifier by measuring the average error rate of the left out parts [9].



(a) *(b)*

Figure 2.9: Example of non-linearly separable features in 2D space. By mapping the features using a non-linear map ϕ, the data becomes linearly separable in a higher-dimensional space. (a) Data in $\mathbb{R}^2$ that is linearly inseparable. (b) The same data mapped to $\mathbb{R}^3$ using a kernel, which makes it linearly separable.

$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$, where $\phi : \mathbb{X} \rightarrow \mathbb{H}$ is a map to a (possibly infinite dimensional) Hilbert space $\mathbb{H}$ [85]. Figure 2.9 illustrates a simple example of a problem that is not linearly separable in the feature space $\mathbb{X}$. Using a kernel, for mapping the data into a higher-dimensional space $\mathbb{H}$, the data becomes linearly separable.

A certain class of kernels, called *Mercer kernels*, have the important property that they are guaranteed to be representable as a dot product in *some* Hilbert space $\mathbb{H}$. Mercer kernels must satisfy the following conditions:

1. They are continuous functions.
2. They are symmetric functions. That is, $K(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_j, \mathbf{x}_i)$.
3. They are positive semi-definite. That is, the $n \times n$ kernel matrix $\mathbf{K}$ must satisfy $\mathbf{v}^T \mathbf{K} \mathbf{v} \geq 0, \; \forall \mathbf{v} \in \mathbb{R}^n$.

Since the input vectors $\mathbf{x}$ are only appearing as dot products in the SVM dual (2.18), the dual can be rewritten as [10, 33]

$$\boldsymbol{\alpha} = \arg\max \ f(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{2.21}$$

$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha} \ , \tag{2.22}$$

$$\text{subject to } 0 \leq \alpha_i \leq \frac{1}{\lambda n} \ , \tag{2.23}$$

where $\mathrm{H}(i,j) = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$. The kernel being expressible as a dot product is an important property, since explicit evaluation of the mapping ϕ can be prohibitively expensive as $\mathbb{H}$ can be of very high (or even infinite) dimensionality. For Mercer kernels, we do not necessarily know the explicit mapping ϕ. However, satisfying the above conditions guarantees that the mapping exists, and for practical use of the kernel, only the scalar values of the dot product have to be known. These scalar values are efficiently computable for a set of commonly used Mercer kernels, including:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) & \text{Radial basis function (RBF) kernel.} \\ (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + a)^b & \text{Polynomial kernel.} \\ \tanh(a\langle \mathbf{x}_i, \mathbf{x}_j \rangle - b) & \text{Sigmoid kernel.} \\ \langle \mathbf{x}_i, \mathbf{x}_j \rangle & \text{Linear kernel.} \end{cases} \tag{2.24}$$

The hyperparameters, σ, a, b control the behavior of the kernel and are usually set using cross-validation.

Although SVMs are usually formulated as binary classifiers, multi-class classification can be made by using the one-versus-all approach [9]. Given a problem involving K classes, multiclass classification can be performed by training K SVMs. One SVM is trained for each class k using training samples from class k as positive examples, and samples from all the other $K-1$ classes as negative examples. Prediction of a test vector $\mathbf{x}_{\text{test}}$ is then performed by

$$y_{\text{test}} = \arg\max_{k} \ f_k(\mathbf{x}_{\text{test}}) \ , \tag{2.25}$$

where $f_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + b$ is the SVM decision function for class k.

An alternative approach for multi-class classification is to train $K(K-1)/2$ different SVMs, where each SVMs is trained on a pair of two classes. This approach is called one-versus-one. The predicted class is then set to be the class with the most "votes" using its decision function. Although this approach helps with class-balance problems suffered by one-versus-all, significantly more computational time is required for both training and testing. While, more sophisticated multi-class approaches exist [9], the above two are by far the most common in the context of SVMs.

2.3 Problems and Previous Approaches

Action recognition is a challenging task that has been tackled in various forms ever since the nominal work of Johansson [44] in 1973, in which he showed that human motion perception can be performed from limited information, using a small set of lights attached to the human body. This section will give a brief overview of previous approaches towards doing human action recognition, and also mentions some existing problems.

Recently, the emergence of cheap and affordable 3D cameras has resulted in improved methods for human action recognition [58]. By allowing the usage of RGB-D data, several difficult in RGB video action recognition tasks have been solved. These tasks include actor-background segmentation and invariance to illumination. A large volume of previous research exists for 3D human action recognition. Roughly, previous method can be divided into three different types: method based on depth maps, methods based on skeletons, and finally, methods that make use of both.

Approaches that leverage pure depth data include the work by Li *et al.* [58] in 2010, which presents a method based on projecting salient depth map poses into 2D. In their method, they create a feature from a bag of 3D points that is sampled from the different possible 2D projections of the depth data. Due to the lack of available 3D action recognition datasets at the time, they created the now well-used benchmark dataset MSRAction3D using an early prototype of the Kinect camera. The dataset contains several action classes that are quite similar spatially. Examples are the actions "*draw x*" and "*draw circle*". The dataset highlighted the need for action recognition methods that are able to handle high inter-class similarities, as well as inter-subject and intra-class variations. Their experimental evaluation indicated that action recognition based on 3D data clearly performs better than previous approaches based on 2D data. Additionally, 3D methods are more robust against occlusions.

Subsequently, the method of space-time occupancy patterns (STOP) was introduced by Vieira *et al.* [99]. In their method, a 4D histogram is used for representing the depth map 3D points. A slightly related method is HON4D, by Oreifej and Liu [72]. They create histograms of oriented 4D normal vectors, and subsequently learn a 4D space that is quantized in a non-uniform manner. This paper also highlighted the need for understanding temporal directionality, where they showed that unlike other approaches [101, 111] their method is able to distinguish between actions such as "*sit up*" and "*sit down*", which are essentially the same action but differ only in the direction of the motion. Yang *et al.* [111] create DMM-HOG, which stacks orthogonally projected depth maps that are then applied to histograms of oriented gradients. Rahmani *et al.* [78] 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) [76]. 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 methods based on the raw 3D data are capable of describing greatly detailed shape information, such methods are not able to capture the correspondence between different parts on the human body.

Next, we have methods that leverage 3D skeletons. One such method is DL-GSGC by Luo *et al.* [60]. DL-GSGC is a dictionary learning approach for sparse coding using regularizers that encourage group sparsity and geometry constraints. They show that their regularizers help with increasing the discriminative capability of their codes. For handling the representation of temporal order in an action, they use max pooling on the codes, together with a temporal pyramid pooling scheme for creating a structure with complimentary sequential representation. Another method based on sparse coding is SSS by Zhao *et al.* [118]. They use distance between pairwise skeletons joints, together with dictionary template learning and sparse coding for learning a representation of gestures. HOJ3D is another approach by Xia *et al.* [108], where they create a sequence of visual words (in fact, postures) using linear discriminant analysis. The visual words are subsequently used for training a hidden Markov model for modeling the generating distribution of postures. Further methods include methodology based on classifying multi-order pose derivatives [114] (MP) using K-nearest neighbor classification, or low-dimensional embeddings of 3D joint positions using principal component analysis [110] (Eigenjoints). A histogram-based method is HOD by Gowayyed *et al.* [34], which works by quantizing the angles between the 3D skeleton joints. The quantized angles are then aggregated using a temporal pyramid, which helps with capturing temporal dependencies of actions. In the work by Ellis *et al.* [31], a human action recognition framework with low latency is created. Their method works by first finding canonical skeletons poses among the training data by leveraging multiple instance learning. Although their method has low computational complexity and applies to real-time classification situations, their scheme is unable to model actions that exhibit temporal structure characteristics, such as "*draw x*", and is limited to more simple action types. Wang *et al.* [103] 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.

Lastly, a set of previous methods leverage both the 3D skeletons and depth data in a simultaneous manner. Wang *et al.* [101] develop an actionlet ensemble (AE), which selects relative joint pairs in a discriminative way that is able to reduce inter-class ambiguity. For capturing temporal order, they use a temporal pyramid, and multiple kernel learning is used for doing the classification. Wang and Wu [100] tackle the problem of temporal misalignment by learning a warping matrix that is used to align the input

RGB-D human action video sequences before carrying out the classification. For each action class, a warping template is learned, and a latent structural SVM is used for predicting the action.

While the availability of depth maps has resulted in a recent boost in performance on benchmark datasets [58, 101], most approaches to human action recognition are however inherently view-dependent [72, 60, 100]. 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 [105, 112], view-invariant features [55, 73, 80, 104] and knowledge transfer between different views [76, 32, 57, 117, 119]. The number of approaches using only depth maps for cross-view action recognition are, however, much fever [78, 108, 102, 77], 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 study falls in the first category of geometric transformations, with the added benefit of being computable using only depth data.

2.4 Applications

Human action recognition can be applied to several tasks in our daily life. An already well-known such task is player interaction in games, which has been thoroughly demonstrated by several titles from the Microsoft Xbox series game consoles [116].

While the usage of action recognition systems in game-related applications brings entertainment and joy into the lives of various humans, one must not forget that in one sense, games are a simulation of the real world. This brings us to the next application, which is human-robot interaction. In order for autonomous robots to be able to successfully understand the world around them, it is crucial to be able to automatically interpret the actions of humans and objects in the surrounding environment.

Health care is yet another use case for action recognition systems, where they can be applied for analysis of rehabilitation exercises, alleviating the busy schedule of a licensed medical professional. Autonomous understanding of the surrounding environment is also important in elder care, where tasks such as automatic fall detection can help mitigate the time until help arrives [115].

Other tasks include surveillance and similar security applications where an alarm could be set upon detecting a certain type of behavior. Finally, video indexing and retrieval are also of interest in the current era, when we have large collections of video databases. Being able to conveniently search these for a target action would lessen the burden of finding videos of interest.

2.5 Summary

In this chapter, we have introduced the task of human action recognition and covered some fundamental concepts that can be used towards solving the task. We have additionally discussed potential applications of the research done in this study, as well as surveyed related research about human action recognition from depth maps. Given a large body of related research, we can conclude that the task of human action recognition is non-trivial and challenging due to intra-class variations, noisy skeleton joints and multi-view camera angles. Here, leveraging depth data greatly helps improving performance and easily solves problems normally faced in RGB-based action recognition, such as subject-background segmentation.

Specifically, we have argued that human action recognition faces several problems. First, intra-class variation is common for human actions, which occurs since a single action can be performed in several ways. This requires methods for action recognition to be flexible enough to allow several representations of the same action class. Inter-subject variation is also naturally occurring, which is caused by varying anatomy between different human subjects, causing the same action to be performed slightly different depending on the person performing the action. This issue requires, again, flexibility in the action representation.

Further, while convenient, action recognition using depth cameras face additional problems not encountered with standard RGB cameras. Noisy depth maps (and skeletons) are two such problems, since unlike RGB cameras, current depth cameras are subject to much larger noise. This means that established methods for RGB action recognition cannot be trivially used with depth cameras. Also, there is the problem of multi-view camera angles, as any human action is not camera-centric, but human-centric. The action should be defined from the subject's point of view, not the camera's. This requires development of features or classification methods robust against viewpoint changes. Additionally, action representations do also need to be able to distinguish the temporal direction of actions, since many actions in daily life are each other's inverse (*cf.* "*sit up*" and "*sit down*").

Finally, human-object interaction should be handled explicitly, which is crucial as certain actions might have similar movements, but differ semantically due to the object being used (*e.g.* "*tennis serve*" vs. "*throw ball*"). Similarly, understanding of human-human interaction should be of interest, which is important for recognition of complicated actions in the real work, such as "*dancing*" or "*playing basketball*", which is usually defined by the interactions between several human subjects, rather than their individual movements.

Systems for human action recognition cannot be fully deployed in the real world for arbitrary tasks until the above points have been solved to a satisfying extent. We can conclude that while the performance of action recognition systems have progressed somewhat in the recent years, it still remains a challenging task worth pursuing.

3

Graph Signal Processing

NOWADAYS, signal processing is a well-known subject that needs no further introduction. Classical results include the discrete Fourier transform (DFT) and the discrete wavelet transform (DWT) for both 1D signals in time and also 2D signals in images [98].

Quite recently, several classical signal processing (CSP) methods have gained proposals for generalization to arbitrary graphs [89]. The classical Fourier transform in CSP is represented by a graph equivalent, which allows generalizations of common CSP operations such as convolution, translation and filter, *etc.*. The structure of natural signals are in general not restricted to a regular grid (*e.g.* anthropometric meshes and sensor networks). CSP is typically limited to signals following regular grid structure, and is not always optimal for analysis of natural signals. Graph signal processing (GSP), on the other hand, allows modeling of signals that follow the graph structure, and is therefore able to offer a more flexible framework for creating natural signals directly adapted to the signal domain using their intrinsic structure. GSP gives us the freedom to design the graph as we please, which offers more flexibility, and allows us to extend previous approaches from CSP in order to incorporate auxiliary information, such as propagation along weighted graph edges. Careful graph design can therefore provide a more suitable framework for naturally occurring signals. However, with power comes a need for carefulness: inappropriate graph designs can conversely lead to inclusion of unwanted signal noise. For an overview of the subject, Shuman *et al.* [89] provides an excellent introduction.

Signal processing on graphs is useful because the flexible representation of various objects that a graph provides. For example, an image can be represented by a graph, where each pixel is a vertex and edges connect nearby vertices together. If the signal associated with each vertex is the pixel intensity, the GSP on such a graph is able to capture properties of the local structure of the image, alleviating tasks such as image denoising [68] or compression [83]. Another type of possible graph to construct is the one mimicking the structure of the human brain [54]. Brain regions from fMRI scans can represent vertices and edges then explain how these regions interact for representing a certain brain state. In addition, using graphs for representing the body might seen natural. As we saw in the previous chapter, the pose of the human body can be represented by a tracked skeleton [88], which essentially is a graph with each body joint being a vertex. Importantly, the early study by Johansson [44] showed that human actions can be defined as a sequences of interactions between parts. The study investigated perception from limited information, where a number of lights were attached to the subject's body. The study showed that capturing the interactions between several parts of the human body is helpful for decreasing the ambiguity between several motion categories. Graphs might therefore provide useful for capturing the interactions of the motion between several body parts. Following this train of thought, one could think of numerous additional ways for using graphs for action recognition. One use case is for using the graph to regularize the feature representation of an action. Since the tracked skeletons are noisy, one could imagine using the information from a graph as prior knowledge of how the feature should be structured regardless of noise. Furthermore, graphs capture only the pair-wise interactions between vertices and are able to create a representation that is invariant to camera view angles. We shall discuss these ideas in more detail in the following chapters.

In this chapter, we will describe several fundamental concepts in GSP, including the graph Laplacian matrix and a framework for spectral graph wavelets. In addition, the end of this chapter presents previous studies related to GSP, and discusses how these relate to the fundamental concepts we have presented. The reader already familiar with the theory of graph signal processing may choose to skip this chapter without affecting the understanding of the main results in the remainder of the thesis. Consequently, this chapter will only present results from previous work. The motivation for leveraging graphs for the purpose of human action recognition in this study is deferred to Chapter 4.

3.1 The Graph Laplacian Matrix

We will in this section briefly describe the graph Laplacian matrix, which is a core part of graph signal processing theory.

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.

The combinatorial (or non-normalized) graph Laplacian matrix is defined as $\mathbf{Ł} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D} = \mathrm{diag}\,\{\mathbf{W1}\}$ is the diagonal degree matrix and $\mathbf{1}$ is the vector of all ones. We let $\{\lambda_\ell, \mathbf{u}_\ell\}_{\ell=0,\dots,N-1}$ denote the eigenvalue and eigenvector pairs of $\mathbf{Ł}$. The spectrum of $\mathbf{Ł}$ carries a frequency interpretation [120], 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 [18]. 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 $\lambda_{\text{max}} = 2$ for the maximum eigenvalue, which is tight when the graph is bipartite [18].

The results presented in the rest of this thesis hold in general for both the combinatorial and normalized version of the graph Laplacian. Indeed, there is not yet any consensus about in which situation either version is to be preferred over the other [95], but we will choose the normalized graph Laplacian due to the trivial upper bound on λ_{max}.

3.2 Graph Fourier Transform

The classical Fourier transform [62] of a function f is defined by the complex exponentials $e^{-j2\pi\xi_k x}$, which are eigenfunctions (with eigenvalue λ_k) of the one-dimensional Laplacian operator

$$\frac{\partial^2}{\partial x^2} e^{-j2\pi\xi_k x} = \lambda_k e^{-j2\pi\xi_k x} \ . \tag{3.1}$$

The Fourier transform is then

$$\widehat{f}(\xi) = \int f(x) e^{-j2\pi\xi x} dx \ . \tag{3.2}$$

Similarly, the graph Fourier transform [89] (GFT) is defined in terms of the eigenvectors of the graph Laplacian matrix **Ł**. 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 $\mathcal{G}$. The graph Fourier transform of $\mathbf{f}$ is then

$$\widehat{f}(\lambda_k) = \sum_{i=1}^{|\mathcal{V}|} f(i) \mathrm{u}_k(i) \ , \tag{3.3}$$

where the graph Laplacian eigenvectors $\{\mathbf{u}_k\}_{k=0,\ldots,|\mathcal{V}|-1}$ are used for calculating the transform. The graph Fourier mode $\widehat{f}(\lambda_k)$ has a frequency interpretation [120], which allows a graph spectral decomposition of $\mathbf{f}$ that follows the graph structure.

Due to the orthogonality of the eigenvectors, the inverse GFT (IGFT) can be defined as

$$f(i) = \sum_{k=0}^{|\mathcal{V}|-1} \widehat{f}(\lambda_k) \mathrm{u}_k(i) \ , \tag{3.4}$$

which together with the GFT allows both analysis and synthesis to be performed of a graph signal.

Like in classical signal processing, the GFT satisfies the Parseval relation [37]

$$\langle \mathbf{f}, \mathbf{h} \rangle = \langle \widehat{\mathbf{f}}, \widehat{\mathbf{h}} \rangle \ , \tag{3.5}$$

which means that filtering of a graph signal can be done by

$$f(i) = \sum_{k=0}^{|\mathcal{V}|-1} h(\lambda_k)\widehat{f}(\lambda_k)\mathrm{u}_k(i) \ , \tag{3.6}$$

where h is a spectral kernel used for filtering the signal.

While translation from a location m by an amount n in classical signal processing is defined by the trivial change of variable $f(m-n)$, this cannot be directly generalized to GSP, due to the signal being discrete [89]. This can, however, the solved by noting that classical translation is a convolution of the signal f with a Kronecker delta δ_{mn} centered at location n:

$$f(m-n) = f * \delta_{mn} \ . \tag{3.7}$$

Similarly, by using a delta signal

$$\delta_n(m) = \begin{cases} 1 & \text{if } m = n \\ 0 & \text{otherwise} \end{cases} \ , \tag{3.8}$$

we can define the translation T_n of a graph signal to a vertex n by

$$T_n\mathrm{f}(m) = (\mathbf{f} * \boldsymbol{\delta}_n) = \sum_{\ell=0}^{N-1} \widehat{\mathrm{f}}(\lambda_\ell)\mathrm{u}_\ell(n)\mathrm{u}_\ell(m) \ , \tag{3.9}$$

where the graph signal convolution $*$ is defined by a multiplication in the graph Fourier domain, and $\widehat{\delta}_n(\ell) = \mathrm{u}_\ell(n)$ [89].

It can be easily verified that Łf, acting as a linear operator on a function $f \in \mathbb{R}^N$, is a difference operator:

$$\text{Ł}f(i) = \sum_{i \sim j} \mathrm{W}(i,j)\left(f(i) - f(j)\right) \ . \tag{3.10}$$

Therefore, it seems natural to encode graph signals in terms of the graph Laplacian eigenvectors, which have been shown to encode a notion of smoothness on a graph [7], and is an approximation of the Laplace-Beltrami operator on a manifold. [6]

Matrix form The entire chain of applying GFT, filtering, and IGFT can be written in matrix form as

$$\mathbf{f}_{\text{out}} = \mathbf{U}\mathbf{H}\mathbf{U}^T\mathbf{f}_{\text{in}} \ , \tag{3.11}$$

where $\mathbf{U}$ is an orthogonal matrix with the eigenvectors of $\mathbf{Ł}$ as columns and $\mathbf{H}$ is a diagonal matrix with each element as $\mathrm{H}(k,k) = h(\lambda_k)$, for some spectral kernel function h. This can easily be verified by checking that each element in $\mathbf{f}_{\text{out}}$ equals the sum in (3.4). Note that the constant function $h(\lambda_k) = 1$ gives the input signal back, while any other function h will cause the input graph signal to be filtered according to h.

Why use eigenvalues for representation of frequency? At first sight, it is perhaps not obvious why the graph Laplacian eigenspace conforms to a frequency interpretation. In this paragraph, we will give a few arguments for why this interpretation holds true.

When performing principal component analysis (PCA) on a data matrix, the eigenvector associated with the largest eigenvalue will act as an orthonormal basis that captures most of the variance of the data. That is, mapping the matrix to the PCA basis will cause most of the elements to have large variation along it.

Next, remember that the linear operator $\mathbf{Ł}f$ is a difference operator, as shown in (3.10). Because $\mathbf{Ł}f$ captures the difference between neighboring vertex signals, the eigenvector $\mathbf{u}_{\text{max}}$ associated with the largest eigenvalue λ_{max} of $\mathbf{Ł}$ will capture the largest variance, or equivalently, frequency, of the graph signal $\mathbf{f}$. Similarly, eigenvectors corresponding to small eigenvalues will capture small variance (small frequency).

Another way of understanding the frequency interpretation of the eigenvalues is through the Courant-Fischer theorem [41], which states that the eigenvalues can be defined in an iterative fashion using the Rayleigh quotient

$$\lambda_0 = \min_{\mathbf{f}} \frac{\mathbf{f}^T \mathbf{Ł} \mathbf{f}}{\|\mathbf{f}\|_2^2} \tag{3.12}$$

$$\lambda_\ell = \min_{\mathbf{f}} \frac{\mathbf{f}^T \mathbf{Ł} \mathbf{f}}{\|\mathbf{f}\|_2^2}, \ \ell = 1, \dots, |\mathcal{V}| - 1 \tag{3.13}$$

$$\text{such that } \mathbf{u}_\ell \perp \text{span}\{\mathbf{u}_0, \dots, \mathbf{u}_{\ell-1}\} \ ,$$

where each subsequent eigenvector $\mathbf{u}_\ell = \mathbf{f}/\|\mathbf{f}\|$ corresponding to the eigenvalue λ_ℓ is constrained to be orthonormal to the preceding ones. It can then

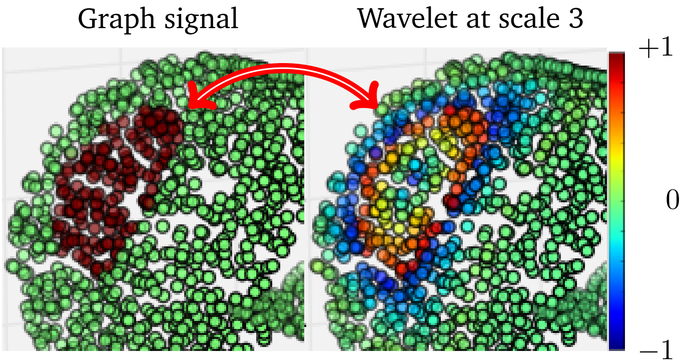


Figure 3.1: SGWT example. A graph signal with an abrupt discontinuity is analyzed on a 3D point cloud of 5000 points, using the SGWT. Wavelet coefficients at scale 3 are shown. Note that the wavelet coefficients exhibit a wavelike, oscillating behavior near the discontinuity. See text for details.

be seen that the term

$$\mathbf{u}_\ell^T \mathbf{Ł} \mathbf{u}_\ell = \frac{1}{2} \sum_{(i,j)\in\mathcal{E}} \mathrm{W}(i,j)(\mathrm{u}_\ell(j) - \mathrm{u}_\ell(i))^2 \tag{3.14}$$

encodes a measure of global smoothness in the graph, which explains why small eigenvalues have eigenvectors that encode a smoother (low-frequency) mapping on the graph.

3.3 Spectral Graph Wavelet Transform

Hammond *et al.* [37] define a spectral graph wavelet transform (SGWT) for graph signals on the eigenspectrum of **Ł**.[1] Their wavelet transform on graphs is defined based on the choice of a kernel function $g : \mathbb{R}^+ \to \mathbb{R}^+$, which can be compared to the Fourier transformed wavelet $\widehat{\psi}$ in the case of classical wavelet transforms [62], and also a scaling function kernel $h : \mathbb{R}^+ \to \mathbb{R}$, which captures low-frequency content.

Specifically, in classical signal processing, a wavelet operator $\psi_{t,n}(x)$ de-

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

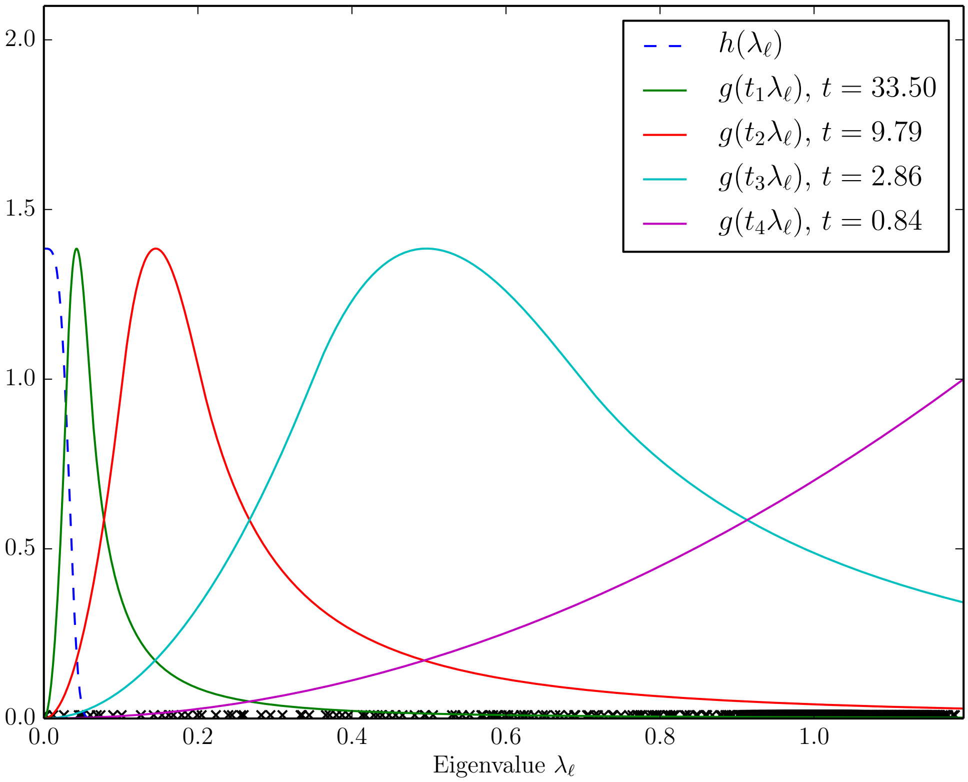


Figure 3.2: SGWT kernels example with the number of wavelet scales $J = 4$. Best viewed in color. Note that the scaling kernel h attenuates high eigenvalues and acts as a low-pass filter. The wavelet kernels $g(t_j\cdot)$, on the other hand, isolate certain frequency bands and act as band-pass filters. The black crosses at the bottom indicate the actual locations of the discrete set of eigenvalues of the graph in Figure 3.1, showing that the eigenvalues do not necessarily appear with uniform density.

scribing the localized frequency at scale t around location n is defined as

$$\psi_{t,n}(x) = \frac{1}{t}\psi\left(\frac{x-n}{t}\right) \ , \tag{3.15}$$

where ψ is a mother wavelet from an orthogonal wavelet family such as Daubechies wavelets [25]. For an even and real-valued mother wavelet ψ, we can rewrite (3.15) as

$$\psi_{t,n}(x) = \frac{1}{2\pi}\int \widehat{\psi}(t\omega)e^{-j\omega n}e^{j\omega x}d\omega \ , \tag{3.16}$$

since translation of a signal is equivalent to multiplication by the complex exponential $e^{-j\omega n}$ in the Fourier domain [62].

Shuman *et al.* define the analogue to (3.16) in the *graph* Fourier domain and write a spectral graph wavelet $\psi_{t,n} \in \mathbb{R}^N$ at scale t localized around vertex n explicitly as a vector

$$\psi_{t,n}(m) = \sum_{\ell=0}^{N-1} g(t\lambda_\ell)\mathrm{u}_\ell(n)\mathrm{u}_\ell(m) \ . \tag{3.17}$$

where the signal translation to vertex n corresponds to multiplication by the graph Laplacian eigenvector entry $\mathrm{u}_\ell(n)$.

The kernel g is analogous to the Fourier transformed mother wavelet $\widehat{\psi}$, and should behave as a band-pass filter, which requires it to satisfy

$$g(0) = 0 \ , \tag{3.18}$$

$$\lim_{x\to\infty} g(x) = 0 \ . \tag{3.19}$$

The scaling function kernel ϕ in classical wavelet theory is realized by the modulated low-pass filter h, which is required to satisfy

$$h(0) > 0 \ , \tag{3.20}$$

$$\lim_{x\to\infty} h(x) = 0 \ . \tag{3.21}$$

Given a graph signal $\mathbf{f}$, we can extract an SGWT coefficient by performing the inner product $\langle \psi_{t,n}, \mathbf{f} \rangle$. The kernel g use for the transform is chosen by Hammond *et al.*to acts as the following band-pass filter [37].

$$g(x) = \begin{cases} x_1^{-\alpha} x^\alpha & \text{for } x < x_1 \\ s(x) & \text{for } x_1 \le x \le x_2 \\ x_2^\beta x^{-\beta} & \text{for } x > x_2 \end{cases} \ . \tag{3.22}$$

Here, $\alpha = \beta = 2$, $x_1 = 1$, $x_2 = 2$ and $s(x)$ is a unique cubic spline that follows the curvature of g. With this formulation, high-frequency information around a vertex will be localized by small scale coefficients (small t), while low-frequency information is captured by larger scale coefficients (large t).

The scaling kernel h in the transform is specified as

$$h(x) = \gamma \exp\left(-\left(\frac{x}{0.6\epsilon} \right)^4 \right) \ . \tag{3.23}$$

This creates a scaling function $\boldsymbol{\phi}_n$ is able to represent low-frequency content of the graph in a stable manner [37]. Note that the scaling function is necessary due to the constraint $g(0) = 0$. Let λ_{max} denote an upper bound on the maximum eigenvalue of the graph Laplacian matrix. Then, the design parameter $\epsilon = \lambda_{\text{max}}/20$, and γ is set so that $h(0) = \max_\lambda g(\lambda)$. We can then define the scaling vector $\boldsymbol{\phi}_n$ similarly to (3.17) as

$$\phi_n(m) = \sum_{\ell=0}^{N-1} h(\lambda_\ell)\mathrm{u}_\ell(n)\mathrm{u}_\ell(m) \; . \tag{3.24}$$

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. The transform coefficients can be expressed compactly as a $(J+1)N$-dimensional vector $\mathbf{c}$ by gathering the wavelet and scaling function vectors in a transformation matrix

$$\mathbf{T} = [\boldsymbol{\Psi}_{t_1}, \ldots, \boldsymbol{\Psi}_{t_J}, \boldsymbol{\Phi}] = [\boldsymbol{\psi}_{t_1,1}, \ldots, \boldsymbol{\psi}_{t_J,N}, \boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_N] \; , \tag{3.25}$$

and then setting $\mathbf{c} = \mathbf{T}^T\mathbf{f}$.

An visual example of wavelet coefficients can be seen in Figure 3.1. In the figure, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ is created on the visible set of 5000 points $\{\mathbf{p}_i\}$ in 3D space. The graph signal is set to be 1 at the red points, and zero everywhere else. This creates an abrupt discontinuity, as best seen visually. The edge weights are set to be binary according to

$$\mathrm{W}(i,j) = \begin{cases} 1 & \text{if } \exp\left(-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|_2^2}{2\sigma^2}\right) < \epsilon \\ 0 & \text{otherwise} \end{cases} \; , \; \forall i \neq j \; , \tag{3.26}$$

where $\sigma = 0.1$ and $\epsilon = 10^{-3}$.

In Figure 3.2, the scaling kernel h and a set of SGWT kernels g_j with the number of scales $J = 4$ can be seen. Note how h behaves as a low-pass filter and attenuates larger eigenvalues (high frequencies), whereas the wavelet kernels tend to isolate certain frequency bands.

Finally, we discuss a few properties of the transform. The transform itself is overcomplete, which means that it generates more wavelet coefficients than there are vertices in the graph. Suppose a signal is representable as a combination of a sparse set of wavelet coefficients. Then the SGWT can be viewed as analogous to sparse coding [109], although the sparsity criterion is not explicitly set. Each wavelet then plays the role of an atom in a sparse dictionary [93]. Compared to sparse coding, however, the SGWT atoms are based on a fixed mathematical structure. This means that they can be computed in an efficient manner. Sparse coding, on the other hand, typically requires finding the solution to a heavy optimization problem [30]. We mention here that there exists some attempts to learn a dictionary that depends a the graph structure [93], although this does not guarantee an efficient implementation. Additionally, the explicit mathematical structure of the spectral graph wavelets yields the another merit of enabling formal analysis of the effects of each wavelet basis.

3.3.1 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 [37]. They approximate the kernels g and h using low-dimensional Chebyshev polynomials

$$g(t_j\lambda) \approx \frac{1}{2}c_{j,0} + \sum_{k=1}^{M_j} c_{j,k}\overline{T}_k(\lambda) \ , \tag{3.27}$$

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 λ_{max} [74].

The approximated transforms are given by

$$\boldsymbol{\Psi}_{t_j}^T\mathbf{f} \approx \tilde{\mathbf{w}}_{t_j,\mathbf{f}} = \frac{1}{2}c_{j,0}\mathbf{f} + \sum_{k=1}^{M_j} c_{j,k}\overline{\mathbf{T}}_k(\boldsymbol{\mathcal{L}})\mathbf{f} \ , \tag{3.28}$$

$$\boldsymbol{\Phi}^T\mathbf{f} \approx \tilde{\mathbf{s}}_{\mathbf{f}} = \frac{1}{2}c_{0,0}\mathbf{f} + \sum_{k=1}^{M_0} c_{j,k}\overline{\mathbf{T}}_k(\boldsymbol{\mathcal{L}})\mathbf{f} \ , \tag{3.29}$$

```
Require: $\mathbf{f}$ : Graph signal;
Ensure: $\mathbf{c}$ : Approximated wavelet coefficients
function FASTSGWT($\mathbf{f}$)
    $\overline{\boldsymbol{\tau}}_0 \leftarrow \mathbf{f}$
    $\overline{\boldsymbol{\tau}}_1 \leftarrow (\boldsymbol{\mathcal{L}} - \mathbf{I})\mathbf{f}$
    for $j = 0, \dots, J$ do
        $\mathbf{r}_j \leftarrow \frac{1}{2} c_{j,0} \overline{\boldsymbol{\tau}}_0 + c_{j,1} \overline{\boldsymbol{\tau}}_1$
    end for
    for $k = 2, \dots, \max_j M_j$ do
        $\overline{\boldsymbol{\tau}}_k \leftarrow 2(\boldsymbol{\mathcal{L}} - \mathbf{I})\overline{\boldsymbol{\tau}}_{k-1} - \overline{\boldsymbol{\tau}}_{k-2}$
        for $j = 0, \dots, J$ do
            if $M_j \geq k$ then
                $\mathbf{r}_j \leftarrow \mathbf{r}_j + c_{j,k} \overline{\boldsymbol{\tau}}_k$
            end if
        end for
    end for
    $\mathbf{c} \leftarrow [\mathbf{r}_0; \dots; \mathbf{r}_J]$
    return $\mathbf{c}$
end function
```

Figure 3.3: Fast SGWT approximation algorithm [37].

with $\overline{\mathbf{T}}_0(\boldsymbol{\mathcal{L}}) = \mathbf{I}$ and $\overline{\mathbf{T}}_1(\boldsymbol{\mathcal{L}}) = \boldsymbol{\mathcal{L}} - \mathbf{I}$. As the approximation accesses $\boldsymbol{\mathcal{L}}$ only through matrix-vector multiplication, it is fast and efficient for sparse graphs. The SGWT approximation algorithm is shown in Fig. 3.3, where the approximated wavelet coefficients are calculated for a graph signal $\mathbf{f}$.

Why is $\tilde{\mathbf{w}}_{t_j,\mathbf{f}}$ a Valid Approximation?

Let us briefly discuss why there is a need to approximate the kernel g and why the approximation works. First, the *exact* wavelet coefficient $\mathbf{w}_{t,\mathbf{f}}(n)$ at scale t and position n is defined as

$$\mathbf{w}_{t,\mathbf{f}}(n) = \langle \boldsymbol{\psi}_{t,n}, \mathbf{f} \rangle \ , \tag{3.30}$$

where $\boldsymbol{\psi}_{t,n}$ is defined in Eq. (3.17).

Next, a function $g : \mathbb{R} \to \mathbb{R}$ can also be expressed as a matrix function [40] $g : \mathbb{R}^{N\times N} \to \mathbb{R}^{N\times N}$

$$g(t\text{Ł}) = \mathbf{U}^T \begin{bmatrix} g(t\lambda_0) & & & \\ & g(t\lambda_1) & & \\ & & \ddots & \\ & & & g(t\lambda_{N-1}) \end{bmatrix} \mathbf{U} \ , \tag{3.31}$$

which follows from the eigendecomposition $\text{Ł} = \mathbf{U}\boldsymbol{\Delta}\mathbf{U}^T$. By explicitly expressing the matrix multiplication, it can be seen that we have

$$(g(t\text{Ł}))(n,m) = \psi_{t,n}(m) \ . \tag{3.32}$$

Then, by (3.30) and (3.32), we have that

$$(g(t\text{Ł})\mathbf{f})(n) = \langle \boldsymbol{\psi}_{t,n}, \mathbf{f} \rangle = \mathbf{w}_{t,\mathbf{f}}(n) \ . \tag{3.33}$$

However, there is a problem. Computing $g(t\text{Ł})\mathbf{f}$ requires us to know the eigenspectrum $\sigma(\text{Ł})$, which takes $\mathcal{O}(N^3)$ time to compute using the QR algorithm. This does not scale well with large graphs, so we need a fast way to approximate $g(t\text{Ł})\mathbf{f}$. This is solved in [37] by expressing $g(t\text{Ł})$ as the Chebyshev polynomial series

$$g(t_j\text{Ł}) \approx p_{t_j}(\text{Ł}) = \frac{1}{2}c_{j,0}\mathbf{I} + \sum_{k=1}^{\infty} c_{j,k} T_k\left((a\mathbf{I})^{-1}(\text{Ł} - a\mathbf{I})\right) \ , \tag{3.34}$$

where $a = \frac{\lambda_{max}}{2}$ in order to make each T_k defined on $[-1,1]$. Expressing g as a Chebyshev polynomial allows us to circumvent having to know the real spectrum, since each T_k can be defined recursively as

$$T_k(x) = \begin{cases} 1 & k = 0 \\ x & k = 1 \\ 2xT_{k-1}(x) - T_{k-2}(x) & otherwise \end{cases} \ . \tag{3.35}$$

This is quite handy, as it allows us to compute T_k *without knowing any actual eigenvalue*!

The reason for choosing the interval $[-1,1]$ is to be able to utilize the following special case for the calculation of each Chebyshev coefficient

$$c_{j,k} = \frac{2}{\pi}\int_0^{\pi} \cos(k\theta) g(t_j a(\cos(\theta)+1)) d\theta \ , \tag{3.36}$$

which does not hold for values outside this interval [37]. For practical computation, (3.36) is approximated by replacing the integral with a sum (by the composite Newton-Cotes formula)

$$c_{j,k} \approx \frac{2}{B} \sum_{b=0}^{B-1} \cos\left(k\frac{b\pi}{B}\right) g\left(t_j a \left(\cos\left(\frac{b\pi}{B}\right) + 1\right)\right) , \tag{3.37}$$

where B is the number of terms used for approximating the infinite integral.

Now, we can express approximate wavelet coefficients $\tilde{\mathbf{w}}_{t,\mathbf{f}}$ by truncating the infinite sum in (3.34) to M_j terms and right-multiplying by $\mathbf{f}$:

$$\mathbf{w}_{t,\mathbf{f}} = g(t_j\mathbf{Ł})\mathbf{f} \approx \tag{3.38}$$

$$\tilde{\mathbf{w}}_{t,\mathbf{f}} = p_{t_j}(\mathbf{Ł})\mathbf{f} \tag{3.39}$$

$$= \frac{1}{2}c_{j,0}\mathbf{f} + \sum_{k=1}^{M_j} c_{j,k} T_k \left(\frac{\mathbf{Ł} - a\mathbf{I}}{a\mathbf{I}}\right) \mathbf{f} \; . \tag{3.40}$$

How do we know that $\tilde{\mathbf{w}}_{t,\mathbf{f}}$ is a good approximation? The authors show in [37] that we have the bound

$$|\mathbf{w}_{t,\mathbf{f}}(n) - \tilde{\mathbf{w}}_{t,\mathbf{f}}(n)| \leq \|\mathbf{f}\| \sup_{x \in [0, \lambda_{max}]} |g(tx) - p_t(x)| \; , \tag{3.41}$$

where $p_t(x)$ is the polynomial approximation of $g(tx)$. So, if $p_t(x)$ is a good approximation (which it tends to be with $M_j \geq 20$), the approximation is good if $\|\mathbf{f}\|$ is small, i.e. the signal on each vertex is close to zero.

3.3.2 Inverse Transform

While we have so far discussed only the analysis part of the transform, synthesis is also possible using the inverse SGWT. The SGWT is an overcomplete transform, as there are more wavelets $\psi_{t_j,n}$ than original vertices in the graph. Representing the transform by a matrix $\mathbf{W} \in \mathbb{R}^{NJ \times N}$, we have that $\mathbf{W}$ maps an input vector $\mathbf{f} \in \mathbb{R}^N$ to a coefficient vector $\mathbf{c} \in \mathbb{R}^{NJ}$ so that

$$\mathbf{Wf} = \mathbf{c} \; . \tag{3.42}$$

We can solve for $\mathbf{f}$ by left-multiplying by the adjoint $\mathbf{W}^*$ so that we get the square matrix $\mathbf{W}^*\mathbf{W}$:

$$\underbrace{\mathbf{W}^*\mathbf{W}}_{\mathbf{A}} \underbrace{\mathbf{f}}_{\mathbf{x}} = \underbrace{\mathbf{W}^*\mathbf{c}}_{\mathbf{b}} \; , \tag{3.43}$$

which is a linear equation system $\mathbf{Ax} = \mathbf{b}$ that can be solved by a standard conjugate gradient solver. So, by having ready expressions for $\mathbf{W}^*$ and $\mathbf{W}^*\mathbf{W}$, the SGWT inverse can be computed. In the paper, they provide Chebyshev series for both the adjoint $\mathbf{W}^*$ and the composition $\mathbf{W}^*\mathbf{W}$. Conjugate gradient is used since calculating the explicit pseudo-inverse [2] is too expensive for large graphs.

3.4 Problems and Previous Approaches

Graph signal processing is still a relatively young field. We have in the previous sections presented a select set of tools for carrying out analysis of signals on graphs, but these are by no means the complete set available. In fact, several alternative approaches towards GSP have been explored in literature. This section will give a brief overview of previous approaches towards GSP, and also mentions some problems left unsolved.

One problem of immediate attention in GSP is the one of computational efficiency. While classical signal processing enjoys fast processing time thanks to the fast Fourier transform, it is still unknown whether there exists a fast graph Fourier transform. While some approximations towards a fast GFT have been attempted, *e.g.* approximately diagonalizing the graph Laplacian matrix into an product of sparse matrices [53], a truly fast and exact implementation à la classical signal processing has not yet been found.

The problem of creating wavelets on graphs have been explored previously by several authors [23, 37, 19, 67, 79]. The work by Crovella and Kolaczyk [23] is one of the earliest, and is applied for the analysis of computer traffic data that resides on unweighted graphs. Their method works in the graph vertex domain, essentially acting as a filter around a local k-hop neighborhood around a node, so that the wavelet value depends only on nodes within a maximum geodesic distance k. As we have previously discussed, Hammond *et al.* [37] created the spectral graph wavelet transform (SGWT). Compared to Crovella and Kolaczyk, the SGWT allows graph signal analysis on the graph Fourier spectrum, and is somewhat analogous to sparse representations such as sparse coding [109, 93], given that a signal

[2] In cases where $\mathbf{W}$ has full column rank, i.e. the columns are linearly independent, the pseudo-inverse can be calculated explicitly as $\mathbf{W}^+ = (\mathbf{W}^*\mathbf{W})^{-1}\mathbf{W}^*$.

can be represented using a sparse set of wavelet coefficients, while being more efficiently computable. Furthermore, an efficient approximation of the SGWT allows the method to scale for large graphs.

Another graph-spectral wavelet method is the work of Coifman and Maggioni [19], which introduces diffusion wavelets based on applying powers of a diffusion operator. Their wavelets are ensured to be fully orthonormal, but their orthogonalization procedure creates a more complicated transform than the SGWT. While orthogonality may be a desirable property for applications such as signal compression, the SGWT offers more flexibility for selection of wavelet scales. Other methods include Narang and Ortega [67], who develop a method for critically sampled filter banks on graphs. Ram *et al.* [79] develop a generalized tree-based wavelet transform that is applicable to functions defined on graphs. While the SGWT necessarily requires the graph to be undirected, Sandryhaila and Moura [84] provide another framework for GSP, which is based on the Jordan normal form of the graph adjacency matrix and is applicable to directed graphs as well. The SGWT, on the other hand, is based on the GFT, which is only applicable to undirected graphs due to the graph Laplacian matrix being symmetric and positive semidefinite. Investigation of whether the SGWT can be modified to work on directed graphs should be of vital importance, and could lead to additional applications of the theory of GSP.

Finally, it is instructive to compare graph signal processing techniques with other works that utilize graphs. One type of work that compares the structure of graphs is graph kernels, which has been popular in approximately the last 15 years [45, 20, 13, 121, 39, 92, 87]. A graph kernel acts as a similarity measure between two graphs, which can then be used for clustering graphs or training an SVM for classifying graph types. The inherent difference is that while GSP aims to model the signals that lie on the graph, where the graph structure tends to be required to be fixed in order to retain signal-to-signal interpretability, graph kernels tend to compare graphs with a differing number of vertices and structure in order to model how similar the structure between graphs is. For example, the graph kernel by Kashima *et al.* [45] compares counts of labeled random walks on the graphs, and is used for classifying the structures of chemical compounds. Related to GSP, the diffusion kernel by Kondor *et al.* [50] is based on the graph Laplacian matrix and differs from graph kernels, since it does not compare graphs, but rather the vertices of a graph. It is a so called structured kernel. The graph-structured relationship between the vertices is shown to lead to increased performance for datasets having a majority of categorical data, something which standard SVM kernels previously had difficulties with [50]. Graph boosting is another type of work that compares substructures on graphs [70, 69]. By representing an image as a graph, the geometric relations between image features can be utilized by combining graph mining and boosting algorithms. Nowozin *et al.* apply this method to the tasks of web-retrieval outlier rejection [70] and later also for action classification from RGB cameras [69], where they extend their method to keep track of the temporal order of the mined features. A difference with GSP is that while their classifier is template-based, and compares mined graphs, GSP focuses on the propagation of signals defined on the vertices of the graph.

3.5 Applications

Graph signal processing has not yet enjoyed as a large amount of applications as its classical equivalent. Presumably, this is due to the relative adolescence of the theory of GSP. Nevertheless, several applications do exist already, including edge-aware image processing [68], depth video coding [48], image compression [83], anomaly detection in wireless sensor networks [29], bridge structure health monitoring [16], brain functional connectivity analysis [54] and mobility pattern prediction [27]. To the best of our knowledge, at time of publication, our conference paper [46] was the first application of GSP to human action recognition. Since then, some related work has emerged in this direction [4].

3.6 Summary

In this chapter, we have covered basic concepts concerning the theory of graph signal processing. The tools presented allow signal analysis of graph-shaped input, along with frequency analysis using the graph Fourier transform and the spectral graph wavelet transform. We have also argued about why eigenvalues enjoy a frequency interpretation when using the graph Laplacian matrix for representing a graph. Finally, we have surveyed some recent research related to graph signal processing, and discussed some emergent problems.

Graphs allow a flexible representation of a various objects using pairwise connections between vertices. As discovered in the early study of Johansson [44], human actions can be well-perceived from limited information; about 10-12 interest points attached to the human body. Graphs are able to capture the relationships between such interest points, which should make graphs a highly suitable candidate for action representation.

The presented framework of GSP is quite flexible, and is applicable for doing frequency analysis of signals defined on undirected graphs. Our interest in exploring graph signal processing for the task of human action recognition lies in the representation of human interest points as a graph, on which frequency analysis can be applied for extracting information useful for describing an action. We will explore this idea further in the following chapters.

4

Representing Actions as Graphs

In this chapter, 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 [88], while the second variant is based on spatio-temporal keypoints [78].

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 chapter 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 Chapter 5.

4.1 Motivation

Actions can be defined as a sequences of interactions between parts. Indeed, an early study by Johansson [44] 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 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.

4.2 Skeleton-based Graphs

As shown in the study by Johansson [44], 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.* [88], 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 [101, 114, 60].

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.* [114] for normalizing skeleton limbs to standard lengths, while still keeping limb angles and positions intact (see

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

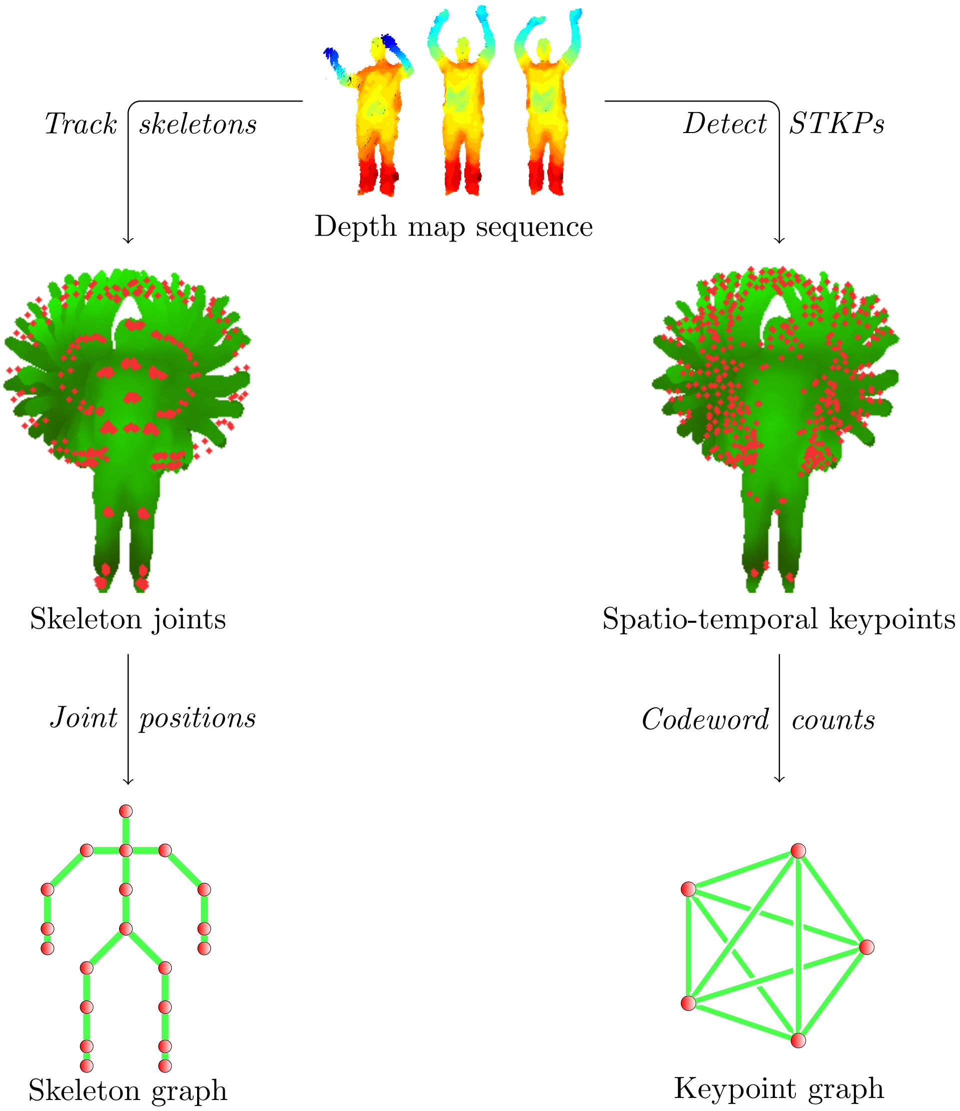


Figure 4.1: Procedure for graph construction. Given a depth map sequence, we detect interest points, such tracked skeleton joints or spatio-temporal keypoints (STKP). The skeleton-based graph construction uses the relative position of the joints, which yields a graph that follows the human skeleton structure. 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.

```
Require: $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_{20}]$ : Skeleton joint positions in a single frame.
Require: $\mathbf{r}$ : Vector of standard joints lengths.
Ensure: $\mathbf{P'}$ : Normalized skeleton joint positions.
function MPNormalize($\mathbf{P}, \mathbf{r}$)
    Let $\mathbf{p}_s$ be the start joint (center hip)
    $\mathbf{p'}_s \leftarrow \mathbf{p}_s$
    for $(i, j)$ in breadth-first search order from $\mathbf{p}_s$ do
        $\mathbf{d} \leftarrow \mathbf{p}_i - \mathbf{p}_j$
        $\mathbf{p'}_j \leftarrow \mathbf{p'}_i + \mathrm{r}(i)\mathbf{d}/\|\mathbf{d}\|_2$
    end for
    return $\mathbf{P'} = [\mathbf{p'}_1, \ldots, \mathbf{p'}_{20}]$
end function
```

Figure 4.2: Skeleton limb length normalization algorithm [114] for transforming the limb lengths of a skeleton $\mathbf{P}$ in a single frame t.

Figure 4.2). The standard lengths $\{\mathrm{r}(i)\}_{i=1,\ldots,20}$ for each joint i can be calculated from training data. Previous research [101, 60] discussed the fact that features extracted by using the relative inter-joint 3D skeleton positions remain largely discriminative. We proceed to describe the location of joint i by constructing the relative position vector

$$\hat{\mathbf{p}}_{t,i} = \mathbf{p}_{t,i} - \mathbf{p}_{t,\text{center hip}} \;, \tag{4.1}$$

since the center hip joint of the tracked 3D skeleton can be considered to be reasonably stationary among different classes of human actions.

4.2.1 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.* [108], 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.* [102], 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

$$\mathrm{v}_{\text{up}}(d) = \operatorname*{median}_{t \in \{1,\dots,T\}} \{\mathrm{v}_{t,\text{up}}(d)\} \, , \; \forall d \, , \tag{4.2}$$

which is gotten by taking the median for each axis independently. The reason for using the median instead of taking 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 [91] in order to create a rotation matrix

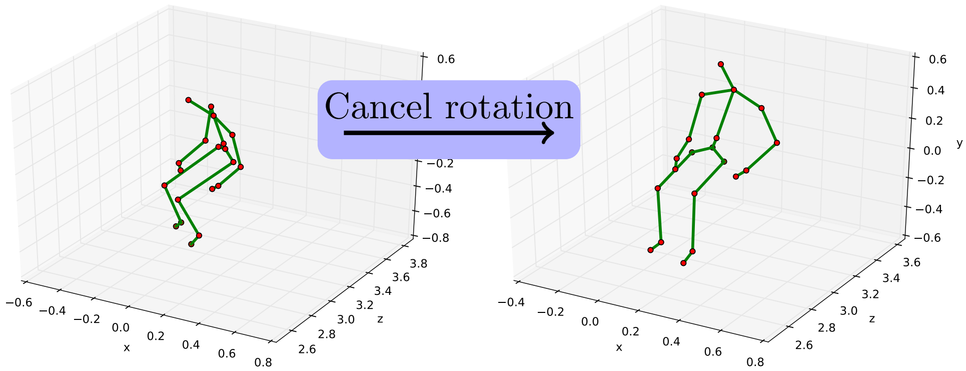


Figure 4.3: Rotation cancellation using the Gram-Schmidt process for a frame of the "*stand up*" action on the left. The right skeleton shows the resulting canonical view facing the camera at $(x, y, z) = (0, 0, 0)$. See text for details.

$\mathbf{R} = [\mathbf{r}_x, \mathbf{r}_y, \mathbf{r}_z]$, where

$$\mathbf{u}_y = \mathbf{v}_{\text{up}} \ , \tag{4.3}$$

$$\mathbf{r}_y = \mathbf{u}_y / \|\mathbf{u}_y\| \ , \tag{4.4}$$

$$\mathbf{u}_x = \mathbf{v}_{t,\text{right}} - \frac{\mathbf{v}_{t,\text{right}}^T \mathbf{u}_y}{\langle \mathbf{u}_y, \mathbf{u}_y \rangle} \mathbf{u}_y \ , \tag{4.5}$$

$$\mathbf{r}_x = \mathbf{u}_x / \|\mathbf{u}_x\| \ , \tag{4.6}$$

$$\mathbf{r}_z = \mathbf{r}_x \times \mathbf{r}_y \ . \tag{4.7}$$

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. 4.3. 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 $\mathbf{R}^{-1}$.

4.2.2 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. 4.1). 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

$$\mathrm{W}(i,j) = \exp\left(-\frac{\|\hat{\mathbf{p}}_{t,i} - \hat{\mathbf{p}}_{t,j}\|_2^2}{2\sigma^2}\right) \tag{4.8}$$

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.

4.3 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.

We use the recently proposed spatio-temporal keypoint (STKP) detector by Rahmani *et al.* [78] (see also Sec. 2.2.1). The reasons for using STKPs are the following. First, the keypoints offer detection repeatability, which means that the keypoints can be detected in different samples of the same action sequence despite noise. Second, the keypoints 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, revealing information about the relevant location of the action in the depth video.

4.3.1 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. 4.1). 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:

$$\mathrm{W}(i,j) = \exp\left(-\sum_{d} \frac{(\mathrm{h}_i(d) - \mathrm{h}_j(d))^2}{\mathrm{h}_i(d) + \mathrm{h}_j(d)} \right) , \tag{4.9}$$

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.

4.4 Summary

In this chapter, we have discussed about how to represent actions as graphs. Our motivation stems from the pioneering study of Johansson [44], who demonstrated that the psychological perception of human actions is possible given the knowledge of a small number of skeleton joints. We have proposed two types of graph constructions. The first one is based on tracked skeleton joints and captures the semantic meaning of body parts and their relationship. For rendering skeletons view-invariant, we also presented a rotation cancellation scheme based on Gram-Schmidt orthonormalization. Finally, the second graph type is based on spatio-temporal keypoints, and describes the shape of the raw point cloud data, providing complementary information to the sometimes noisy skeleton joints.

5

Spectral Graph Sequences (SGSs)

In this chapter, we present our proposed feature descriptor (SGS) for temporal sequences of graphs based on the spectral graph wavelet transform (SGWT) [37]. The overview of our method can be seen in Fig. 5.1. We assume that we have a 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 Chapter. 4. 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.

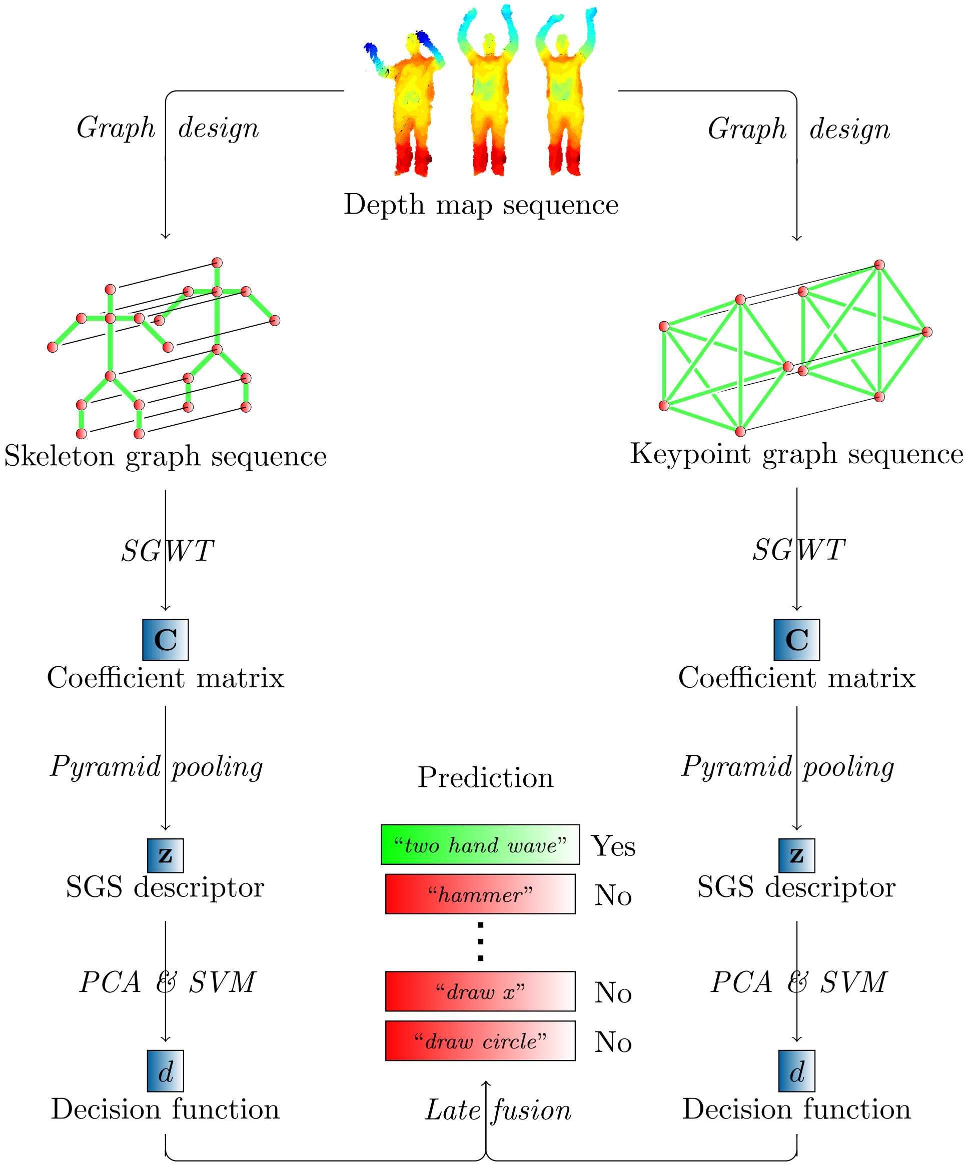


Figure 5.1: Overview of the proposed action recognition system. Given an input depth map sequence, SGS descriptors based on graphs from both skeletons and keypoints are calculated. An SVM is trained for each descriptor and their decision functions are finally combined using late fusion. Note that the 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.* [88].

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

5.1 Graph Design

We consider a temporal sequence of T graph signals $\mathbf{f}_1, \dots, \mathbf{f}_T$, 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, several previous works have employed graph kernels for creating an implicit vector mapping into a reproducing kernel Hilbert space [39, 92, 13, 121, 87, 13, 45, 20]. Graph kernels compare graphs with different structures and different number of vertices, such as chemical compounds [45]. We instead assume the structure of the graph to be constant, and look at the propagation of signals defined on the vertices of the graph. This approach is related to the structural SVM kernel by Kondor *et al.* [50]. One important difference is that while they consider each vertex to be a datum, and compare their relationships using the graph structure, we consider the whole graph to be a datum, where the vertices represent interest points and edges connect them in order to express the natural structure of the input. We will now explain our approach in the following.

As graph signals are scalars by definition [89], 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 $\mathcal{G}_1, \dots, \mathcal{G}_T$. The choice of the graph $\mathcal{G}$ can be *e.g.* one of the two we previously discussed in Chapter 4. Since we assume the signals to be embedded on a common graph $\mathcal{G}$, we let 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 predefined local edge weights, set by some distance kernel $\exp(-\text{dist}(v_{i_t}, v_{j_t}))$ (see Chapter 4). We assume strong signal correlation across the temporal direction ($\exp(-\text{dist}(v_{i_t}, v_{i_{t+1}})) \approx 1$), so we set temporal edge weights to unity. The augmented weight matrix $\mathbf{W}_{\text{aug}}$ therefore has the following fixed sparse block structure:

$$\mathbf{W}_{\text{aug}} = \begin{pmatrix} \mathbf{W} & \mathbf{I} & & & \\ \mathbf{I} & \mathbf{W} & \mathbf{I} & & \\ & \ddots & \ddots & \ddots & \\ & & \mathbf{I} & \mathbf{W} & \mathbf{I} \\ & & & \mathbf{I} & \mathbf{W} \end{pmatrix} , \tag{5.1}$$

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 pairwise interactions within a frame.

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

5.2 Graph Wavelet Coefficient Extraction

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.* [37], which is a generalization of classical wavelet transforms onto arbitrary graphs. We briefly describe the method here; details are covered in Sec. 3.3. 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 [37]

$$\psi_{t,n}(m) = \sum_{\ell=0}^{N-1} g(t\lambda_\ell)\mathrm{u}_\ell(n)\mathrm{u}_\ell(m) , \tag{5.2}$$

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. Given a graph signal $\mathbf{f}$, SGWT coefficients are extracted by the matrix-vector multiplication $\boldsymbol{\Psi}_{t_j}^T\mathbf{f}$, where $\boldsymbol{\Psi}_{t_j} = [\boldsymbol{\psi}_{t_j,1}, \ldots, \boldsymbol{\psi}_{t_j,|\mathcal{V}|}]$.

We proceed to create the normalized graph Laplacian matrix $\boldsymbol{\mathcal{L}}_{\text{aug}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}_{\text{aug}}\mathbf{D}^{-1/2}$, where $\mathbf{D} = \text{diag}(\mathbf{W}_{\text{aug}}\mathbf{1})$ and $\mathbf{1}$ is the vector of all ones. The matrix $\boldsymbol{\mathcal{L}}_{\text{aug}}$ has an eigenbasis that conforms to harmonic analysis of graph signals [89], and has a maximum eigenvalue $\lambda_{\text{max}} = 2$ [18]. Hammond *et al.* [37] presented a fast approximation of the SGWT based on truncated Chebyshev polynomial series [74], which avoids expensive eigendecomposition of $\boldsymbol{\mathcal{L}}_{\text{aug}}$. 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 $[0, \lambda_{\text{max}}]$. We acquire the transform coefficients by calculating the approximate SGWT [37]

$$\boldsymbol{\Psi}_{t_j}^T\mathbf{f}_{\text{aug}} \approx \frac{1}{2}c_{j,0}\mathbf{f}_{\text{aug}} + \sum_{k=1}^{M_j} c_{j,k}\overline{\mathbf{T}}_k(\boldsymbol{\mathcal{L}}_{\text{aug}})\mathbf{f}_{\text{aug}} \,, \tag{5.3}$$

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.

```
Require: $\mathbf{f}$ : Graph signal; $\mathbf{f} = [\mathbf{f}_1; \dots ; \mathbf{f}_T]$
Ensure: $\mathbf{C}$ : Approximated wavelet coefficients
 1: function FastSGWT($\mathbf{f}$)
 2:     $\overline{\boldsymbol{\tau}}_0 \leftarrow \mathbf{f}$
 3:     Calc. $\mathbf{v} \leftarrow (\boldsymbol{\mathcal{L}} - \mathbf{I})\mathbf{f}$ using alg. in Fig. 5.3.
 4:     $\overline{\boldsymbol{\tau}}_1 \leftarrow \mathbf{v}$
 5:     for $j = 0, \dots, J$ do
 6:         $\mathbf{r}_j \leftarrow \frac{1}{2} c_{j,0} \overline{\boldsymbol{\tau}}_0 + c_{j,1} \overline{\boldsymbol{\tau}}_1$
 7:     end for
 8:     for $k = 2, \dots, \max_j M_j$ do
 9:         Calc. $\mathbf{v} \leftarrow (\boldsymbol{\mathcal{L}} - \mathbf{I})\overline{\boldsymbol{\tau}}_{k-1}$ using alg. in Fig. 5.3.
10:         $\overline{\boldsymbol{\tau}}_k \leftarrow 2\mathbf{v} - \overline{\boldsymbol{\tau}}_{k-2}$
11:         for $j = 0, \dots, J$ do
12:             if $M_j \geq k$ then
13:                 $\mathbf{r}_j \leftarrow \mathbf{r}_j + c_{j,k} \overline{\boldsymbol{\tau}}_k$
14:             end if
15:         end for
16:     end for
17:     $\mathbf{R} \leftarrow [\mathbf{r}_0, \dots, \mathbf{r}_J]$
18:     $\mathbf{C} \leftarrow \mathbf{R}$, reshaped to store $\boldsymbol{\psi}_{t_j,n}^T \mathbf{f}$ on row $t$.
19:     return $\mathbf{C}$
20: end function
```

Figure 5.2: Fast SGWT approximation with incorporated efficient matrix-vector multiplication steps on lines 3 and 9. Note that we assume the graph signal $\mathbf{f} = [\mathbf{f}_1; \dots ; \mathbf{f}_T]$ to have a time series structure consisting of T frames, unlike the standard SGWT approximation 3.3.

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

5.2.1 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 [37]. Since our graph has a special sparsity structure (5.1), we can further optimize this step. Fig. 5.2 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. 5.3. 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 $\mathbf{W}$.

While the naïve algorithm requires $\mathcal{O}(N^2T)$ 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.

5.3 Pyramid Pooling

Since wavelets have zero mean, taking the average of the coefficients does not yield any information [95]. 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 $p : \mathbb{R}^{t \times DN(J+1)} \rightarrow \mathbb{R}^{DN(J+1)}$ 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

$$p_{\text{max}}(\mathbf{C}) = \left[\max_t |\mathrm{C}(t,i)|\right]_{i=1,\ldots,DN(J+1)} , \tag{5.4}$$

$$p_{\text{mean}}(\mathbf{C}) = \left[\frac{1}{T}\sum_{t=1}^{T} |\mathrm{C}(t,i)|\right]_{i=1,\ldots,DN(J+1)} . \tag{5.5}$$

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 [60, 34, 101], 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 $\{\mathbf{z}_k\}_{k=1,\ldots,K}$.

5.4 PCA & SVM

Most natural signals are sparse [30]. 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.

5.5 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

$$f(\mathbf{x}) = \frac{1}{2}d_{\text{skeleton}}(\mathbf{x}) + \frac{1}{2}d_{\text{keypoint}}(\mathbf{x}) . \tag{5.6}$$

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 [42], we have found simple averaging to yield good enough results.

5.6 Feature Vector

The resulting feature descriptor encodes the spectral content of a sequence of graphs, so we name it a *spectral graph sequence* (SGS). As computing the SGWT approximation [37] 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 [60, 100].

5.7 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 [120]. 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 [1]. 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. 5.3 needs to be modified accordingly to ensure that the exactness of the calculation of $(\boldsymbol{\mathcal{L}} - \mathbf{I})\overline{\boldsymbol{\tau}}_{k-1}$

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

still holds true.

5.8 Interpretation and Effect of Coefficients and Edges

Unlike methods based on *e.g.* sparse coding [60] or deep learning [76], our method allows for some direct interpretation of the effects of the descriptor. In this section, we present some insights about the information captured by the SGWT coefficients and the edges of our graph structure.

The normalized graph Laplacian is a difference operator that satisfies [89]

$$(\boldsymbol{\mathcal{L}}\mathbf{f})(i) = \frac{1}{\sqrt{d_i}} \sum_{(v_j,v_i)\in\mathcal{E}} \mathrm{W}(i,j) \left(\frac{\mathrm{f}(i)}{\sqrt{d_i}} - \frac{\mathrm{f}(j)}{\sqrt{d_j}} \right) . \tag{5.7}$$

For our graph structure (5.1), this becomes

$$\begin{aligned}(\boldsymbol{\mathcal{L}}\mathbf{f})(i_t) = \frac{1}{\sqrt{d_{i_t}}} \left(2\frac{\mathrm{f}(i_t)}{\sqrt{d_{i_t}}} - \frac{\mathrm{f}(i_{t-1})}{\sqrt{d_{i_{t-1}}}} - \frac{\mathrm{f}(i_{t+1})}{\sqrt{d_{i_{t+1}}}} \right) + \\ \frac{1}{\sqrt{d_{i_t}}} \sum_{(v_{j_t},v_{i_t})\in\mathcal{E}} \exp(-\mathrm{dist}(v_{i_t}, v_{j_t})) \left(\frac{\mathrm{f}(i_t)}{\sqrt{d_{i_t}}} - \frac{\mathrm{f}(j_t)}{\sqrt{d_{j_t}}} \right) ,\end{aligned} \tag{5.8}$$

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

$$(\boldsymbol{\mathcal{L}}\mathbf{f})(i_t) = \Delta^2_{\text{temporal}}(i_t) + \Delta^2_{\text{local}}(i_t) . \tag{5.9}$$

Therefore, since the SGWT is essentially a frequency-modulated graph Laplacian matrix acting as an operator on a signal [37], 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 (5.2), the coefficient $\boldsymbol{\psi}^T_{t_j,i}\mathbf{f}$ at scale t_j and vertex i satisfies

$$\begin{aligned}\boldsymbol{\psi}^T_{t_j,i}\mathbf{f} &= \sum_{\ell=0}^{N-1} g(t_j\lambda_\ell) \sum_j \mathrm{u}_\ell(i)\mathrm{u}_\ell(j)\mathrm{f}(j) \qquad (5.10)\\ &= \sum_j \mathrm{f}(j) \sum_{\ell=0}^{N-1} g(t_j\lambda_\ell)\mathrm{u}_\ell(i)\mathrm{u}_\ell(j) . \qquad (5.11)\end{aligned}$$

Following Shuman *et al.* [89], if we assume that g is an order K polynomial $g(\lambda) = \sum_{k=0}^{K} c_k \lambda^k$, we can simplify (5.11) to

$$\sum_j \mathrm{f}(j) \sum_{k=0}^{K} c_k (\boldsymbol{\mathcal{L}}^k)(i,j) \; . \tag{5.12}$$

Inserting (5.9) then leads to

$$\begin{aligned} c_0 \mathrm{f}(i) + \sum_{k=1}^{K} c_k \boldsymbol{\mathcal{L}}^{k-1} (\boldsymbol{\Delta}^2_{\text{temporal}} + \boldsymbol{\Delta}^2_{\text{local}})(i) \\ = \hat{\Delta}^2_{\text{temporal}}(i) + \hat{\Delta}^2_{\text{local}}(i) \; , \end{aligned} \tag{5.13}$$

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 [37]. 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 [37]. 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 Section 3.3.1).

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

$$\begin{aligned} \frac{1}{T} \sum_t |\hat{\Delta}^2_{\text{temporal}}(i_t) + \hat{\Delta}^2_{\text{local}}(i_t)| \leq \\ \frac{1}{T} \sum_t \left(|\hat{\Delta}^2_{\text{temporal}}(i_t)| + |\hat{\Delta}^2_{\text{local}}(i_t)| \right) \; . \end{aligned} \tag{5.14}$$

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. 5.4. 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 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

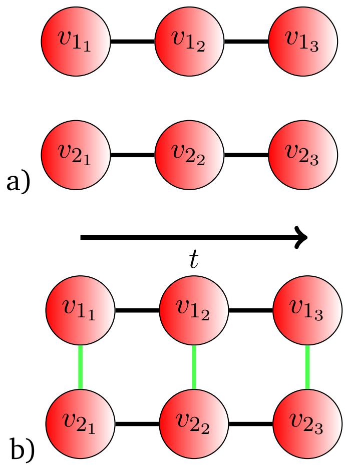


Figure 5.4: Graphs corresponding to a time sequence of $T = 3$ graph signals. (a) Without local edges. (b) With local edges. See text for details.

one. Applying the combinatorial 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 [89].

5.9 Summary

In this chapter, we have presented a framework for human action recognition from depth maps using graph signal processing techniques. Our proposed method represents an action as a graph and captures the relationship between vertices over time using both local and temporal edges. We also presented an efficient algorithm for calculating the approximate SGWT coefficients that takes advantage of our graph's block sparsity structure. Finally, we performed analysis of the interpretation and effects of the system, showing that our system captures second order information of the input graph signal.

6

Experimental Evaluation

In this chapter, the proposed method is evaluated on five publicly available datasets: MSRAction3D [58], MSRActionPairs3D [72], UCF-Kinect [31], N-UCLA Multiview Action3D [103] and Online RGBD Action [113]. Accuracies of previous work are obtained from literature. In addition to the experiments, the end of the chapter 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.

6.1 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 (5.4) and mean (5.5) 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 [97]. 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.

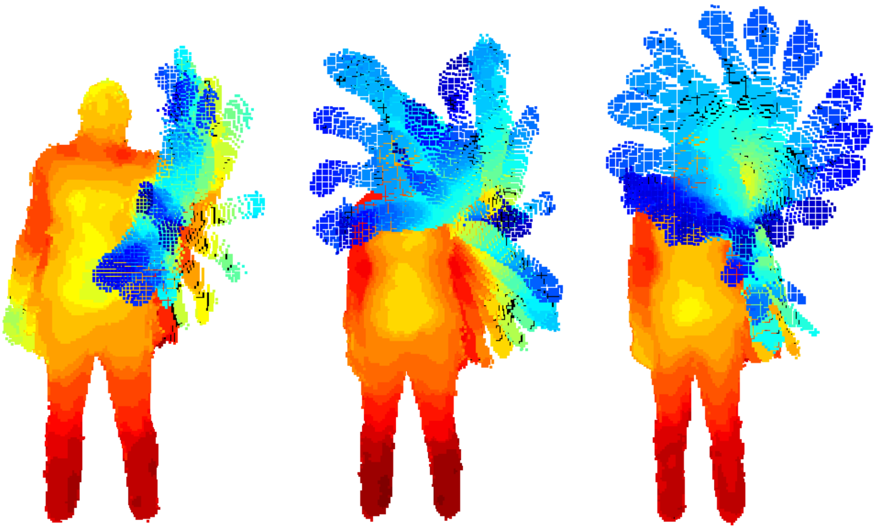


Figure 6.1: Examples from the MSRAction3D dataset of frontal view actions "*hammer*" (left), "*draw x*" (middle) and "*draw circle*" (right).

Table 6.1: Recognition performance on the MSRAction3D dataset.

Method	Accuracy (%)
DL-GSGC [60]	96.7
MMTW [100]	92.7
MP [114]	91.7
SGS($p_{\mathbf{mean}}$, skeleton-view-dep.)	**91.4**
HOD [34]	90.2
HON4D [72]	88.9
AE [101]	88.2
SGS($p_{\mathbf{max}}$, skeleton-view-dep.)	**86.3**
SGS($p_{\mathbf{mean}}$, skeleton-view-inv.)	**83.5**
SSS [118]	81.7
SGS($p_{\mathbf{max}}$, skeleton-view-inv.)	**79.4**
SGS($p_{\mathbf{mean}}$, keypoint)	**73.9**
Canonical poses [31]	65.7
HMM [61]	63.0
Motion Templates + DTW [65]	54.0

Table 6.2: Recognition performance on the MSRAction3D dataset for the three different subject configurations on the three action sets as in Li *et al.*. [58] 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 6.1.

Method	Test 1				Test 2				Cross-subject test			
	AS1	AS2	AS3	Avg.	AS1	AS2	AS3	Avg.	AS1	AS2	AS3	Avg.
DL-GSGC [60]	100	98.7	100	99.6	100	98.7	100	99.6	97.2	95.5	99.1	97.3
SGS($p_{\mathbf{max}}$, 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($p_{\mathbf{mean}}$, 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 [111]	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 [99]	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 [110]	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 [108]	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 [58]	89.5	89.0	96.3	91.6	93.4	92.9	96.3	94.2	72.9	71.9	79.2	74.7

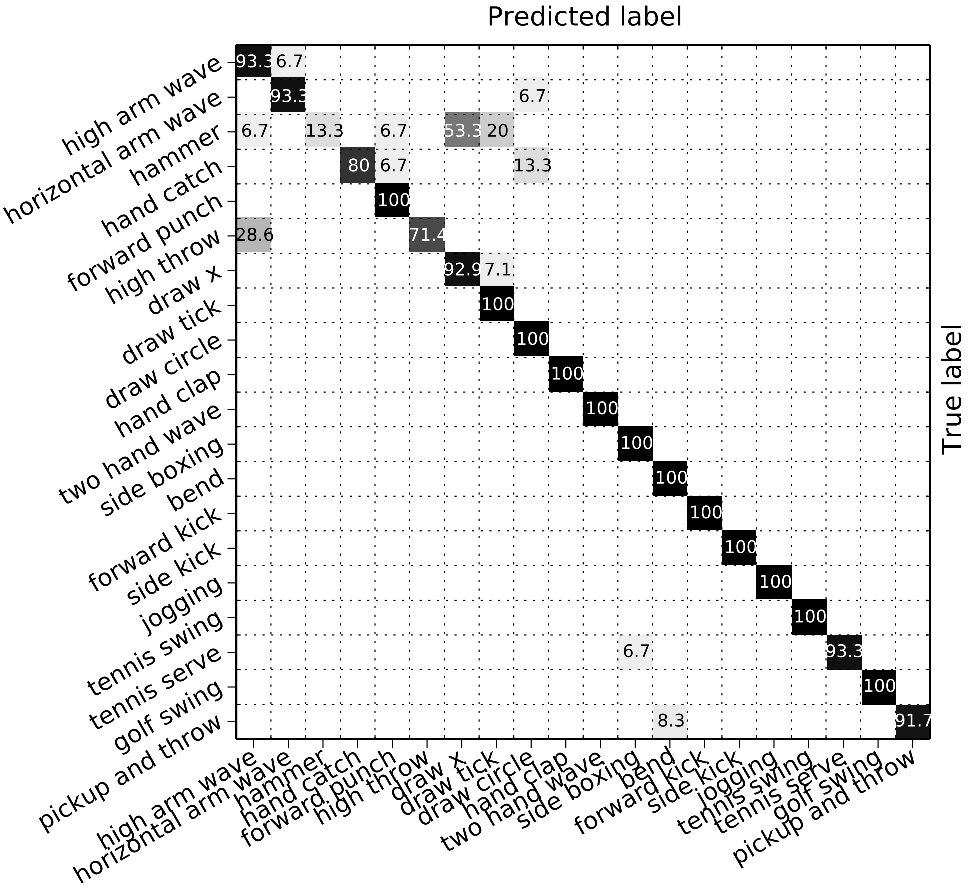


Figure 6.2: Confusion matrix for using our method on the MSRAction3D dataset. Each cell shows classification accuracy (%) from white (0) to black (100) in the cross-subject setting. The average accuracy is 91.4%.

6.2 Datasets and Results

6.2.1 MSRAction3D

The MSRAction3D dataset [58] consists of 20 actions that are being performed by 10 subjects. Each action is repeated by each subject $1-3$ times, resulting in a dataset of 557 depth videos. Further, each action repetition is not necessarily performed in the same manner, which increases intra-class variance of the actions. The challenge of this dataset is that some action classes are quite spatially similar. Examples of such actions are "*draw x*" and "*draw circle*". Thus, a method for tackling this dataset needs to be able to both handle intra-class variance and inter-class similarity for successful action recognition. Ever since the release of this dataset in 2010, this dataset has emerged as a standard benchmark dataset for evaluating methods for 3D human action recognition, with many results available from previous action recognition approaches. Although this dataset contains both discriminative depth maps and tracked 3D skeletons, the dataset has remained challenging due to both large amounts of noise in the depth maps and skeletons, as well as due to the mixture of intra-class variance and inter-class similarity as mentioned above. Our experiments are run similarly to previous research in the cross-subject setting, by using samples from half of the subjects (*i.e.* subjects $1, 2, 3, 4, 5$) for training, and the rest for testing. In this dataset, there are some frame where the skeleton tracking fails. This causes the skeleton joints in such frames to be incorrectly placed at the origin of the 3D coordinate system. We repair such invalid values by using standard inter-frame linear interpolation. Coordinates are determined to be invalid only when the coordinates $(x, y, z) = (0, 0, 0)$, which is what is output by the Kinect when the subject is either closer than 40 cm, or in case no depth value is able to be measured (*e.g.* due to occlusions or specular surfaces).

The best parameters were $K = 4$ and $J = 50$. The feature dimension was reduced by PCA from 45900 to 152. The best results were gotten using *view-dependent* skeleton-based graphs and can be seen in Table 6.1. The confusion matrix is shown in Fig. 6.2. 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*"). From the results, we can see that although both being effective, mean pooling is performing slightly better than max pooling. 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 due to some action sequences being too short in the dataset, we could not test our descriptor with parameter $K > 4$. Our method achieves perfect classification accuracy for most actions, and is also able to clearly distinguish between actions that are spatially similar, such as "*draw x/circle*" (see Fig. 6.1). Some error modes include our method mistaking the action "*hammer*" for "*draw x*". We believe that the reason is both of these two classes being characterized by similar accelerating movement directions along the 3D space coordinate axes. The SGS descriptor fails to distinguish these actions sometimes because while it is able to capture different ranges of accelerations, the crude structure of the temporal pyramid makes it difficult to capture the fine temporal order that describes the occurrence of these accelerations. See Figures 6.3, 6.4 and 6.5 for examples of successful predictions and mistakes. We can conclude that classes such as "*high throw*" and "*high arm wave*" are very similar spatially. This makes discrimination between them using only skeleton information very challenging, as we do not take into account information such as the shape of the hand, which should differ between the two actions.

Overall, our method results in performance that is comparable to most previous approaches for the task of frontal-view action recognition. Although, we note that our method is unable to achieve the same performance as the high-performing DL-GSGC [60] method, which is based on regularized sparse coding. Still, DL-GSGC requires finding the solution to a heavy optimization problem, while our method enjoys fast computational speed that is linear in the sequence length. 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 [28]. Further, DL-GSGC is designed to handle spatially similarity in a discriminant manner, while our method is unable to handle such similarities with the standard SGWT kernel, as we discussed above. A possible remedy would be to attempt learning SGWT kernels from the graph signals in a discriminant manner [93].

Compared to MMTW [100], our method falls just short in performance. However, our method works well with a temporal pyramid that performs a simple uniform division along the temporal axis, while MMTW discriminatively learns a non-uniform warping template for each class. Therefore, it is reasonable to think that the warping templates learned by MMTW are complementary to our method, and could be a target for exploration in 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 [72] perform well using only spatio-temporal point cloud data, they represent their feature using a spatial histogram, which implicitly encodes the spatial locations.

Previous approaches to 3D human action recognition have also reported results on a different configuration of the MSRAction3D dataset. Three action sets were defined by Li *et al.* [58] so that actions that are visually similar are grouped together. The action sets aim to test the recognition performance when we can only train on a small set of visually similar actions. We conducted experiments using this setup, and the results are listed in Table 6.2. From the results, we can see that contrary to the evaluation on the full dataset, max pooling is seen to perform a bit better than mean pooling. This indicates that the choice of mean or max pooling might depend on datasets. In the action set scenario we have fewer classes, and our method results in performance that is closer to DL-GSGC, while enjoying faster computational speed.

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 6.3, 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. 5.4 for doing dimensionality reduction. We also note that an added

Table 6.3: 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

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. 5.7. 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. 5.8.

Finally, we conduct an experiment where we apply dynamic time warping (DTW) [81] 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 [101, 65]. Wang *et al.* [101] 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 [66], estimation of the parameters requires a large number of available frames [51], 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. (4.1)), and the optimal warping function between the sequences is found by dynamic programming [81]. 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 [8].

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 [56]. 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 [37], 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.

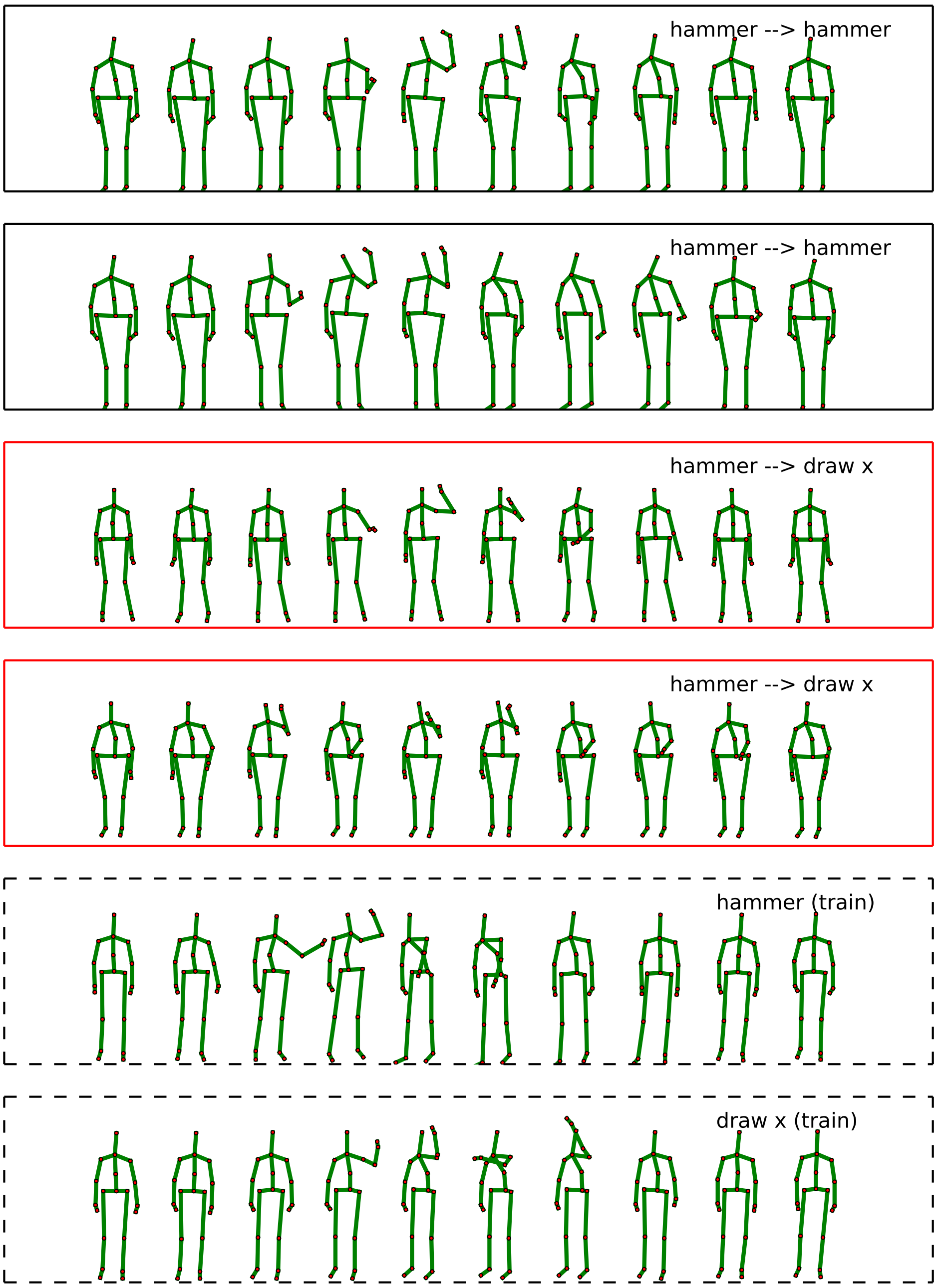


Figure 6.3: Example skeleton sequences and predictions on the MSRAction3D dataset for the action *"hammer"* (notation: `ground-truth --> prediction`). Rows 1,2 show successful predictions. Rows 3,4 show mistakes (red). The last two rows (dashed) show training examples for the two actions. Note the spatial similarity between the actions.

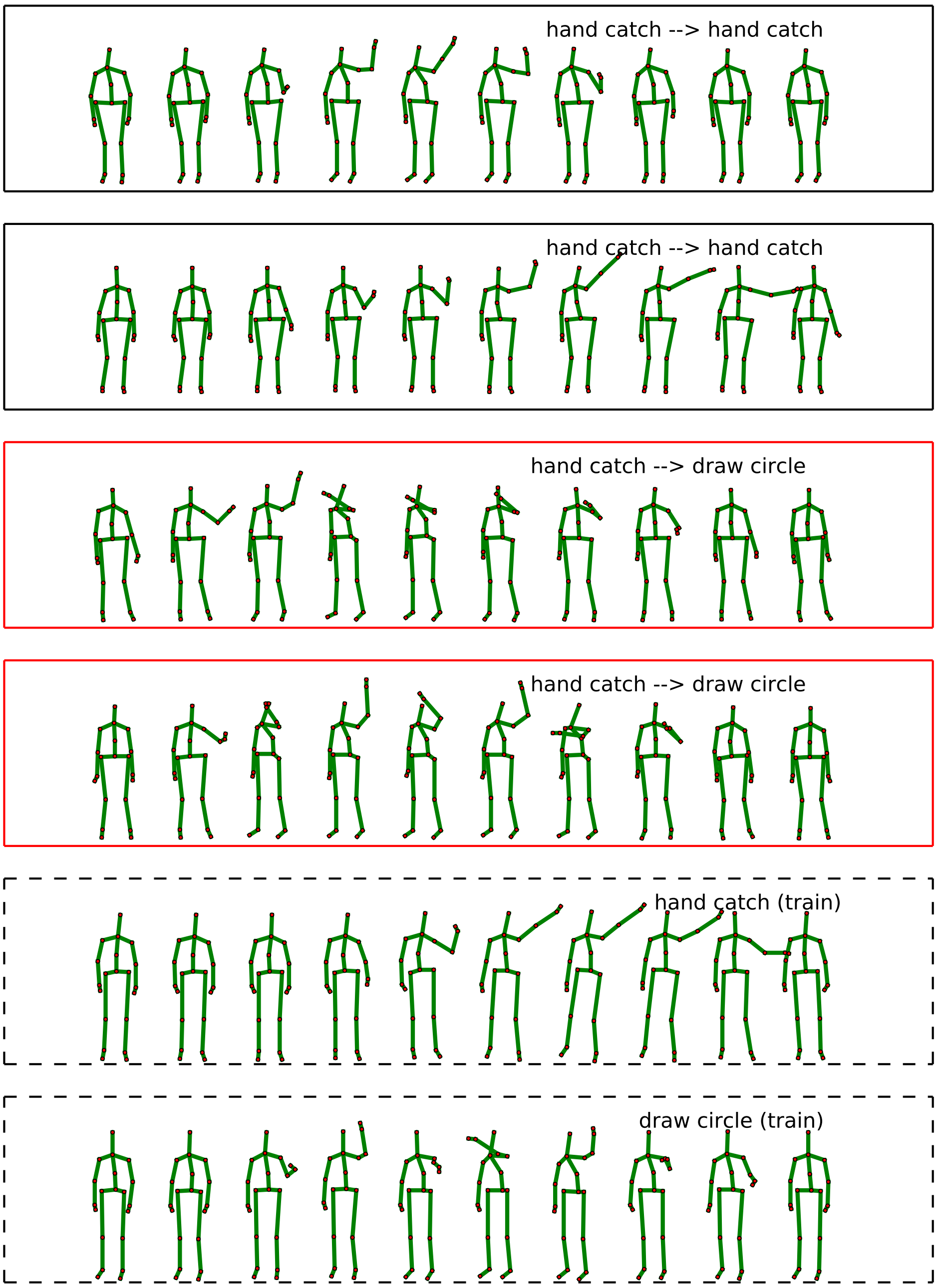

hand catch --> hand catch
hand catch --> hand catch
hand catch --> draw circle
hand catch --> draw circle
hand catch (train)
draw circle (train)

Figure 6.4: Example skeleton sequences and predictions on the MSRAction3D dataset for the action “*hand catch*” (notation: `ground-truth --> prediction`). Rows 1,2 show successful predictions. Rows 3,4 show mistakes (red). The last two rows (dashed) show training examples for the two actions. Note the spatial similarity between the actions.

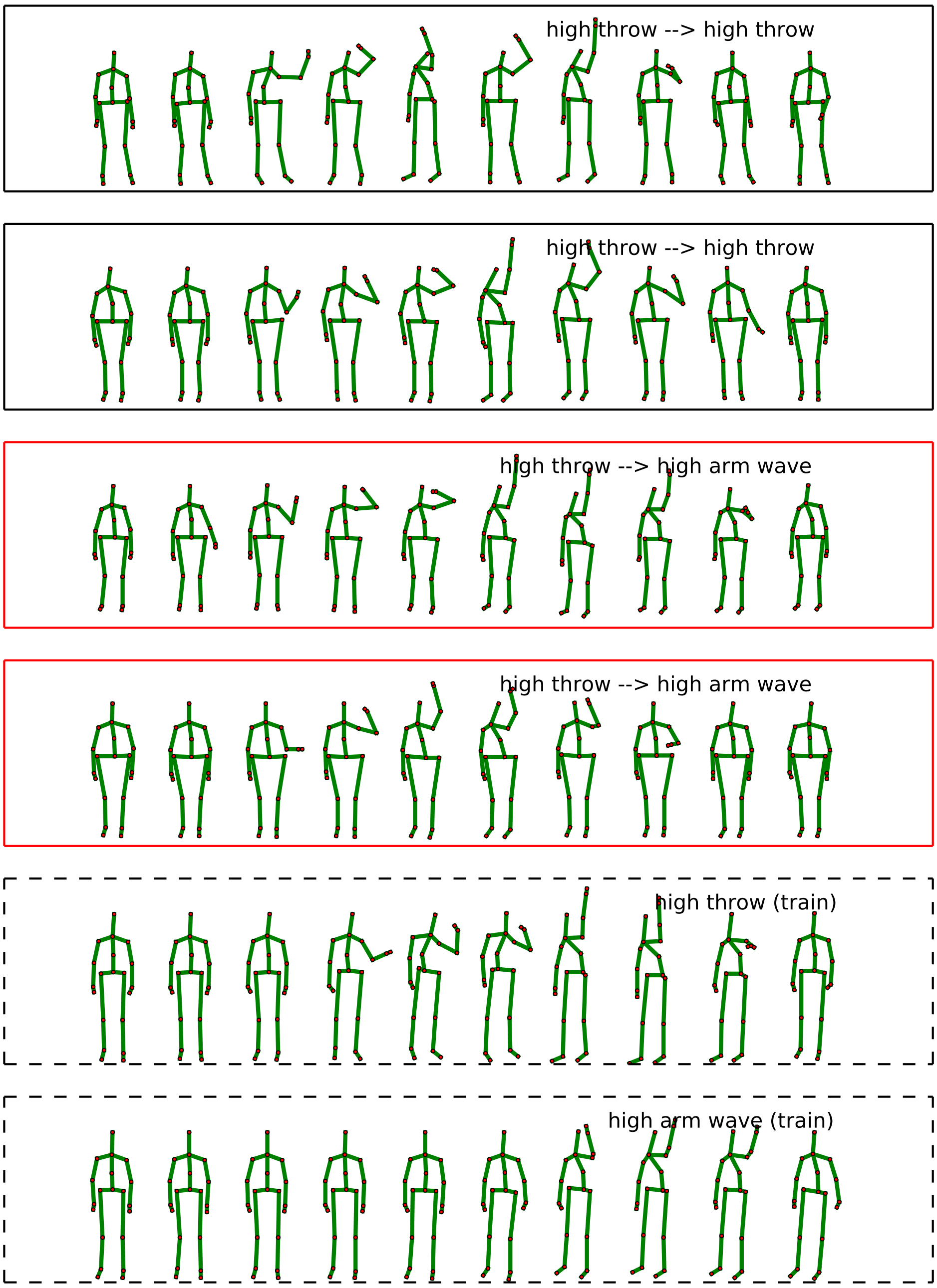

high throw --> high throw
high throw --> high throw
high throw --> high arm wave
high throw --> high arm wave
high throw (train)
high arm wave (train)

Figure 6.5: Example skeleton sequences and predictions on the MSRAction3D dataset for the action "*high throw*" (notation: `ground-truth --> prediction`). Rows 1,2 show successful predictions. Rows 3,4 show mistakes (red). The last two rows (dashed) show training examples for the two actions. Note the spatial similarity between the actions.

Table 6.4: Recognition performance on the MSRActionPairs3D dataset.

Method	Accuracy (%)
HON4D [72]	96.7
SGS(p_{mean}, skeleton-view-dep.)	**96.0**
SGS(p_{max}, skeleton-view-dep.)	**93.1**
AE [101]	82.2
DMM-HOG [111]	66.1

6.2.2 MSRActionPairs3D

The MSRActionPairs3D dataset [72] was created for the special purpose of evaluating the capability of action recognition methods to handle differentiation of motion directionality. The dataset consists pairs of actions that are visually similar in motion, but differ only in directionality. Examples of such actions are "*pick up box*" and "*put down box*". The actions in the dataset were performed three times each by ten subjects. It contains six action pairs in total. Similar to previous research, we evaluate our method in the cross-subject setting. The first half of the subjects 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). The feature dimension was reduced by PCA from 3720 to 80. Table 6.4 lists our results on MSRActionPairs3D. Comparing to HON4D [72], our method achieves comparable performance, despite using only skeleton information. Moreover, our approach is able to recognize the temporal directionality of actions using only a simple temporal pyramid (uniform temporal quantization), while HON4D divides the 4D space into a data-driven discriminative non-uniform quantization. To illustrate the importance of the temporal pyramid, we note that our proposed approach gets accuracy 56.6% with $K = 1$ and 86.3% with $K = 2$. This confirms that for recognizing motion directionality, the effect of using the temporal pyramid pooling scheme is important.

Table 6.5: 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 [114]	98.5
Canonical poses [31]	95.9

6.2.3 UCF-Kinect

The UCF-Kinect dataset [31] contains pre-segmented actions suitable for games. Examples of included actions are "*climb ladder*", "*leap*" and "*twist left*". In total, the dataset contains 1280 action sequences, which are divided into 16 action classes. The dataset contains 16 subjects, who perform each action five times each. Unlike the MSRAction3D and MSRActionPairs3D datasets, UCF-Kinect contains skeletons with only 15 joints. The center hip joint that we use for calculating relative skeleton joint positions is missing. Therefore, we approximate the center hip joint by taking the average 3D position of the left and right hip joints. Our experimental evaluation is run as in the work of Ellis *et al.* [31], where they report 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 $J = 43$. The feature dimension was reduced by PCA from 18480 to 127. Table 6.5 shows our results on UCF-Kinect. Compared to the previous approach based on canonical poses [31], we can see that our proposed method achieves higher performance. Our method also performs slightly better than MP [114]. The experimental results indicate that our proposed approach is capable of recognizing human actions related to games, where all tracked skeleton joints of the body are used for defining the action.

6.2.4 N-UCLA Multiview Action3D

The N-UCLA Multiview Action3D dataset [103] aims to capture daily actions performed by humans from multiple camera angles, such as "*throw trash*",

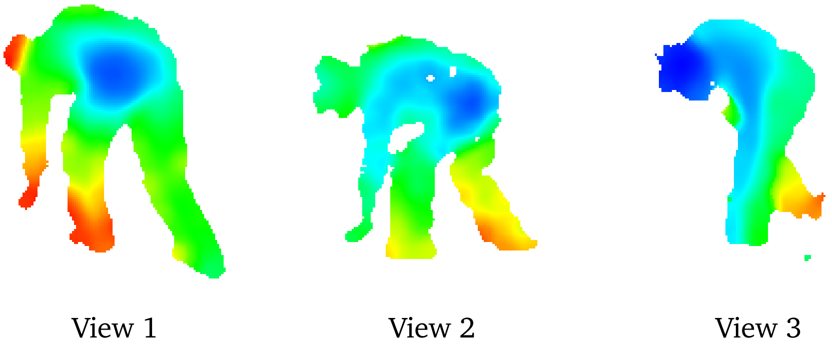


Figure 6.6: Cross-view examples of the action "*pick up with one hand*" from three different views in the N-UCLA Multiview Action3D dataset.

Table 6.6: 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 [76]	75.8
SGS(p_{max}, skel.-view-dep.)	**74.7**
AOG [103]	73.3
nCTE [36]	68.6
SGS(p_{mean}, skel.-view-dep.)	**64.9**
CVP [117]	60.6
DVV [57]	58.5
Hankelets [55]	45.2

"*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. 6.6). Further, several actions include interaction with objects, such as "*drop trash*"

Table 6.7: 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

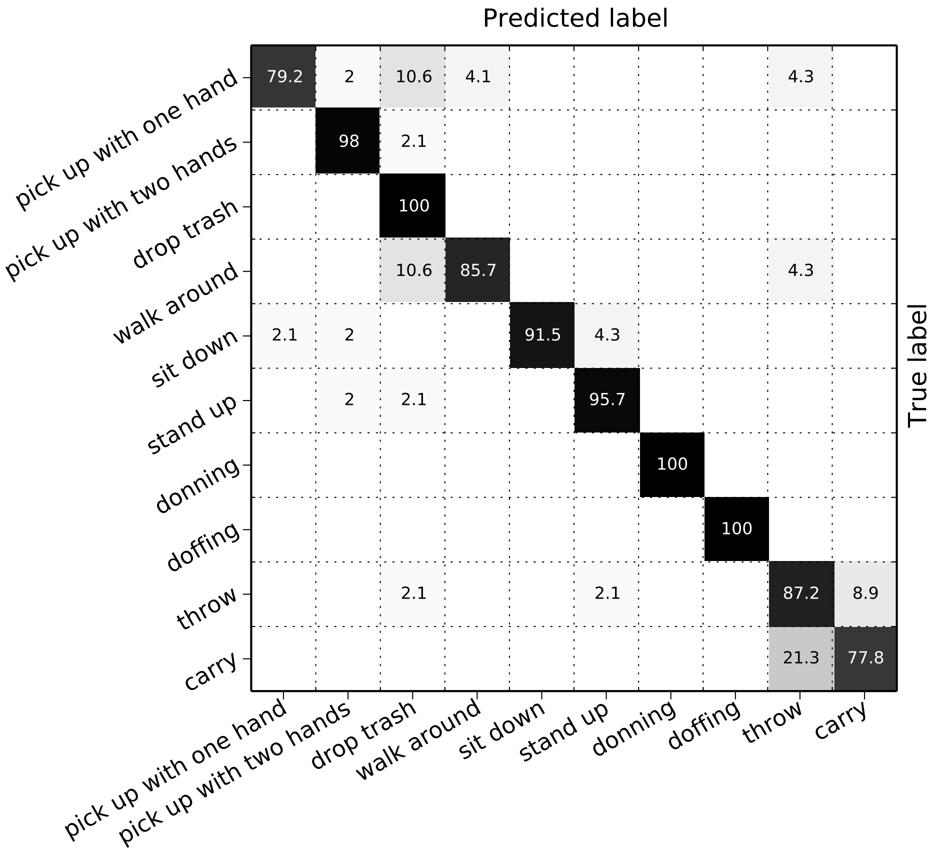


Figure 6.7: Confusion matrix for using our method on the N-UCLA Multiview Action3D dataset. Each cell shows classification accuracy (%) from white (0) to black (100) in the cross-view setting. The average accuracy is 90.8%.

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 6.6. Accuracies of previous work are due to results by Rahmani *et al.* [76]. The confusion matrix is shown in Fig. 6.7. 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 [76], 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 [38]. Figures 6.8, 6.9 and 6.10 show some examples of correct predictions and mistakes. We can see that the actions look quite different from the different view angles, which makes it challenging to generalize the notion of each action across the available views. Especially, note the difference between the visual appearance of this dataset and MSRAction3D (*e.g.* Fig. 6.3). The actions in N-UCLA Multiview Action3D are taken from a wide range of views, while the ones in MSRAction3D are always taken from a frontal angle. Further, the difference between certain actions are very subtle, such as "*walk around*" and "*drop trash*".

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 6.7. 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.* [78], 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 [63]). 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.

6.2.5 Online RGBD Action

The Online RGBD Action dataset [113] 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

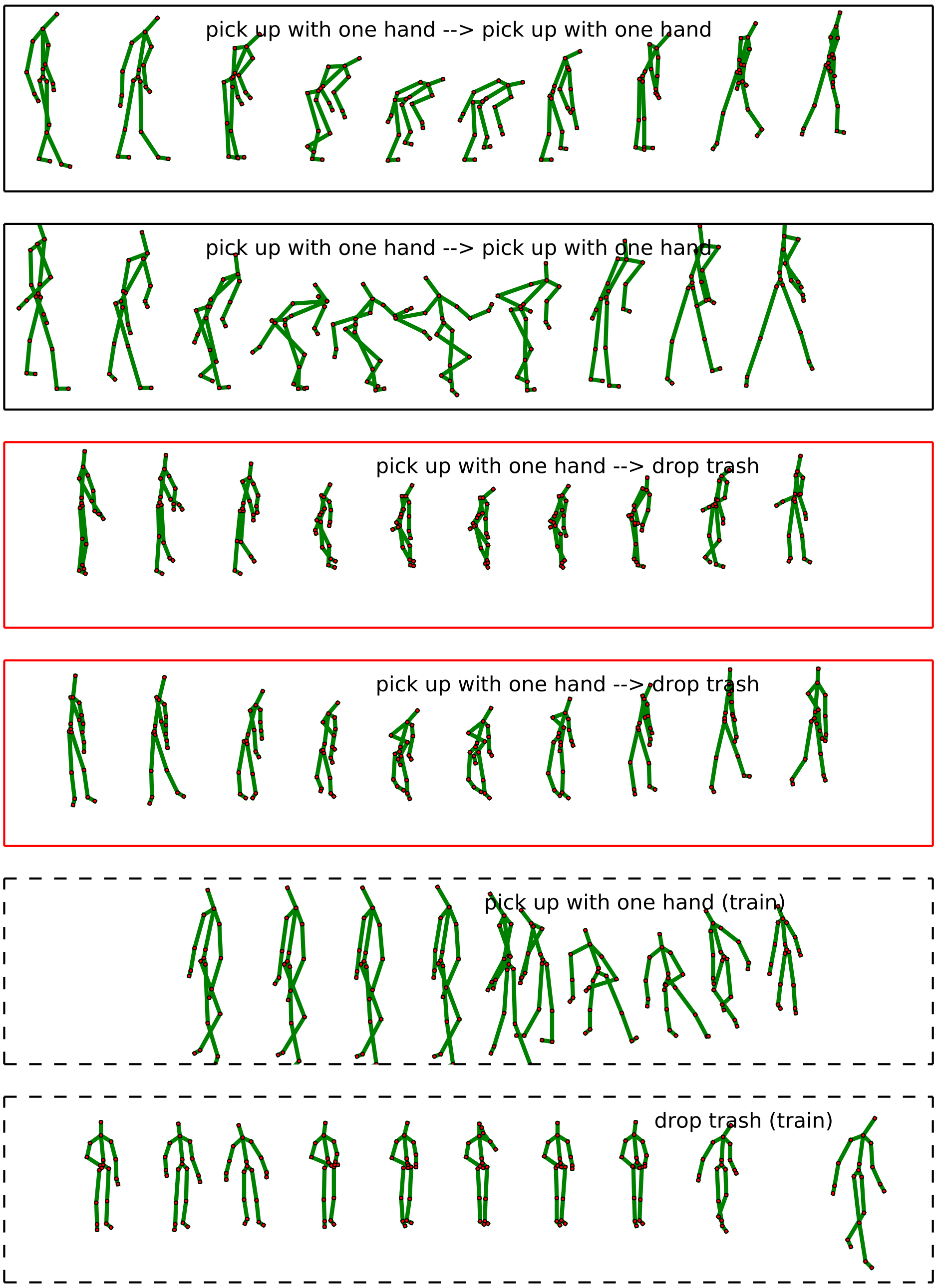

pick up with one hand --> pick up with one hand
pick up with one hand --> pick up with one hand
pick up with one hand --> drop trash
pick up with one hand --> drop trash
pick up with one hand (train)
drop trash (train)

Figure 6.8: Example skeleton sequences and predictions on the N-UCLA Multiview Action3D dataset for the action "*pick up with one hand*" (notation: `ground-truth --> prediction`). Rows 1,2 show successful predictions. Rows 3,4 show mistakes (red). The last two rows (dashed) show training examples for the two actions.

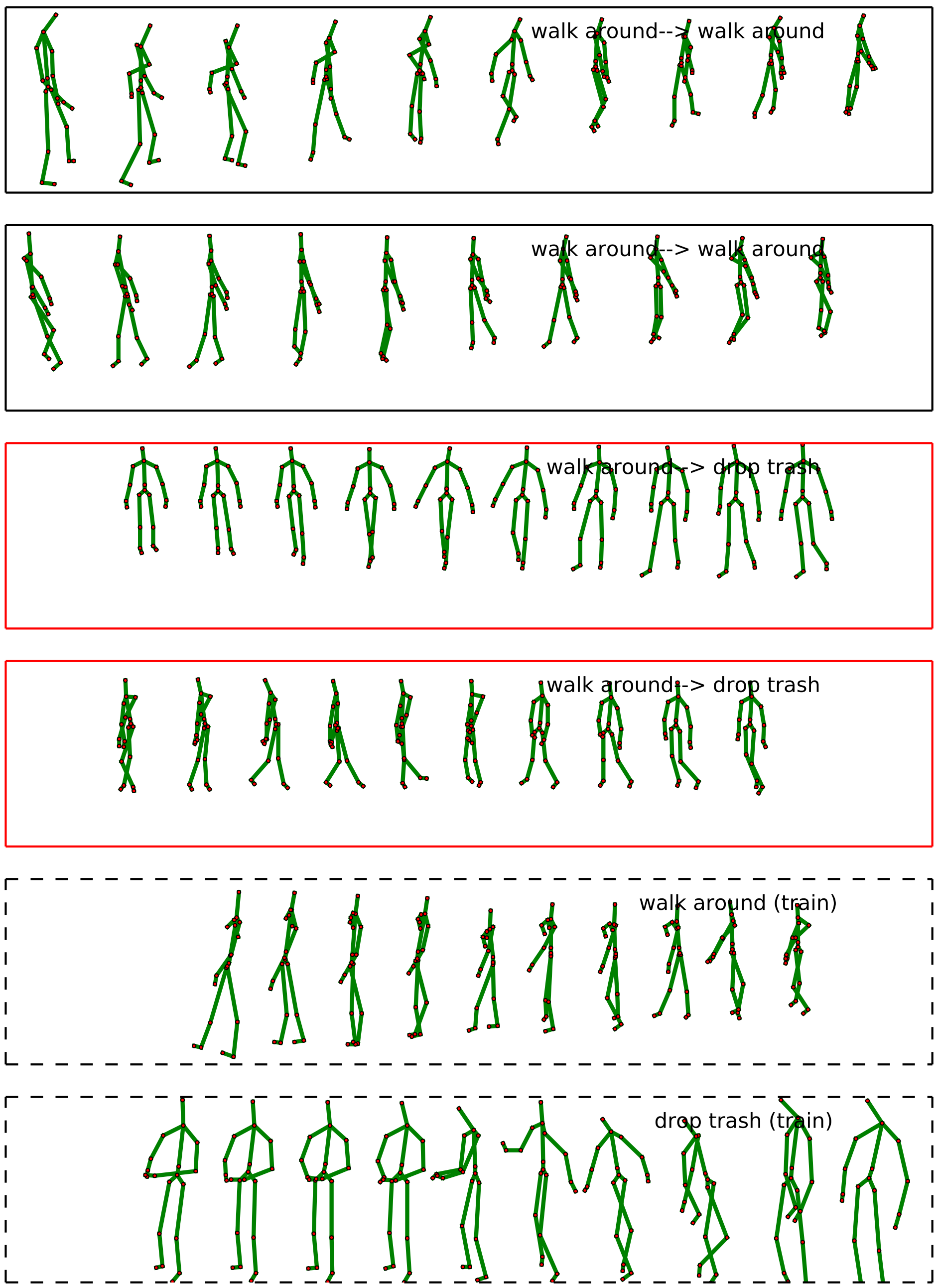

walk around--> walk around
walk around--> walk around
walk around--> drop trash
walk around--> drop trash
walk around (train)
drop trash (train)

Figure 6.9: Example skeleton sequences and predictions on the N-UCLA Multiview Action3D dataset for the action "*walk around*" (notation: `ground-truth --> prediction`). Rows 1,2 show successful predictions. Rows 3,4 show mistakes (red). The last two rows (dashed) show training examples for the two actions.

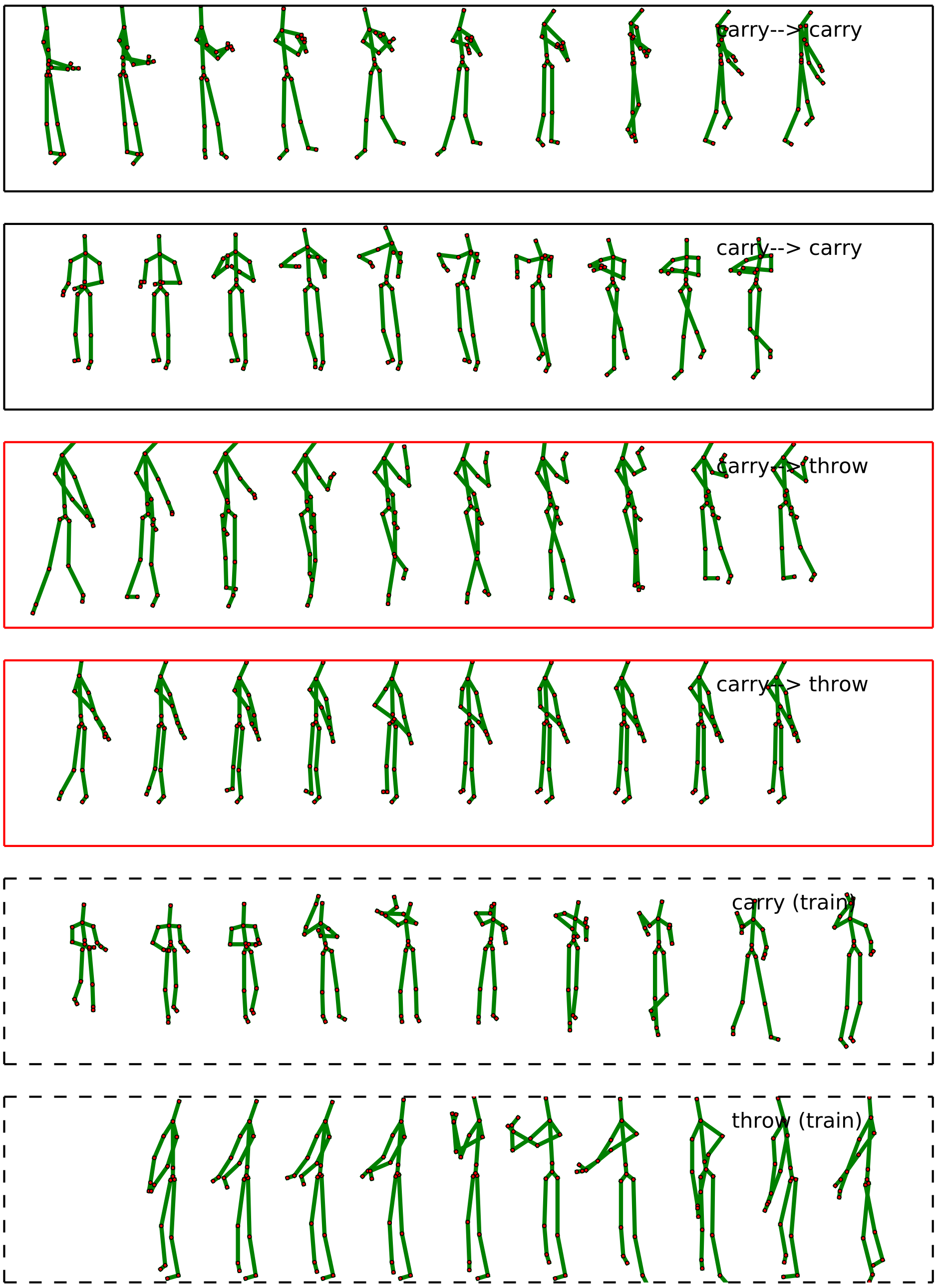


Figure 6.10: Example skeleton sequences and predictions on the N-UCLA Multiview Action3D dataset for the action "*carry*" (notation: `ground-truth --> prediction`). Rows 1,2 show successful predictions. Rows 3,4 show mistakes (red). The last two rows (dashed) show training examples for the two actions.

Table 6.8: Recognition performance on the Online RGBD Action dataset in the SameEnv setting. The plus sign indicates late fusion.

Method	Accuracy (%)
SGS($p_{\mathbf{max}}$, skel.-view-dep.+keypoint)	**72.3**
Orderlet mining [113]	71.4
AE [101]	66.0
SGS($p_{\mathbf{max}}$, keypoint)	**64.7**
DSTIP+DCSF [107]	61.7
SGS($p_{\mathbf{max}}$, skel.-view-dep.)	**59.4**
HOSM [26]	49.5
Eigenjoints [110]	49.1
MP [114]	38.4

Table 6.9: Recognition performance on the Online RGBD Action dataset in the CrossEnv setting. The plus sign indicates late fusion.

Method	Accuracy (%)
Orderlet mining [113]	66.1
AE [101]	59.8
SGS($p_{\mathbf{max}}$, skel.-view-dep.+keypoint)	**57.1**
SGS($p_{\mathbf{max}}$, skel.-view-dep.)	**46.4**
HOSM [26]	50.9
SGS($p_{\mathbf{max}}$, keypoint)	**42.0**
Eigenjoints [110]	35.7
MP [114]	28.5
DSTIP+DCSF [107]	21.5

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.

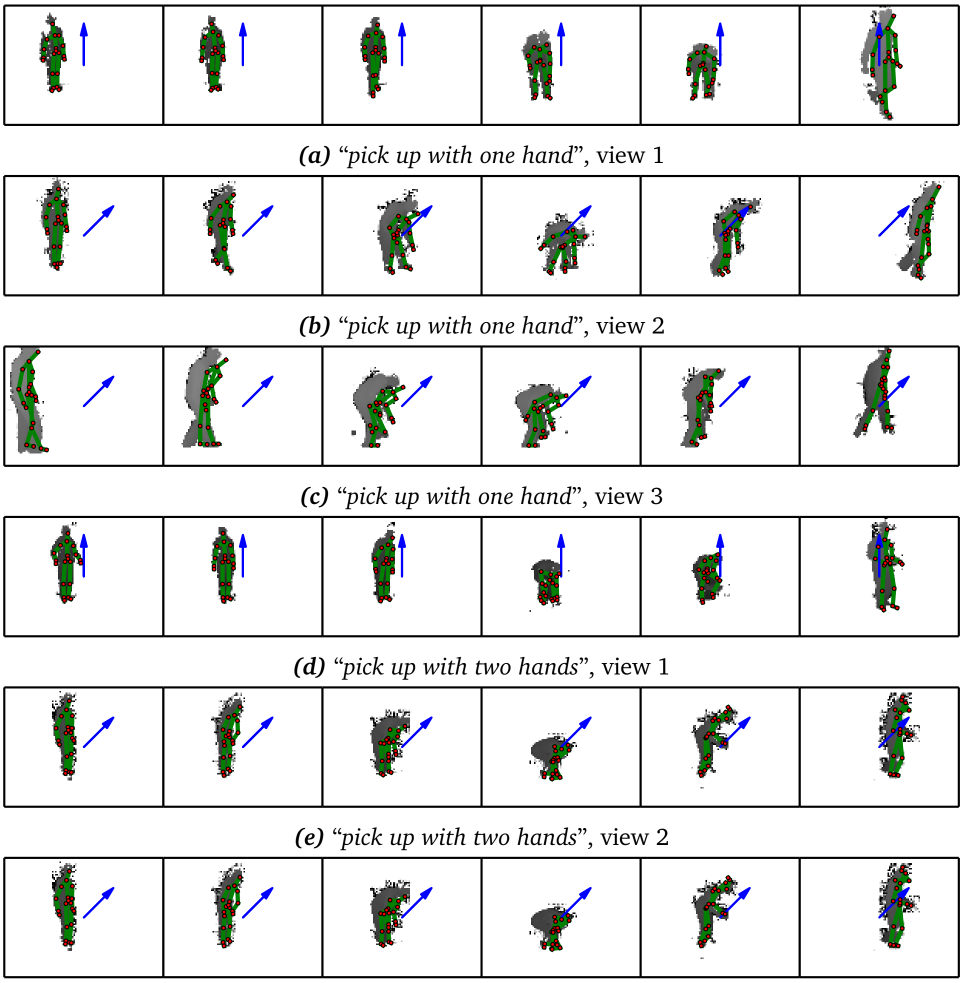
We follow the two experimental configurations used by Yu *et al.* [113]: *SameEnv* and *CrossEnv*. SameEnv reports the two-fold cross-validation accuracy of S1 and S2, while CrossEnv trains on S1 $\cup$ S2 and tests on S3. For simplicity, we evaluate only max pooling and view-dependent skeleton graphs for this experiment. The results are shown in Tables 6.8 and 6.9.

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 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 [113] 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 [107], which works well in the SameEnv setting, but fails to generalize to new environments, as can be seen in Table 6.9.

We further note that Yu *et al.* [113], 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 [101].

6.3 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. 4.2.1. One concern is that the rotation cancellation is largely determined by the estimated up vector $\mathbf{v}_{\text{up}}$ in (4.2). In this section, we briefly investigate the robustness of the up vector estimates for different action classes in the N-



(a) “*pick up with one hand*”, view 1

(b) “*pick up with one hand*”, view 2

(c) “*pick up with one hand*”, view 3

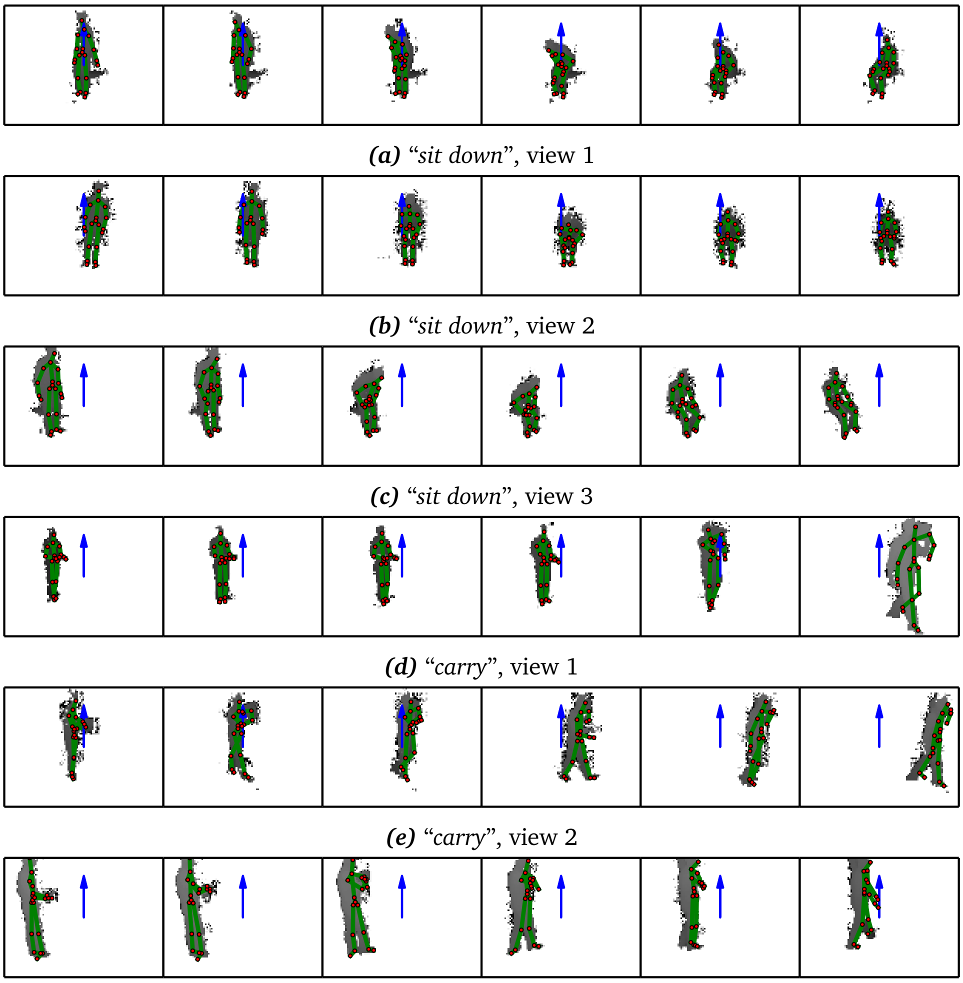
(d) “*pick up with two hands*”, view 1

(e) “*pick up with two hands*”, view 2

(f) “*pick up with two hands*”, view 3

Figure 6.11: Visualization of estimated up vectors (blue arrow) on example frames from the N-UCLA Multiview Action3D dataset. Best viewed in color.

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. 6.11 and 6.12. 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



(a) *"sit down"*, view 1

(b) *"sit down"*, view 2

(c) *"sit down"*, view 3

(d) *"carry"*, view 1

(e) *"carry"*, view 2

(f) *"carry"*, view 3

Figure 6.12: Visualization of estimated up vectors (blue arrow) on example frames from the N-UCLA Multiview Action3D dataset (continued). Best viewed in color.

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. 6.7, and the up vector is estimated in a sensible way for most other action classes.

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. 6.12c. For action sequences where this assumption does not hold, such as in Fig. 6.11b and 6.11c, the estimated vector becomes of lower quality, which is an inherent weakness of the current method, and should be investigated in future work.

6.4 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. 6.13. Qualitative examples on aggregated point clouds are shown in Fig. 6.14.

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

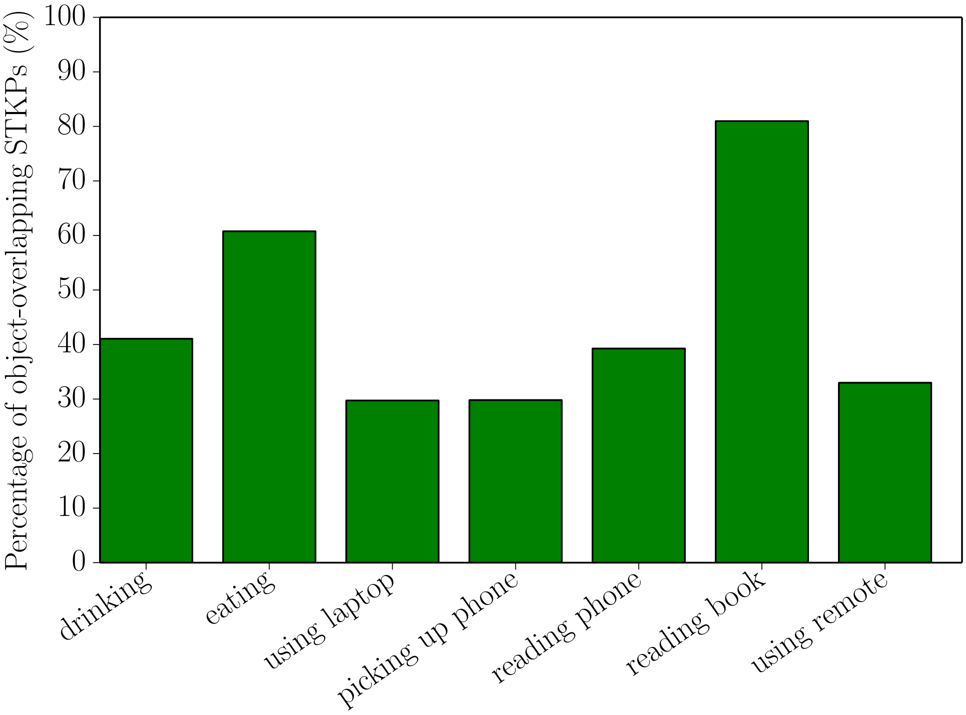


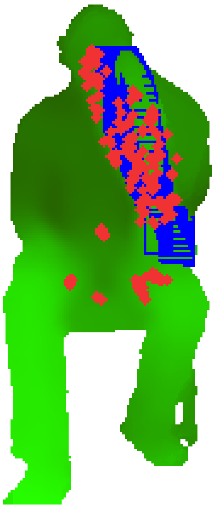
Figure 6.13: The percentage of detected STKPs in the subset S1 ∪ S2 of the Online RGBD Action dataset, that overlap with ground-truth bounding boxes for each action class. The grand mean is 44.95%.

verified by the experimental results in Table 6.8.

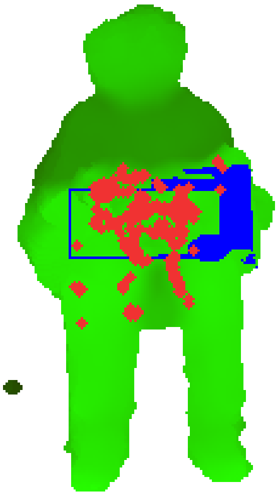
6.5 Parameter Analysis

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. 6.15, and for the N-UCLA Multiview Action3D dataset in Fig. 6.17 and 6.19. 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. 6.19 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



(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 6.14: 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.

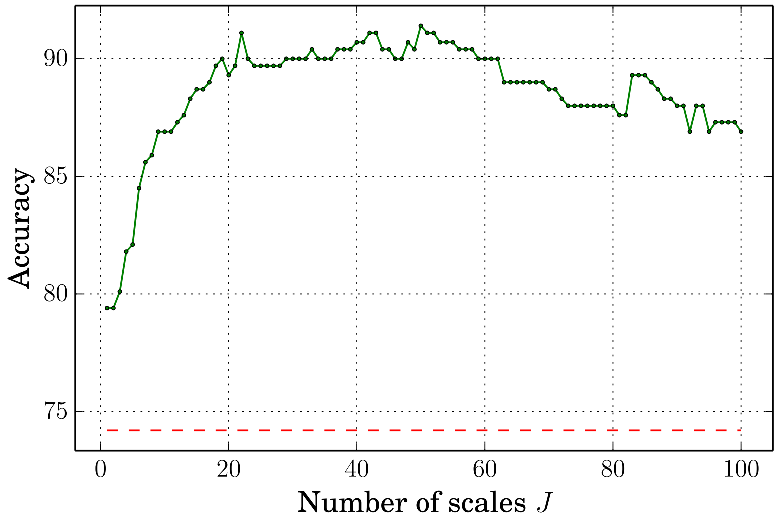


Figure 6.15: Test set accuracy as a function of the number of wavelet scales J on the MSRAction3D dataset using skeleton-based graphs. The dashed line shows accuracy without using the SGWT.

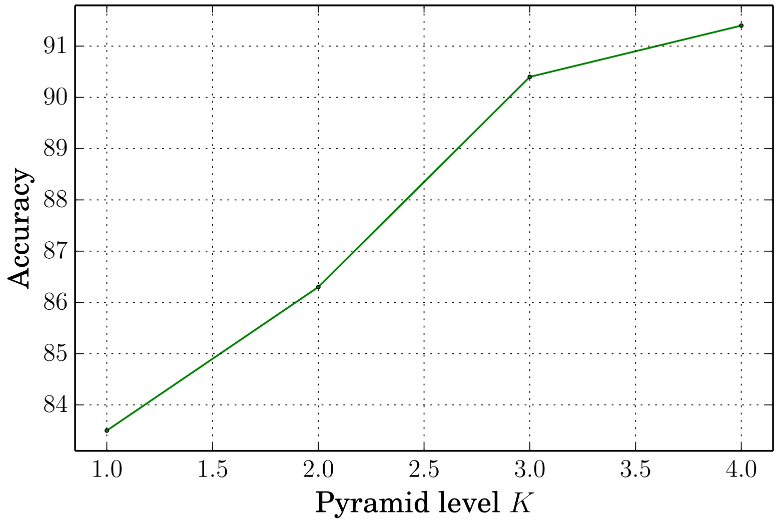


Figure 6.16: Test set accuracy as a function of the number of pyramid levels K on the MSRAction3D dataset using skeleton-based graphs.

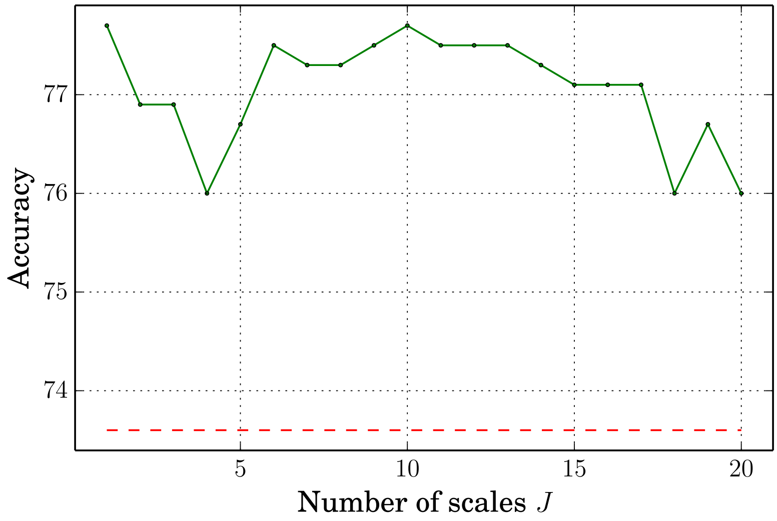


Figure 6.17: Test set accuracy as a function of the number of wavelet scales J on the N-UCLA Multiview Action3D dataset using keypoint-based graphs. The dashed line shows accuracy without using the SGWT.

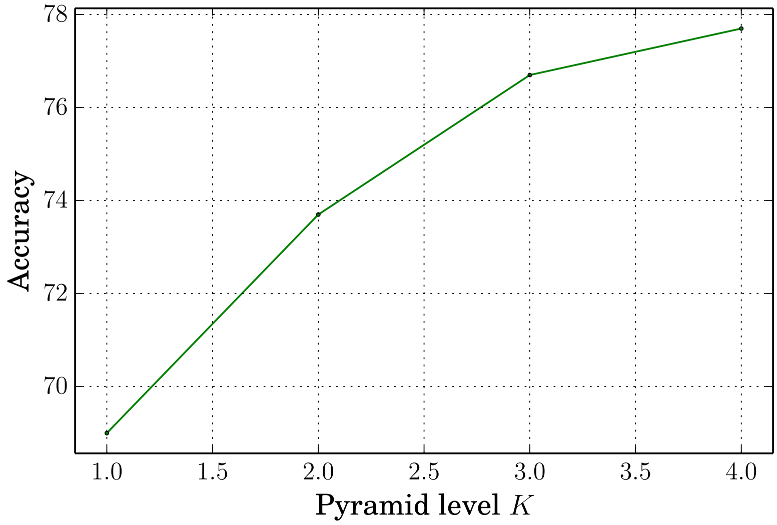


Figure 6.18: Test set accuracy as a function of the number of pyramid levels K on the N-UCLA Multiview Action3D dataset using keypoint-based graphs.

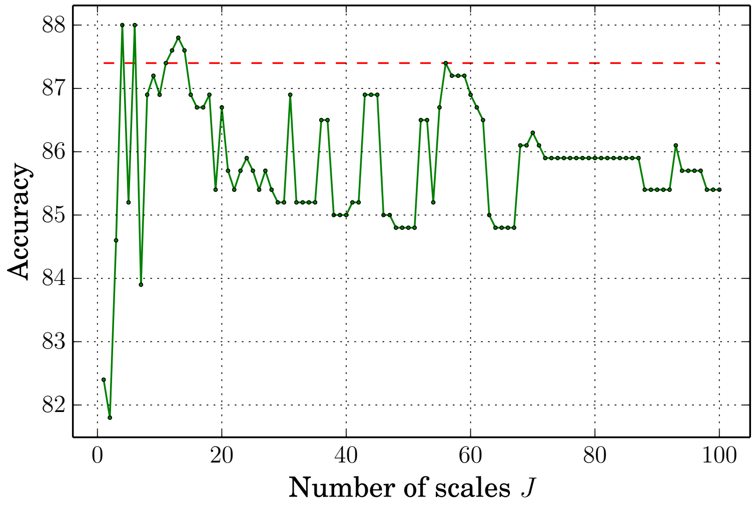


Figure 6.19: Test set accuracy as a function of the number of wavelet scales J on the N-UCLA Multiview Action3D dataset using skeleton-based graphs. The dashed line shows accuracy without using the SGWT.

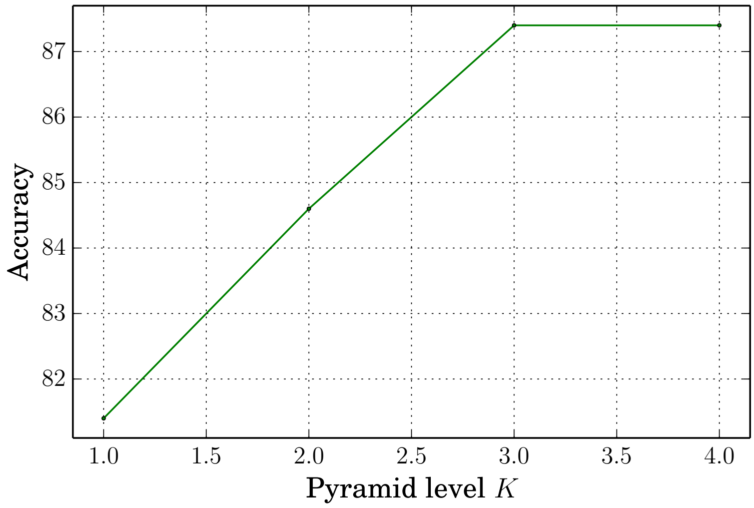


Figure 6.20: Test set accuracy as a function of the number of pyramid levels K on the N-UCLA Multiview Action3D dataset using skeleton-based graphs.

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 [37]. 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 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. 6.16, and for N-UCLA Multiview Action3D in Fig. 6.18 and 6.20. 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.

We note that when not using temporal pyramids ($K = 1$), the SGS feature becomes blind to the temporal order of the action. That is, it will not know in which order the information captured by the coefficients occur. The capability of capturing such motion directionality is very important for actions such as the ones in MSRActionPairs3D. Our results have indicated that leveraging a temporal pyramid (setting $K > 1$) enables us to capture this type of temporal order to some extent. However, mistakes are still being made, which leads us to think that perhaps a non-uniform partition of the temporal axis is needed for improved performance, such as for example in HON4D [72] and MMTW [100]. This might also be important for distinguishing between actions that display a temporal order that is quite local and fine-grained, such as for the MSRAction3D action classes "*hammer*" and "*draw x*". The downside of increasing the parameter K is however that is doubles the size of the descriptor for each increased step. So in practice, a decision of compromise between accuracy and computational speed might have to be made.

6.6 Descriptor Properties

In this section, we discuss some properties of our proposed descriptor.

The classical Laplace operator captures second-order information. As the graph Laplacian matrix $\boldsymbol{\mathcal{L}}$ act as a graph-analog to this operator, we can see that our proposed SGS descriptor is capable of capturing several scales of differential second order information, for each vertex and 3D coordinate axis. The SGWT kernel g, in turn, determines the extents of these scales. On the contrary, the SGWT scaling function kernel h captures aggregated low-frequency information, such as the mean 3D position of the skeleton joints. Based on these properties, we argue that for skeleton-based graphs, the SGS descriptor is able to capture information that discriminates action classes that are defined by specific ranges of acceleration along each 3D skeleton joint. On the other hand, this illustrates the weakness of SGS, which results in difficulty separating action classes that are defined by very similar acceleration properties. This weakness is supported by the experimental evaluation

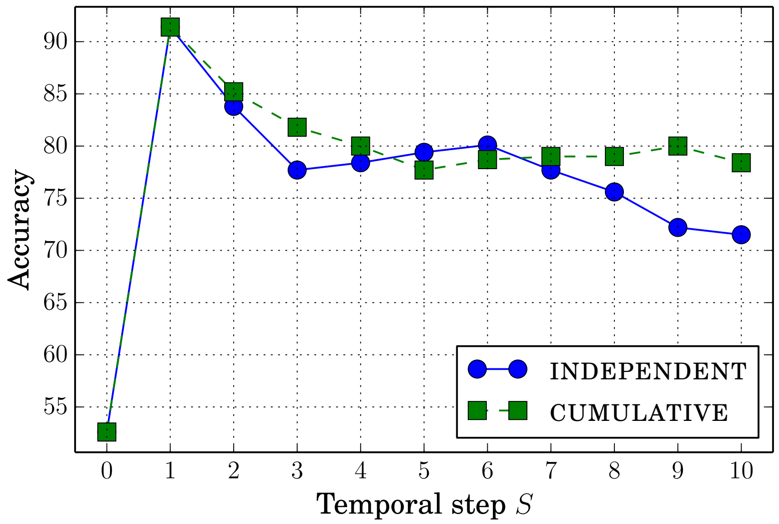


Figure 6.21: Accuracy on the MSRAction3D dataset as a function of the temporal edge connectivity in the augmented graph.

on MSRAction3D, where we could see that "*hammer/draw x/draw tick*" were often mistakenly classified. Indeed, around the same spatial location, these actions also exhibit similar ranges of acceleration.

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.

6.7 Temporal Edge Connectivity

The augmented graph in our proposed method is structured as a sequence of graphs, where the temporal edges connect only to the immediate previous and next frames in the action sequence. By looking at the structure of the weight matrix $\mathbf{W}_{\text{aug}}$ (Eq. (5.1)), we can see that it has a Markov-like property, as each temporal edge is only connected to immediate neighbors of the frame.

One might wonder if changing the temporal connectivity of the graph so that we get a higher-order Markov structure is useful. In order to investigate this, we conduct an experiment where the temporal connectivity is changed.

We investigate two cases. First, instead of connecting a vertex v_{i_t} at time t with a vertex $v_{i_{t+1}}$ at time $t+1$, we connect v_{i_t} to $v_{i_{t+S}}$. Basically $S = 1$ gives us the standard behavior of our method, and $S = 2$ will create temporal edges that skip every other frame. We call this scenario INDEPENDENT. Second, we connect v_{i_t} to all vertices $v \in \{v_{i_{t+s}} : s = 1, \ldots, S\}$, in order to see what effect we will get with jointly having temporal edges with larger step S. We call this scenario CUMULATIVE.

Results are shown in Fig. 6.21. We can see that adding extra edges that skip certain frames deteriorates the performance. Essentially, with these extra edges, our analysis of the graph capturing higher-order properties, such as acceleration, becomes invalid (see Sec. 5.8). It is not clear whether the higher-order information induced by the extra temporal edges is meaningful for action recognition, whereas acceleration certainly is, as was also discussed in previous work [114].

Setting $S = 0$ corresponds to removing all temporal edges, as self-connecting edges are not supported by the graph Laplacian matrix. Note that performance is vastly decreased when removing the temporal edges altogether, which illustrates the importance of capturing temporal information for action recognition.

6.8 Hand-crafted Higher-order Graph Signal

The SGWT captures higher-order information from the graph signal on our augmented graph (see Sec. 5.8). In order to better understand the effect of the graph signal on the graph, we manually create a graph signal that captures higher-order information from the skeleton 3D joint positions, such as velocity and acceleration. From the standard 3D joint position $\hat{\mathbf{p}}_{t,i}$ used for the skeleton-based graphs, we approximate velocity as $\mathbf{v}_{t,i} = \hat{\mathbf{p}}_{t+1,i} - \hat{\mathbf{p}}_{t-1,i}$, and acceleration as $\mathbf{a}_{t,i} = \hat{\mathbf{p}}_{t+2,i} + \hat{\mathbf{p}}_{t-2,i} - 2\hat{\mathbf{p}}_{t,i}$. Using these features, we are interested in seeing if they are useful as graph signals for our task. We investigate all combinations of the above three features for the skeleton-based graphs, by concatenating the feature vectors.

Results are shown in Table. 6.10. The reduced performance is most likely due to the graph being designed for the 3D space, based on the structure of the skeleton joint positions. This graph is then not representative for how

Table 6.10: Recognition performance on the MSRAction3D dataset using hand-crafted higher-order graph signals. The plus sign denotes concatenation.

Graph Signal	Accuracy (%)
pos.+vel.+acc.	67.7
pos.+vel.	66.7
pos.+acc.	63.6
vel.+acc.	63.2
pos.	91.4
vel.	57.0
acc.	57.0

the intrinsic interactions between velocity and acceleration occurs between the joints. The graph signal given by the hand-crafted velocity and acceleration then induces noise into the features as it does not conform to the graph structure. An alternative graph structure that captures a structure more suitable for these hand-crafted augmented features might be necessary, although it should be noted that the SGWT already captures higher-order information such as acceleration from the skeleton joint 3D positions, as was shown in Sec. 5.8. From this experiment, we can conclude that it is important that the graph structure reflects the natural interactions of the signals defined on it, in order for our method to be effective.

6.9 Computational Time

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

6.10 Summary

In this chapter, we have performed experimental evaluation of the proposed action recognition framework. We conducted experiments for frontal-view cross-subject action recognition using three publicly available datasets, and also cross-view action using a standard public benchmark dataset. Additionally, experiments on a public human-object interaction dataset were conducted in order to illustrate the complementary effects of the keypoint-based graphs. Our results indicate that our method is efficient for frontal-view action recognition, where it gains performance comparable to state-of-the-art approaches. More importantly, we demonstrated that our framework is suited quite well for cross-view action recognition, where our method significantly outperformed previous approaches by a 19.8% increase in accuracy. By achieving state-of-the art results for human-object interaction in the same environment setting, and getting comparable results in the cross-environment setting, we also demonstrated that our method is useful for capturing the shape of the spatio-temporal point cloud, for tasks where only the skeleton information is not enough for achieving competitive results. Together with measurements of computational time, showing the efficiency of our method, we can conclude that the proposed approach is effective for recognizing various natural human actions across different subjects and views.

We have demonstrated that our proposed method solves various of the previous problems we have discussed in action recognition (see Sec. 2.5). Namely, our framework is able to handle inter-class variation of actions quite well, as was demonstrated by gaining a competitive 91.4% accuracy on the MSRAction3D dataset. The same experiments also showed that our method is able to recognize actions regardless of subject variations and noisy skeleton measurements. Particularly, our method was shown to work well for handling cross-view camera angles, due to the view-invariance of keypoint graphs and rotation cancellation for skeletons. This merit was demonstrated by a 90.8% accuracy on the N-UCLA Multiview Action3D dataset. Finally, our method was shown to capture temporal directionality through experiments on the MSRActionPairs3D dataset, where we got 96.0% accuracy.

We showed that while our graph representation is undirected, the temporal pyramid pooling approach allows our method to capture temporal directionality. The method was also shown to capture some human-object interaction information by getting 72.3% accuracy on the Online RGBD Action dataset.

While our method has proved effective for some problems in human action recognition, there are still some issues left to tackle. One such task is *explicit* human-object interaction, which can arguably prove useful for distinguishing between actions such as "*high throw*" and "*high arm wave*". While our experiments on the Online RGBD Action dataset showed that our method is able to capture some *implicit* human-object interaction, this does not work for all desirable actions, as our keypoint-based graph is currently only capturing information about the spatio-temporal point cloud at areas where there is large movement. The shape of the hand while throwing an object is not captured by our current system if the movement is small or slow. Finally, we have not explored human-human interaction, which should be an important point of future work for autonomous understanding of activities that involve complex interactions between several entities. It would be interesting to explore whether such interactions could be captured by a graph.

7

Conclusions and Future Work

THIS chapter summarizes the findings of this thesis, draws conclusions, and provides discussion regarding future prospects.

7.1 Conclusions

In this dissertation, 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 [37] 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. By the graph structure, our method captures the temporal interaction between depth map interest points and achieved 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 resulted 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 achieved 72.3% accuracy on the Online RGBD Action dataset. A 96.0% and 98.8% accuracy on the MSRActionPairs3D and UCF-Kinect datasets was also achieved, respectively.

We can conclude that skeleton-based graphs worked well for the recognition of both frontal- and cross-view actions due to them capturing second-order information together with a semantic labeling of the skeleton joints. Keypoint-based graphs, on the other hand, were suitable for cross-view action recognition, where they were shown to improve over the baseline method. Combining both methods through late fusion showed that they are complementary, leading to a significant improvement in recognition accuracy for cross-view action recognition. Similar effects for keypoint-based graphs were also shown for capturing implicit human-object interaction, where late fusion also showed the two graph types to be complementary.

To summarize, our main contributions in this thesis are the following:

- A framework for human action recognition using graph signal processing. Unlike previous work based on *e.g.* temporal pyramids only, our method is able to capture the interactions between interest points throughout time thanks to the graph structure.

- A simple, but efficient skeleton rotation cancellation method based on Gram-Schmidt orthonormalization.

- A view-invariant graph-based action representation that is shown to significantly advance the state-of-the-art for cross-view action recognition.

- An efficient algorithm for calculating the spectral graph wavelet transform that takes advantage of the explicit sparsity structure of our graph.

7.2 Future Work

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. While this study has focused on action recognition, the proposed framework is in general applicable to any time series of graphs.

Although this study of using graph wavelets for action recognition has shown some promising results, it is still in its infant stage, with several interesting future directions open for exploration. The optimized selection of the wavelet kernel g in Section 3.3 could lead to increased performance and is therefore an important point of future work. Especially, deeper understanding of why a certain kernel might provide better performance should be beneficial for accelerating the discovery of further applications of graph signal processing in the future. Likewise, as was shown by the experimental results, the wavelet kernel used in this work is able to handle various action types, but is less successful at handling spatial similarity. Making our method discriminative by learning the SGWT kernel from data could be an interesting approach towards tackling this issue [93]. Other frameworks for processing graph signals could also be explored, together with a more detailed analysis of the suitability of graph signals for action recognition.

This dissertation has focused on the study of simple human actions. While research in recent years has pushed the limits of this task, it is our belief that the performance of approaches to this problem has not yet saturated, and the future holds several major challenges facing this task. Namely, we consider here two major obstacles for future research to overcome. First, large scale datasets are scarce due to the relative rarity of 3D camera data. This has hindered progress in this field compared to others, such as large-scale image recognition, where deep learning methods have thrived in the past years. Second, for successful deployment of action recognition systems to the robotics or security industry, we need scalable and efficient depth map-based methods for recognizing not only simple actions, but *activities* such as human-object and human-human interaction. Quite recently, there has been some development in this direction [86], which is bound to result in a bright future for research in automated human behavior analysis.

List of Publications

Journal paper (peer-reviewed)

- T. Kerola, N. Inoue, and K. Shinoda. Cross-view Human Action Recognition from Depth Maps Using Spectral Graph Sequences. *Computer Vision and Image Understanding (CVIU)*. Accepted for publication. In press. Oct 16, 2016, pp. 1–19. Journal Impact Factor (2015): 2.134

International conference (peer-reviewed)

- T. Kerola, N. Inoue, and K. Shinoda. Spectral graph skeletons for 3D action recognition. In *Asian Conference on Computer Vision (ACCV)*, Singapore, Nov 1-5, 2014, pp. 417–432. Acceptance rate: 28% (227 papers accepted)

Domestic conference

- T. Kerola, N. Inoue, and K. Shinoda. Spectral graph wavelets for skeleton-based 3D action recognition. In *Technical Committee on Pattern Recognition and Media Understanding (PRMU)*, Tohoku Univ., Sendai, Feb 19–20, 2015, pp. 131–136.

Bibliography

[1] Kinect for Windows Sensor Components and Specifications. `https://msdn.microsoft.com/en-us/library/jj131033.aspx`. Accessed 2016-05-24.

[2] The Microsoft Kinect peripheral for the Xbox 360. License: Public domain. `https://en.wikipedia.org/wiki/Kinect#/media/File:Xbox-360-Kinect-Standalone.png`, August 2011. Accessed 2016-05-18.

[3] Jake K Aggarwal and Michael S Ryoo. Human activity analysis: A review. *ACM Computing Surveys (CSUR)*, 43(3):16, 2011.

[4] Tamal Batabyal, Andrea Vaccari, and Scott T Acton. Ugrasp: A unified framework for activity recognition and person identification using graph signal processing. In *ICIP*, 2015.

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006*, pages 404–417. Springer, 2006.

[6] Mikhail Belkin and Partha Adviser-Niyogi. Problems of learning on manifolds. 2003.

[7] Mikhail Belkin, Irina Matveeva, and Partha Niyogi. Regularization and semi-supervised learning on large graphs. In *Learning theory*, pages 624–638. Springer, 2004.

[8] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is “nearest neighbor” meaningful? In *International conference on database theory*, pages 217–235. Springer, 1999.

[9] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738.

[10] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[11] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[12] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL `http://dx.doi.org/10.1023/A:1010933404324`.

[13] Horst Bunke and Kaspar Riesen. Recent advances in graph-based pattern recognition with applications in document analysis. *Pattern Recognition*, 44(5):1057–1067, 2011.

[14] Horst Bunke and Kaspar Riesen. Towards the unification of structural and statistical pattern recognition. *Pattern Recognition Letters*, 33(7): 811–825, 2012.

[15] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[16] S. Chen, F. Cerda, P. Rizzo, J. Bielak, J. Garrett, and 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):2879–2893, June 2014.

[17] Wei Chen and Jason J Corso. Action detection by implicit intentional motion clustering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3298–3306, 2015.

[18] Fan RK Chung. *Spectral graph theory*, volume 92 of *CBMS Regional Conference Series in Mathematics*. American Mathematical Soc., 1997.

[19] Ronald R Coifman and Mauro Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, 2006.

[20] Michael Collins and Nigel Duffy. Convolution kernels for natural language. In *Advances in neural information processing systems*, pages 625–632, 2001.

[21] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, 2002.

[22] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[23] Mark Crovella and Eric Kolaczyk. Graph wavelets for spatial traffic analysis. In *INFOCOM*, 2003.

[24] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.

[25] Ingrid Daubechies et al. *Ten lectures on wavelets*, volume 61. SIAM, 1992.

[26] Wenwen Ding, Kai Liu, Fei Cheng, and Jin Zhang. Learning hierarchical spatio-temporal pattern for human activity prediction. *Journal of Visual Communication and Image Representation*, 35:103–111, 2016.

[27] Xiaowen Dong, Antonio Ortega, Pascal Frossard, and Pierre Vandergheynst. Inference of mobility patterns via spectral graph wavelets. In *ICASSP*, 2013.

[28] Xiaowen Dong, Dorina Thanou, Pascal Frossard, and Pierre Vandergheynst. Laplacian matrix learning for smooth graph signal representation. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 3736–3740. IEEE, 2015.

[29] Hilmi E Egilmez and Antonio Ortega. Spectral anomaly detection using graph-based filtering for wireless sensor networks. In *ICASSP*, 2014.

[30] Michael Elad. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer Publishing Company, Incorporated, 1st edition, 2010. ISBN 144197010X, 9781441970107.

[31] Chris Ellis, Syed Zain Masood, Marshall F Tappen, Joseph J Laviola Jr, and Rahul Sukthankar. Exploring the trade-off between accuracy and observational latency in action recognition. *International Journal of Computer Vision*, 101(3):420–436, 2013.

[32] Ali Farhadi and Mostafa Kamali Tabrizi. Learning to recognize activities from the wrong view point. In *ECCV*. 2008.

[33] T. Fletcher. Support vector machines explained. *Tutorial paper., Mar*, 2009.

[34] Mohammad A Gowayyed, Marwan Torki, Mohamed E Hussein, and Motaz El-Saban. Histogram of oriented displacements (hod): describing trajectories of human joints for action recognition. In *IJCAI*, 2013.

[35] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Computer Vision, 2005 Tenth IEEE International Conference on*, volume 2, pages 1458–1465. IEEE, 2005.

[36] Ankur Gupta, Julieta Martinez, James Little, and Robert 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*, pages 2601–2608, 2014.

[37] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[38] Jungong Han, Ling Shao, Dong Xu, and J. Shotton. Enhanced computer vision with Microsoft Kinect sensor: A review. *IEEE Transactions on Cybernetics*, 43(5):1318–1334, Oct 2013.

[39] Linus Hermansson, Tommi Kerola, Fredrik Johansson, Vinay Jethava, and Devdatt Dubhashi. Entity disambiguation in anonymized graphs using graph kernels. In *CIKM*. ACM, 2013.

[40] Nicholas J. Higham. *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008. ISBN 978-0-898716-46-7.

[41] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[42] Nakamasa Inoue and Koichi Shinoda. A fast and accurate video semantic-indexing system using fast map adaptation and gmm supervectors. *Multimedia, IEEE Transactions on*, 14(4):1196–1205, 2012.

[43] Thorsten Joachims. *Machine Learning: ECML-98: 10th European Conference on Machine Learning Chemnitz, Germany, April 21–23, 1998 Proceedings*, chapter Text categorization with Support Vector Machines: Learning with many relevant features, pages 137–142. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. ISBN 978-3-540-69781-7. doi: 10.1007/BFb0026683. URL `http://dx.doi.org/10.1007/BFb0026683`.

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

[45] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized kernels between labeled graphs. In *ICML*, volume 3, pages 321–328, 2003.

[46] Tommi Kerola, Nakamasa Inoue, and Koichi Shinoda. Spectral graph skeletons for 3D action recognition. In *Asian Conference on Computer Vision (ACCV), Singapore, Nov 1-5*, pages 417–432, 2014.

[47] Tommi Kerola, Nakamasa Inoue, and Koichi Shinoda. Cross-view human action recognition from depth maps using spectral graph sequences. *Computer Vision and Image Understanding (CVIU)*, pages 1–19, 2016.

[48] Woo-Shik Kim, Sunil K Narang, and Antonio Ortega. Graph based transforms for depth video coding. In *ICASSP*, 2012.

[49] George Kimeldorf and Grace Wahba. Some results on tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33(1):82–95, 1971.

[50] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML*, volume 2, pages 315–322, 2002.

[51] Kaustubh Kulkarni, Georgios Evangelidis, Jan Cech, and Radu Horaud. Continuous action recognition based on sequence alignment. *International Journal of Computer Vision*, 112(1):90–114, 2015.

[52] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. IEEE, 2006.

[53] Luc Le Magoarou and Rémi Gribonval. Are there approximate fast fourier transforms on graphs? In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[54] Nora Leonardi and Dimitri Van De Ville. Wavelet frames on graphs defined by fmri functional connectivity. In *ISBI*, 2011.

[55] Binlong Li, Octavia I Camps, and Mario Sznaier. Cross-view activity recognition using hankelets. In *CVPR*, 2012.

[56] Lei Li and B Aditya Prakash. Time series clustering: Complex is simpler! In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 185–192, 2011.

[57] Ruonan Li and Todd Zickler. Discriminative virtual views for cross-view action recognition. In *CVPR*, 2012.

[58] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. Action recognition based on a bag of 3d points. In *CVPR Workshops*, 2010.

[59] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444, 2002.

[60] Jiajia Luo, Wei Wang, and Hairong Qi. Group sparsity and geometry constrained dictionary learning for action recognition from depth maps. In *ICCV*, 2013.

[61] Fengjun Lv and Ramakant Nevatia. Recognition and segmentation of 3-d human action using hmm and multi-class adaboost. In *Computer Vision–ECCV 2006*, pages 359–372. Springer, 2006.

[62] Stephane Mallat. *A wavelet tour of signal processing: the sparse way*. Academic press, 2008.

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

[64] Aditya Krishna Menon. Large-scale support vector machines: algorithms and theory. *Research Exam, University of California, San Diego*, 2009.

[65] Meinard Müller and Tido Röder. Motion templates for automatic classification and retrieval of motion capture data. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 137–146. Eurographics Association, 2006.

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

[67] Sunil K Narang and Antonio Ortega. Perfect reconstruction two-channel wavelet filter banks for graph structured data. *IEEE Transactions on Signal Processing*, 60(6):2786–2799, 2012.

[68] Sunil K Narang, Yung Hsuan Chao, and Antonio Ortega. Graph-wavelet filterbanks for edge-aware image processing. In *Statistical Signal Processing Workshop (SSP)*. IEEE, 2012.

[69] Sebastian Nowozin, Gokhan Bakir, and Koji Tsuda. Discriminative subsequence mining for action classification. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[70] Sebastian Nowozin, Koji Tsuda, Takeaki Uno, Taku Kudo, and Gokhan BakIr. Weighted substructure mining for image analysis. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[71] Thierry Oggier, Michael Lehmann, Rolf Kaufmann, Matthias Schweizer, Michael Richter, Peter Metzler, Graham Lang, Felix Lustenberger, and Nicolas Blanc. An all-solid-state optical range camera for 3d real-time imaging with sub-centimeter depth resolution (swissranger). In *Optical Systems Design*, pages 534–545. International Society for Optics and Photonics, 2004.

[72] Omar Oreifej, Zicheng Liu, and WA Redmond. Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences. In *CVPR*, 2013.

[73] Vasu Parameswaran and Rama Chellappa. View invariance for human action recognition. *International Journal of Computer Vision*, 66(1): 83–101, 2006.

[74] George M Phillips. *Interpolation and approximation by polynomials*, volume 14. Springer Science & Business Media, 2003.

[75] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, March 1986. ISSN 0885-6125. doi: 10.1023/A:1022643204877. URL `http://dx.doi.org/10.1023/A:1022643204877`.

[76] Hossein Rahmani and Ajmal 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*, pages 2458–2466, 2015.

[77] Hossein Rahmani and Ajmal Mian. 3d action recognition from novel viewpoints. In *CVPR, June*, 2016.

[78] Hossein Rahmani, Arif Mahmood, Du Q Huynh, and Ajmal Mian. HOPC: Histogram of oriented principal components of 3D point-clouds for action recognition. In *ECCV*. 2014.

[79] Idan Ram, Michael Elad, and Israel Cohen. Generalized tree-based wavelet transform. *IEEE Transactions on Signal Processing*, 59(9): 4199–4209, 2011.

[80] Cen Rao, Alper Yilmaz, and Mubarak Shah. View-invariant representation and recognition of actions. *International Journal of Computer Vision*, 50(2):203–226, 2002.

[81] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

[82] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986. ISBN 0070544840.

[83] Aliaksei Sandryhaila and José M. F. Moura. Nearest-neighbor image model. In *ICIP*, 2012.

[84] Aliaksei Sandryhaila and José M. F. Moura. Discrete signal processing on graphs. *IEEE Transactions on Signal Processing*, 61(7):1644–1656, 2013.

[85] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[86] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016.

[87] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561, 2011.

[88] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[89] D. I Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre 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):83–98, May 2013.

[90] Khurram Soomro, Haroon Idrees, and Mubarak Shah. Action localization in videos through context walk. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3280–3288, 2015.

[91] Gilbert Strang. *Introduction to linear algebra*. Wellesley-Cambridge Press, Wellesley (Mass.), 2009.

[92] Elena Stumm, Christopher Mei, Simon Lacroix, Juan Nieto, Marco Hutter, and Roland Siegwart. Robust visual place recognition with graph kernels. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[93] Dorina Thanou, David I Shuman, and Pascal Frossard. Parametric dictionary learning for graph signals. In *IEEE GlobalSIP*, 2013.

[94] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.

[95] Nicolas Tremblay and Pierre Borgnat. Graph wavelets for multiscale community mining. *Signal Processing, IEEE Transactions on*, 62(20): 5227–5239, 2014.

[96] Pavan Turaga, Rama Chellappa, Venkatramana S Subrahmanian, and Octavian Udrea. Machine recognition of human activities: A survey. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18 (11):1473–1488, 2008.

[97] Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[98] Martin Vetterli and Jelena Kovačević. *Wavelets and subband coding*. Prentice Hall PTR Englewood Cliffs, New Jersey, 1995.

[99] Antonio W Vieira, Erickson R Nascimento, Gabriel L Oliveira, Zicheng Liu, and Mario FM Campos. Stop: Space-time occupancy patterns for 3d action recognition from depth map sequences. In *CIARP*. 2012.

[100] Jiang Wang and Ying Wu. Learning maximum margin temporal warping for action recognition. In *ICCV*, 2013.

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

[102] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Learning actionlet ensemble for 3d human action recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(5):914–927, 2014.

[103] Jiang Wang, Xiaohan Nie, Yin Xia, Ying Wu, and Song-Chun Zhu. Cross-view action modeling, learning and recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2649–2656, 2014.

[104] Daniel Weinland, Remi Ronfard, and Edmond Boyer. Free viewpoint action recognition using motion history volumes. *Computer Vision and Image Understanding*, 104(2):249–257, 2006.

[105] Daniel Weinland, Edmond Boyer, and Remi Ronfard. Action recognition from arbitrary views using 3d exemplars. In *ICCV*, 2007.

[106] Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Learning to track for spatio-temporal action localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3164–3172, 2015.

[107] Lu Xia and JK 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*, pages 2834–2841, 2013.

[108] Lu Xia, Chia-Chih Chen, and JK Aggarwal. View invariant human action recognition using histograms of 3d joints. In *CVPR Workshops*, 2012.

[109] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[110] Xiaodong Yang and YingLi Tian. Eigenjoints-based action recognition using naive-bayes-nearest-neighbor. In *CVPR Workshops*, 2012.

[111] Xiaodong Yang, Chenyang Zhang, and YingLi Tian. Recognizing actions using depth motion maps-based histograms of oriented gradients. In *ACM MM*, 2012.

[112] Alper Yilmaz and Mubarak Shah. Actions sketch: A novel action representation. In *CVPR*, 2005.

[113] Gang Yu, Zicheng Liu, and Junsong Yuan. Discriminative orderlet mining for real-time recognition of human-object interaction. In *Asian Conference on Computer Vision*, pages 50–65. Springer, 2014.

[114] Mihai Zanfir, Marius Leordeanu, and Cristian Sminchisescu. The moving pose: An efficient 3d kinematics descriptor for low-latency action recognition and detection. In *ICCV*, 2013.

[115] Chenyang Zhang, Yingli Tian, and Elizabeth Capezuti. *Computers Helping People with Special Needs: 13th International Conference, ICCHP 2012, Linz, Austria, July 11-13, 2012, Proceedings, Part I*, chapter Privacy Preserving Automatic Fall Detection for Elderly Using RGBD Cameras, pages 625–633. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-31522-0. doi: 10.1007/978-3-642-31522-0_95. URL `http://dx.doi.org/10.1007/978-3-642-31522-0_95`.

[116] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE MultiMedia*, 19(2):4–10, April 2012. ISSN 1070-986X. doi: 10.1109/MMUL.2012.24. URL `http://dx.doi.org/10.1109/MMUL.2012.24`.

[117] Zhong Zhang, Chunheng Wang, Baihua Xiao, Wen Zhou, Shuang Liu, and Cunzhao Shi. Cross-view action recognition via a continuous virtual path. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2690–2697, 2013.

[118] Xin Zhao, Xue Li, Chaoyi Pang, Xiaofeng Zhu, and Quan Z Sheng. Online human gesture recognition from motion data streams. In *ACM MM*, 2013.

[119] Jingjing Zheng and Zhuolin Jiang. Learning view-invariant sparse representations for cross-view action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3176–3183, 2013.

[120] Xiaofan Zhu and Michael Rabbat. Approximating signals supported on graphs. In *ICASSP*, 2012.

[121] Xiaojin Zhu, Jaz Kandola, John Lafferty, and Zoubin Ghahramani. Graph kernels by spectral transforms. In Olivier Chapelle, Bernhard Schoelkopf, and Alexander Zien, editors, *Semi-Supervised Learning*, pages 277–291. MIT Press, 2006.