2014

Exploring sparsity, self-similarity, and low rank approximation in action recognition, motion retrieval, and action spotting

Chuan Sun
University of Central Florida

EXPLORING SPARSITY, SELF-SIMILARITY, AND LOW RANK APPROXIMATION IN ACTION RECOGNITION, MOTION RETRIEVAL, AND ACTION SPOTTING

by

CHUAN SUN
M.S. Central South University, China, 2009

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2014

Major Professor: Hassan Foroosh

ABSTRACT

This thesis consists of 4 major parts. In the first part (Chapters 1 – 2), we introduce the overview, motivation, and contribution of our works, and extensively survey the current literature for 6 related topics. In the second part (Chapters 3 – 7), we explore the concept of "Self-Similarity" in two challenging scenarios, namely, the Action Recognition and the Motion Retrieval. We build three-dimensional volume representations for both scenarios, and devise effective techniques that can produce compact representations encoding the internal dynamics of data. In the third part (Chapter 8), we explore the challenging action spotting problem, and propose a feature-independent unsupervised framework that is effective in spotting action under various real situations, even under heavily perturbed conditions. The final part (Chapters 9) is dedicated to conclusions and future works.

For action recognition, we introduce a generic method that does not depend on one particular type of input feature vector. We make three main contributions: (i) We introduce the concept of Joint Self-Similarity Volume (Joint SSV) for modeling dynamical systems, and show that by using a new optimized rank-1 tensor approximation of Joint SSV one can obtain compact low-dimensional descriptors that very accurately preserve the dynamics of the original system, e.g. an action video sequence; (ii) The descriptor vectors derived from the optimized rank-1 approximation make it possible to recognize actions without explicitly aligning the action sequences of varying speed of execution or difference frame rates; (iii) The method is generic and can be applied using different low-level features such as silhouettes, histogram of oriented gradients (HOG), etc. Hence, it does not necessarily require explicit tracking of features in the space-time volume. Our experimental results on five public datasets demonstrate that our method produces very good results and outperforms many baseline methods.

For action recognition for incomplete videos, we determine whether incomplete videos that are often discarded carry useful information for action recognition, and if so, how one can represent such mixed collection of video data (complete versus incomplete, and labeled versus unlabeled) in a unified manner. We propose a novel framework to handle incomplete videos in action classification, and make three main contributions: (i) We cast the action classification problem for a mixture of complete and incomplete data as a semi-supervised learning problem of labeled and unlabeled data. (ii) We introduce a two-step approach to convert the input mixed data into a uniform compact representation. (iii) Exhaustively scrutinizing 280 configurations, we experimentally show on our two created benchmarks that, even when the videos are extremely sparse and incomplete, it is still possible to recover useful information from them, and classify unknown actions by a graph based semi-supervised learning framework.

For motion retrieval, we present a framework that allows for a flexible and an efficient retrieval of motion capture data in huge databases. The method first converts an action sequence into a self-similarity matrix (SSM), which is based on the notion of self-similarity. This conversion of the motion sequences into compact and low-rank subspace representations greatly reduces the spatiotemporal dimensionality of the sequences. The SSMs are then used to construct order-3 tensors, and we propose a low-rank decomposition scheme that allows for converting the motion sequence volumes into compact lower dimensional representations, without losing the nonlinear dynamics of the motion manifold. Thus, unlike existing linear dimensionality reduction methods that distort the motion manifold and lose very critical and discriminative components, the proposed method performs well, even when inter-class differences are small or intra-class differences are large. In addition, the method allows for an efficient retrieval and does not require the time-alignment of the motion sequences. We evaluate the performance of our retrieval framework on the CMU mocap dataset under two experimental settings, both demonstrating very good retrieval rates.

For action spotting, our framework does not depend on any specific feature (e.g. HOG/HOF, STIP, silhouette, bag-of-words, etc.), and requires no human localization, segmentation, or framewise tracking. This is achieved by treating the problem holistically as that of extracting the internal dynamics of video cuboids by modeling them in their natural form as multilinear tensors. To extract their internal dynamics, we devised a novel Two-Phase Decomposition (TP-Decomp) of a tensor that generates very compact and discriminative representations that are robust to even heavily perturbed data. Technically, a Rank-based Tensor Core Pyramid (Rank-TCP) descriptor is generated by combining multiple tensor cores under multiple ranks, allowing to represent video cuboids in a hierarchical tensor pyramid. The problem then reduces to a template matching problem, which is solved efficiently by using two boosting strategies: (i) to reduce the search space, we filter the dense trajectory cloud extracted from the target video; (ii) to boost the matching speed, we perform matching in an iterative coarse-to-fine manner. Experiments on 5 benchmarks show that our method outperforms current state-of-the-art under various challenging conditions. We also created a challenging dataset called Heavily Perturbed Video Arrays (HPVA) to validate the robustness of our framework under heavily perturbed situations.

Dedicated to My Parents and My Country

ACKNOWLEDGMENTS

First and foremost, I would like to acknowledge my sincere gratitude to my great advisor Dr. Hassan Foroosh for his precious guidance, encouragement and support in the past years. I sincerely appreciate his kindness, patience, insights, and wisdom to help me make progress in the academic research. I believe the knowledge and skills that I have learned from him will continue to shape me in my future life. Also, I would like to thank the support from the National Science Foundation under grants IIS-1212948 and IIS-091686.

I would like to thank Dr. Rahul Sukthankar, Dr. Marshall Tappen, Dr. Charlie Hughes, and Dr. Mike Moshell, for serving as my PhD committee members and for their valuable suggestions on my research work. I am deeply honored to have them as my committee members. Also, I would like to express my gratitude to Dr. Haiyan Nancy Hu. I appreciate the opportunity to learn and study in her lab during the three months at the beginning of my PhD program.

I would like to thank Dr. Imran N. Junejo who provided valuable suggestions for my research. Also, I want to thank my labmates Nazim Ashref, Muhammad Ali, etc., and friends Junyao Zhang, Guang Shu, Cong Li etc., for their friendship and support.

I would like to thank my home country China. During my 5-year study in US, the Sleeping Lion is awakening.

The most special thanks goes to my girl friend Yuyu Fan, whom I want to marry from the bottom of my heart.

Last but not least, I owe everything to my parents. Without their selfless dedication and sacrifices, I will not come this far.

TABLE OF CONTENTS

LIST OF FIGURES xiv

LIST OF TABLES xvii

CHAPTER 1: LITERATURE REVIEW 1

Recurrence Plot Theory 1

Self-Similarity Matrix 3

Low-Rank Tensor Approximation 4

Action Recognition 5

Motion Retrieval 6

Action Spotting 8

CHAPTER 2: OVERVIEW AND CONTRIBUTIONS 9

Overview 9

Contributions 10

CHAPTER 3: REPRESENTATION OF VIDEO DYNAMICS WITH SELF SIMILARITY 12

Preliminaries of Self-Similarity Matrix (SSM) 12

Joint Self-Similarity Matrix (jSSM) 16

Joint-SSV Construction 17

Motivation 17

Construction Procedure 18

Properties of Joint-SSV 20

Joint-SSV Concretization 21

Silhouette-based Joint-SSV 21

HOG3D-based Joint-SSV 23

Tracked Points Based Joint-SSV 24

CHAPTER 4: MULTILINEAR APPROXIMATION OF JOINT-SSV 27

Motivation 27

Tensor Preliminaries 27

Rank-1 Tensor Approximation Algorithm 28

Feature Properties and Discussion 31

CHAPTER 5: ACTION RECOGNITION USING JOINT-SSV 33

Motivation 33

Two Action Classification Methods 35

Cross-correlation metric based classification 36

Gaussian process based classification 37

Experiments 39

Weizmann dataset 41

KTH dataset 42

UCF sports dataset 43

CMU mocap dataset 44

UCF-CIL dataset 45

CHAPTER 6: ACTION RECOGNITION FOR INCOMPLETE VIDEOS 46

Background and Motivation 46

Sparse video recovery 49

Incompleteness 49

Recovery procedure 50

Compact representations 52

Classification 53

Experiments 54

280 parameter combinations 54

Results and discussions 56

CHAPTER 7: MOTION RETRIEVAL USING MOTION SEQUENCE VOLUME 60

Motion Capture Data Representation 62

Terminology 62

Skeletal model 62

Motion Sequence Volume (MSV) Construction 65

MSV Tensor Approximation 67

Similarity Measure 69

Two Experiments 70

Experiment 1 71

Experiment 2 72

Time and space complexity 74

CHAPTER 8: ACTION SPOTTING VIA RANK-BASED TENSOR CORE PYRAMID . . 79

Two-Phase Decomposition (TP-Decomp) 80

TP-Decomp procedure 81

Theoretical Insights of TP-Decomp 82

Two Operators F_{tk} and F_{cp} 82

Robustness of TP-Decomp under Perturbations 83

Properties 90

Rank-TCP Descriptor 91

Motivation 92

Construction Procedure 93

Boosting Strategies for matching 95

Trajectory-assisted Space Reduction 96

Coarse-to-fine Matching 99

Matching Measure 100

Experimental Evaluation 101

CCWebVideo Dataset 107

CMU Action Spotting Dataset 107

MSR Action Dataset I and II 108

Heavily Perturbed Video Arrays (HPVA) Dataset 108

Analysis 109

CHAPTER 9: CONCLUSIONS AND FUTURE WORK 111

Conclusions 111

Future Work 113

APPENDIX A: MULTILINEAR TENSOR FUNDAMENTALS 115

Multilinear tensor fundamentals 116

LIST OF REFERENCES 117

LIST OF FIGURES

Figure 3.1: Four Lorenz curves and corresponding SSM representations 14

Figure 3.2: The fusion of two three-dimensional trajectories 15

Figure 3.3: Visualization of the Joint-SSV 20

Figure 3.4: Convert silhouette features into time series 21

Figure 3.5: A sequence of SSMs for *Bend* action in Weizmann dataset 22

Figure 3.6: HOG3D descriptor extraction and volume partition 22

Figure 3.7: Build SSMs for CMU Mocap dataset and UCF-CIL dataset 26

Figure 4.1: Rank-1 approximation $\hat{\mathcal{A}} = \lambda U^{(1)} \circ U^{(2)} \circ U^{(3)}$ for a Joint SSV $\mathcal{A}$ 29

Figure 4.2: Rank-1 tensor approximation error for Joint SSV. 30

Figure 5.1: The flow of our action recognition framework 35

Figure 5.2: Screenshots for different action classes of 5 public datasets 40

Figure 5.3: Recognition rate under different HOG3D block depths 42

Figure 6.1: Occluded frames for the KTH *running* action 48

Figure 6.2: Occluded frames for the KTH *running* action 49

Figure 6.3: Occluded frames for the UCF Sports *Swingbench* action 58

Figure 6.4: The k-NN (k = 5) sparse weighted graph for the KTH dataset 59

Figure 6.5: The k-NN and ϵ-NN sparse weighted graphs for the UCF Sports dataset . . . 59

Figure 7.1: The overview of our retrieval framework for motion capture data. 61

Figure 7.2: The tracked joint positions and their indices for forming the initial vector . . 63

Figure 7.3: SSMs built for *Run* and *Kick* actions in the CMU Mocap dataset 64

Figure 7.4: Selected Mocap motion sequences from CMU Mocap dataset 75

Figure 7.5: The constructed Motion Sequence Volume for the selected 5 motion sequences 76

Figure 7.6: Comparisons for the decomposed primary vector and the secondary vector . 77

Figure 7.7: The confusion matrices of motion retrieval for CMU mocap dataset 78

Figure 8.1: The illustration of Tucker decomposition followed by CP decomposition. . . 80

Figure 8.2: Comparison of the resulting quadruples 87

Figure 8.3: Comparison of the resulting quadruples for *running* 88

Figure 8.4: Comparison of the resulting quadruples for *benchswing* 89

Figure 8.5: The point cloud formed by the dense trajectories 95

Figure 8.6: The Precision-Recall Curve 99

Figure 8.7: Video numbered 1 to 4 of MSR I dataset 102

Figure 8.8: (Cont.) Video numbered 5 to 8 of MSR I dataset 103

Figure 8.9: (Cont.) Video numbered 9 to 12 of MSR I dataset 104

Figure 8.10: (Cont.) Video numbered 13 to 16 of MSR I dataset 105

Figure 8.11: Our HPVA dataset 106

LIST OF TABLES

Table 5.1: Recognition rate comparison for 3 datasets under the JSSV-hog3d scheme . . 43

Table 5.2: Recognition rate comparison for 5 datasets between our 3 different schemes 45

Table 6.1: The classification error rates of our two benchmarks 53

Table 7.1: Comparison of retrieval rate from 4 selected motion classes 71

Table 8.1: Average Precision (AP) comparison 101

Table 8.2: Average precision for HPVA results 107

Table 8.3: Time (hours) spent in 3 template matching scenarios for 5 benchmarks 107

CHAPTER 1: LITERATURE REVIEW

In this chapter, we provide brief literature review for 6 topics, namely, the recurrence plot theory, self-similarity matrix, low-rank tensor approximation, action recognition, motion retrieval, and action spotting, respectively. These topics are closely related to our work in this thesis. In the corresponding chapters, we will provide detailed information about the motivation, related works, or backgrounds of each topics. The main thread running through all those topics is the tensor based multilinear algebra.

Recurrence Plot Theory

Recurrence Plots (RP) were initially proposed in [1] to demonstrate the dynamics of trajectories for dynamical systems in phase space. Traditionally, it is difficult to visualize the phase space, because higher-dimensional phase spaces can only be projected into 2D or 3D subspaces for visualization. The RP, on the contrary, allows researchers easily interpret or diagnose a dynamicial system. The information it reveals are interpretable about time scales, which are otherwise inaccessible.

Recurrence Quantification Analysis (RQA) was later proposed to measure the various structures in RPs [2]. The quantification is useful for RAQ because they provide important information for the classification of different types of dynamics in the phase space. There are some important recurrence quantification measures, such as Recurrence Rate (RR), Determinism (DET), Divergence (DIV), Entropy (ENTR), etc. Those measures can be computed for each diagonal line parallel to the main diagonal, and thus can be found as a function of the distance to the main diagonal [3]. RQA was applied in applications in various fields such as physiology [2, 4], geology [5], and economy [6, 7].

Researchers were also interested in learning the relationship between RPs and dynamical invariants [8, 9]. Cross Recurrence Plots (CRPs) are the extension of RPs. The aim of CRP is to interpret the dependencies between two different dynamical systems [10, 11]. As a generalization of linear version of cross correlation function, CRP can be adopted to non-stationary and short time series [1], and it had been successfully used in climate analysis and geological systems [12, 11, 13]. However, CRP method has several drawbacks. For example, CRP is not appropriate to analyze the synchronization of oscillators. Meanwhile, CRP may not be suitable to detect small changes in coupling strength [10].

As an effective alternative, the Joint Recurrence Plot (JRP) was proposed in [1]. It is the Hadamard product of the recurrence plots of the considered sub-systems. It has several advantages compared with CRP. The JRP is well defined if two systems have different dimensions. In addition, the JRP is invariant under coordinate permutation and thus can be used to detect phase synchronization. The JRP can also be calculated using a fixed amount of nearest neighbors. Each individual RP which contributes to the final JRP can then be computed using the same number of nearest neighbors.

During the past decades, the recurrence plot theory has been applied in numerous applications. One of the first application was the heart beat intervals analysis [14], which revealed features for cardiac transplant patients and cardiomyopathy patients. Also, many studies used the RQA in order to monitor disease [15] or to detect cardiac arrhythmia [12]. For example, the RQA method was used to analyze DNA sequence of the genome *caenorhabditis elegans* [16], which showed the long-range correlations in introns and intergenic regions. In addition, the recurrence plot theory has also been used in economics to identify chaos in time series [17]. In [7], RQA was used to find correlations between currencies, which may be difficult to analyze in raw exchange data.

Self-Similarity Matrix

The notion of “self-similarity” has received significant attention in the recent years. The work in [18] describes a gait recognition technique based on the image self-similarity of a walking person and classify the movement patterns of different people. Some works [19, 20] show the effective use of the self-similarity in recognizing different types of biological periodic motions.

The method that is closely related to ours is that of Junejo et al. [21]. The authors exploit the notion of image self-similarities, as proposed by [22]. For a given action sequence, [21] first extracts some low level features. The distances between extracted features for all pairs of time frames are computed and this results in a Self-Similarity Matrix (SSM). Each action sequence is thus reduced to a 2D SSM, and the authors then proceed to extracting some useful features from these SSMs and use it to train the action recognition system.

The concept of self-similarity is also closely related to the statistical co-occurrence of pixel intensities across images captured by Mutual Information [23]. The paper in [24] proposed an image matching method based on internal self-similarity property of images, and explored various definitions of self-similarity to find the best one for image matching.

In terms of video analysis, there are mainly two types of self-similarity based descriptors in the literature, namely the local self-similarity descriptor (LSS) and the global self-similarity descriptor (GSS). In [25], the authors introduced the concept of LSS descriptors that capture internal geometric layouts of local self similarities with videos, and these descriptors are estimated on a dense grid of points in the video data at multiple scales. This type of descriptor captures the internal layout of local regions and can be compared across images which appear substantially different at the pixel level [26]. Built upon this method, [27] explored instead the structure of similarities between all pairs of time-frames in a sequence, and made only mild assumptions about the rough localization of a person in the frame, instead of relying on structure recovery and correspondence estimation. Myriads of other applications are also proposed based on this LSS descriptor [26, 28, 29, 30].

In contrast, in [26] the authors explored instead the global self-similarity and its advantages over the local ones, and proposed two global descriptors: the bag-of-correlation-surfaces and self-similarity hypercubes. The paper further declared that GSS may outperform LSS and is computationally more efficient. However, since the GSS descriptor in [26] was applied merely on Pascal VOC 2007 and ETHZ Shape Classes datasets rather than popular action recognition datasets, we are unable to conclude that on human action datasets GSS still outperforms LSS.

Low-Rank Tensor Approximation

The tensor theory has attracted increasing attention in recent years. For example, tensor approximation has been applied in signal processing [31, 32], computer vision [33, 34, 35, 36, 37, 38], data mining [35, 39, 40], neuroscience [41, 42, 43, 44], etc.

In image and video analysis, [45] presents a multilinear independent component analysis (ICA) method to solve statistical independence problems by encoding the input image as a general tensor. The face recognition problem also often boils down to multilinear discriminant analysis [46, 47]. In [48], the authors propose an algorithm for face representation based on the tensor algebra and differential geometry.

Tensor theory also plays a special role in action recognition. For example, [49] extends the classical canonical correlation analysis (CCA) into that of high-order tensors and proposes a tensor canonical correlation analysis (TCCA) which can extract descriptive correlation features of two videos in the joint space-time domain. The work in [50] presents an action recognition framework based on the idea of non-negative tensor factorization. A recent study in [51] represents videos as a tangent bundle on a Grassmann manifold, and videos are expressed as third order tensors and factorized to a set of tangent spaces.

However, we observed that, those tensor related action recognition methods are either feature-dependent [50], or rely on complex models that are very computationally demanding [49, 51, 52].

Action Recognition

Action recognition has continued to be an active area of research and has thus rightfully attracted much attention from the researchers over the years. Important application domains, such as automatic video indexing and archiving, video surveillance, human-computer interaction, augmented reality, user interface design, and human factors would benefit immensely from a robust and efficient solution to this problem.

There are many factors that make this a challenging problem, including the large variations in performing an action by different people, whether by varying the postures, or the execution speed, illumination variations in the sequences, occlusions and disocclusions, distracting background motions, and perspective effects and camera motion. As a consequence, current methods often resort to restricted and simplified scenarios with simple backgrounds, simpler kinematic action classes, static cameras or limited view variations.

Various approaches have been proposed over the years for action recognition. On the basis of representation, they can be categorized as: time evolution of human silhouettes [53, 54], space-time shapes [55, 56, 57, 58], dense trajectories [54, 59, 60, 61], and local 3D patch analysis [62, 63, 64, 65, 66], generally coupled with some machine learning techniques.

All these works rely primarily on effective feature extraction. These feature extraction methods can be roughly divided into the following four categories: motion based [67, 68], appearance based [57, 69, 70], space-time volume based [55, 56, 57], and space-time interest points or local features based [65, 71, 72, 73, 74]. *Motion based* methods generally compute optical flow from a given action sequence, followed by appropriate feature extraction. However, these methods are known to be very susceptible to noise and easily lead to inaccuracies. *Appearance based* methods are prone to differences in appearance between the training dataset and the testing sequences. *Volume or shape based* methods mostly require highly detailed silhouette extraction, which may not be possible in real-world noisy video dataset.

In comparison with these approaches, the *space-time interest point (STIP) based* methods [63, 72, 73] are more robust to noise and camera movement and also seem to work quite well with low resolution inputs. However, these methods rely solely on the discriminative power of individual local space-time descriptors. Information related to the global spatio-temporal distribution is ignored, and smooth motions cannot be captured using STIP methods. In addition, issues like optimal space-time descriptor selection and codebook clustering algorithm selection have to be addressed, with fine-tuning various parameters, which is highly data dependent [75].

Motion Retrieval

Generation of human motion capture dataset is a very time consuming and an expensive process, but it is also a very critical application for animation and movie industry [76, 77]. An equally important problem is to *identify* or *retrieve* an action sequence that might already be present in the motion capture dataset. Additional complications occur when a particular action sequence in the database has many variants [76, 78]. These variations can be caused by individual differences in expression, posture, motion clothing, perspective effects and camera motions. Also, actions frequently involve and depend on manipulating objects, which adds another layer of variability.

For example, we might have two instances of a kick motion sequence, which may seem similar visually but may differ significantly if compared numerically on a frame-by-frame basis. This work proposes a novel method that aims to address some of the above mentioned issues.

With the growth of available motion databases, indexing and retrieval of motion data has received an increasing attention in the literature. Various approaches have been proposed in the past few years. The work in [79] constructed a hierarchical tree of clusters of motions by using the extracted keyframes for each motion, with deeper levels of the tree corresponding to joints deeper in the skeletal hierarchy. The DTW-based indexing technique in [80] is applied to mocap editing operations such as time-warping, filtering, or motion-warping. [81] proposed a retrieval strategy by precomputing a match web as an efficient searchable representation of all possibly similar motion segments. [76] presented an approach where they first define various kinds of geometric features that are claimed to be invariant under spatial variations. Then, they perform an adaptive segmentation to achieve invariance under temporal deformations. Two motion clips are then considered as similar if they have similar progression of geometric features.

Recently, [82] propose an automated method for identifying logically similar motions in a data set and use them to build a continuous and intuitively parameterized space of motions. However, they require time correspondences for matching two sequences. [83] use a motion pattern discovery and matching scheme that decomposes human motions into a part-based, hierarchical motion representation. A fast string match algorithm is then used for matching. [84] propose a method for global similarity searches based on kd-tree-based local neighborhood searches. The work in [85] applied SVM on extracted geometric motion vectors for human motion classification. However, since the motion retrieval problem always involves a tradeoff between accuracy and efficiency, these methods either focus more on "accuracy" with less "efficiency", or vice versa [86].

Action Spotting

The aim of action spotting is to spatiotemporally detect and localize a given query action within a larger search video. The intra-class variance and scene clutter make action spotting difficult. Some previous works combine tracking and classification for action localization, or treat action spotting and recognition in a joint manner [87, 88].

We observe several drawbacks hindering the performance of existing techniques that rely on extracting specific features. For example, *tracking-based* methods [89, 90] must track multiple body parts or joints and classify actions based on stable motion trajectories. But the tracker initialization and its robustness often impede a fully automatic operation, and manual intervention is therefore inevitable. *Contour/silhouette based* methods [91, 92] seek to extract features from the 3D space-time body shape. But it is often hard to perform robust segmentation for complex videos. Also, the silhouette is ambiguous if the body limbs are in front of the body. For *optical flow-based* methods [93, 94], the dense flow estimates are unreliable when the scene is under camera motion, such as zooming, pan, translation, or along occluding boundaries or in the presence of foreground clutter. *Spatiotemporal gradients based* methods [95, 74] seek to generate compact representations to characterize video spatiotemporal structures. But the presence of foreground clutter in a video could still contaminate dimensionality measurements and thus make the matching fail [87]. *Space-time interest points based* methods [96, 97] are usually adopted to construct global bag-of-words descriptors. Although showing merits, they may fail to detect interest points within shadows, along object occluding boundaries, or in highly dynamic clutters.

In addition, action can sometimes be represented by dense templates of image-based measurements such as the optical flow, spatiotemporal gradients, etc. These measurements are searched in a video using a sliding window formulation, avoiding problematic preprocessing operations such as localization, tracking, and segmentation. But these methods can be computationally costly.

CHAPTER 2: OVERVIEW AND CONTRIBUTIONS

In this chapter, we first present an overview of our frameworks for four major applications, namely, the action recognition, the action recognition for incomplete video, the motion retrieval, and the action spotting. We then summarize our contributions to these four applications.

Overview

To tackle the problem of action recognition, we analyze the concept of self-similarity (SSM), followed by the detailed descriptions of an effective representation called the Joint Self-Similarity Volume (Joint-SSV). The flow of our action recognition framework is as follows. *First*, given an input action video, we extract either low-level features like silhouettes in a frame-by-frame manner, or middle-level features like HOG3D from the partitioned video blocks, or some tracked feature points. *Second*, we transform the feature vector in each frame into an SSM. From the sequence of SSMs we then construct a symmetric and unique 3D structure, which we refer to as the Joint-SSV. This volume holds characteristic information about action dynamics. However, in order to exploit it more efficiently, and handle its large dimension, it is decomposed into three compact and discriminative vectors, two of which are identical (due to symmetry). These descriptor vectors characterize the internal dynamics of an action. *Finally*, these vectors are used for measuring distances to a reference set of vectors for final classification

To tackle the problem of action recognition for incomplete video, we adopt a framework that involves five steps. The first step is the mixing of incomplete and complete videos. To generate a large amount of incomplete videos, we sparsely sample the complete videos under various sparsity settings. Regarding an incomplete video as a three-way incomplete tensor, our second step is to recover incomplete videos using an existing tensor completion algorithm. Different sparsity parameters are tested to handle incompleteness under sparsity settings. At the third step, we generate lower-dimensional representation using a rank-1 tensor decomposition algorithm. The generated compact vectors are fed into the fourth step to form the feature space. In the final step, we use graph-based semi-supervised learning for binary-class action classification.

To tackle the problem of motion retrieval, our framework is as follows. *First*, we set up skeletal model for motion sequence poses for initial representation; *Second*, we convert motion sequence into a series of self-similarity matrix representations in temporal dimension under Euclidean distance metric, thereby creating a Motion Sequence Volume (MSV) structure that encodes the internal dynamics of a motion sequence. *Third*, the structure is decomposed into three low-rank compact vectors using an optimal iterative algorithm. *Finally*, we employ the cross correlation based similarity measure for the final retrieval phase.

To tackle the problem of action spotting, we first treat all the video cuboids involved as three-way tensors in multilinear algebra, and propose a natural yet effective technique called Two-Phase Decomposition (TP-Decomp). *Second*, we explore the rank, a critical factor in determining the tensor dynamics, and observe experimentally that, using the combination of multiple cores outperform that of using a single core. This motivate us to establish a Rank-based Tensor Core Pyramid (Rank-TCP) descriptor that can hierarchically represent a video, yet preserve discriminative ability. Finally, we adopt two effective boosting strategies for template matching.

Contributions

We make multiple contributions in the four fields. For action recognition, (i) we propose a generic framework that can be applied using various features such silhouette, 3d tracked points, HOG/HOF, etc. (ii) we extend the concept of SSM to higher dimensional time-space volumes to characterize action dynamics. (iii) we introduce an optimized low rank tensor approximation algorithm to obtain compact description of high-dimensional time-space self similarity representation.

For action recognition in incomplete video, (i) We cast the action classification problem for a mixture of complete and incomplete data as a semi-supervised learning problem of labeled and unlabeled data. (ii) We introduce a two-step approach to convert the input mixed data into a uniform compact representation. (iii) Exhaustively scrutinizing 280 configurations, we experimentally show on our two created benchmarks that, even when the videos are extremely sparse and incomplete, it is still possible to recover useful information from them, and classify unknown actions by a graph based semi-supervised learning framework.

For motion retrieval, (i) we propose a retrieval framework that does not require temporal alignment in contrast to the conventional methods. (ii) we propose a novel scheme of subspace dimensionality reduction for fast motion sequence retrieval based on rank-1 tensor decomposition. (iii) as a byproduct of low rank decomposition, reduced time-complexity, and hence fast indexing and retrieval is achieved to handle very large databases.

For action spotting, the contributions of the proposed work are summarized as follows. (i) we propose an unsupervised framework for action spotting in videos that does not depend on any specific feature (e.g. HOG/HOF, STIP, silhouette, bag-of-words, etc.), and our solution requires no human localization, segmentation, or frame-wise tracking. (ii) we devise a Two-Phase Decomposition procedure, and both theoretically and experimentally verify its advantages and feasibility. (iii) we propose a Rank-based Tensor Core Pyramid (Rank-TCP) algorithm, whose hierarchical structure enables fast template matching, and the resulting descriptor enables compact representations. (iv) we introduce two very effective boosting strategies that can prune a considerable amount of irrelevant search spaces in template matching.

CHAPTER 3: REPRESENTATION OF VIDEO DYNAMICS WITH SELF SIMILARITY [1]

In this chapter, we introduce in detail how to construct a three-dimensional volume structure called "Joint-SSV" that can fully characterize the dynamics of a video volume. Then we provide three types of concretization schemes for building Joint-SSV. Each of them has distinct properties, and can be applied under various situations.

Preliminaries of Self-Similarity Matrix (SSM)

Let us first introduce some preliminaries of Self-Similarity Matrix. SSM provides important insights into the dynamics of a vector in both spatial and temporal dimensions, which is especially advantageous in a high dimensional space, as stated in the early research in [98].

A SSM can be expressed by a $N \times N$ matrix:

$$R_{i,j}(\epsilon, v) = \Theta(\epsilon - \|v_i - v_j\|_p), i, j \in [1, N],$$

where N is the length of a vector v, and ϵ is a threshold distance. The threshold ϵ is a tuning parameter that changes the characteristics of the SSM dynamics. The $\| \cdot \|$ is a predefined norm, and the function $\Theta(\cdot)$ can be the Heaviside function (i.e. $\Theta(x) = 0$ if $x < 0$, and $\Theta(x) = 1$ otherwise), or other proper filter functions [98]. In this thesis, we will adopt a different function for Θ, as defined shortly.

[1]The content in this chapter and the next chapter was published in the paper: Chuan Sun, Imran Junejo, and Hassan Foroosh, "Action Recognition using Rank-1 Approximation of Joint Self-Similarity Volume", IEEE International Conference on Computer Vision (ICCV, Spain) 2011: 1007-1012.

Concretely, for a row vector $v = (v_1, v_2, ..., v_n)$, whose component v_i is a column vector such that $v_i \in \mathbb{R}^{m\times 1}$ and $v_i = (v_{i1}, v_{i2}, ..., v_{im})^T$, the SSM can be explicitly expressed by

$$M_\epsilon(v) = \begin{pmatrix} 0 & d_{12} & d_{13} & \cdots & d_{1n} \\ d_{21} & 0 & d_{23} & \cdots & d_{2n} \\ \vdots & \vdots & \vdots & & \vdots \\ d_{n1} & d_{n2} & d_{n3} & \cdots & 0 \end{pmatrix},$$

where d_{ij} is the distance between vector components v_i and v_j under distance norm p such that $d_{ij} = \|v_i - v_j\|_p$.

SSM behaves differently given different distance norms p and the thresholds ϵ. In this chapter we set $\epsilon = 0$ for a complete representation for the jSSM representation which is defined below. We use $M(v)$ to represent the resulting SSM of vector v thoughout our work, and we use the ℓ_p-norm defined by $d_{ij} = \{\sum_{k=1}^{m} |v_{ik} - v_{jk}|^p\}^{\frac{1}{p}}$ as the distance metric. This metric gives the Manhattan distance and Euclidean distance when $p = 1$ and $p = 2$, respectively. Note that the SSM holds the following three properties:

1. Symmetry: $R_{i,j} = R_{j,i}$
2. Non-negativity: $R_{i,j} \geq 0$
3. Triangle inequality: $R_{i,k} <= R_{i,j} + R_{j,k}$

If all elements of vector v are identical, namely $v_i = v_j$ for all $i, j \in [1, N]$, the SSM would become an all-zero matrix. We name this type of vector and its resulting SSM the "constant vector" and "constant SSM", respectively. Based on the definition, we observe that a non-constant vector $v \in \mathbb{R}^d$ uniquely corresponds to a SSM M_ϵ^v under threshold ϵ and distance metric p.

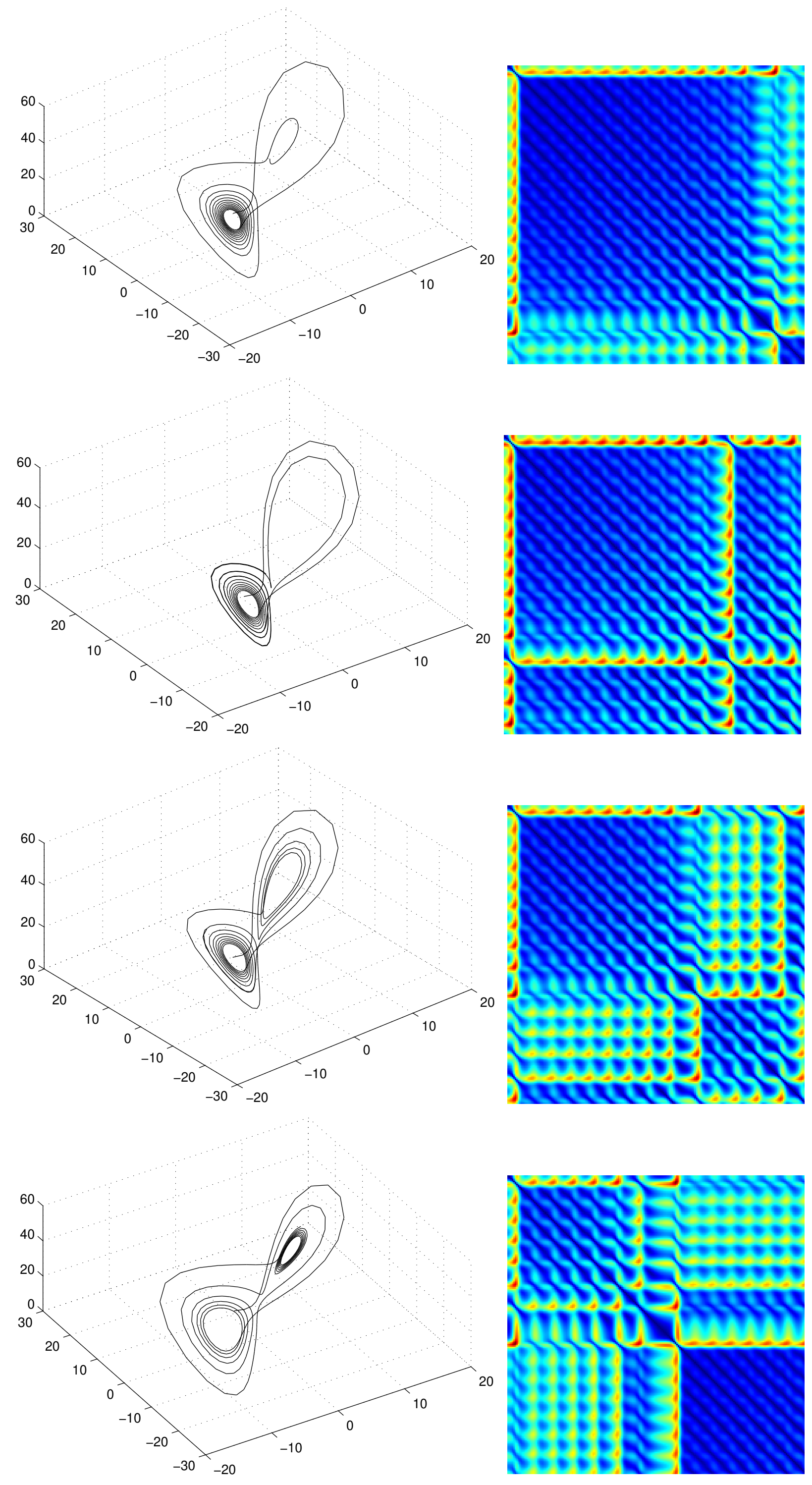


Figure 3.1: Four parameterized Lorenz curves and their corresponding SSM representations.

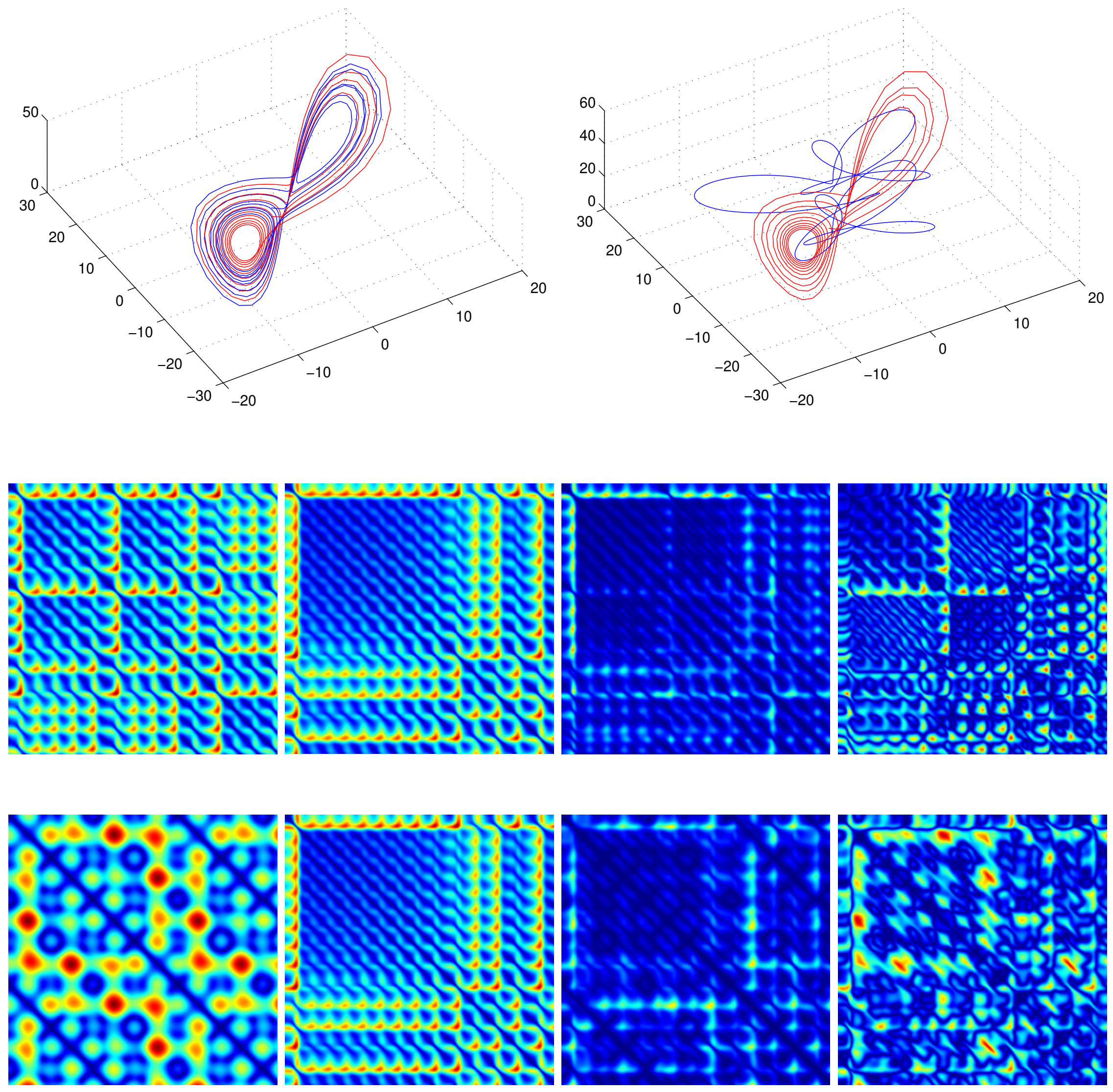


Figure 3.2: The fusion of two three-dimensional trajectories (better be viewed in color). ($1st$ row) The first figure shows the fusion between the trajectories generated by two Lorenz attractors, while the second figure is for the fusion results between a Lorenz curve and a parameterized “Butterfly curve”. ($2nd$ and $3rd$ row) The $1st$ column shows the SSM for the blue trajectory; The $2nd$ column shows the SSM for the red trajectory; The $3rd$ column shows the Joint SSM computed by $SSM_{red} \circ SSM_{blue}$; The $4th$ column shows the SSM computed by the gradient operator $|SSM_{red} - SSM_{blue}|$.

We demonstrate the intuition of SSM by adopting the Lorenz attractor, which is a three-dimensional dynamical system representation that exhibits chaotic flow, noted for its figure-eight shape, as shown in Fig.3.1. The four curves are generated using four sets of different initialization parameters. The curves show how the state of a dynamical system evolves over time in a complex, non-repeating pattern, which, to some extent, is difficult to interpret directly in the three-dimensional space. We observe that the SSM representation is not only able to reveal the subtle internal variations within each dynamical system, but it can also uncover the external discriminating patterns amongst different attractors.

Joint Self-Similarity Matrix (jSSM)

SSM reveals the dynamics of one individual vector. What if we want to encode the mutual dynamics between two vectors? Inspired by the Joint Recurrence Plot (JPR) theory [98], we extend the concept of SSM to the joint SSM, which aims to describe the interaction between two vectors. For notational purpose, we use the short-hand "jSSM" to denote the joint SSM, and present the following definition.

Definition 1. The jSSM is defined as

$$JR_{i,j}^{v,w}(\epsilon_v, \epsilon_w, v, w) = \Theta(\epsilon_v - \|v_i - v_j\|_{p1})\Theta(\epsilon_w - \|w_i - w_j\|_{p2}),$$

in which $i, j \in [1, N]$, ϵ_v and ϵ_w are two internal thresholds, p_1 and p_2 are two distance norms.

The jSSM will be used in our volume construction procedure. It defines an operation that generates one "fused" SSM out of two existing SSMs. The motivation for this extension is that, $JR_{i,j}^{v,w}$ can be viewed as defining the relationship between two trajectories, and represent their interaction in a uniform manner. We illustrate this intuition in the Fig.3.2, in which the mutual dynamics between two different well-known trajectories are shown. In other words, a recurrence will take place if a point v_j on the first trajectory v returns to the neighborhood of a former point v_i, and simultaneously a point w_j on the second trajectory w returns to the neighborhood of a former point w_i. For this reason, this extension is advantageous when we want to fuse two SSMs, whose resulting SSM specifically encodes the mutual dynamics of the input trajectories.

Joint-SSV Construction

In this section, we first discuss our motivation behind the Joint-SSV. Then, we introduce in detail how to build the Joint-SSV, and finally state several properties.

Motivation

For action recognition, SSM has been experimentally proven to be a very useful representation [18, 19, 20, 21]. Amongst those prior works, the one in [21] is closely related to ours. However, each action video in [21] is simply reduced to one single SSM, and many useful and salient local structures may potentially be lost. On the other hand, spatiotemporal volume based analysis, in contrast to the bag-of-features (BoF) based approaches, is preferable not only because local descriptors can be extracted, but also many local contextual information at spatiotemporal key-points can be preserved. In other words, spatial and temporal saliency information embedded in the volume plays a key role in characterizing the volume. For both reasons, our aim in this section is to embed the SSM representation into a characteristic 3D volume, namely, the Joint-SSV, to assist action recognition.

Joint-SSV can be virtually constructed from any feature vector. To demonstrate the generic nature of our method, we consider three types of feature that are used frequently in action recognition, and introduce in detail how we build corresponding volume with regard to each type.

Construction Procedure

The Joint-SSV construction procedure consists of two steps. The first step is to generate a sequence of matrices via Laplacian operation. The second step is to build a 3D volume out of a sequence of fused jSSMs. We now describe the two steps in detail.

Inspired by the Laplacian kernel for edge enhancement in digital image processing, we adopt the one-dimensional Laplacian operator to capture the differences amongst a sequence of SSMs. The reason of using the Laplacian operator stems from three facts:

1. Within an image, the Laplacian operator is able to find the fine details, highlight the edges, and enhance features with sharp discontinuities;
2. As a second order derivative based operator, the Laplacian is known to have stronger response to fine details than the first order derivative (a.k.a. gradient operator) [99];
3. Laplacian is an isotropic operator, implying that self-similarity is not dependent on the direction along the temporal axis.

Let Ψ be a sequence of feature vectors

$$\Psi = \{V_1, V_2, ..., V_N\}$$

with $V_i \in \mathbb{R}^d$. Let $\Gamma : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ be the operator that maps a vector $V_i \in \mathbb{R}^d$ to a SSM $\Gamma(V_i)$. We apply Γ on each element of Ψ, resulting a sequence of SSMs

$$\mathcal{M} = \{M_1, M_2, \cdots, M_N\},$$

where $M_i = \Gamma(V_i)$ and $M_i \in \mathbb{R}^{d \times d}$. We first apply the gradient operator ∇ to $\mathcal{M}$. This yields a resulting sequence

$$\mathcal{G} = \nabla \mathcal{M} = \{G_2, G_3, \cdots, G_N\},$$

where $G_i = dM_i/dt = M_i - M_{i-1}$ and $G_i \in \mathbb{R}^{d \times d}$. The size of $\mathcal{G}$ becomes $N - 1$ since we ignore M_1 for consistent representation. Then, we apply the gradient operator ∇ to $\mathcal{G}$. This yields another sequence

$$\mathcal{L} = \nabla \mathcal{G} = \{L_3, L_4, \cdots, L_N\},$$

where $L_i = \nabla G_i = G_i - G_{i-1}$ and $L_i \in \mathbb{R}^{d \times d}$. Similarly, we ignore the first elements of $\mathcal{G}$. The size of $\mathcal{L}$ becomes $N - 2$. Notice that $\mathcal{L} = \nabla \mathcal{G} = \nabla^2 \mathcal{M}$ and $L_i = M_i - 2M_{i-1} + M_{i-2}$.

The resulting sequence $\mathcal{L}$ carries the 2nd order difference information of $\mathcal{M}$. Analogous to the case when Laplacian operator is applied to one-dimensional vectors, the "edges", "sharp points", or "steep changes" in the SSM sequence $\mathcal{M}$ is enhanced. Since $\mathcal{M}$ corresponds to the spatial dimension of video frames, the temporal dynamics is considered to be embedded in $\mathcal{L}$.

The $\mathcal{M}$ and $\mathcal{L}$ encodes the spatial and temporal dynamics, respectively. Our motivation of defining the concept of Joint-SSV here is to create a representation that encodes both spatial and temporal dynamics in a coherent way, namely

Definition 2. Given a sequence of feature vectors $\Psi = \{V_1, V_2, \cdots, V_N\}$ with $V_i \in \mathbb{R}^d$ and $i \in [1, N]$, the Joint-SSV of Ψ is formed by concatenating a sequence of $N - 2$ matrices $\mathcal{S} = \{S_3, S_4, \cdots, S_N\}$, where $\mathcal{S}$ is obtained by element-wisely fusing $\mathcal{M}$ and $\mathcal{L}$ of Ψ, i.e.,

$$S_j = M_j \circ L_j, \quad j \in [3, N],$$

where $\circ$ is the element-wise multiplication operator between the two matrices M_J and L_j.

According to **Definition 1**, both M_j and L_j are SSMs, and the resulting S_j is thus a jSSM. The $\mathcal{S}$ is a collection of jSSMs, and the Joint-SSV is the concatenation of all jSSMs in $\mathcal{S}$.

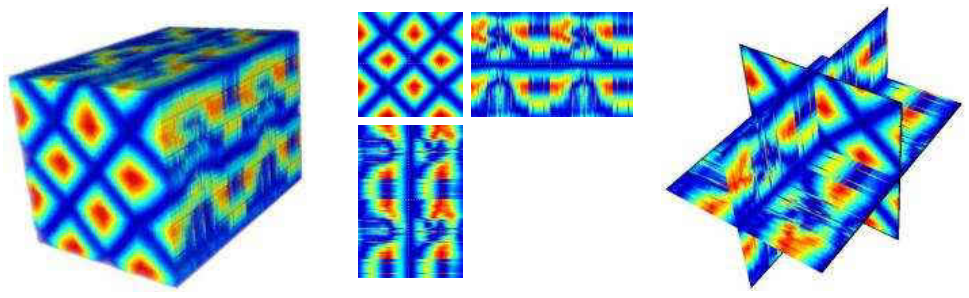


Figure 3.3: Visualization of the symmetric Joint-SSV. The middle figure shows its cut in three directions. The right figure shows the X-section of the volume

Properties of Joint-SSV

The Joint-SSV has two distinct properties. First, the Joint-SSV is a symmetric 3D structure. The reason is that for a given vector Ψ, all elements of its $\mathcal{M}$, $\mathcal{G}$, and $\mathcal{L}$ consist of symmetric matrices. Second, the Joint-SSV encodes both spatial and temporal dynamics of an action video, whose spatial dynamics is frame-wisely encoded in SSMs, while its temporal dynamics is first enhanced by the Laplacian operator, then encoded by the concatenation of all fused jSSMs. In a word, the construction of Joint-SSV enables us to combine both the inter-frame dynamics and the intra-frame dynamics in an video.

We illustrate a Joint-SSV in Fig.3.3. The central figure of Fig.3.3 shows its cut in three directions. The X-section representation demonstrates its internal dynamics. It is worth noticing that this volume might also encode redundant information, and we are facing the problem of further reducing its dimensionality and removing its redundancy. For human action videos, we will provide three schemes to construct the volume from various features, either low-level feature like silhouettes and tracked points, or mid-level feature like the HOG3D descriptor.

Joint-SSV Concretization

In practice, to make full use of the Joint-SSV, we need to build Joint-SSV upon a certain feature. In this section, we present three concretized Joint-SSV and their construction procedures.

Silhouette-based Joint-SSV

Human silhouette in an action has been extensively explored in the action recognition literature. The advantage of silhouette-based approaches is that silhouette as a feature can be easily extracted from raw video frames using object localization and background substraction. Silhouettes can be easily extracted from static or uniform action background. However, it is harder or even impractical for more challenging action sequences with dynamic background. For this reason, we only test this scheme on Weizmann dataset, which provides well-extracted silhouette features.

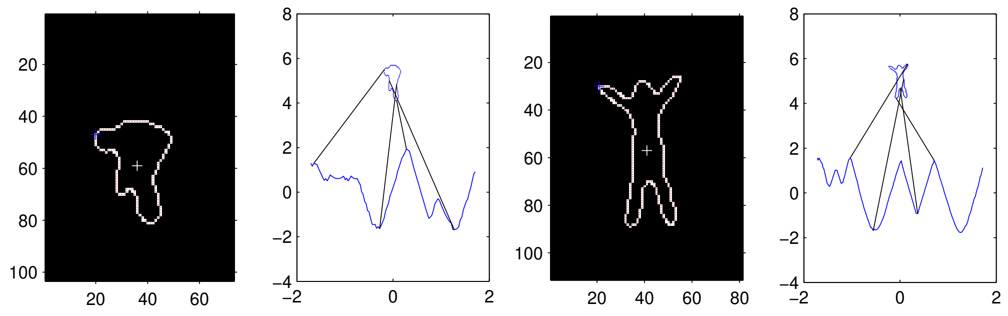


Figure 3.4: Convert silhouette features to time series using the method in [100] for *Bend* and *Jack* action from the Weizmann dataset

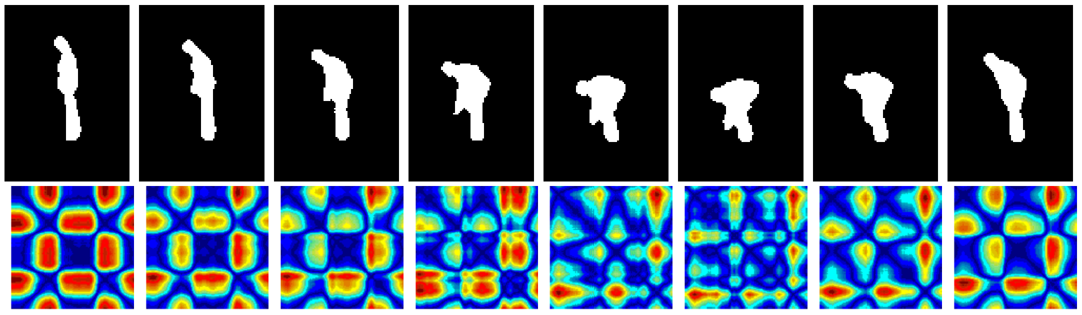


Figure 3.5: A sequence of computed SSMs for frames selected from the the *Bend* action in the Weizmann dataset. Note that all above SSMs are of identical dimension.

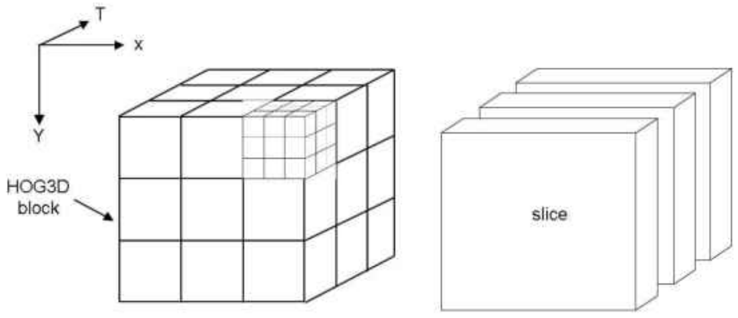


Figure 3.6: (Left) Extracting HOG3D feature descriptor after the dense sampling for the action volume in ROI and the partitioning of the volume into blocks; (Right) All blocks with the same temporal location form a *slice*. Each slice is further vectorized to a vector feeding into the Joint SSV construction procedure

To answer the important question of how to represent silhouette shapes efficiently and robustly, various solutions such as shape moments [101], Fourier descriptors [102], and shape context [103] are proposed. In our framework, we first represent the silhouette in each frame using SSMs, and we transform an action sequence into a volume frame by frame. In detail, we extract the contour from silhouette in each frame and transform the contour into time series using the method in [100], as shown in Fig.3.4. The time series are normalized to zero mean and unit variance before being fed into the framework as input vectors to generate the Joint SSV. Fig.3.5 shows a sequence of generated SSMs for the *Bend* action in the Weizmann dataset. It can be observed that both salient and subtle differences between silhouette contours are reflected in SSMs.

HOG3D-based Joint-SSV

We employ the dense representation as in [104], and use HOG3D descriptor [105] at densely distributed locations within a Region of Interest (ROI) centered around the actor, and partition the volume into regular overlapping *blocks*, and all blocks are then partitioned into small regular *cells*. Histograms of 3D gradient orientations, generated using dodecahedron based quantization [105] with 6 orientation bins, for cells within a block, are then computed, and concatenated to form a block descriptor. Here we name all blocks within the same temporal location a *slice*, as shown in Fig.3.6.

We used the same configuration for defining ROIs but a different block setup as in [104]. We used $2^{\kappa} \times 2^{\kappa} \times 2^{\tau}$ pixel blocks subdivided into $2 \times 2 \times 2$ cells, and computed the HOG3D descriptor for each block. Note that κ and τ are parameters that control the size of blocks. We let κ range from 2 to 4. Otherwise, the larger the κ is, the less the number of blocks for each slice will be, which may be disadvantageous for the computation of Joint SSVs. The τ ranges from 1 to 5. It can control the depth of the generated volume.

Slices overlap with each other between consecutive ones, yielding a redundant representation, which enhances the discriminative power [104]. Within each slice, all blocks are concatenated in row order into a block sequence. This sequence is a vector used for building the self-similarity matrix. Using all slices, we then construct a Joint SSV out of SSMs and Joint SSMs using the procedure described in Section 4.

Tracked Points Based Joint-SSV

We evaluate our framework also on the CMU Mocap dataset and the UCF-CIL dataset [2] containing tracked points at limbs for human actions. Both datasets contain tracked points for different joints of human body for each action frame.

Human pose can be represented using a simplified model of human *skeleton*, composed of bones that are connected by *joints*. The position of all joints at a given time is known as a *pose*, described as a vector $p \in \mathbb{R}^{3\times|J|}$, where $|J|$ is the number of joints in the skeletal model and each joint requires 3 elements to describe its 3D position. A mocap sequence can be then formally described as a time-dependent sequence of poses. This can be represented by a 2D matrix $S \in \mathbb{R}^{T\times(3\times|J|)}$, where T is the number of poses (frames) in the mocap sequence.

The CMU mocap dataset uses a skeletal model of 32 joints (i.e., $|J| = 32$). It provides a detailed configuration for the joints such as the thumb joint. Since 13 joints are sufficient to represent the skeleton for a motion action [106], we select 13 key points, and then concatenated them into a vector in the following order: *left knee, right knee, left foot, right foot, hip, left elbow, right elbow, left shoulder, right shoulder, neck, head, left hand*, and *right hand*, as shown in Fig.3.7. Note that the concatenation order of the key points actually does not matter much in building the final volume. In this way, each frame of an action sequence can be initially represented by a vector of size 13, which is used in the subsequent Joint SSV construction stage.

The UCF-CIL dataset also provides the joint position information for each frame of actions. As shown in Fig.3.7, this dataset provides 11 manually marked limb positions. We build SSMs and Joint SSVs in a similar way to the CMU Mocap dataset.

Many popular vision-based or statistic-based action recognition methods may substantially fail

[2] http://cil.cs.ucf.edu/ucf-cilactiondataset.html

when dealing with these two datasets because of the discrete tracked points. However, our JSSV-pos model is equally capable of discriminating actions in a uniform and flexible manner based on a set of tracked points. We will elaborate the application of our JSSV-pos to those two datasets in Chapter 7.

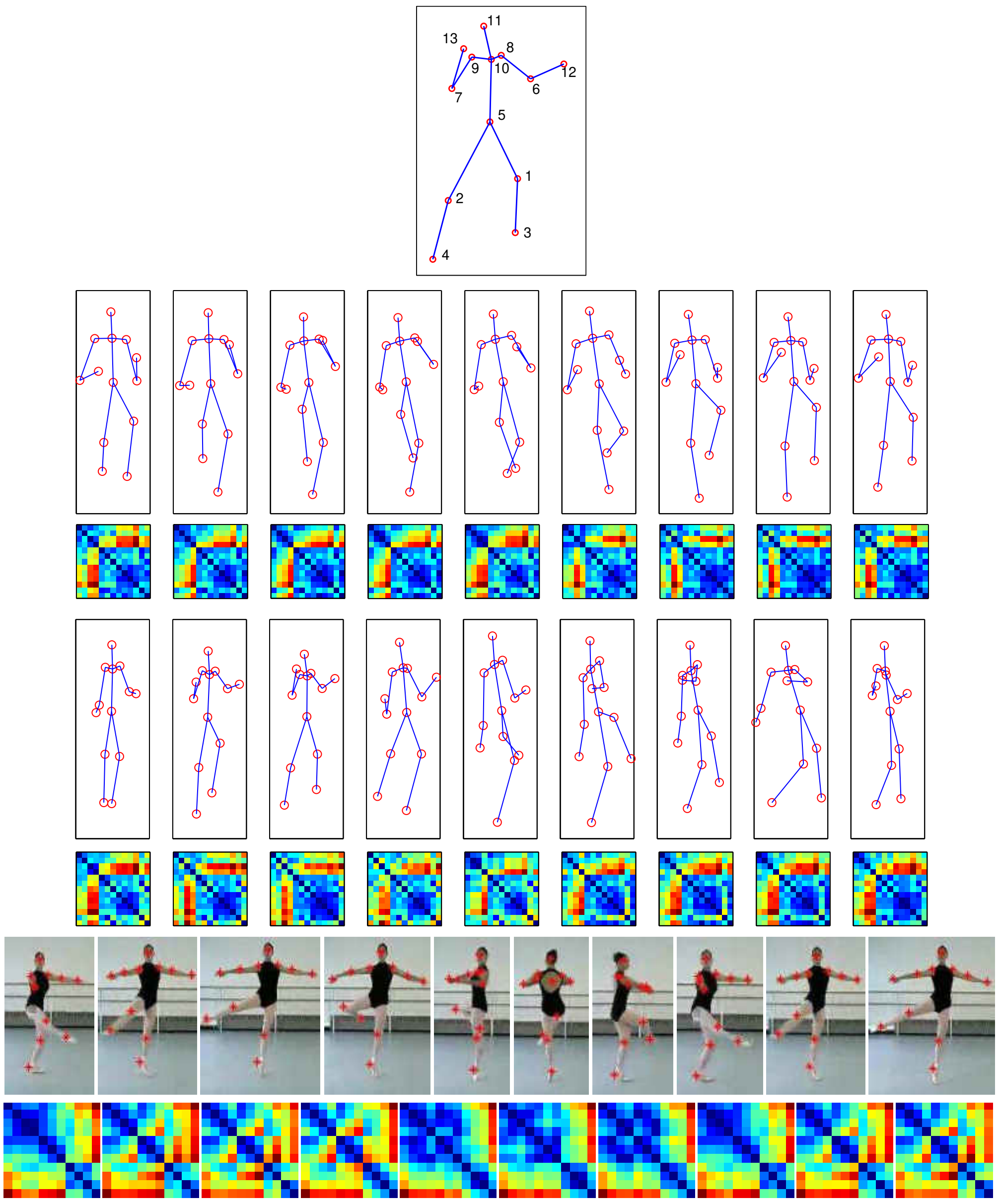


Figure 3.7: Building SSMs for both CMU Mocap dataset and the UCF-CIL dataset. (1st row) The *Run* and *Kick* action in skeleton representation in CMU Mocap dataset. The skeleton in the center with indices shows the tracked point position and their indices for forming the initial vector; (2nd row) The SSMs correspond to the *Run* and *Kick* actions. All SSMs are of the dimension 13×13 and are generated from a vector by concatenating the tracked limb points; (3rd row) The *fouette* action in UCF-CIL dataset represented by the manually marked 11 key limb positions; (4th row) The SSMs corresponding to the *fouette* action. All SSMs are of the dimension 11×11.

CHAPTER 4: MULTILINEAR APPROXIMATION OF JOINT-SSV

In this chapter, we first describle the motivation of the low-rank decomposition for Joint-SSV. Then, we provide some necessary preliminaries of multilinear tensor algebra. Next, we discuss in detail the low-rank approximation algorithm for Joint-SSV. Finally, we exposit the advantages of our algorithm over traditional dimensionality reduction techniques.

Motivation

Numerous methods have been proposed in recent years for mapping from high-dimensional space to low-dimensional space. Here we address the problem of obtaining low-dimensional representation of Joint-SSV from a different perspective. Since the Joint-SSV is a characteristic, unique, and a symmetric 3D structure, we treat the Joint-SSV as a tensor in multilinear algebra, and aim to perform multilinear projection to generate its compact representation. In this section, we first introduce some preliminaries of tensor, then present an algorithm that approximate the Joint-SSV to its low-dimensional alternative, and finally analyze the usefulness of the proposed algorithm in dimensionality reduction.

Tensor Preliminaries

To better assist our elaboration, we briefly provide some necessary preliminaries for tensor approximation. A high-order tensor is denoted as $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. The n-mode product of a tensor $\mathcal{A}$ by a matrix $U \in \mathbb{R}^{J_n \times I_n}$, denoted by $\mathcal{A} \times_n U$, is defined by a tensor with entries [107]

$$(\mathcal{A} \times_n U)_{i_1 \cdots i_{n-1} j_n i_{n+1} \cdots i_N} = \sum_{i_n} a_{i_1 \cdots i_N} u_{j_n i_n}.$$

The Forbenius norm of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is defined as $\|\mathcal{A}\| = \sqrt{\langle \mathcal{A}, \mathcal{A} \rangle}$. The rank of a tensor $\mathcal{A}$ is defined as the minimum number of rank-1 tensors that sum up to $\mathcal{A}$ [108], and a tensor is said to be rank-1 if it can be expressed as an outer product of a number of vectors. The n-rank of $\mathcal{A}$, denoted by $R_n = rank_n(\mathcal{A})$, is the dimension of the vector space spanned by the n-mode vectors. According to the description of HOSVD in [31], any tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ can be expressed as the product

$$\mathcal{A} = \mathcal{B} \times_1 U^{(1)} \times_2 U^{(2)} \times_3 \cdots \times_N U^{(N)}, \tag{4.1}$$

with the properties that $U^{(n)} = (U_1^{(n)} U_2^{(n)} \cdots U_n^{(n)})$ is a unitary $(I_n \times I_n)$ matrix.

Rank-1 Tensor Approximation Algorithm

The problem of tensor rank-$(R_1, R_2, \cdots, R_N)$ approximation for a real Nth-order tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ can be formulated as finding a tensor $\hat{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ satisfying $rank_1(\hat{\mathcal{A}}) = R_1$, $rank_2(\hat{\mathcal{A}}) = R_2$, $\cdots$, $rank_N(\hat{\mathcal{A}}) = R_N$ such that the least-square function $\hat{\mathcal{A}} = arg\min_{\hat{\mathcal{A}}} \|\mathcal{A} - \hat{\mathcal{A}}\|_2$ can be minimized. The desired tensor can be represented as

$$\hat{\mathcal{A}} \approx \mathcal{A} = \mathcal{B} \times_1 U^{(1)} \times_2 U^{(2)} \times_3 \cdots \times_N U^{(N)}, \tag{4.2}$$

where $U^{(1)} \in \mathbb{R}^{I_1 \times R_1}$, $U^{(2)} \in \mathbb{R}^{I_2 \times R_2}$, $\cdots$, $U^{(N)} \in \mathbb{R}^{I_N \times R_N}$ and $\mathcal{B} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$, and $U^{(i)}$ has orthonormal columns for $1 \leq i \leq N$.

Let $\mathcal{A}$ be a Joint-SSV. We consider $\mathcal{A}$ as a 3-order tensor in multilinear algebra, and we attempt to find an approximation of $\mathcal{A}$, denoted by $\hat{\mathcal{A}}$, that satisfies the following objective function

$$\hat{\mathcal{A}} = arg\min_{\hat{\mathcal{A}}} \|\mathcal{A} - \hat{\mathcal{A}}\|_2.$$

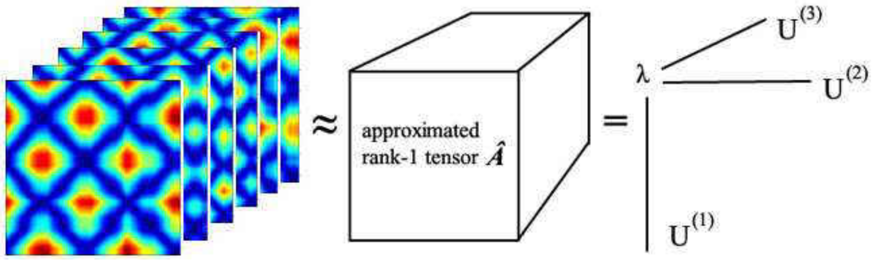


Figure 4.1: Rank-1 approximation $\hat{\mathcal{A}} = \lambda U^{(1)} \circ U^{(2)} \circ U^{(3)}$ for a Joint SSV $\mathcal{A}$

To obtain an optimal rank-1 approximation for the Joint-SSV, we adopt the alternating least squares (ALS) algorithm in our work in an iterative fashion similar to [109, 110]. Although the truncation of the high-order SVD (HOSVD) of a given tensor may lead to a good rank-($R_1, R_2, ..., R_N$) approximation, it is known that this will not necessary generate the best possible (least-squares) approximation under the given n-mode rank constraints [109]. In addition, although the rate of convergence of ALS-based method has not yet been analyzed in the literature, it is proved in [111] that linear convergence of ALS can be achieved in a neighborhood of the optimal solution by the technique of Generalized Rayleigh Quotient (GRQ).

The rank-1 approximation of $\mathcal{A}$ yields the fact that the tensor $\mathcal{B}$ in Eq.(4.1) is a tensor of dimension $1 \times 1 \times \cdots \times 1$, which is equivalent to a numerical scalar. Since $U^{(i)}$ have orthonormal columns, we have $\|U^{(i)}\|_2 = 1$ for $1 \le i \le N$, and Eq.(4.2) can then be transformed into

$$\mathcal{A} \times_1 U^{(1)^T} \times_2 U^{(2)^T} \cdots \times_N U^{(N)^T} = \lambda.$$

This expression leads to an efficient computation compared with the conventional HOSVD-based approximation methods. In our work, we only consider the case when $N = 3$ since we deal with three-dimensional Joint-SSV. The algorithm for rank-1 approximation of 3-rd order tensor Joint-SSV is described in **Algorithm** 1. For clarity of presentation, we denote $U^{(1)}, U^{(2)}$ and $U^{(3)}$ as

α, β, and γ, respectively. The $\overline{\times}_i$ for $i = 1, 2, 3$ denotes the multiplication between a tensor and a vector in mode-i of that tensor, whose result is also a tensor, namely,

$$\mathcal{B} = \mathcal{A}\overline{\times}_i\alpha \Longleftrightarrow (\mathcal{B})_{jk} = \sum_{i=1}^{I} \mathcal{A}_{ijk}\alpha_i.$$

The value of $U^{(n)} \in \mathbb{R}^{I_n \times R_n}$ was initialized with the truncation of the HOSVD in [31]. However, the computation of HOSVD is very expensive in that for the matrix unfoldings $A_{(n)}(1 \leq n \leq N)$, the columns of the column-wise orthogonal matrices span the space of the dominant left singular vectors. In our work, we instead adopt the uniformly distributed random numbers (though columns are not necessarily orthonormal) for initialization. Starting with random initial values for α, β, and γ, the algorithm alternately updates one variable while fixing the other two and iteratively achieves the optimal approximation. The iteration stops when the difference between $\mathcal{A}$ and $\hat{\mathcal{A}}$ arrives at a sufficiently small value (i.e., 10^{-10}), as illustrated in Fig.4.2. We observed that when the iteration number is 5, it is experimentally sufficient to obtain a rank-1 tensor $\hat{\mathcal{A}} = \lambda\alpha \circ \beta \circ \gamma$ that achieves the optimal approximation of the original Joint-SSV $\mathcal{A}$.

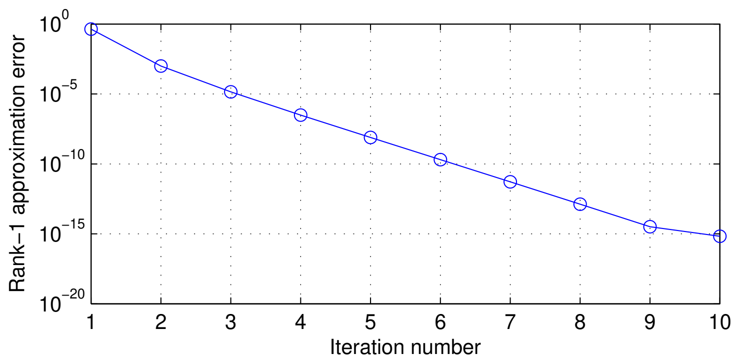


Figure 4.2: Rank-1 tensor approximation error for Joint SSV.

Algorithm 1: Joint-SSV rank-1 approximation

```
input  : A 3-order tensor Joint-SSV $\mathcal{A} \in \mathbb{R}^{I\times J\times K}$, and an iteration termination threshold $\epsilon$
output: Three vectors $\alpha$, $\beta$, and $\gamma$ that minimize $\|\mathcal{A} - \lambda\alpha \circ \beta \circ \gamma\|_2$, where $\alpha \in \mathbb{R}^{I\times 1}$, $\beta \in \mathbb{R}^{J\times 1}$,
        $\gamma \in \mathbb{R}^{K\times 1}$, and $\|\alpha\|_2 = \|\beta\|_2 = \|\gamma\|_2 = 1$
Initialize $\alpha^{(0)}, \beta^{(0)}, and\ \gamma^{(0)}$;
while $\|\mathcal{A} - \lambda^{(t)}\alpha^{(t)} \circ \beta^{(t)} \circ \gamma^{(t)}\|_2 \geq \epsilon$ do
    $\widetilde{\alpha}^{(t+1)} = \mathcal{A}\overline{\times}_2\beta^{(t)}\overline{\times}_3\gamma^{(t)}$;
    $\widetilde{\beta}^{(t+1)} = \mathcal{A}\overline{\times}_1\alpha^{(t)}\overline{\times}_3\gamma^{(t)}$;
    $\widetilde{\gamma}^{(t+1)} = \mathcal{A}\overline{\times}_1\alpha^{(t)}\overline{\times}_2\beta^{(t)}$;
    $\alpha^{(t+1)} = \widetilde{\alpha}^{(t+1)}/\|\widetilde{\alpha}^{(t+1)}\|$;
    $\beta^{(t+1)} = \widetilde{\beta}^{(t+1)}/\|\widetilde{\beta}^{(t+1)}\|$;
    $\gamma^{(t+1)} = \widetilde{\gamma}^{(t+1)}/\|\widetilde{\gamma}^{(t+1)}\|$;
    $\lambda^{(t+1)} = \mathcal{A}\overline{\times}_1\alpha^{(t+1)}\overline{\times}_2\beta^{(t+1)}\overline{\times}_3\gamma^{(t+1)}$;
end
```

Linear methods, such as principal component analysis (PCA) and multidimensional scaling, aim to perform a linear mapping of the data to a lower dimensional space such that the variance in the low-dimensional representation is maximized. Nonlinear techniques, including Isomap, locally linear embedding (LLE), Laplacian eigenmaps, etc, construct a low-dimensional data representation using a cost function that retains local properties of the data. However, those techniques often become inadequate when handling multidimensional data, because they result in very high-dimensional vectors, or break the natural structure and correlation in the original data [112].

Our method can be categorized into multilinear projection that maps from a high-dimensional tensor space to a low-dimensional vector space [112], because the rank-1 tensor approximation enables the mapping from the high-dimensional Joint-SSV $\mathcal{A}$ to low-dimensional final vectors α, β, and γ. Our method differs from these existing dimensionality reduction methods in two aspects:

1. Rather than representing input data as vectors, we find the low-dimensional representation by performing mode-wise iteration until convergence, without going through vectorization,

as shown in **Algorithm 1**

2. We holistically operate on natural tensorial representation of the Joint-SSV, and the structure and correlation in the original volume are then preserved.

In addition, since the Joint-SSV is a symmetric structure, we have $I = J$, which yields $\alpha = \beta$ in **Algorithm 1**. Therefore, only two out of three vectors are responsible for the low-dimensional representation. The dimension of the original Joint-SSV is $I^2 \times K$, while the dimension of the final feature vectors used for classification is $2I + K$. It is the vector α (or β) that captures the spatial dynamics of the Joint-SSV, and the vector γ that encodes the temporal dynamics.

CHAPTER 5: ACTION RECOGNITION USING JOINT-SSV[1]

In this chapter, we discuss how to perform action recognition using our Joint-SSV based representation, with the help of low rank decomposition as dimensionality reduction technique. We verify the feasibility of our framework using two classificaton metrics, namely, the cross-correlation based, and the Gaussian Process based metric. We experiment on five datasets: the Weizmann, the KTH, the UCF sports, the CMU Mocap, and the UCF-CIL dataset.

Motivation

We address action recognition from a different perspective compared to traditional approaches. In contrast to [18, 19, 20, 21] where they reduce an action sequence to a 2D SSM, our formulation constructs a 3-order tensor for each action sequence, which we refer to as the Joint Self-Similarity Volume (Joint-SSV). We propose that a Joint-SSV for a human action is sparse, and hence propose a new optimized rank-1 tensor approximation to represent it.

This approach has multiple advantages: *first*, avoids the need to pre-align videos; *second*, leads to huge dimensionality reduction, and hence a significant saving in memory and computational time, because the reference database is just a collection of rank-1 tensors; and *finally*, we only need one rank-1 tensor per action in our reference database and therefore require no training. We evaluate three different schemes of constructing the Joint-SSV, namely the HOG3D-based, the silhouette-based, and the tracked point based schemes, on five public datasets. The results are very promising, and experimentally outperform most baseline methods.

[1]The content in this chapter was published in the paper: Chuan Sun, Imran Junejo, and Hassan Foroosh, "Action Recognition using Rank-1 Approximation of Joint Self-Similarity Volume", IEEE International Conference on Computer Vision (ICCV, Spain) 2011: 1007-1012.

In [113] the authors provide an excellent survey about the current action recognition methodologies. They state that almost all current methods in the literature can be classified into two categories: single-layered approaches and hierarchical approaches. The former category are methods that can represent and recognize actions directly based on image sequences, which themselves are classified into space-time approaches and sequential approaches. The latter category represents human actions using constructed example-based or state model-based models, and describes action sequences in terms of simpler human activities called subevents. We observe that our framework can be categorized as single-layered approach as our volume generation scheme relies on sequential dependency.

Our framework is shown schematically in Fig.5.1. We construct a SSM for each frame of the video sequence using a certain type of feature vector. We then extract joint SSMs (jSSM) from this sequence of SSMs, leading to a Joint Self-Similarity Volume (Joint-SSV). Joint-SSV is then decomposed into its rank-1 approximation vectors using an optimized iterative tensor approximation algorithm. This yields a set of compact vector descriptors that are discriminative between different actions. To evaluate our method on human action recognition, we used five different public datasets.

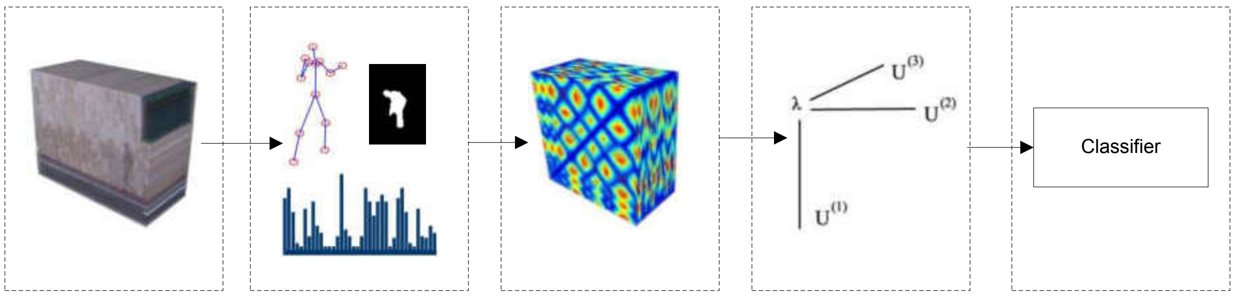


Figure 5.1: The flow of our action recognition framework. *First*, given an input action video, we extract either low-level features like silhouettes in a frame-by-frame manner, or middle-level features like HOG3D from the partitioned video blocks, or some tracked feature points. *Second*, we transform the feature vector in each frame into an SSM. From the sequence of SSMs we then construct a symmetric and unique 3D structure, which we refer to as the Joint Self-Similarity Volume. This volume holds characteristic information about action dynamics. However, in order to exploit it more efficiently, and handle its large dimension, it is decomposed into three compact and discriminative vectors, two of which are identical (due to symmetry). These descriptor vectors characterize the internal dynamics of an action. *Finally*, these vectors are used for measuring distances to a reference set of vectors for final classification

Two Action Classification Methods

To validate the ability of our decomposed vectors for action recognition, we consider two classification methods, namely the cross-correlation metric based classification (X-corr based), and the Gaussian Process based classification (GP-based). The first one adopts the k Nearest Neighborhood (k-NN) approach, which is straightforward and widely-used in many applications: an action is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors. The second one is a latent variable based model with a nonlinear probabilistic mapping from latent positions, and is a stable approach with respect to different parameter settings, and it has a very good ability of generalization [114][115].

The motivation of introducing the GP-based classification method lies in the following facts. The k-NN approach in X-corr based method relies on the assumption that input feature vectors are embedded in linear Euclidean distance space, which is straightforward and easy to compute. While the GP-based classification assume that the feature vectors can be modeled by parameterized Gaussian process, which is more parameterically involved and general. Therefore, in order to demonstrate that our framework can indeed produce discriminative feature vectors, we show results under both classification methods.

Cross-correlation metric based classification

Let Ψ_i and Ψ_j be the two initial vectors. By dimensionality reduction via rank-1 tensor approximation, we obtain their corresponding approximated vector pairs $v^{(i)} = \{U_i^{(1)}, U_i^{(2)}, U_i^{(3)}\}$ and $v^{(j)} = \{U_j^{(1)}, U_j^{(2)}, U_j^{(3)}\}$, respectively. Specifically, for both $v^{(i)}$ and $v^{(j)}$, we know that $U_i^{(1)} = U_i^{(2)}$ and $U_j^{(1)} = U_j^{(2)}$ due to volume symmetry. Note that $U_i^{(1)}$ and $U_j^{(1)}$ are not necessarily of equal length since they are determined by the spatial dimension of their corresponding Joint SSMs, and neither are $U_i^{(3)}$ and $U_j^{(3)}$. To tackle the length inequality, we normalize $U_i^{(1)}$ and $U_j^{(1)}$ (as well as $U_i^{(3)}$ and $U_j^{(3)}$) to zero mean and unit variance, and make them of equal length as described below.

We define the cross-correlation based metric $\mathcal{D}_{xcorr}$ as

$$\mathcal{D}_{xcorr}(\Psi_i, \Psi_j) = \sum_{k=1}^{3} d(U_i^{(k)}, U_j^{(k)}), \tag{5.1}$$

with

$$d = \max \mathcal{C}(U_i^{(k)}, U_j^{(k)}),$$

where $k = 1, 2, 3$. The function $\mathcal{C}$ returns the vector containing the cross-correlation values between the components $U_i^{(k)}$ and $U_j^{(k)}$, namely

$$\mathcal{C}(u,v) = \mathcal{C}_t^{vu} = \frac{C_k^{vu}}{\sqrt{C_0^{vv} \cdot C_0^{uu}}},$$

where C_0^{vv} and C_0^{uu} are the variances of v_i and u_i, and C_t^{vu} is the cross-covariance of u_i and v_i defined by

$$C_t^{vu} = \frac{1}{N} \sum_{i=t+1}^{N} (v_i - \mu_v)(u_{i-t} - \mu_u),$$

and μ_v and μ_u are the mean values of v_i and u_i, respectively.

For various motion sequences, their execution frames are probably of unequal length, leaving those two components of unequal size, then we zero-pad the shorter vector to the length of the longer vector. By this similarity measure, the more similar $U_i^{(k)}$ and $U_j^{(k)}$ are, the larger the value of $\mathcal{D}$ will be.

Gaussian process based classification

For notational convenience, we use the following shorthands. The training data is denoted by $n \times d$ matrix $\mathbf{X} = [\mathbf{x_1}, ..., \mathbf{x_n}]$, the target value is denoted by $n \times 1$ vector $\mathbf{y} = [y_1, ..., y_n]^T$, where $y_i \in \{-1, 1\}$ and the latent function values are summarized by $\mathbf{f} = [f_1, ..., f_n]^T$ with $f_i = f(\mathbf{x_i})$. Let $\mathcal{D} = \{(\mathbf{X_i}, \mathbf{y_i}) | i = 1, ..., n\} = (\mathbf{X}, \mathbf{y})$ be the observed data. Test data can be referred to by asterisk, namely, $\mathbf{f}_*$ is the latent function values for test data $\mathbf{X}_* = [\mathbf{x}_{*,1}, .., \mathbf{x}_{*,m}]$. Covariances between $\mathbf{f}$ and $\mathbf{f}_*$ are written by $[\mathbf{K}_{**}]_{ij} = k(\mathbf{x}_{*,i}, \mathbf{x}_{*,j})$, $[\mathbf{K}_*]_{ij} = k(\mathbf{x}_i, \mathbf{x}_{*,j})$, $[\mathbf{k}_*]_i = k(\mathbf{x}_i, (x)_*)$, and $k_{**} = k(x_*, x_*)$, where $[\cdots]_{ij}$ denotes the entry ij of the matrix.

A GP [116] is a stochastic process fully specified by a *mean function* $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ and a positive definite *covariance function* $k(\mathbf{x}, \mathbf{x}') = \mathbb{V}[f(\mathbf{x}), f(\mathbf{x}')]$. We put a Gaussian process prior on the latent function $\mathbf{f}$ such that any number of data evaluated from this function has a multivariate Gaussian density, that is, $\mathbb{P}(\mathbf{f}|\mathbf{X}, \theta) = \mathcal{N}(\mathbf{f}|\mathbf{m}_0, \mathbf{K})$.

Given data set $\mathcal{D}$, we wish to find the correct class label for a new data x_* by computing the class probability $\mathbb{P}(y_*|x_*, \mathcal{D})$. The marginalization over the training set latent variables for making predictions is

$$\mathbb{P}(\mathbf{f}_*|\mathbf{X}_*, \mathbf{y}, \mathbf{X}, \theta) = \int \mathbb{P}(\mathbf{f}_*, \mathbf{f}|\mathbf{X}_*, \mathbf{y}, \mathbf{X}, \theta) d\mathbf{f} = \int \mathbb{P}(\mathbf{f}_*|\mathbf{f}, \mathbf{X}_*, \mathbf{X}, \theta)\mathbb{P}(\mathbf{f}|\mathbf{y}, \mathbf{X}, \theta) d\mathbf{f},$$

where

$$\mathbb{P}(\mathbf{f}_*|\mathbf{f}, \mathbf{X}_*, \mathbf{X}, \theta) = \mathcal{N}(\mathbf{f}_*|\mathbf{K}_*^T\mathbf{K}^{-1}\mathbf{f}, \mathbf{K}_{**} - \mathbf{K}_*^T\mathbf{K}^{-1}\mathbf{K}_*),$$

and

$$\mathbb{P}(\mathbf{f}|\mathbf{y}, \mathbf{X}, \theta) \propto \mathbb{P}(\mathbf{y}|\mathbf{f}, \mathbf{X}, \theta)\mathbb{P}(\mathbf{f}|\mathbf{X}, \theta) = \{\prod_{i=1}^{n} \mathbb{P}(y_i|f_i, \theta)\}\mathbb{P}(\mathbf{f}|\mathbf{X}, \theta).$$

Finally, the predictive class membership probability $p_* := \mathbb{P}(y_* = 1|\mathbf{x}_*, \mathcal{D}, \theta)$ is obtained by integrating out the test set latent variables f_*

$$\mathbb{P}(y_*|\mathbf{x}_*, \mathbf{y}, \mathbf{X}, \theta) = \int \mathbb{P}(y_*|f_*)\mathbb{P}(f_*|\mathbf{x}_*, \mathbf{y}, \mathbf{X}, \theta) df_*.$$

In our experiments, we adopted the GPML package[2] for Gaussian Process learning and inference. There are mainly four key factors in GP learning: mean function (MF), covariance function (CF), inference function (IF), and likelihood function (LF). For different applications, each factor has various optional functions to choose from. For example, in GPML, there are 9 MF, 21 CF, 6 IF, and 6 LF. For those 6804 function combinations, we filter out those combinations that are inappropriate for classification, and ignore those invalid combinations. For example, exact inference in IF and Gaussian likelihood in LF should not be combined for classification in our Gaussian Process learning, yet they can be combined for regression. This elimination process results in a total of 754

[2]http://www.gaussianprocess.org

candidate combinations. We will describe how we perform experiments under this classification method in the next section.

Experiments

We evaluated our method on 5 well-known public datasets: Weizmann, KTH, UCF sports, CMU Mocap dataset, and the UCF-CIL dataset. Our goal is to evaluate the feasibility of our framework on various datasets with different Joint SSV schemes. For all the results reported in this section, we performed the recognition using the leave-one-out cross validation based on the two classification methods.

We perform Gaussian Process (GP) based classification using the following steps. First, for the 754 function combinations in the GP parameter settings, we perform our final action classification by randomly selecting 100 combinations. This is motivated by the fact that, in the absence of any prior information, all combinations are equally likely to provide the best classification rate. In addition, those 100 random combinations can largely cover almost all potential GP parameter combinations. Second, since each dataset has its own corresponding JSSV type, we perform action classification under these random 100 combinations for each type of JSSV, and report our best rates for that type. Lastly, we make comparisons with both our X-corr based method and other methods reported in the literature.

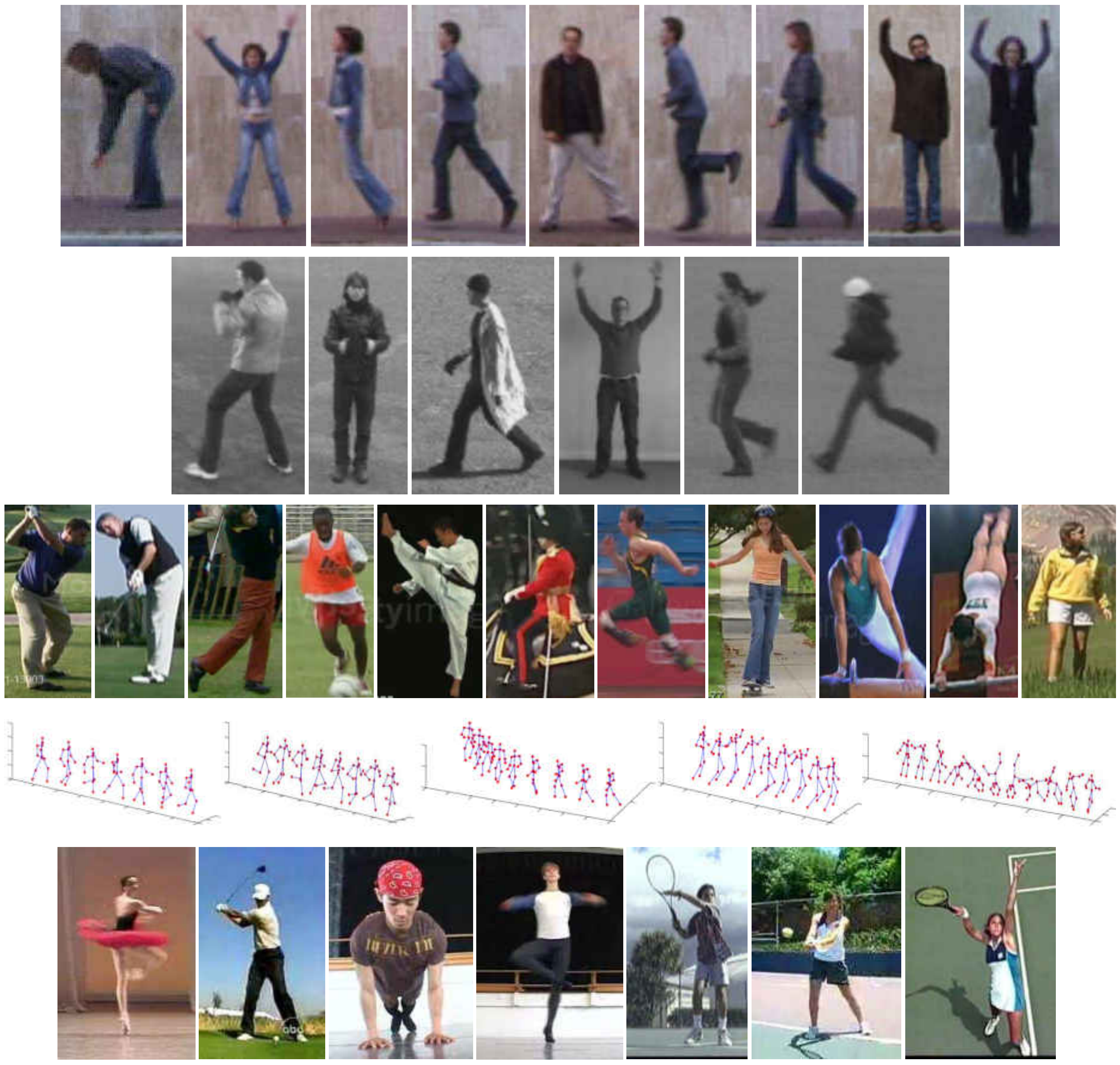


Figure 5.2: Screenshots for different action classes of 5 public datasets. (1st row) The Weizmann dataset; (2nd row) The KTH dataset; (3rd row) The UCF sports dataset; (4th row) The CMU Mocap dataset; (Last row) The UCF-CIL dataset. Note that for the CMU Mocap dataset, the actions shown here denote *kick, run, walk-turn, jump, and cartwheels*, respectively

Weizmann dataset

The Weizmann dataset [3] consists of videos of 10 different actions performed by 9 actors. Each video clip contains one subject performing a single action. The 10 different action categories are: walking, running, jumping, gallop sideways, bending, one-hand-waving, two-hands-waving, jumping in place, jumping jack, and skipping. Each of the clips lasts about 2 seconds at 25Hz with image frame size of 180×144.

We evaluated two schemes, namely the JSSV-silh and the JSSV-hog3d, separately, using the two classification methods, respectively. The JSSV-pos scheme is unavailable for this dataset since there is no body joint point information available. For the JSSV-silh scheme, we used the provided well-extracted silhouettes in the dataset to build input vectors for the whole framework; and for the JSSV-hog3d scheme, we extract the ROI using the silhouettes by fitting a bounding box around each of them. To be consistent, all ROIs in our experiments are scaled and concatenated to form a $128 \times 64 \times t$ volume, where t is the frame number in sequence. We evaluated various block size setups (Table 5.1).

For the X-corr based classification, we were able to achieve a recognition rate of 100% for JSSV-silh. We also observed that when $\kappa = 4$ and $\tau = 3$ (i.e. block size: $16 \times 16 \times 8$), the JSSV-hog3d scheme yields the best recognition rate of 100%, as shown in Table 5.2. For the GP-based classification method, we use the same generated feature vectors as in the X-corr based method, and scanned through the randomly chosen 100 combinations. For the JSSV-silh and JSSV-hog3d scheme, we achieved our best classification rates of 100% and 97.3%, respectively.

[3]http://www.wisdom.weizmann.ac.il/ṽision/SpaceTimeActions.html

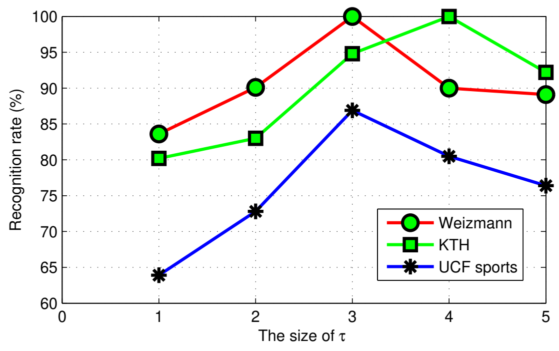


Figure 5.3: Recognition rate under different HOG3D block depths 2^τ for three datasets using JSSV-hog3d for the X-corr based classification method

KTH dataset

The KTH dataset [4] consists of 6 actions performed by 25 actors in four different scenarios. We followed the evaluation procedure in [104] but used slightly different settings for block size. We extracted the ROIs using the bounding boxes provided by [71]. Since both the JSSV-silh and JSSV-pos schemes are unavailable for this dataset, we evaluated only the JSSV-hog3d scheme on this dataset under various block size configurations, as shown in Fig.5.3.

For the X-corr based classification method, when τ is small, the block depth is small, making the final decomposed vectors non-discriminating for classification. But as τ grows, the recognition rate grows accordingly. This also agrees with our intuition that larger blocks contain more cells, and capture more stable gradient information compared with the smaller ones. But as the block size becomes larger, more redundant information is introduced, leading a reduced recognition rate beyond an optimal size.

Especially, our best recognition rate of 100% is achieved when $\kappa = 4$ and $\tau = 4$, This outperform-

[4]http://www.nada.kth.se/cvap/actions/

s both the result in [104] (92.4%), which has similar experimental configuration to us, and the state-of-the-art in [117] (94.5%), while using the GP-based classification method yields the best classification rate of 96.5%.

Table 5.1: Recognition rate comparison for 3 action datasets under the JSSV-hog3d scheme using both X-corr based and GP-based classification methods. For demonstration purposes, we also list our recognition rates under only four random GP parameter settings. The κ and τ are parameters controlling the block size $2^\kappa \times 2^\kappa \times 2^\tau$

		Weizmann					KTH					UCF sports				
		$\tau = 1$	$\tau = 2$	$\tau = 3$	$\tau = 4$	$\tau = 5$	$\tau = 1$	$\tau = 2$	$\tau = 3$	$\tau = 4$	$\tau = 5$	$\tau = 1$	$\tau = 2$	$\tau = 3$	$\tau = 4$	$\tau = 5$
	κ=2	70.5	78.4	82	80.1	71.1	70.5	73.9	64.8	76.3	60.2	68	64.7	69.8	72	48.6
X-corr	κ=3	75	76.5	86.2	87.3	85.4	75.5	70.4	84.8	88	84	63.9	72.8	**86.9**	80.5	76.4
	κ=4	83.6	90.1	**100**	90	89.1	80.2	83	94.8	**100**	92.2	68.2	81.9	64.8	77.5	76.1
	κ=2	76.2	82.4	69.6	70.2	70.9	83.9	90.6	92.4	95.6	67	69.8	84.3	68	81.1	81.8
GP-para1	κ=3	86.6	81.3	72.5	81.3	71.2	92	84.7	95.7	82.3	80.9	82.7	69.4	74.7	72.4	81.6
	κ=4	68.5	91.6	83.2	93.9	87.2	90.2	73.6	81.4	93.1	83.6	75.3	83.4	71	72.4	70.4
	κ=2	67.4	70.5	92	81	91.5	72.9	71.2	72.4	68.2	85.4	73.1	84.9	80.2	79.7	73.3
GP-para2	κ=3	73	83	85.2	67.9	84.8	75.1	82.6	87.1	81.4	82.5	84.6	80.9	76.6	79.2	77.4
	κ=4	77.5	68.4	81.1	72.5	70.5	79.9	70.5	81.2	91.7	92.3	72.2	74.8	72.9	73.9	74.8
	κ=2	88.1	88.3	68.8	91.9	94	75.5	76.6	80.5	85.7	67.7	76.2	68	86.9	87.8	77.8
GP-para3	κ=3	95.5	91.9	89.7	81.8	72.1	91.4	83.2	91.7	77	79.9	77.8	74.4	86.8	75.1	69.4
	κ=4	78.5	70.8	67.8	94.2	75.7	68.5	72.1	86.2	76.5	93	84.2	75.6	72.3	75.9	69.1
	κ=2	92.1	69.5	77.6	77.7	86.8	85.6	81	71.4	89.6	69.9	76.7	77.5	73.5	73.1	81.9
GP-para4	κ=3	84.3	89.8	77.6	72.9	69.5	75.5	73.8	82.3	69.6	78.7	68.5	80.4	80.8	79.2	69.3
	κ=4	89.3	72.9	78.2	83	73.6	70	70.2	89.7	75.4	84.5	71.6	72.7	77.5	69.9	78.1

UCF sports dataset

The UCF sports dataset contains 11 actions: *golf swing (back, front, side), kicking (front, side), riding horse, run, skate boarding, swing bench, swing (side), and walk.* Similar to the KTH dataset, this dataset does not provide silhouette or tracked point information. And due to dynamic backgrounds and potential moving viewpoints, it seems impractical to consistently extract reliable silhouettes from it, meaning we cannot apply our JSSV-silh and JSSV-pos schemes. Fortunately, this dataset provides the well-extracted bounding boxes for extracting the ROIs from each action sequences, which enable us to apply our JSSV-hog3d scheme.

We choose 10 example videos for each action class. For consistency in our experiments, since in the UCF sports dataset there are different number of example videos for different actions, for those actions with less than 10 examples such as "golf-swing-back", "golf-swing-side", and "golf-swing-front", we increased the number of available video examples by adding a horizontally flipped version of selected existing videos. This resulted in 110 example videos in total.

For X-corr based classification method, as shown in Table 5.1, our best recognition rate of 86.9% was achieved when $\kappa = 3$ and $\tau = 3$, which is comparable with the state-of-the-art in [118] (87.27%). For the GP-based classification method, we were able to achieve the best rate of 88.5%, which also outperforms our X-corr based method.

CMU mocap dataset

We used motion capture data from CMU dataset to evaluate the performance of our framework. A total of 164 sequences corresponding to 12 actions (*walkturn, golf, fjump, flystroke, jjack, jump, carwheels, drink, kick, walk, bend, run*) were used. This dataset is different from all previous datasets in that the only information stored are tracked body joints location rather than frame pixels. Therefore, only the JSSV-pos scheme is applicable.

For the X-corr based classification method, our overall recognition rate was 93.1%, which outperforms the method in [119] (92.0%), but [119] adopted a different evaluation setting using only 5 actions, each performed by 3 actors. Note that our approach also outperforms the method in [21] (90.5%), which also explored several SSM-based approaches. For the GP-based classification method, we were able to achieve the highest rate of 94.8%, which even outperforms the X-corr based classification method.

UCF-CIL dataset

The UCF-CIL dataset consists of 56 sequences of 8 actions (4 ballet fouette, 12 ballet spin, 6 push-up, 8 golf swing, 4 one-handed tennis backhand stroke, 8 two-handed tennis backhand stroke, 4 tennis forehand stroke, and 10 tennis serve). Each action is performed by different subjects. Compared with the CMU Mocap dataset, all the joint positions of the human body in each frame are manually annotated frame-by-frame, rather than being captured by motion capture system.

This dataset provides 11 joint positions for every frame of the action sequence. Similar to the construction procedure of the CMU Mocap dataset, we built the JSSV-pos volume using the 11 annotated points in the following order: head, right shoulder, right elbow, right hand, left shoulder, left elbow, left hand, right knee, right foot, left knee, and left foot. Similar to the CMU Mocap dataset, only the JSSV-pos scheme is valid for this dataset. For the X-corr based and GP-based classification method, our best recognition rates were 87.1% and 90.7%, respectively.

Table 5.2: Recognition rate comparison for 5 datasets between our 3 different schemes (JSSV-silh / JSSV-hog3d / JSSV-pos) and other methods in the literature. The symbol '-' means the rate is unavaliable under corresponding schemes.

Methods	Weizmann	KTH	UCF sports	CMU Mocap	UCF-CIL
X-corr based	**100.0/100.0/-**	-/**100.0**/-	-/86.9/-	-/-/93.1	-/-/87.1
GP-based	**100.0**/97.3/-	-/96.5/-	-/**88.5**/-	-/-/**94.8**	-/-/90.7
Other methods	Schindler [120] **100.0**	Gilbert [117] 94.5	Kovashka [118] 87.27	Shen [119] 92.0	Shen [121] **95.83**
	Zhang [122] 97.8	Lin [71] 93.4	Klaser [123] 86.7	J.Imran [21] 90.5	
	J.Imran [21] 95.3	Schindler [120] 92.7	Wang [124] 85.6		
	Niebles [72] 90.0	Weinland [104] 92.4			
	Liu [125] 89.3	Liu [125] 82.8			

CHAPTER 6: ACTION RECOGNITION FOR INCOMPLETE VIDEOS [1]

Background and Motivation

Action recognition in video data has been applied in many areas such as video retrieval, annotation, surveillance, and human computer interaction, etc. Despite its extensive study, it still remains one of the most challenging problems in the literature. For action recognition, many approaches have been proposed and high recognition rates have been reported for various datasets in the literature. However, we noticed that many existing methods lack some level of realism, in the sense that they include only video data that are complete, i.e. have no missing parts, occlusions, or noise, and are densely sampled. In a world inundated with videos (e.g. Youtube), vast majority of data may be one way or another subject to some level of incompleteness (as defined shortly). The purpose of this paper is thus to determine to what extent incomplete data are useful and carry helpful information for action recognition.

For clarity, we use the word "complete" throughout this paper to refer to raw, uncontaminated, occlusion-free, or densely sampled video, and "incomplete" to refer to video that is contaminated, corrupted, occluded, sparsely sampled, or simply missing data (pixels or blocks). We observe that most existing recognition methods require action data to be complete and they would fail to achieve the same recognition performance if the video is incomplete. For instance, methods based on the spatiotemporal descriptors, such as the space time interest point (STIP) descriptor [118][126], the HOG3D descriptor [127], and the bag-of-words based approach [128], etc., would fail to detect sufficient features under high degree of missing data (e.g. when using a compressive sensing device). Some approaches have also been proposed to handle occlusions in *tracking* or

[1] The content in this chapter was accepted in the paper: Chuan Sun, Hassan Foroosh, "Should We Discard Sparse or Incomplete Videos?", IEEE International Conference on Image Processing 2014.

detection [129][130][131], but they may still fail under severe occlusions. Also, if a large portion of the video element is sparsely sampled, the performance of bag-of-words based approach [128] will inevitably be hampered due to insufficient video words.

Researchers prefer complete and clean video data rather than incomplete or corrupted data due to mere convenience. As a result, it is conceivable to assume that potentially useful information is lost when we discard incomplete data in a dataset. This leads to two important questions our study aims to answer: (1) Is it always wise to simply discard incomplete video just because they are incomplete? (2) Is it possible to make full use of incomplete data in action classification?

Many factors can cause a video to be incomplete. Occlusions and block loss (e.g. in ATM video) are some common examples. For example, part-based detectors have been explored in handling the partial occlusion in images [132]. They alleviate the occlusion side effect using the unoccluded parts to determine the human pose. In [133] the authors integrates the part-based detectors to the sliding-window detectors, and propose a human detection method to handle image partial occlusion. However, those methods have to identify the occluded regions inside the sliding window detectors when partial occlusion appears, and have to determine whether an occlusion has occurred, and if so, where it is located. Another recent interesting work is in [131]. The authors propose a hybrid approach using the partitioning of a dense 3D HOG representation in a hierarchical classifier to handle both viewpoint changes and occlusions, and shows some promising recognition results.

It is worth noting that most of these contributions have focused on handling occlusions in tracking or detection problems, and less attention has been paid to the effect of incompleteness of data on human action classification. Based on the discussion in Section 3, we focus on binary classification of sparsely sampled videos. We list our contributions as follows: (1) We cast the action classification problem for a mixture of complete/incomplete data as a semi-supervised learning problem of labeled/unlabeled data, for which we propose a graph-based solution. The classification requires neither the localization of the occluded regions, nor the configuration of part-based detectors. (2) We introduce innovative steps to convert the input mixed data into a uniform representation that can be used by the above graph-based semi-supervised learning method. The steps are (a) tensor completion using Canonical Polyadic decomposition, (b) low-rank tensor representation of tensors to generate vectorized features. (3) We experimentally show that it is possible and a good idea to mix incomplete and complete data to boost classification performance. Therefore, a large number of videos, often discarded as useless, do indeed carry significant information. What has been hindering this from happening in the past is merely lack of good unifying and homogeneous representation of such data, which we introduce in this paper.

To study these two questions, we propose a framework including four steps. The first is the mixing of incomplete and complete videos. To generate a large amount of incomplete videos, we sparsely sample the complete videos under various sparsity settings. Secondly, we regard an incomplete video as a three-way incomplete tensor, and recover incomplete videos by a solid tensor completion algorithm. Thirdly, we build lower dimensional representation using a rank one tensor decomposition algorithm. Finally, we apply graph based semi-supervised learning for action classification. Our experiments show that the proposed framework is very effective, and to our best knowledge, is the first attempt to address this challenging problem in the literature.

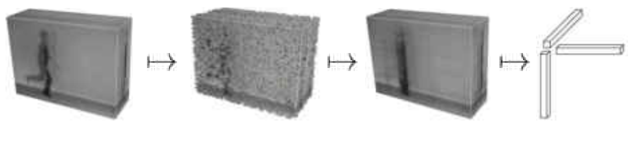


Figure 6.1: The process of generating the sparse representation. We add sparse occlusions to a complete video. This result in an incomplete occluded tensor of unknown rank. Its complete version under specified rank is then recovered. We then perform a generalized tensor rank-1 decomposition algorithm to obtain its compact representation

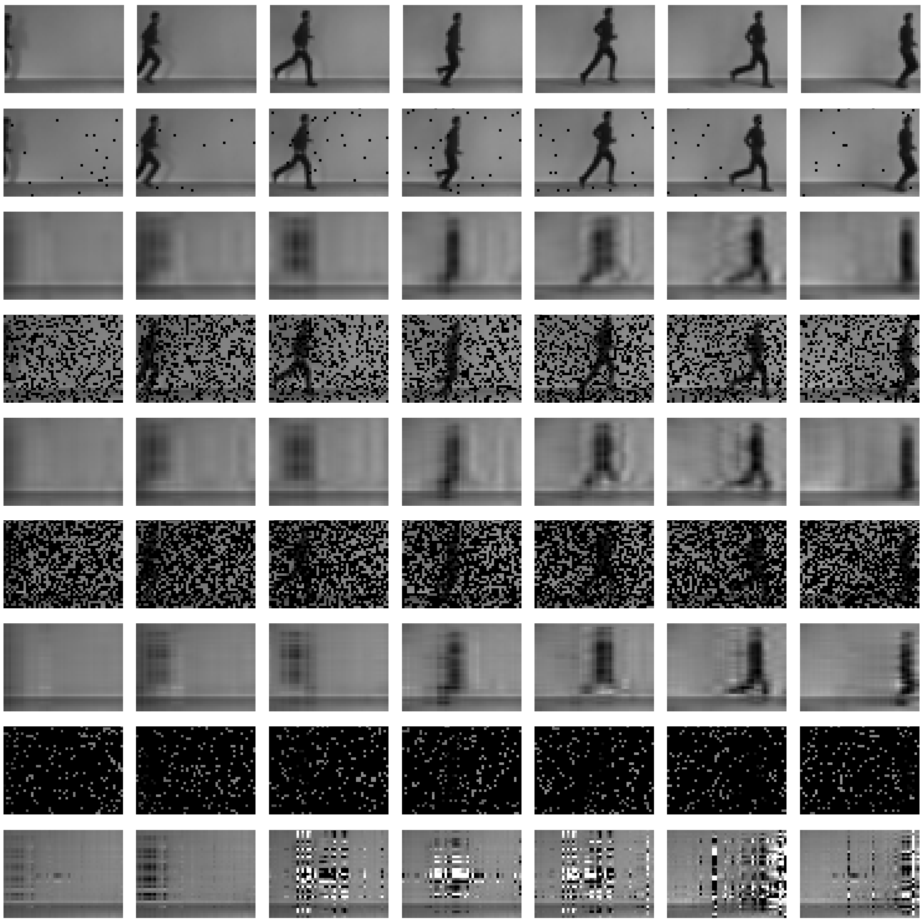


Figure 6.2: Occluded frames under different occlusion settings for the KTH *running* action. The $1st$ row shows the original complete frames; The $2nd, 4th, 6th, 8th$ rows shows $1\%, 31\%, 61\%$, and 91% pixels are occluded; The $3rd, 5th, 7th, 9th$ row shows corresponding recovered frames. We fix the rank value R = 16.

Sparse video recovery

Incompleteness

We first summarize 3 types of incomplete video. (1) The *first* involves the sparseness and randomness in compressive sensing [134, 135, 136, 137, 138, 139, 140]. In sparsely sampled video, a number of pixels in random locations would be missing. As compressive sensing is becoming popular, it is likely that more and more sparsely sampled videos will be generated. (2) The *second* is caused by partial occlusion lasting throughout all frames. The work in [131] explored this type by artificially placing occluders on some predefined parts in the human body. This method however needs to calculate the overlapping amount with the occluding object, and complex local partitioning is required to ensure robustness. (3) The *third* is the partial occlusion with short duration. This behaves like spatiotemporal "holes" and is directly related to the video completion problem, which is the process of filling in missing pixels or replacing undesirable pixels [141]. To sum up, the first is closely related to compressive sensing. The last two are special cases of the first. Since the second and the third type are well studied in inpainting and video completion, we focus on the first in this paper.

Recovery procedure

Regarding the sparse video $\mathcal{V}$ under sparse sampling mask $\mathcal{W}$ as an incomplete tensor $\mathcal{X}$, the sparse video recovery is equivalent to sparse tensor completion. Let $\mathcal{V} \in \mathbb{R}^{I\times J\times T}$ be an incomplete video in gray scale, and T is the frame number. Let $\mathcal{W} \in \mathbb{R}^{I\times J\times T}$ be a sparse weight tensor represent $\mathcal{V}$'s missing entries. $\mathcal{W}$ acts as a binary mask filtering out certain portion of $\mathcal{V}$. To specify video sparsity, we follow the approach in [142]. Let $\mathcal{X}$ be a three-way tensor of size $I \times J \times K$ with rank R. For the missing entries in the sparse tensor, tensor recovery can be defined as minimizing the error function:

$$f_w(A,B,C) = \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}\{w_{ijk}(x_{ijk} - \sum_{r=1}^{R} a_{ir}b_{jr}c_{kr})\}^2,$$

where $x_{ijk} = \sum_{r=1}^{R} a_{ir}b_{jr}c_{kr}$ for all $i \in [1,I], j \in [1,J]$, and $k \in [1,K]$. The factor matrices A, B, and C are of size $I \times R$, $J \times R$, and $K \times R$, respectively. It leads to a weighted least square Canonical Polyadic (CP) decomposition problem [142], which is a generalization of the SVD to tensors [142] in multilinear algebra. $\mathcal{W}$ is a nonnegative weight sparse tensor defined as $w_{ijk} = 1$

if x_{ijk} is known, and $w_{ijk} = 0$ if x_{ijk} is missing, and is of the same dimension as $\mathcal{X}$. The objective function can further be generalized as:

$$f_{\mathcal{W}}(A^{(1)}, A^{(2)}, A^{(3)}) = \|\mathcal{W} * (\mathcal{X} - \langle A^{(1)}, A^{(2)}, A^{(3)} \rangle)\|^2,$$

where the $\langle A^{(1)}, A^{(2)}, A^{(3)} \rangle$ defines a $I_1 \times I_2 \times I_3$ tensor whose elements are given by:

$$(\langle A^{(1)}, A^{(2)}, A^{(3)} \rangle)_{i_1 i_2 i_3} = \sum_{r=1}^{R} a^{(1)}_{i_1 r} a^{(2)}_{i_2 r} a^{(3)}_{i_3 r}.$$

Let $\mathcal{Y} = \mathcal{W} * \mathcal{X}$ and $\mathcal{Z} = \mathcal{W} * \langle A^{(1)}, A^{(2)}, A^{(3)} \rangle$, the gradient of the objective function can be computed by the partial derivatives of $f_{\mathcal{W}}$ with respect to each element of the factor matrices, namely

$$\frac{\partial f_{\mathcal{W}}}{\partial A^{(n)}} = 2(\mathcal{Z}_{(n)} - \mathcal{Y}_{(n)}) A^{(-n)},$$

where $A^{(-n)} = A^{(3)} \odot A^{(2)} \odot A^{(1)}$ for $n = 1, 2, 3$ and $\odot$ is the Khatri-Rao product. The optimal factor matrices can then be computed iteratively using the gradient descent method.

To recover an incomplete video, we need to consider three critical parameters. The *first* one is the sparsity of $\mathcal{W}$. It can be denoted by the percentage of missing entries. The smaller the percentage, the less sparse the $\mathcal{W}$ will be. Meanwhile, the larger the percentage, the more costly to recover the incomplete video. The *second* is the randomness of $\mathcal{W}$. The sparse element locations are generated from a standard normal distribution with small perturbation noises. The *third* is the rank of $\mathcal{V}$, which is defined as the smallest number of rank-1 tensors that generate $\mathcal{V}$ as their sum [143].

Fig.6.2 illustrates the frames from the recovered actions in KTH dataset under a fix rank $R = 16$. We set different occlusion percentages ranging from 1% to 90%. We observe that the frames can be nicely recovered if the occlusion percentage is under 80%. Severe occlusion occurs above this percentage, making the recovered frames full of noise, or even meaningless (see the $9th$ row in

Fig.6.2). This is because the original video structure is heavily corrupted, and the recovery algorithm fails to factorize its factor matrices. Note that, although under 1% occlusion, the recovered frames in the $3rd$ row look as blurred as the $5th$ row, this stems from the nature of the recovery procedure, which is not intended to generate element-wise (pixel-wise) recovery.

Compact representations

After incomplete video recovery, we are facing two issues: (1) The three-way tensor is highly costly to be directly used in classification due to high dimensionality. (2) It is hard to extract dominant features due to heavy contamination.

To overcome both issues, we extend the rank-1 tensor decomposition method proposed in [144] to generate the low-dimensional compact representations. The outputs are one scalar value and three compact one-dimensional vectors, which discriminately represent the decomposed video. In the context of [144], the tensors degenerate to symmetric tensor due to frame symmetry. In our scenario, however, we handle instead the general asymmetric tensors due the randomness and sparseness of the weight tensor $\mathcal{W}$.

Starting with random initial values for α, β, and γ, the algorithm alternately updates one variable while fixing the other two and iteratively achieves the optimal decomposition. By Algorithm 3, we transform any 3D tensor $\mathcal{A}$ into three compact vectors α, β, and γ, such that $\|\mathcal{A} - \lambda\alpha \circ \beta \circ \gamma\|_2$ is less than a predefined sufficiently small threshold value ϵ, and that $\|\alpha\|_2 = \|\beta\|_2 = \|\gamma\|_2 = 1$. We linearly concatenate those three vectors to form a single feature vector f such that $f \in \mathbb{R}^{(I+K+J)\times 1}$. Therefore, the dimensionality of the feature vector is now reduced from $I \times K \times J$ to $I + K + J$. The details can be referred in [144].

Table 6.1: The classification error rates of our two benchmarks. The first and second table illustrate the error rates (%) for benchmarks stemming from KTH and UCF Sports, respectively. We extensively scrutinize 280 possible parameter combinations for our classification-with-incomplete-data scenario. Note that “r8” means the evaluation is under rank “R=8”

P	P=1%				P=20%				P=30%				P=40%				P=60%				P=80%				P=90%			
Q	r4	r8	r12	r16	r4	r8	r12	r16	r4	r8	r12	r16	r4	r8	r12	r16	r4	r8	r12	r16	r4	r8	r12	r16	r4	r8	r12	r16
Q=90%	14.7	17.6	13.3	10.3	20.2	15.2	12.6	11.2	11.1	15.5	11.1	4.6	22.3	15.6	13.7	10.4	26.5	18.6	22.2	10.3	47.7	39.8	36.6	35.7	49.9	45.2	48.5	46.8
Q=70%	8.4	9.8	7.1	8.7	13.4	12.0	7.3	10.7	14.0	12.5	8.4	5.7	8.9	12.7	10.3	8.5	19.9	15.6	18.3	17.8	31.3	15.4	20.1	17.7	41.1	48.2	49.4	47.1
Q=50%	7.9	9.3	5.0	6.7	14.3	13.3	8.5	17.4	7.8	5.8	9.4	4.3	9.7	12.5	6.7	7.2	9.6	9.7	5.9	6.2	22.6	18.1	13.2	14.3	47.5	41.7	45.9	48.6
Q=30%	5.3	5.2	9.4	6.1	7.8	9.2	5.2	13.1	3.1	6.2	9.5	6.9	8.6	8.5	16.7	14.0	9.7	6.5	6.5	3.8	17.6	14.1	15.0	9.4	49.6	43.8	44.5	43.9
Q=10%	10.2	12.7	9.4	5.1	8.1	10.4	6.7	5.8	8.3	6.5	9.1	3.2	11.5	20.4	6.6	14.8	9.2	7.3	10.1	2.0	9.6	14.7	11.2	8.0	48.4	40.1	44.2	39.2

P	P=1%				P=20%				P=30%				P=40%				P=60%				P=80%				P=90%			
Q	r4	r8	r12	r16	r4	r8	r12	r16	r4	r8	r12	r16	r4	r8	r12	r16	r4	r8	r12	r16	r4	r8	r12	r16	r4	r8	r12	r16
Q=90%	16.8	14.6	13.2	9.3	15.2	15.3	14.8	11.3	18.0	15.4	11.0	13.6	22.2	19.7	17.8	17.4	26.5	19.7	22.1	17.3	35.6	33.2	32.6	30.8	49.6	42.2	47.6	48.8
Q=70%	15.5	16.9	12.0	14.8	13.6	12.9	9.4	8.8	16.0	19.5	20.5	17.9	20.9	17.8	15.3	13.7	24.9	23.6	20.5	18.4	34.3	36.3	32.2	30.9	44.1	39.3	47.7	49.6
Q=50%	17.8	14.8	12.9	13.2	14.1	14.6	13.5	13.3	17.8	18.9	19.8	23.2	24.8	28.8	20.9	18.2	25.8	22.8	18.9	20.2	42.8	38.4	33.2	30.0	48.6	46.6	44.8	38.6
Q=30%	15.5	15.2	16.0	17.9	17.5	18.2	13.2	9.0	18.2	17.5	19.7	12.9	18.9	17.4	25.4	40.0	42.7	36.5	36.4	37.7	47.7	40.4	35.1	36.4	49.6	43.9	42.7	48.9
Q=10%	14.0	13.2	20.1	15.6	18.0	20.1	8.8	15.7	20.1	15.2	14.1	16.4	20.4	17.3	15.8	17.9	40.1	47.2	30.0	35.9	38.3	44.4	45.1	43.0	49.2	46.2	49.1	48.0

Classification

We make three assumptions before classification: (1) There are many sparse videos available. (2) It might be expensive or impractical to label the sparse videos due to heavy occlusions. (3) Complete videos are labeled, whereas sparse ones are unlabeled. The reasons for those three are straightforward. On the one hand, the essence of our framework lies in the conjecture that using unlabeled incomplete videos would be helpful for classification. We believe that it might be hard to directly use sparse videos in classification if the occluders are dense, but their recovered versions could make this possible. On the other hand, the semi-supervised learning (SSL) uses readily available unlabeled data to improve classification rate given labeled data is scarce or limited. If sparse videos dominate the dataset and they happen to be unlabeled, the SSL will be our best choice.

The first key component in our SSL is to build a sparse weighted graph out of video data. Graph based models are ideally suited to represent data based on pairwise information such as similarities, distances, and relations [145]. Practically, instead of a fully connected dense graph, we prefer sparse graphs such as the k-NN graph and the ϵ-NN graph for video graph construction. We use the Gaussian kernel similarity function to measure the similarity between graph nodes, i.e., $s(x,y) = exp(-\|x-y\|^2/(2\sigma^2))$. Fig.6.4 show the constructed k-NN graphs for KTH dataset. Since many graph-based methods can be viewed as estimating a function f on the graph [145][146], we then follow the method in [146] and adopt *transductive learning* via regularized least squares to predict labels on the unlabeled videos.

Experiments

To fully characterize how the incomplete videos affect the classification stage, we extensively evaluate a total of 280 possible parameter combinations. Here we emphasize that, we exhaustively examine all the 280 cases, primarily because we want to spot the true underlying factors behind our challenging classification-with-incomplete-data scenario.

However, to our best knowledge, there is no existing “incomplete video dataset” available for benchmark purpose, we thus create two special benchmarks from the KTH and the UCF Sports dataset. To decide how the SSL method behaves under different sparsity settings, we chose 2 action classes from the KTH dataset (*running* and *handwaving*). A total of 100 videos belong to these two actions, of which there are 49 running actions and 51 handwaving actions. The second benchmark stems from the UCF Sports dataset, in which a total of 40 actions are considered, 20 *walkingfront* actions and 20 *benchswing* actions. Our benchmarks are distinct from conventional ones because the many different sparsity and rank settings were generated from these videos.

280 *parameter combinations*

Let n be its total video number for each benchmark. We randomly selected n_c out of n and regarded them as labeled complete observations, while for the rest $n_o = n - n_c$ videos, we generated sparse videos under various settings, and considered them as unlabeled observations. The sparsity and randomness of n_o videos were specified by tensor rank

$$R = \{4, 8, 12, 16\}$$

and sparseness percentages

$$P = \{1\%, 20\%, 30\%, 40\%, 60\%, 80\%, 90\%\}.$$

The combination of both parameters yielded a total of $4 \times 7 = 28$ settings for each video, both in the complete and the sparse set.

For each of the n_o sparse videos, following the steps in Section 2, we recover its complete version under all 28 settings. Each recovered video yields a three-dimensional tensor. We then generated its rank-1 compact representations using the Algorithm 1, yielding its final feature vectors. It is worth noting that these final feature vectors are not necessarily of the same length due to different video sizes. Before the vectors are fed into the final graph based semi supervised learning stage, a z-normalization was conducted to make them of identical dimensions.

Another critical parameter

$$Q = \{10\%, 30\%, 50\%, 70\%, 90\%\}$$

was also configured throughout our experiments to denote the percentage of the n_o sparse videos out of all videos, namely, $Q = n_o/(n_o{+}n_c)$. Q was very necessary in SSL learning, since practically, we cannot know in advance how many sparse videos should be used for better classification. Under each setting, we chose $n_o = Q \times n$ sparse videos randomly rather than deterministically (Fig.6.4).

To build the sparse weighted graph, we chose between the k-NN and the ϵ-NN graph. Note that as we increase the k or ϵ, the number of edges increase accordingly (Fig.6.4). Practically, we fixed

$k = 5$ and $\epsilon = 0.3$ throughout our experiments. If under a certain setting the classification rate is larger for the k-NN graph structure than for the ϵ-NN structure, then we chose k-NN graph, and vice versa.

Results and discussions

There were totally 28 sparsity settings for each sparse video. If we take into consideration the 5 percentage ratios given by Q and the 2 graph structures in SSL framework, there will be totally $28 \times 5 \times 2 = 280$ experimental configurations for each benchmark. Note that every single configuration was involved with randomness: the sparsity itself was random, the selection of whether a video should be considered as sparse video was also made random. We thus performed classification on every single configuration for 20 times (10 for k-NN graph, the other 10 for ϵ-NN graph). We computed the average classification rate for each structure, and chose the larger value between these two. The classification rate comparisons for KTH and UCF Sports is shown in Table 6.1.

How does rank value affect error rates? The rank R is the primary factor that affects the incomplete video recovery. The gradient descent method benefits from a larger rank value, and hence increases recovery precision. Experimentally, we observed in most cases that larger rank does help reduce classification error rate (a.k.a, boosting classification rates). Also, it is worth noting that in Fig.6.2, we use fixed rank value $R = 16$, which makes the recovered frames more authentic than other ranks smaller than 16. Note that larger ranks also introduces higher computational burden.

How does the "incompleteness" affect error rates? We studied a whole range of possible occlusion percentages P (Table 6.1), and observed that as P increases, the classification error rates follow an overall subtle (but not monotonic) increasing pattern. Due to randomness, lower Q in some cases (e.g., 60% and 80% for UCF sports) leads to higher error rates. But the overall trend can be readily perceived. Firstly, under heavy occlusion ($P \geqslant 90\%$), a large portion of video pixels is missing and video's structure is heavily corrupted. The recovered incomplete videos no longer contribute positively to the classification rates that are slightly better than random binary selection (i.e. 50% probability). Secondly, within the range 1% ~ 80%, the incomplete videos are under either mild or large occlusions. By properly combining occlusion percentage P and unlabeled data ratio Q, their classification rates can be acceptable, or even very high. To sum up, *incomplete videos, once properly recovered, indeed show merits in action classification. Especially, in the cases where only few complete videos are available, it is feasible (or even necessary) to include available incomplete videos, rather than simply ignore or discard them.*

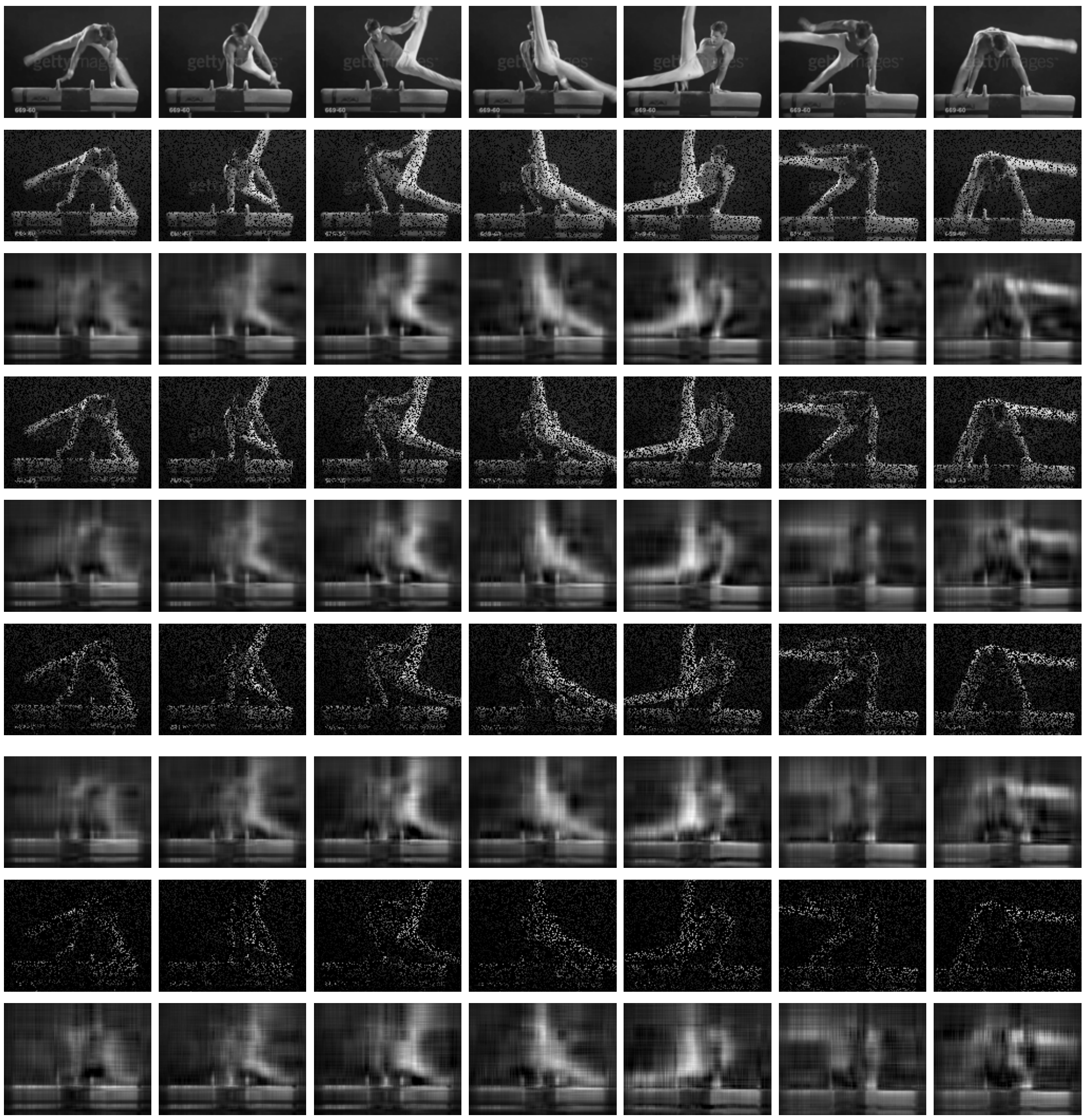


Figure 6.3: Occluded frames under different occlusion settings for the UCF Sports *Swingbench* action. The $1st$ row shows the original complete frames; The $2nd, 4th, 6th, 8th$ rows shows $20\%, 40\%, 60\%$, and 80% pixels are occluded; The $3rd, 5th, 7th, 9th$ row shows corresponding recovered frames. We fix the rank value R = 16.

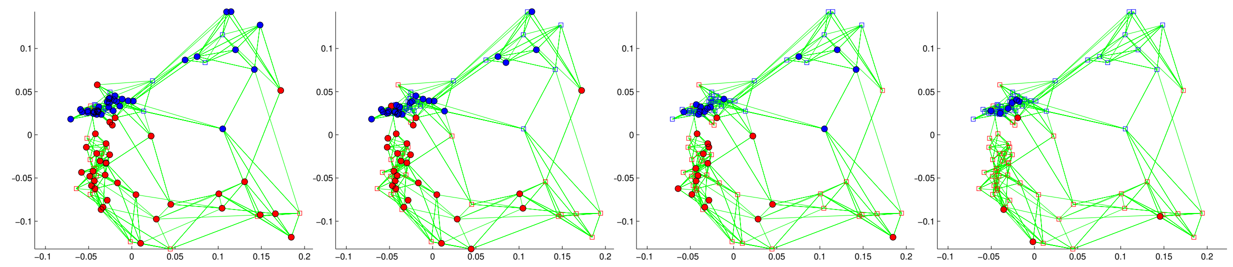


Figure 6.4: The constructed k-NN ($k = 5$) sparse weighted graph for the KTH dataset. For a total of 100 videos, there are two classes involved: 49 *running* actions (Red) and 51 *handwaving* actions (Blue). All circle dots denote complete videos, and all square dots denote sparse incomplete videos. The percentage of sparse unlabeled videos, from left to right, are 30%, 50%, 70%, and 90%, respectively. (Zoom in for better view)

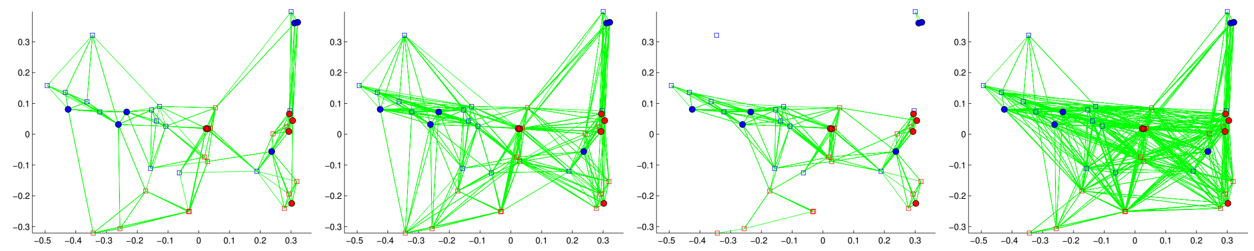


Figure 6.5: The constructed k-NN and ϵ-NN sparse weighted graphs for the UCF Sports dataset. For a total of 40 videos, there are two classes involved: 20 *walkfront* actions (Red) and 20 *swing-bench* actions (Blue). All circle dots denote complete videos, and all square dots denote sparse incomplete videos. The percentage of sparse unlabeled videos used is 30%. (1st) k-NN graph with $k = 5$; (2nd) k-NN graph with $k = 10$; (3rd) ϵ-NN graph with $\epsilon = 0.3$; (4th) ϵ-NN graph with $\epsilon = 0.5$

CHAPTER 7: MOTION RETRIEVAL USING MOTION SEQUENCE VOLUME[1]

We propose a framework for motion retrieval by exploring the notion of "self-similarity", which has received significant attention recently. The work in [18] describes a gait recognition technique based on the visual self-similarity of a walking person to classify the movement patterns of different people. [147] shows the effective use of the self-similarity in recognizing different types of biological periodic motions. In [148], the authors explore self-similarities of action sequences over time to capture the structure of temporal similarities and dissimilarities within an action sequence. For a given action sequence, they compute the distances between action representations for all pairs of time-frames and store the results in a Self-Similarity Matrix (SSM). They consider the temporal similarities between frames of image sequences. The method in [148] also exploits the notion of image self-similarity. For a given action/motion sequence, they first extract some low level features. The distances between extracted features for all pairs of time frames are computed and this results in a SSM. Each action sequence is thus reduced to a 2D SSM matrix, and the authors then proceed to extracting some useful features from these SSMs and use them to train their action recognition system.

[1]The content in this chapter was published in the paper: "Motion Sequence Volume based Retrieval for 3D Captured Data", Computer Graphics Forum (CGF), Volume 30, Issue 7, pages 1953-1962, September 2011.

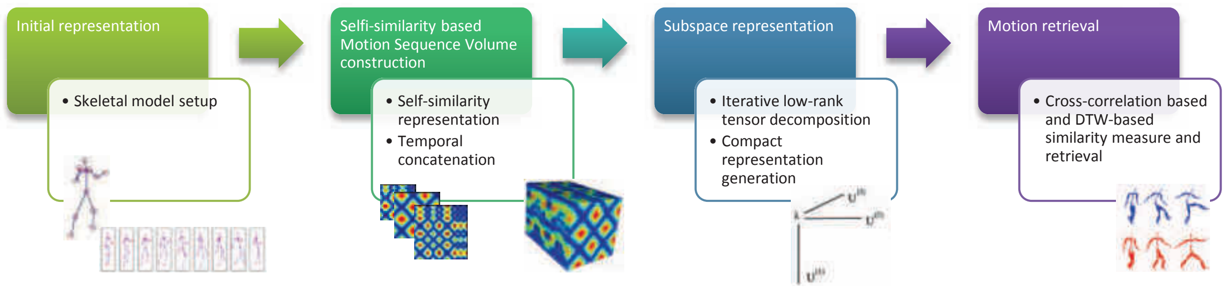

Initial representation
• Skeletal model setup
Selfi-similarity based Motion Sequence Volume construction
• Self-similarity representation
• Temporal concatenation
Subspace representation
• Iterative low-rank tensor decomposition
• Compact representation generation
Motion retrieval
• Cross-correlation based and DTW-based similarity measure and retrieval

Figure 7.1: The overview of our retrieval framework for motion capture data.

We make multiple contributions to the motion retrieval field. The first contribution is that, we propose a retrieval framework that does not require time alignment in contrast to the conventional methods. We employ the notion of self-similarity derived from the recurrence plot theory, and generate a unique, symmetric structure to summarize every motion sequence. *Second*, we propose a novel scheme of subspace dimensionality reduction for fast motion sequence retrieval based on rank-1 tensor decomposition. Compared with the recent approaches that involve dynamic time warping (DTW) for building score-based correspondences between related events, our tensor subspace decomposition condenses an action into its meaningful representation by reducing it to its lowest rank feature vectors. *Third*, as a byproduct of low rank decomposition, reduced time-complexity, and hence fast indexing and retrieval is achieved to handle large databases.

Fig.7.1 gives an overview of our proposed framework. *First*, we set up skeletal model for motion sequence poses for initial representation; *Second*, we convert motion sequence into a series of self-similarity matrix representations in temporal dimension under Euclidean distance metric, thereby creating a Motion Sequence Volume (MSV) structure that encodes the internal dynamics of a motion sequence. *Third*, the structure is decomposed into three low-rank compact vectors using an optimal iterative algorithm. *Finally*, we employ the cross correlation based similarity measure for the final retrieval phase.

Motion Capture Data Representation

Motion sequence retrieval has been extensively explored for over a decade. The task is to query a motion sequence and find the sequences whose distance to the query is either below a threshold τ or among the k smallest. Our primary motivation is to devise an effective data representation and framework that identifies similar motions that are numerically dissimilar. Our secondary motivation is to propose a method that allows the animators to quickly find similar motions within a large motion capture database.

Terminology

Human pose can be represented using a simplified model of human *skeleton*, composed of bones that are connected by *joints*. Motion capture can then be regarded as the process of recording a temporal sequence of 3-dimensional joint positions. The position of all joints at a given time is known as a *pose*, described as a vector $p \in \mathbb{R}^{3\times|J|}$, where $|J|$ is the number of joints in the skeletal model and each joint requires 3 elements to describe its 3D position. A mocap sequence can be then formally described as a time-dependent sequence of poses. This can be represented by a 2D matrix $S \in \mathbb{R}^{T\times(3\times|J|)}$, where T is the number of poses (frames) in the mocap sequence.

Skeletal model

The CMU mocap dataset uses a skeletal model of 32 joints (i.e., $|J| = 32$). It provides a detailed configuration for the joints such as the thumb joint. However, efficient retrieval or classification does not require detailed joints information when performing comparison between motion classes. For instance, the variation in thumb joint position is obviously less important than that of the femur joint position. Also, since it is believed that 13 joints are sufficient to represent a motion action

[106], we evaluate our motion capture sequences using a skeletal model consisting of 13 most significant joints, namely the *left knee, right knee, left foot, right foot, heep, left elbow, right elbow, left shoulder, right shoulder, neck, head, left hand*, and *right hand*. We concatenate these joints into a vector in the order specified by indices from 1 to 13, as shown in Fig.7.3.

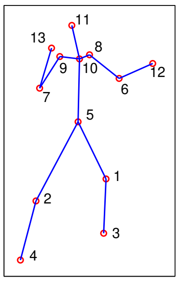


Figure 7.2: The tracked joint positions and their indices for forming the initial vector

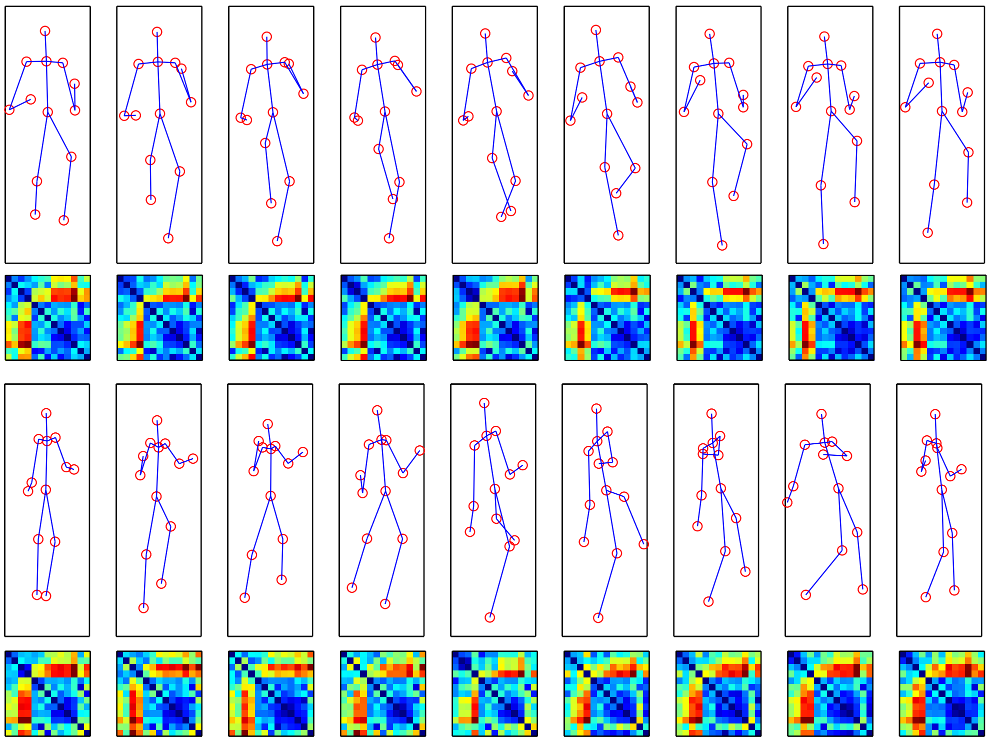


Figure 7.3: SSMs built for *Run* and *Kick* actions in the CMU Mocap dataset by stacking the 13 key limb joints. All SSMs are of dimension 13×13 and are generated from the vector by concatenating the tracked limb points

In this way, each frame of an action sequence can be initially represented by a vector of size 13, which is used in the subsequent Motion Sequence Volume construction stage. In Fig.7.4, we show 5 out of 12 motion sequences from the CMU Mocap dataset. It can be observed that apparently $walk$ and run exhibit similar motion patterns. The $walkturn$ motion is also similar to $walk$ motion except that the human body turns by some angle while walking. While traditional motion retrieval approaches sometimes fail to discriminate these pairs of near-identical patterns, our method is capable of finding their characteristic differences by making it possible to focus on a subspace of the motion volume.

Motion Sequence Volume (MSV) Construction

Since the 3D human motion data is typically high-dimensional and nonlinear, operations on such data often involve dimensionality reduction as a critical step before further processing. Forbes et al. [149] used the weighted PCA in searching of motion data. Barbic et al. [150] used PCA to cut a long motion data stream into single behavioral segments. The method in [151] detects key motion poses in locations where local variations are sharp with a Multi-Dimensional Scaling (MDS) technique. Due to the fact that each frame encoded by a high-dimensional vector has a nonlinear relationship with other frames, [152] adopted the Local Linear Embedding (LLE), an unsupervised non-linear dimensionality reduction technique, to compute a low-dimensional, neighborhood-preserving embedding for motion poses. The approach attempts to discover nonlinear structures in high-dimensional data by exploring the local symmetries of linear reconstructions.

Unfortunately, popular low-dimensional embedding approaches like PCA and MDS tend to create distortions in nonlinear manifolds due to their linearity, and thus hinder the detailed dynamics of the motion. As a result, they are suitable for datasets where inter-class distances are sufficiently large to distinguish different motion sequences. On the other hand, the LLE method is highly susceptible to local minima in optimization, and tends to emphasize preserving purely local geometry and ignores distant inputs.

To tackle this problem, we perform the subspace projection and dimensionality reduction from a different perspective. The main motivation is to deal with the nonlinearity of the high-dimensional input motion manifold by using the multilinear algebra and the recurrent plot theory. Starting from the construction of self-similarity matrix, our technique builds an order-3 tensor from the input motion, and then performs an optimal rank-1 tensor decomposition to reduce a motion sequence into its compact and discriminative representation.

Based on our skeletal model representation, we compute the SSM for each pose p_i in the motion sequence. In order to characterize the dynamics of the input motion sequence based on the SSMs, we construct a 3D structure called Motion Sequence Volume (MSV) by linearly concatenating the intermediate SSMs in temporal dimension. We define the MSV for pose p under distance metric q as

$$\mathcal{V}^{(q)}(p) = \{\mathcal{M}^{(q)}(p_1), \mathcal{M}^{(q)}(p_2), ..., \mathcal{M}^{(q)}(p_t)\}, \tag{7.1}$$

where t is the temporal dimension of the pose, and $\mathcal{V}^{(q)}(p)$ denotes the concatenation of the SSM of each individual pose p_i.

It can be observed that for different motion sequences, the resulting MSVs might be of different temporal dimensions, but they all have identical spatial dimension since their components $\mathcal{M}^{(q)}(p_i)$ are computed under the same skeletal model configuration. More specifically, the dimension of all MSV is $13 \times 13 \times t$ in our method, where t denotes their corresponding temporal dimension.

Fig.7.5 illustrates 5 different MSVs computed using Euclidean metric for 5 motion sequences in Fig.7.4. We use xyz axis to denote the spatiotemporal dimension of the MSV structure. xy-plane encodes the spatial dimensions of SSMs, z direction depicts the temporal dimension. As we can see, the similarity and hence confusion between $walk$ and run can be largely reflected in their MSV structure. Especially, we can observe that the yz-plane of both motions have similar spatial patterns given their different motion durations. Similar observations can be made for *walk* and *walkturn* sequences.

MSV Tensor Approximation

As the MSV is a characteristic, unique, and a symmetric 3D structure, we developed an iterative algorithm using the tensor theory to extract the most compact and optimized discriminative features from it. The motivation for this subspace decomposition is twofold:

1. It reduces the dimensionality of the problem by exploiting the redundancy in the symmetric structure of the motion manifold;
2. It does not distort the manifold by imposing linearity as it is common practice in existing dimensionality reduction techniques.

This latter motivation, is important, as also verified experimentally later, for preserving the dynamics of the motion in this process of dimensionality reduction.

We now discuss the intuition for each decomposed vector and how each vector can be used to interpret spatial-temporal variations of human motion. In algorithm 1, $\rho^{(t+1)}$ is updated using both $\rho^{(t)}$ and $\varepsilon^{(t)}$ by $\overline{\times}_2$ and $\overline{\times}_3$ operators, leaving $\mathcal{A}$ unaffected in mode-1 dimension; while $\varepsilon^{(t+1)}$ is updated using the two $\rho^{(t)}$ by $\overline{\times}_1$ and $\overline{\times}_2$ operators, leaving $\mathcal{A}$ unaffected in mode-3 dimension. It is known from tensor theory and symmetry property of the MSV that the mode-1 and mode-2 of the MSV is identically responsible for spatial dimension, and the mode-3 for the temporal dimension. In other words, the primary vector ρ *mainly* encodes the dynamics of the spatial variance of MSV while the secondary vector ε *mainly* encodes the temporal variance.

By using the algorithm, each motion sequence in the database is transformed to three low-dimensional vectors, namely, two identical vectors $U^{(1)}, U^{(2)}$ mainly corresponding to the spatial dimensions of the motion sequence, and another vector $U^{(3)}$ mainly corresponding to the temporal dimension. The retrieval can then be performed at the event level rather than at the frame level, which leads to huge reduction in memory consumption and run time for large-scale databases, while preserving the dynamics of the motion manifolds, as shown in our experiments.

In order to illustrate the difference between the decomposed vectors for motions from the same action classes, we randomly chose 6 samples for *walk*, 8 samples for *walkturn*, and 7 sample for *cartwheel* from the CMU Mocap dataset. Samples within the same motion classes have various frames and execution speeds, as shown in the second row of Fig.7.6. Take the *walk* for example. Although its 6 samples are performed by various actors under different speeds, their decomposed vectors ($U^{(1)}$ and $U^{(3)}$), show strong similarities. This is also true for the *walkturn* and *cartwheel* classes. Especially, the *cartwheel* class involves more complex body movements and orientation variations than *walk*, and hence the decomposed secondary vectors $U^{(3)}$ vary a lot compared with *walk* and *walkturn*. However, we can still identify the striking similarities when comparing their curves.

Another noticeable example is the comparison between *walk* and *walkturn*. Logically, both motion classes are very similar except that the *walkturn* motion class involves a turning angle. In other words, the spatial configurations of the body skeleton joints at a specific frame for both motion classes are very similar but these spatial configurations vary to some extent along the time dimension. Therefore, the primary vectors for both motion classes are slightly similar. However, the secondary vectors encode the subtle variations between those two motion classes, as illustrated also in Fig.7.6.

Note that it may not be straightforward to visualize or compare directly the similarities of motion sequences within the same class from their generated MSVs (Fig.7.5). Their decomposed vectors $U^{(1)}$ and $U^{(3)}$, however, clearly demonstrate strong similarities given different execution speeds and body joints moving patterns. Specifically, within the same motion class, the primary vector $U^{(1)}$ of motion samples largely overlap. It is worth noticing also that even for motion samples from different classes, their primary vectors may still overlap to some extent, but their secondary vectors will instead largely capture the motion dynamics.

Similarity Measure

For motion retrieval, finding motion sequences that are similar to a given query motion p is highly dependent on the similarity measure. Variations in the same action class are allowable as long as we can correctly identify the matches. However, it is known that similar motions may be numerically dissimilar because corresponding poses may have various joint orientations and angular velocities.

As a result, traditional methods based on aggregating frame-wise scores, often fail to distinguish between numerical similarity and visual similarity. In other words, they fail to differentiate motions that are different versions of the same class of action [81]. On the other hand, linear subspace decomposition methods that achieve dimensionality reduction, distort the non-linear motion manifold, and hence cause the opposite problem of concealing the dissimilarity between two close, but different classes of actions.

We now describe our similarity measure as follows. Let $p^{(i)}$ and $p^{(j)}$ be the two initial motion sequences. By dimensionality reduction via rank-1 tensor decomposition, we obtain their corresponding approximated vector pairs $v^{(i)} = \{U_i^{(1)}, U_i^{(2)}, U_i^{(3)}\}$ and $v^{(j)} = \{U_j^{(1)}, U_j^{(2)}, U_j^{(3)}\}$, respectively. Specifically, for both $v^{(i)}$ and $v^{(j)}$, their first two components are vectors of size 13, and the size of the third component is equal to the motion sequence frame number.

We define two similarity metrics for motion retrieval, namely the cross-correlation based metric $\mathcal{D}_{xcorr}$ and the dynamic time warping based metric $\mathcal{D}_{dtw}$, respectively. The $\mathcal{D}_{xcorr}$ is defined as

$$\mathcal{D}_{xcorr}(p^{(i)}, p^{(j)}) = \sum_{k=1}^{3} d(U_i^{(k)}, U_j^{(k)}), \tag{7.2}$$

with

$$d = \max C(U_i^{(k)}, U_j^{(k)}),$$

where $k = 1, 2, 3$ and r is the component dimension. The function C returns the vector containing the cross-correlation values between the components $U_i^{(k)}$ and $U_j^{(k)}$. For various motion sequences, their execution frames are probably of unequal length, leaving those two components of unequal size, then we zero-pad the shorter vector to the length of the longer vector. By this similarity measure, the more similar $U_i^{(k)}$ and $U_j^{(k)}$ are, the larger the value of $\mathcal{D}$ will be.

The other metric $\mathcal{D}_{dtw}$ is defined as

$$\mathcal{D}_{dtw}(p^{(i)}, p^{(j)}) = \sum_{k=1}^{3} DTW(U_i^{(k)}, U_j^{(k)}), \tag{7.3}$$

where the function DTW is used for measuring the similarity between the two input sequences. Note that although DTW-based comparisons are widely used in motion retrieval, our usage here has particular advantage that the computation is performed on our decomposed vectors rather than on initial motion data as in conventional methods [81].

Two Experiments

We consider the motion sequences corresponding to 12 actions (*walkturn, golf, fjump, flystroke, j-jack, jump, cartwheels, drink, kick, walk, bend, run*) from the CMU mocap database. This amounts to a total of 196 motion clips with the duration length ranging from 90 to 1000 frames per clip. The motion actions are chosen so that they span various classes to illustrate the generality of our technique. For example, the *walk* and *run* actions are similar in nature representing different degrees of a single locomotion class, whereas *jump*, *kick*, etc are distinct motion types. All experiments were implemented with Matlab with 4GB memory and a 2.66 GHz Core2Duo processor.

Experiment 1

To efficiently measure the similarity of the query motion with motions in our dataset, we first performed the motion retrieval using the given query motion q under similarity measure in both Eq.(7.2) and Eq(7.3). Since the degree of similarity actually represents the relevance between the queried motion and all retrieved candidate motions, all the retrieved motion clips were sorted according to their degree of similarity to the queried motion.

Table 7.1: Comparison of retrieval rate from 4 selected motion classes

	walk	run	jump	kick	Overall
Ours using $\mathcal{D}_{dtw}$	93.0	95.2	92.0	100.0	95.1
Ours using $\mathcal{D}_{xcorr}$	95.7	97.2	83.4	86.8	90.8
Deng [83]	92.1	98.0	87.4	78.0	88.9
Kovar [81]	82.0	85.3	86.7	75.2	82.3
Forbes [149]	74.4	80.0	63.1	65.5	70.8
Liu [153]	80.0	93.1	70.9	63.9	76.9

In other words, for each query, we maintained a sorted list consisting of all motions in descending order denoted by $\mathcal{Q} = \{Q_i\}$, where $i = 1, 2, ..., N$ with N being the number of all motion sequences. Let $\mathcal{C} = \{c_j\}$, where $j = 1, 2, ..., J$, be the set of motion classes. Here $J = 12$ since we considered 12 motion classes. Let the function f denote the correspondence between a motion Q_j and its class c_i, namely $c_i = f(Q_j)$. For a query motion q, let CQ_k be the set of classes for the top k candidate motions, i.e., $CQ_k = \{Q_1, Q_2, ..., Q_k\}$, where we set $k = 20$. Our aim is to establish how many motions in CQ_k have the identical motion class as $f(q)$. For example, the $walk$ and run motions sometimes cannot be correctly differentiated because they are visually different but numerically similar. Intuitively, given a *walk* motion for querying, we want our retrieved results have more candidates for *walk* rather than *run*, and have less or even no candidates for *golf* motion.

To evaluate the retrieval accuracy of our approach, we conducted experiments similar to the settings in [81] and [83]. Our objective was to query the same motion in two different types of datasets. The first one was the labeled dataset $D_{labeled}$ with the same motion class, and the second one was a mixture of unlabeled motions $D_{unlabeled}$ with various classes. The $D_{labeled}$ can be considered as the ground-truth compared with the $D_{unlabeled}$. In order to make comparison with the experiments in [81] and [83], we collected a motion dataset consisting of 4 motion categories, namely *walk*, *run*, *jump*, and *kick*, with 80 sequences and 4570 frames from the CMU mocap dataset. Other motion categories were not included in this experiment either because their numerical experimental results were unavailable, or because they were tested in a different dataset.

The work in [154] used the *True-Positive Ratio* as an accuracy criterion that was defined as the percentage of the top k retrieved candidates from the mixed dataset $D_{unlabeled}$ that were in the correct labeled dataset $D_{labeled}$. Here we also used this criterion and compared our approach with several related works, as shown in Table 7.1. Those four motion categories were correctly retrieved under the two similarity measures ($\mathcal{D}_{xcorr}$ and $\mathcal{D}_{dtw}$) at high mean rates (90.8% and 95.1%). Especially, the dynamic time warping based similarity measure outperformed the corss-correlation based similarity measure in this testing scenario.

Experiment 2

It is known that confusion matrix is a good representation for rate comparison, which contains information about actual and predicted classified (or retrieved) data performed by a classification (or retrieval) system. For a retrieval system, each column of this matrix represents the instances in a retrieved motion class, while each row represents the instances in an actual motion class. An important advantage of a confusion matrix is that it is easy to see if the system is confusing two classes (i.e. commonly mislabeling one as another). For this reason, as an alternative of true-positive ratio, the confusion matrix is adopted herein to evaluate the efficiency of our method on more than 4 motion classes compared with the previous experiment.

Below are the details of how the experiment was conducted. Here, the input is only labeled motions instead of the two separate datasets $D_{unlabeled}$ and $D_{labeled}$ as in the first experiment. We considered not only the number of correctly retrieved motions, but also the classes of mistakenly retrieved motions. Fig.7.7 shows the two confusion matrices of mean retrieval rates using the similarity measures in Eq.(7.2) and Eq.(7.3) for all the 196 motion clips with 12 classes. The diagonal of this matrix denotes the correct retrieval rate for a certain motion sequence, and the off-diagonal elements denote the rate for mistakenly retrieved classes. For the *flystroke* motion in the first confusion matrix of Fig.7.7, for instance, 95.8% of all the retrieved motion clips are correctly retrieved as *flystroke*, whereas only 4.2% are incorrectly retrieved as $jjack$ motion, and for the *golf* motion, 100% of all retrieved motions are correctly identified. This means that, for the *golf* action our method retrieves the motion with no confusion.

Since logically the *walkturn* and the *walk* are very similar but numerically the former one involves a turning angle while walking, as shown in Fig.7.6, it may not be easy to differentiate them using conventional retrieval methods. But both confusion matrices demonstrate that our method can nicely differentiate them with very low confusion (< 5%). Another special pair is the *fjump* and the *jump*. It is worth mentioning that both logically and numerically those two categories are extremely similar: the *fjump* involves jumping forward at a certain distance; while the *jump* involves landing on the same location. The two confusion matrices show that our approach can still have somewhat acceptable retrieval rates.

Moreover, the overall mean retrieval rates of our method in this experiment is 93.1% and 91.8%, respectively, for the two similarity measures. It can be noticed that the overall performance of cross-correlation based measure is slightly better than the dynamics time warping measure, but its performance falls behind when handling some motion categories like *walkturn, flystroke, jjack*, and *kick*.

Time and space complexity

Since we reduce each motion sequence to low-rank vectors for similarity measure, the retrieval can be performed at the event level rather than at the frame level. This can largely reduce the run time for large-scale databases, while preserving the dynamics of the motion manifolds, as shown in our experiments. In particular, the time complexity required to build and store our index structure is *linear, O(n)* compared with DTW-based strategies, which are *quadratic, O(n^2)*, with regards to the frame number n in the database.

Based on the mocap sequence terminology in Section 7, the space complexity for an original motion dataset of m clips can be expressed as $S = \sum_{i=1}^{m} 3 \times h \times T_i$, where h is the number of joints in skeletal model ($h = 13$ in this work), and T_i is the temporal duration for the ith motion clip. Using our low-rank subspace decomposition in **Algorithm** 1, the space complexity becomes $S' = \sum_{i=1}^{m}(h+T_i)$ since the length of $U^{(1)}$, $U^{(2)}$, and $U^{(3)}$ are h, h, T_i, respectively. The compression ratio is thus

$$R = S'/S = \frac{m}{3\sum_{i=1}^{m} T_i} + \frac{1}{3h}.$$

As we can see, a smaller R corresponds to a larger saving in space complexity. In order to reduce R, one can either increase the quantity $\sum_{i=1}^{m} T_i$ or h. Intuitively, the quantity $\sum_{i=1}^{m} T_i$ is relatively large for large capture datasets. As modern motion capture datasets grow increasingly large, our technique becomes particularly suitable in the applications where both retrieval efficiency and accuracy are required.

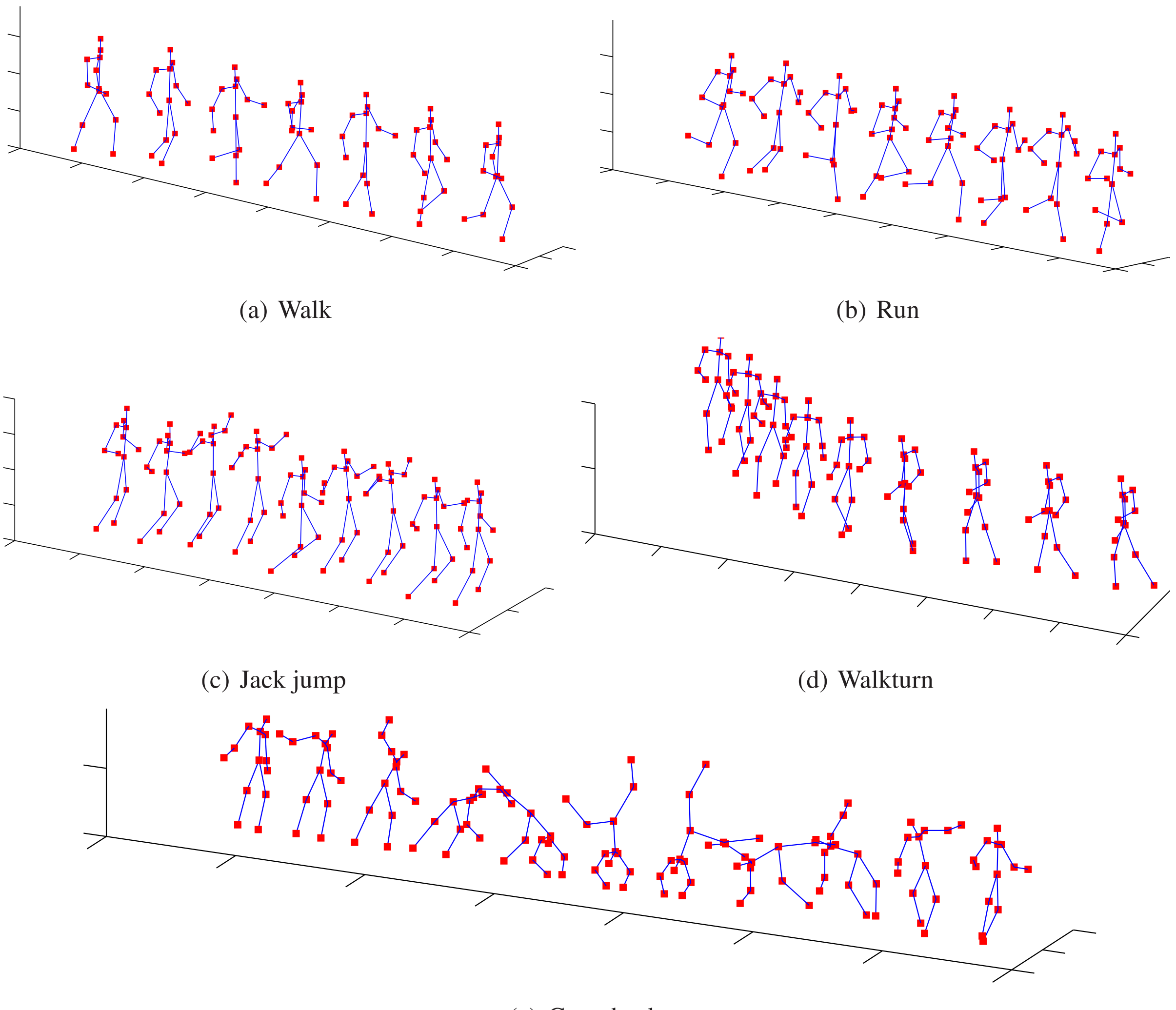
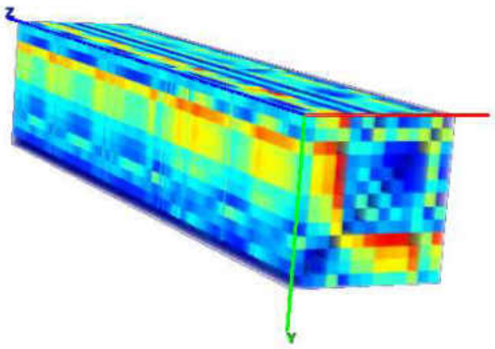
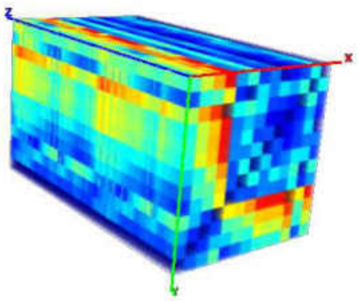
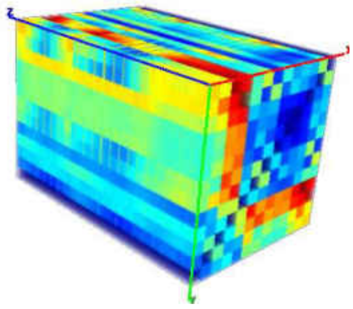
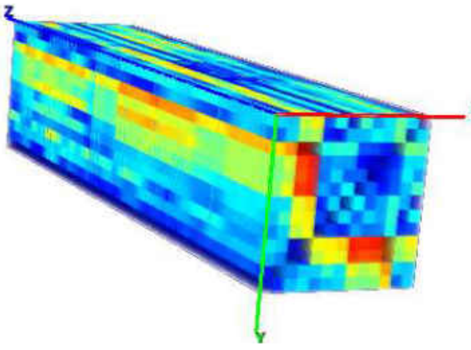

(a) Walk
(b) Run
(c) Jack jump
(d) Walkturn
(e) Cartwheel

Figure 7.4: Five selected Mocap motion sequences out of 12 motion classes from CMU Mocap dataset (*walkturn, golf, fjump, flystroke, jjack, jump, cartwheels, drink, kick, walk, bend, run*). Each motion sequence is performed by various actors. Each pose is represented by a skeletal model consisting of 13 joints.


(a) Walk MSV


(b) Run MSV


(c) Jack Jump MSV


(d) Walkturn MSV

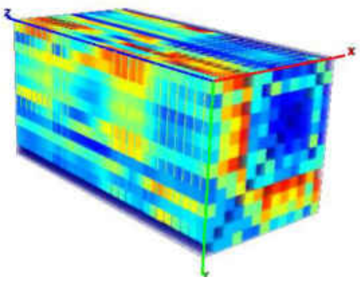

(e) Cartwheel MSV

Figure 7.5: The constructed Motion Sequence Volume for the selected 5 motion sequences in Fig.7.4. Each slice in xy-plane is the Euclidean metric based SSM of dimension 13×13. The z direction represents the temporal dimension of the motion sequence. Each volume structure is symmetric, encoding the internal dynamics of motion.

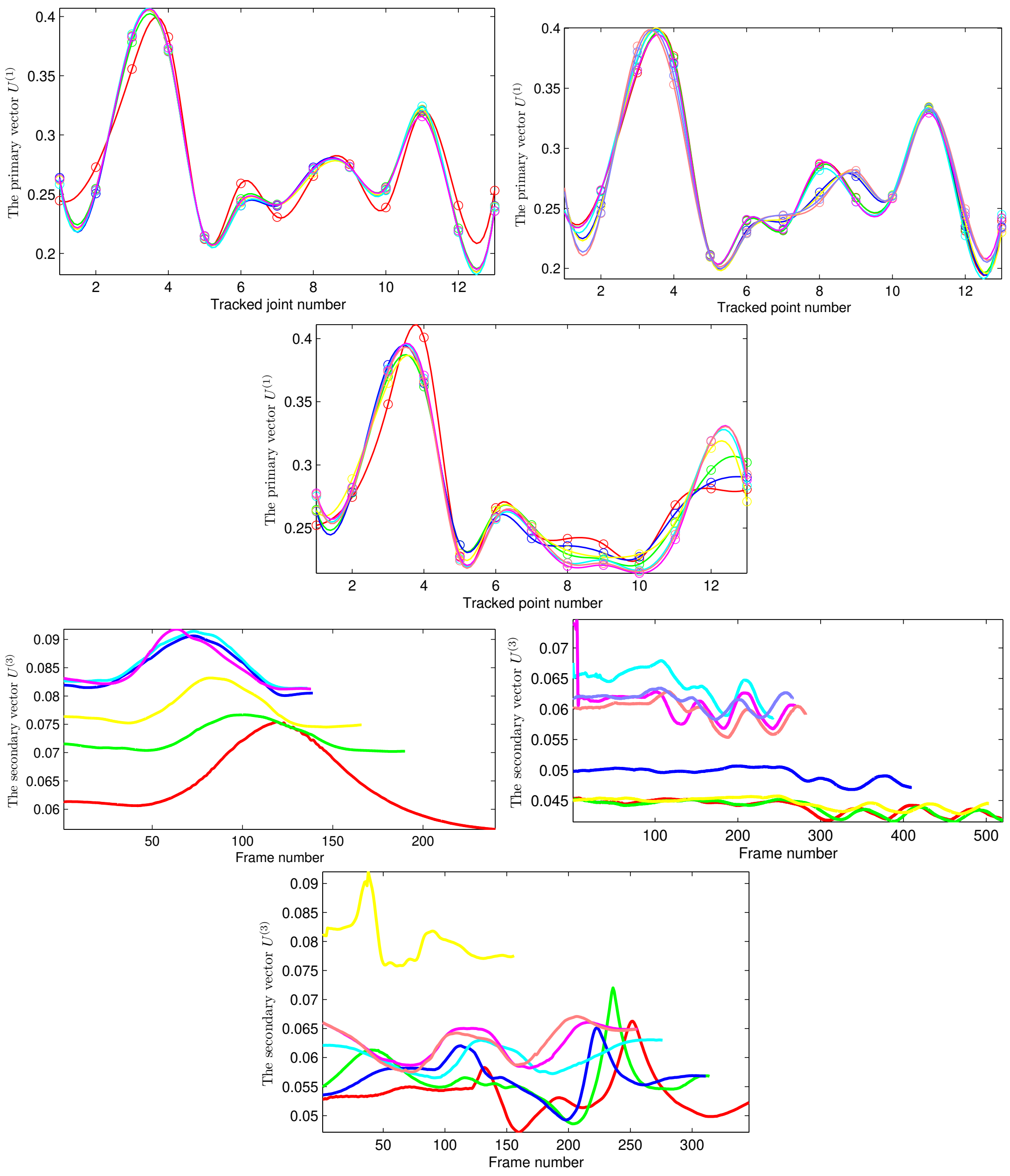


Figure 7.6: Comparisons for the decomposed *primary vector* $U^{(1)}$ and the *secondary vector* $U^{(3)}$ between 3 classes of motions. The columns from left to right correspond to *walk* (6 samples), *walkturn* (8 samples), and *cartwheels* (7 samples), respectively. The first and the second row correspond to the comparisons for the *primary vector* and *secondary vector*, respectively. In each column, the curves in the same color denote the same motion samples within identical class (Better to view in color).

	walkturn	golf	fjump	flystroke	jjack	jump	cartwheels	drink	kick	walk	bend	run
walkturn	**95.8**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.2	0.0	0.0
golf	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
fjump	0.0	0.0	**93.2**	0.0	0.5	5.9	0.0	0.0	0.0	0.5	0.0	0.0
flystroke	0.0	0.0	0.0	**95.8**	4.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
jjack	0.0	0.0	0.0	0.0	**97.6**	0.0	0.0	0.0	0.0	2.4	0.0	0.0
jump	0.0	0.0	16.7	0.0	0.0	**79.2**	0.0	0.0	0.0	4.2	0.0	0.0
cartwheels	0.0	2.4	0.0	0.0	2.4	0.0	**95.2**	0.0	0.0	0.0	0.0	0.0
drink	0.0	0.0	0.0	0.0	0.0	4.2	0.0	**83.3**	0.0	0.0	0.0	12.5
kick	0.0	0.0	1.9	0.0	0.0	0.0	0.0	0.0	**96.3**	1.9	0.0	0.0
walk	0.8	0.0	0.4	0.0	0.0	0.0	0.0	0.0	0.4	**96.1**	0.0	2.3
bend	0.0	0.0	11.1	0.0	2.8	0.0	0.0	0.0	2.8	0.0	**83.3**	0.0
run	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.6	0.0	**98.4**

	walkturn	golf	fjump	flystroke	jjack	jump	cartwheels	drink	kick	walk	bend	run
walkturn	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
golf	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
fjump	0.0	0.0	**94.6**	0.0	0.0	5.4	0.0	0.0	0.0	0.0	0.0	0.0
flystroke	0.0	0.0	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
jjack	0.0	0.0	0.0	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0
jump	0.0	0.0	25.0	0.0	0.0	**75.0**	0.0	0.0	0.0	0.0	0.0	0.0
cartwheels	0.0	0.0	0.0	14.3	0.0	0.0	**85.7**	0.0	0.0	0.0	0.0	0.0
drink	0.0	25.0	0.0	0.0	0.0	0.0	0.0	**75.0**	0.0	0.0	0.0	0.0
kick	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**100.0**	0.0	0.0	0.0
walk	2.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**93.0**	0.0	4.7
bend	0.0	0.0	16.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**83.3**	0.0
run	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.8	0.0	**95.2**

Figure 7.7: The confusion matrices of motion retrieval for CMU mocap dataset using two similarity measures. (Left) Cross-correlation based measure; (Right) Dynamic time warping based measure

CHAPTER 8: ACTION SPOTTING VIA RANK-BASED TENSOR CORE PYRAMID [1]

A natural way to represent a video is to holistically encode its internal dynamics in a three-way tensorial manner, rather than the commonly-used ad-hoc vectorization. The multilinear tensor analysis, subsuming conventional linear analysis as a special case, emerges as a unifying powerful mathematical framework suitable for addressing problems in many fields [33]. In computer vision, the past decades witnessed many successfully applications in directions such as face recognition [33], action recognition [50], action categorization and detection [155], etc.

Our motivation is to spot actions in a data-driven manner without relying on commonly-used features, and without requiring human localization, segmentation, or frame-wise body parts tracking.

To achieve this, we formulate the problem as that of discovering the internal dynamics of a three-way time-space tensor. In a nutshell, we treat all video cuboids involved as three-way tensors. We then introduce a new multilinear tensor decomposition called Two-Phase Decomposition (TP-Decomp) tailored for action spotting, by combining the Tucker decomposition with CANDECOMP/PARAFAC (CP for short) decomposition [107] in a natural yet effective way. This is then used to establish a Rank-based Tensor Core Pyramid (Rank-TCP) descriptor using multiple tensor cores under multiple ranks, which is basically a new tensor-based hierarchical video representation. At the final template matching stage, we adopt two effective boosting strategies that requires no human localization, segmentation, or frame-wise tracking.

[1]The content in this chapter was published in the paper: Chuan Sun, Marshall Tappen, Hassan Foroosh, "Feature-Independent Action Spotting Without Human Localization, Segmentation or Frame-wise Tracking", IEEE CVPR, 2014.

Two-Phase Decomposition (TP-Decomp)

The Tucker and CP decomposition are two powerful techniques that decompose tensors onto modes [107]. However, they have very different characteristics (the Tucker algorithm is a form of higher-order PCA that decomposes a tensor into a core tensor multiplied by a matrix along each mode; while the CP decomposition factorizes a tensor into a sum of component rank-1 tensors [107]), and have been mostly treated as two distinct algorithms independently applied to various fields [33, 156, 157].

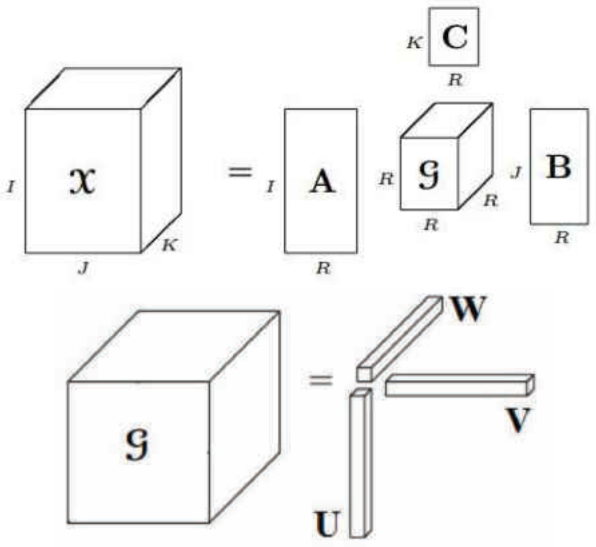


Figure 8.1: The illustration of Tucker decomposition followed by CP decomposition.

We establish a procedure called TP-Decomp that combines these two powerful techniques in a natural yet very effective way. Its resulting vectors shows remarkably good performance in terms of robustness and discriminative ability.

TP-Decomp procedure

Given a query video $\mathcal{Q} \in \mathbb{R}^{I\times J\times K}$ and a large target video $\mathcal{S}$, where I and J correspond to the spatial dimension of $\mathcal{Q}$, and K corresponds to the temporal dimension. The objective of action spotting is to find the best match for $\mathcal{Q}$ out of all sub-volumes of $\mathcal{S}$. We formulate this problem in a tensor-based framework. For the preliminaries of tensor definitions and operations, please refer to [107].

Since query video $\mathcal{Q}$ is a 3-way tensor, we first apply the Tucker decomposition to $\mathcal{Q}$, resulting in a smaller-sized tensor $\mathcal{G}$ and three factor matrices

$$\mathcal{Q} \approx \mathcal{G} \times_1 A \times_2 B \times_3 C = \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} g_{pqr} a_p \circ b_q \circ c_r, \tag{8.1}$$

where $A \in \mathbb{R}^{I\times P}$, $B \in \mathbb{R}^{J\times Q}$, and $C \in \mathbb{R}^{K\times R}$ are the orthogonal factor matrices. The $\overline{\times}_i$ operator denotes the multiplication between a tensor and a vector in mode-i of that tensor, whose result is also a tensor, namely, $\mathcal{A} = \mathcal{B}\overline{\times}_i \alpha \Longleftrightarrow (\mathcal{A})_{jk} = \sum_{i=1}^{I} \mathcal{B}_{ijk}\alpha_i$.

The first-phase operator F_{tk} is defined as follows.

Definition 1: The F_{tk} operator is a mapping that transforms the input tensor $\mathcal{Q} \in \mathbb{R}^{I\times J\times K}$ into a tensor $\mathcal{G} \in \mathbb{R}^{R\times R\times R}$, namely, $F_{tk}(\mathcal{Q}) = \mathcal{G} = \mathcal{Q} \times_1 A^T \times_2 B^T \times_3 C^T$, where $R < \min\{I, J, K\}$, and A^T, B^T, C^T are transposes of the factor matrices in Eq.(8.1).

Recall that the CP decomposition factorizes a 3-order tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ into a sum of component rank-1 tensors $\mathcal{X} \approx \sum_{r=1}^{R} u_r \circ v_r \circ w_r$, where R is a positive integer and $u_r \in \mathbb{R}^I$, $v_r \in \mathbb{R}^J$, and $w_r \in \mathbb{R}^K$ for $r = \{1, ..., R\}$. If $P = Q = R = 1$, then the CP decomposition degenerates to the rank-1 decomposition.

We define the second-phase operator F_{cp} as follows.

Definition 2: The F_{cp} operator is a mapping that transforms a cubical $\mathcal{G} \in \mathbb{R}^{R\times R\times R}$ into a quadruple, namely, $F_{cp}(\mathcal{G}) = [\![\lambda_R; U_R, V_R, W_R]\!]$, where λ is a scalar, and U, V, W are three vectors of sizes $1 \times R$.

We emphasize that, it is this second phase operator F_{cp}, which operates directly on $\mathcal{G}$, that distinguishes our TP-Decomp from conventional approaches. Schematically, the transformation flow is as follows:

$$\mathcal{Q} \xrightarrow{F_{tk}} \mathcal{G}_R \xrightarrow{F_{cp}} [\![\lambda_R^{\mathcal{Q}}; U_R^{\mathcal{Q}}, V_R^{\mathcal{Q}}, W_R^{\mathcal{Q}}]\!].$$

For notation purpose, we define a mapping function Ξ that maps the input volume $\mathcal{Q}$ to a quadruple

$$\Xi : \mathbb{R}^{I\times J\times K} \longrightarrow \{\mathbb{R}; \mathbb{R}^{1\times I}, \mathbb{R}^{1\times J}, \mathbb{R}^{1\times K}\}.$$

Theoretical Insights of TP-Decomp

Given an input query video $\mathcal{Q}$, in the first phase, the core $\mathcal{G}$ encodes most of the action dynamics. In the second phase, $\mathcal{G}$ is further reduced to compact representation via rank-1 decomposition. We mainly focus on why the TP-Decomp is robust to perturbations.

Two Operators F_{tk} and F_{cp}

In the first phase of TP-Decomp, the core $\mathcal{G}$ can be obtained by various methods such as the higher-order SVD (HOSVD) [158] or the higher-order orthogonal iteration (HOOI). They can be viewed as natural extensions to the Singular Value Decomposition (SVD) and Principal Component Analysis (PCA). In our framework, we adopt the HOOI as our building block to obtain the core $\mathcal{G}$, as shown in **Algorithm 1**. The initialization step applies HOSVD for one time. After that, multiple iterations are performed until convergence.

In the second phase of TP-Decomp, the F_{cp} operator is a mapping that transforms a cubical $\mathcal{G} \in \mathbb{R}^{R\times R\times R}$ into a quadruple. It is effective because F_{cp} is theoretically guaranteed by the fact that, for a 3-way tensor, the property of *rotational uniqueness* holds for CP decomposition [159], i.e, there is *one and only one* possible combination of rank-one tensors that sums to $\mathcal{G}$.

Algorithm 2: Higher-Order Orthogonal Iteration for Third-order Tensor $\mathcal{Q}$.

```
input  : A query video $\mathcal{Q} \in \mathbb{R}^{I_1\times I_2\times I_3}$ treated as a three-mode tensor
output: Tensor core $\mathcal{G} \in \mathbb{R}^{R\times R\times R}$ and three matrices $A^{(1)} \in \mathbb{R}^{I_1\times R}, A^{(2)} \in \mathbb{R}^{I_2\times R}, A^{(3)} \in \mathbb{R}^{I_3\times R}$
/* Initialize $A^{(i)} \in \mathbb{R}^{I_i\times R}$ for $i = 1,2,3$ using HOSVD */;
$A^{(1)} \leftarrow R$ leading left singular vectors of $\mathcal{Q}_{(1)}$ ;
$A^{(2)} \leftarrow R$ leading left singular vectors of $\mathcal{Q}_{(2)}$ ;
$A^{(3)} \leftarrow R$ leading left singular vectors of $\mathcal{Q}_{(3)}$ ;
/* Start the iterative fitting on each mode */;
while $\|\mathcal{Q} - \mathcal{G} \times_1 A^{(1)} \times_2 A^{(2)} \times_3 A^{(3)}\|_2 \leq \epsilon$ do
    $\mathcal{U} \leftarrow [\![\mathcal{Q}; I_1, A^{(2)T}, A^{(3)T}]\!]$ ;
    $A^{(1)} \leftarrow R$ leading eigenvectors of $\mathcal{U}_{(1)}\mathcal{U}_{(1)}^T$ ;
    $\mathcal{V} \leftarrow [\![\mathcal{Q}; A^{(1)T}, , I_2, A^{(3)T}]\!]$ ;
    $A^{(2)} \leftarrow R$ leading eigenvectors of $\mathcal{V}_{(2)}\mathcal{V}_{(2)}^T$ ;
    $\mathcal{W} \leftarrow [\![\mathcal{Q}; A^{(1)T}, A^{(2)T}, I_3]\!]$ ;
    $A^{(3)} \leftarrow R$ leading eigenvectors of $\mathcal{W}_{(3)}\mathcal{W}_{(3)}^T$ ;
end
$\mathcal{G} \leftarrow \mathcal{Q} \times_1 A^{(1)T} \times_2 A^{(2)T} \times_3 A^{(3)T}$ ;
```

Robustness of TP-Decomp under Perturbations

The HOOI procedure involves three tensor unfoldings, $\mathcal{U}_{(1)} \in \mathbb{R}^{I_1\times I_2 I_3}, \mathcal{V}_{(2)} \in \mathbb{R}^{I_2\times I_1 I_3}$, and $\mathcal{W}_{(3)} \in \mathbb{R}^{I_3\times I_1 I_2}$. We now only analyze $\mathcal{U}_{(1)}$ since the other two are similar. Let us denote the unfolding on the first mode as

$$M = \mathcal{U}_{(1)}.$$

Its counterpart under perturbation is denoted by M_ϵ as follows

$$M_\epsilon = M + E,$$

where E is a perturbation matrix. If $\mathcal{Q}$ is perturbed by noise, E is a Gaussian random matrix with zero mean and positive variance. However, if $\mathcal{Q}$ is perturbed by Gaussian blurring, then E is the difference between the convoluted matrix and the original.

By SVD, M can be decomposed as

$$M = U\Sigma V^T,$$

where the left and right singular matrices $U \in \mathbb{R}^{I_1 \times I1}$ and $V \in \mathbb{R}^{n \times I1}$ are orthogonal unitary matrices, which induce a rotation of the input data. The matrix $\Sigma \in \mathbb{R}^{n \times n}$ has diagonal form:

$$\Sigma = diag(\sigma_1, \sigma_2, \dots, \sigma_n),$$

where $\{\sigma_i\}$ are singular values of M and satisfy

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_R \geq \sigma_{R+1} \geq \dots \geq \sigma_n,$$

where $\sigma_1 \neq 0$. Now, by setting all but the first R singular values in M to 0, we obtain a truncated rank-R matrix M_R as:

$$M_R \equiv U\Sigma_R V^T, \Sigma_R = diag(\sigma_1, \sigma_2, \dots, \sigma_R, 0, \dots, 0) \in \mathbb{R}^{n \times n}.$$

When R is chosen properly, the condition number $\frac{\sigma_1}{\sigma_r}$ of M_R will be moderate.

In **Algorithm 1**, within each iteration, only the R leading eigenvectors of the unfolded matrices $\mathcal{U}_{(1)}\mathcal{U}_{(1)}^T$ are considered, implying that only R sigular values in Σ are considered. In fact, according to the Eckart-Young theorem [160], this truncated SVD provides the best rank-R approximation in Frobenius norm [161]:

$$\|M - M_R\|_F = \min_{X:rank(X)=R} \|M - X\|_F.$$

For this reason, selecting the R left eigenvectors of $\mathcal{U}_{(1)}\mathcal{U}_{(1)}^T$ implies an optimal rank-R approximation for $\mathcal{U}_{(1)}$ to approximate the $\mathcal{Q}$ in the first mode. Similarly, the same operations for $\mathcal{V}_2\mathcal{V}_{(2)}^T$ and $\mathcal{W}_3\mathcal{W}_{(3)}^T$ are performed, and optimal rank-R approximation can be achieved for $\mathcal{Q}$ in the second and third mode, respectively.

The core is obtained by iteratively alternating one mode while fixing the other two. In this way, the first phase operator F_{tk} produces an approximated version of $\mathcal{Q}$, denoted by $\widehat{\mathcal{Q}}$. As an intermediate tensor, the $\widehat{\mathcal{Q}}$ can be further decomposed into a core and three factor matrices as follows:

$$\mathcal{Q} \approx \widehat{\mathcal{Q}} = [\![\mathcal{G}; A^{(1)}, A^{(2)}, A^{(3)}]\!].$$

We now show that, the difference between $\mathcal{Q}$ and $\widehat{\mathcal{Q}}$ is equivalent to the difference between $\mathcal{Q}$ and the core $\mathcal{G}$. Since $A^{(i)}$ are orthonormal matrices, i.e., $\|A^{(i)}\| = 1$, and $i = 1, 2, 3$, we have $\|[\![\mathcal{G}; A^{(1)}, A^{(2)}, A^{(3)}]\!]\|^2 = \|\mathcal{G}\|^2$. Then, the difference between $\mathcal{Q}$ and $\widehat{\mathcal{Q}}$ is:

$$\begin{aligned}
&\|\mathcal{Q} - \widehat{\mathcal{Q}}\|^2 \\
&= \|\mathcal{Q} - [\![\mathcal{G}; A^{(1)}, A^{(2)}, A^{(3)}]\!]\|^2 \\
&= \|\mathcal{Q}\|^2 - 2\langle \mathcal{Q}, [\![\mathcal{G}; A^{(1)}, A^{(2)}, A^{(3)}]\!]\rangle + \|[\![\mathcal{G}; A^{(1)}, A^{(2)}, A^{(3)}]\!]\|^2 \\
&= \|\mathcal{Q}\|^2 - 2\langle \mathcal{Q} \times_1 A^{(1)T} \times_2 A^{(2)T} \times_3 A^{(3)}, \mathcal{G}\rangle + \|\mathcal{G}\|^2 \qquad (8.2) \\
&= \|\mathcal{Q}\|^2 - 2\langle \mathcal{G}, \mathcal{G}\rangle + \|\mathcal{G}\|^2 \\
&= \|\mathcal{Q}\|^2 - 2\|\mathcal{G}\|^2 + \|\mathcal{G}\|^2 \\
&= \|\mathcal{Q}\|^2 - \|\mathcal{G}\|^2
\end{aligned}$$

This means that, if $\widehat{\mathcal{Q}} \to \mathcal{Q}$, then $\mathcal{G} \to \mathcal{Q}$. Also, after many rounds of iteration until convergence, the generated tensor core $\mathcal{G}$ is a full/dense tensor, satisfying an all-orthogonality property between all its slices across three modes [158]. As such, the core $\mathcal{G}$ can be viewed as a compressed version of $\mathcal{Q}$ for two reasons: (1) The dimension of $\mathcal{G}$ could be much smaller than $\mathcal{Q}$ since $R < \min\{I_1, I_2, I_3\}$. (2) The core encodes the data variation and internal dynamics of $\mathcal{Q}$ by the rank-R truncated SVD in each mode during the iteration.

Our second phase operator F_{cp} further decomposes the core $\mathcal{G}$ to compact, unique, yet effective representation, by directly converting a core of size $R \times R \times R$ into a quadruple of size $3R+1$. This step leads to huge dimensionality reduction of the core.

Given a query Q, in terms of the perturbed video $\mathcal{Q}'$, neglecting all but the first R components for each unfolded matrix in $\mathcal{Q}'$ in each mode is justified, since the perturbations introduced by E are only of the same order of magnitude as the $n - R$ smallest eigenvalues, whereas the first R components in Σ typically capture the underlying structure and primary internal dynamics [162].

In other words, using the truncated SVD in the Algorithm, it is guaranteed that, the effects of the perturbations (noise, Gaussian blurring, etc) will be largely filtered out during many rounds of iterations, because noise perturbations in E affect mostly the small eigenvalues in SVD. The primary underlying action dynamics are encoded by large eigenvalues, which are well-preserved during iterations. This is the underlying reason for the robustness of TP-Decomp. If, however, the perturbation in E is to intense that larger eigenvalues are also affected, then it will lead to degraded performance in action spotting.

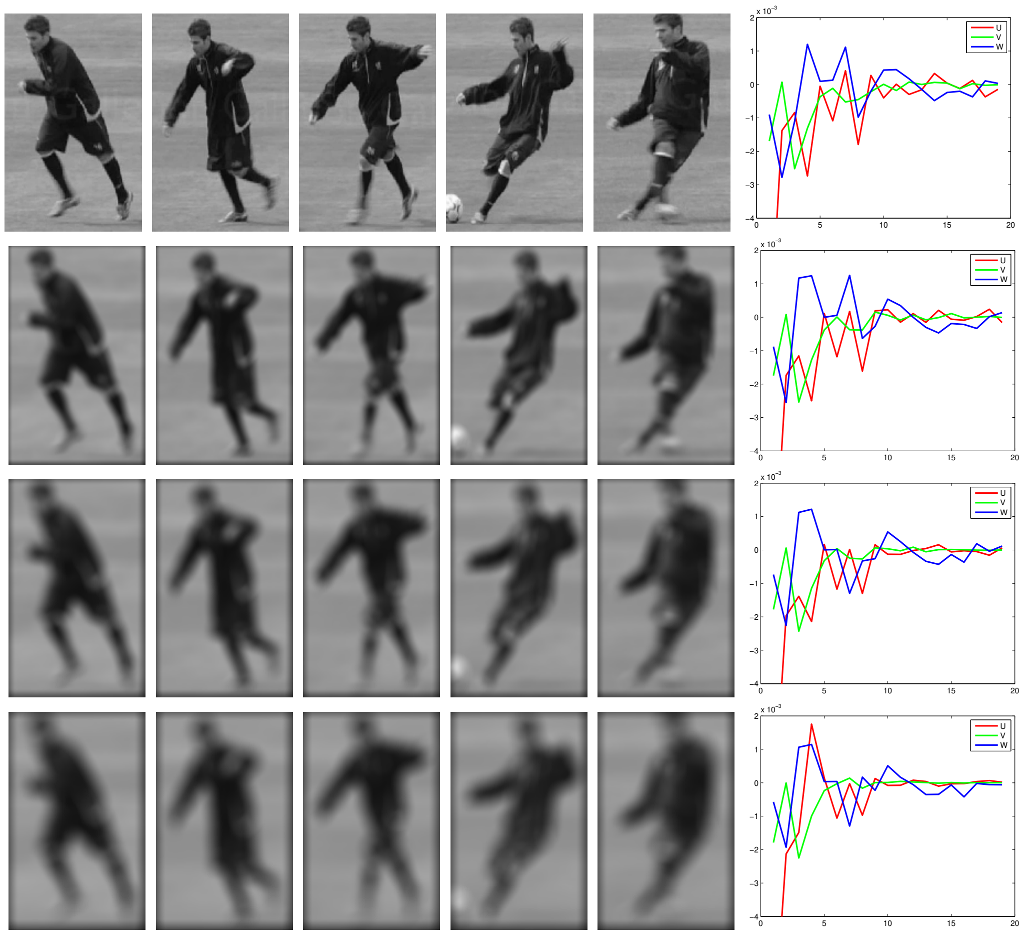


Figure 8.2: Comparison of the resulting quadruples for *kicking* actions under *Gaussian blurring* perturbation. The $2nd-4th$ rows correspond to different blur amount. All curves in the last column show the dynamics of the resulting tuple.

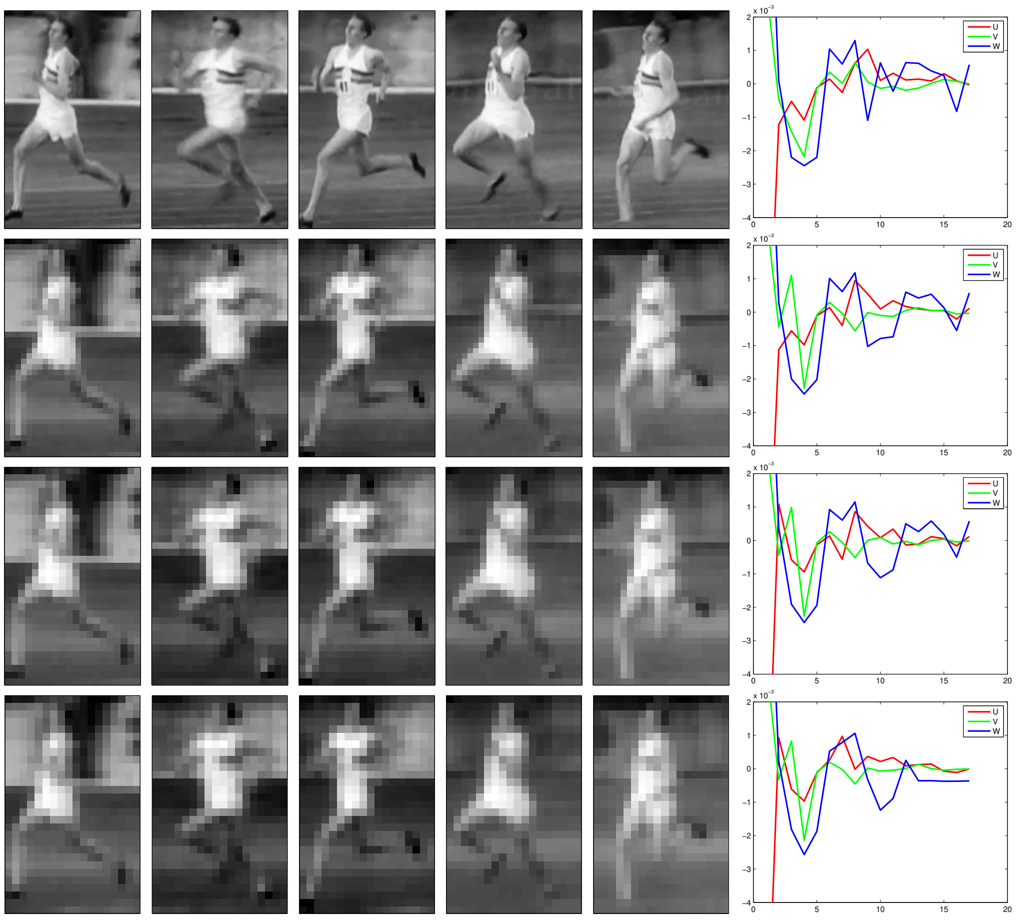


Figure 8.3: Comparison of the resulting quadruples for *running* actions under *down-sampling* perturbation. For scaling, the $6th - 8th$ rows correspond to scaling ratios of $10\%, 8\%$, and 7%, respectively. All curves in the last column show the dynamics of the resulting tuple

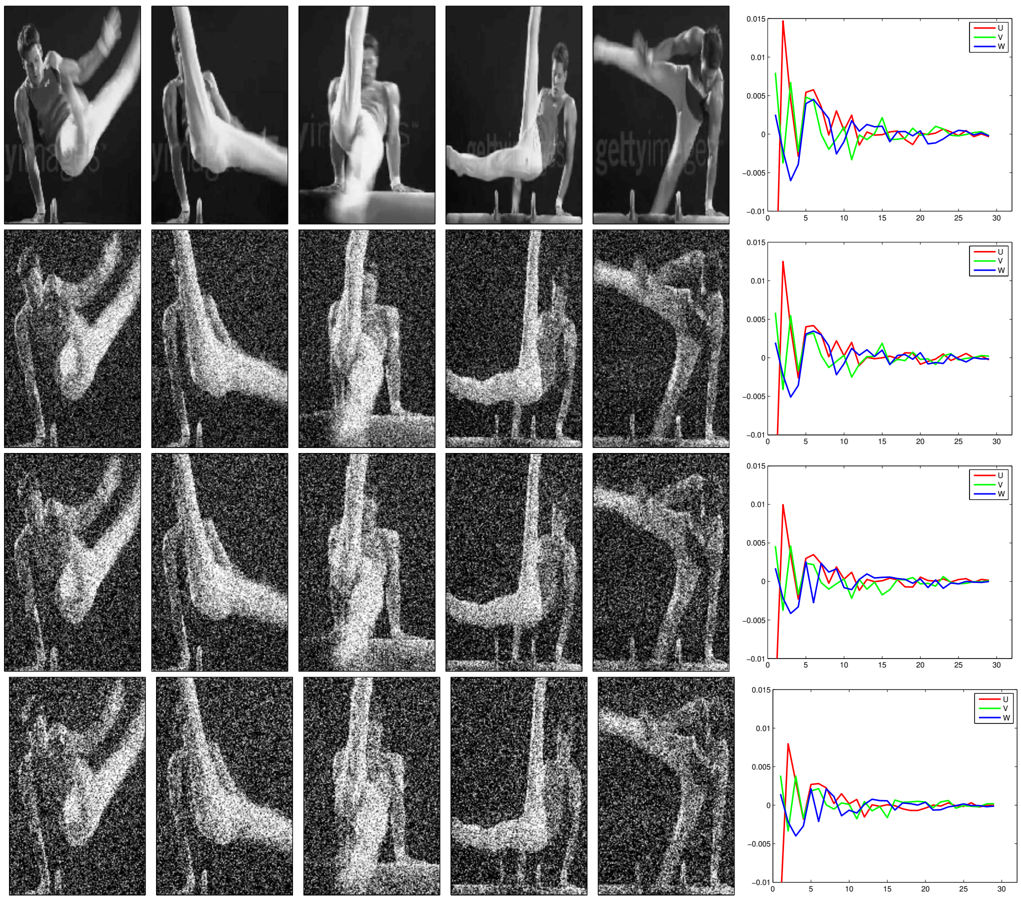


Figure 8.4: Comparison of the resulting quadruples for *benchswing* actions under *noise* perturbation. For noise perturbation, the $10th-12th$ rows correspond to noise variance values of $0.04, 0.07$, and 0.1, respectively. All curves in the last column show the dynamics of the resulting tuple

Properties

TP-Decomp has three good properties. We mainly focus on the robustness of TP-Decomp stated in Property 3.

Property 1: The TP-Decomp produces very compact representations. By contrast, the dimensionality is reduced from $I \times J \times K$ for $\mathcal{Q}$, to R^3 for $\mathcal{G}$, and finally to $3R$ for $\Xi_R^{\mathcal{Q}}$, namely

$$IJK \rightarrow R^3 \rightarrow 3R.$$

Property 2: The TP-Decomp is invariant to spatiotemporal dimension permutation for query video. Holistically treating video cuboid as tensor, we show no priority on spatial dimension over temporal dimension. This is in contrast to the approaches that handle spatial and temporal dimensions separately. By rotational uniqueness property [159], when permutating the spatiotemporal dimension of a query video, both the tensor core and the factored matrices remain invariant. This leads to stable quadruples in F_{cp}. Practically, this property is useful to spot the cubiods that are rotated, mirrored, or transposed along certain axis in the target video, leading to some level of view invariance.

Property 3: Under proper rank R, the TP-Decomp is robust to Gaussian blurring, downscaling, and noise perturbation. What happens when the target video is degraded by intensive blurring, low resolution down-sampling, or drastic noise perturbation? Many approaches will possibly fail to answer this due to feature dependency. For example, heavy Gaussian blurring removes high frequency intensities in image. It may affect optical flow extraction that highly depends on pixel gradients. The low resolution down-sampling removes intermediate pixels in frames. It may affects the HOG/HOF features whose statistical performance almost entirely depends on redundant subimage patches. The heavy noise perturbation adds random contaminations in video frames. It may affect the effective extraction of the silhouettes or STIP features in space-time. In addition, bag-of-features or bag-of-words paradigms, which rely on statistical clustering of sufficient descriptors, may also be hard to classify the codebooks under heavily contaminated situations.

For illustration purposes, we tested TP-Decomp on 3 action classes: *kicking* (KTH), *running* (UCF Sports), and *benchswing* (UCF Sports), as shown in Fig.8.2,8.3,8.4. We applied TP-Decomp on $\mathcal{Q}$ under rank $R = 20$, and plotted the resulting triplets as curves. For the Gaussian blurring degradation, under various smoothing kernels, we found that the resulting quadruple of *kicking* largely captures the action dynamics, with merely slight fluctuations in quadruple. For down-sampling, we downscaled the *running* video into various low resolution ones. Even when over 90% of the frame pixels were missing, the *running* dynamics was still well-preserved. For noise perturbation, we contaminated the *benchswing* with Gaussian noises. Even when the frames were heavily contaminated, the resulting triplets remained resilient to noise.

To fully investigate how the TP-Decomp performs for action spotting under heavy perturbation, we create a challenging dataset called Heavily Perturbed Video Arrays (HPVA). We will describe it in detail in experiment section.

Rank-TCP Descriptor

In this section, we first describle our motivation of why we use hierarchical representation of multiple core. Then, we elaborate the detailed procedure of how to build the pyramid representations.

The core dimension is solely decided by the rank R, yet the final quadruple is solely decided by the core. Hence, the rank R is the ultimate factor that affects both core and quadruple dynamics.

Algorithm 3: Rank-TCP descriptor construction

```
input : A query video $\mathcal{Q} \in \mathbb{R}^{I \times J \times K}$ where $I$ and $J$ correspond to spatial dimension of frames, and
        $K$ corresponds to temporal dimension
output: Tuple $\{\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}\}$.
Initialize $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$ to empty vectors;
for index $e$ from $2$ to $7$ do
    Rank $R = 2^e$;
    if $R \le \min\{I, J, K\}$ then
        Apply $F_{tk}$ operator to $\mathcal{Q}$ using rank $R$;
        Compute core $\mathcal{G}_R = \mathcal{Q} \times_1 A^T \times_2 B^T \times_3 C^T$ ;
        Apply $F_{cp}$ operator to $\mathcal{G}_R$;
        Compute quadruple $[\![\lambda_R; U_R, V_R, W_R]\!]$ ;
        Concatenate $U_R$ to the rear of $\boldsymbol{U}$ ;
        Concatenate $V_R$ to the rear of $\boldsymbol{V}$ ;
        Concatenate $W_R$ to the rear of $\boldsymbol{W}$ ;
    end
end
```

However, there is no prior that we can count on about which R leads to good representation for action dynamics in core/quadruple, because determining tensor rank is NP-hard [163], and the behavior of higher-order SVD is far beyond well-understood [33]. In the extreme cases, for instance, if R is too small ($R \le 3$), the core will be too small to capture sufficient information, whereas if R is too large, the F_{tk} operator may become undefined by *Lemma 1*; and the F_{cp} operator may not be unique by *Lemma 2*, as shown in the Appendix.

For this reason, we experimentally validate how ranks affect the TP-Decomp on a realistic big dataset called CCWebVideo, as stated in Section 5. Our results on five classes (2471 videos) show that, combining quadruples from multiple ranks outperform that of individual ranks (Fig.8.6). This is what motivates us to establish a Rank-based Tensor Core Pyramid to fully characterize an action.

Construction Procedure

We establish a new tensorial coarse-to-fine pyramid using multiple cores under multiple ranks. S-maller ranks correspond to cores of smaller size, lying above the larger ones in the pyramid. Under each candidate rank R, we apply the TP-Decomp on the query tensor $\mathcal{Q}$, yielding its corresponding quadruple $\Xi_R^{\mathcal{Q}} = [\![\lambda_R^{\mathcal{Q}}; U_R^{\mathcal{Q}}, V_R^{\mathcal{Q}}, W_R^{\mathcal{Q}}]\!]$.

Cores with lower ranks coarsely encode dynamics in $\mathcal{Q}$, whereas cores with higher ranks encode dynamics $\mathcal{Q}$ more precisely. But in terms of computational burden, the larger the R, the longer the TP-Decomp takes. Practically, we consider the following candidate ranks

$$R = \{4, 8, 16, 32, 64, 128\}.$$

This choice stems from practical concerns. All 6 ranks are unevenly distributed in a sense that we lean towards lower rank spectrum while not losing higher ranks. This choice can weight the computational burden of TP-Decomp.

For the 6 pyramid layers, there are totally $6 \times 3 = 18$ vectors generated. At the ith layer, the size of the quadruple is $3 * 2^i + 1$. Our final feature descriptor is the concatenation of the quadruple in all pyramid layers. We concatenate the U, V, W vectors at each level, yielding three long vectors $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$, namely

$$\begin{cases} \boldsymbol{U} = [U_4, U_8, ..., U_{128}] \\ \boldsymbol{V} = [V_4, V_8, ..., V_{128}] \\ \boldsymbol{W} = [W_4, W_8, ..., W_{128}]. \end{cases}$$

The triplet $\{\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}\}$, with each element of size $\sum_i 2^i = 252, i = [2, \cdots, 7]$, will be fed into the final template matching phase. If a candidate rank R is larger than the size of query tensor, namely

$R > \min\{I, J, K\}$, then we let R_{max} be the largest candidate rank, which is smaller than the query tensor size. Then each element in the triplet $\{\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}\}$ is of size $\sum_i 2^i, i = \{2, \cdots, log_2 R_{max}\}$.

Experimental results in CCWebVideo dataset shows that combining multiple ranks outperforms s-ingle rank representations (Fig.8.6). Intuitively, by combining multiple cores, the pyramid achieves a rich and redundant representation for video.

The procedure to build the Rank-TCP descriptor is shown in **Algorithm 1**. The storage space saved in TP-Decomp is from $I \times J \times K$ to $3R + 1$. For Rank-TCP descriptor, the space complexity of the final quadruple is $\sum_{e=2}^{\min\{I,J,K\}} (3 * 2^e + 1)$. In the case that $\min\{I, J, K\} \geq 128$, the total size for the pyramid is 756.

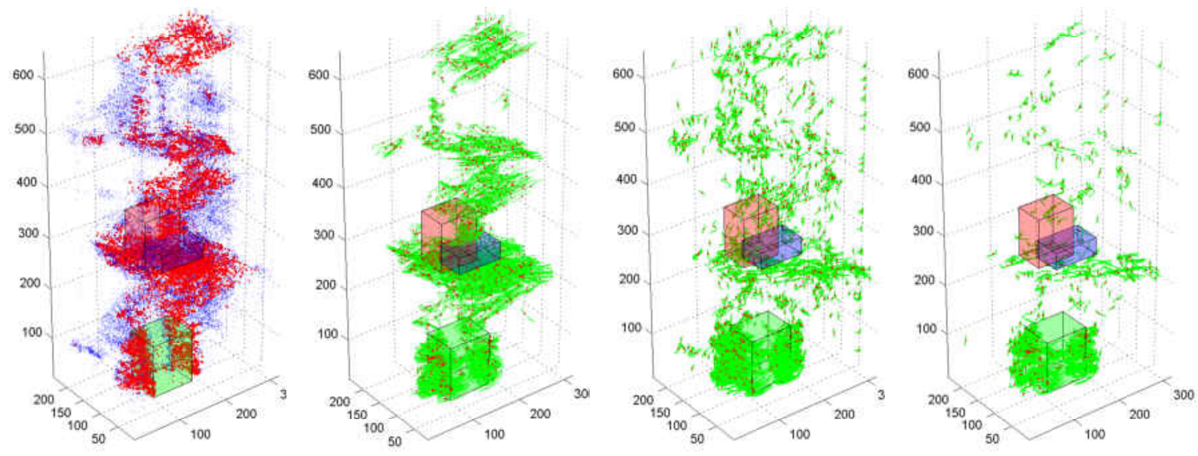


Figure 8.5: $(1st)$ The point cloud formed by the dense trajectories extracted from a video of M-SR I dataset. Red points map to top trajectories by length. $(2nd)$ Top trajectories thresholded by the mean length of all trajectories. $(3rd)$ Top trajectories thresholded by the mean unsigned total curvature of all trajectories. $(4th)$ Our filtered trajectories. The red dots denote the average locations of trajectories. The red, green, and blue boxes denote the ground truth volume for *clapping, waving*, and *boxing*, respectively (zoom in for better view)

Boosting Strategies for matching

Although costly, template matching can avoid problematic preprocessing operations in localization, tracking, and segmentation [164]. The computational burden can be reduced by various techniques such as branch-and-bound [165] or voting algorithms [166].

In our framework, two strategies are adopted to boost the template matching: (1) to reduce the space complexity, we employ the dense trajectory in [167], and prune the search space using cues derived from trajectories. (2) to reduce the time complexity, we use a coarse-to-fine strategy assisted by the Rank-TCP descriptor.

Trajectory-assisted Space Reduction

Motion is the most reliable and informative source of information for action analysis [168]. The method using dense trajectories to compute local descriptor in [168] is one of the state-of-the-art approaches for action recognition. The work in [167] further improves the work of [168] and shows high reliability and robustness in handling motion.

Given a target video, we first extract its dense trajectories by the method in [167] using default parameters. Instead of being used in codebooks for k-means clustering as in [168, 167], the dense trajectories are treated as reliable cues to prune the huge matching space. Our underlying intuition of this trajectory-assisted matching agrees with that of [169]: despite the huge number of candidate cuboids needing search, only very few contain the true motion of interest (MOI). Instead of exhaustively evaluating them all, we target only the best few.

Inspecting thoroughly, we observe the motion annotated by trajectories can be roughly divided into 8 categories:

1. (a) limb movement around MOI, such as the hands in handwaving, the head and feet in jumping, etc.
2. (b) camera motion, such as zoom, pan, translation, vibration, etc.
3. (c) non-MOI motion inside blobs.
4. (d) patch motion inside MOI, such as motion caused by appearance (e.g. cloth) in bending or jumping, non-limb body region motion, etc.
5. (e) background clutter, such as cars on street, crowd, remote or close pedestrians, etc.
6. (f) foreground motion, such as a pedestrian moving from left to right across the scene, etc.

7. (g) local motion propagation, such as a fountain in background, etc.

8. (h) regional random movement of noisy patches.

Out of those 8 categories, we are especially interested in the first. The category (b) is effectively canceled out by method in [167], and few trajectories are responsible for it. Further observation reveals that, short trajectories stem from $C_{short} = \{c, d, e\}$, long trajectories stem from $C_{long} = \{a, d, e, f\}$, curved trajectories stem from $C_{curved} = \{a, e, g, h\}$, and relatively straight trajectories stem from $C_{straight} = \{c, d, e, f\}$. Notice that

$$C_{long} \bigcap C_{curved} = \{a\}.$$

This observation suggests that, a long curved trajectory indicates a possible MOI around this trajectory. This inspires us that, the length and the curvature of trajectory is possibly a very good cue to spot the MOI.

In practice, given a trajectory denoted by $T = \{p_i\}, i = 1, ..., n$ and a neighborhood m, we calculate its tangent orientation based discrete curvature [170] at point p_i by

$$k(p_i) = \frac{\angle\left(p_{i-m}p_i, p_ip_{i+m}\right)}{|p_{i-m}p_i| + |p_ip_{i+m}|}.$$

Since curvature is signed, the total curvature of a S-shaped trajectory is possibly zero. We then use unsigned total curvature $\kappa = \sum |k(p_i)|$ to reflect the total “bendness” of a trajectory. Along with its total length $l = \sum |p_{i+1} - p_i|$, we define

$$\varsigma = \kappa l$$

as our metric to measure the trajectories, and define the threshold as $\theta = mean(\varsigma_j), j = 1, ..., t$, where t is the total number of extracted trajectories. This metric is unsupervised, simple, yet powerful enough to prune a significant amount of irrelevant trajectories, as shown in Fig.8.5. The mean position of all survived trajectories are used as search locations in template matching.

Actually, our metric is somehow connected with the flow field dynamics. Let us call the motion annotated by a dense trajectory as “motionlet”, the smallest motion unit inside a spatialtemporal volume. Each motionlet involves a transmission of energy in the form of trajectory flow. If we regard the spatialtemporal video volume as a fluid energy field, then long trajectories reflect those motionlets that traverse long distances, and carry more fluid energy than that of short trajectories. But the energy itself does not necessarily suggest a true MOI because noise or clutters also carry energy.

On the other hand, curved trajectories correspond to the motionlets that carry curl (or vorticity) information. The curl around the flow field suggests the rotation, density, or magnitude of the flow. Given the same flow energy, the curl can provide a cue of how the energy flows, in which direction, and in what speed. Our treatment of the curved trajectories using unsigned total curvature is a natural quantization for the curl of flow field, because total “bendness” of a trajectory provides a good way to simulate its curl.

To illustrate the effectiveness of this metric, we plot all the trajectory cloud information for 16 videos in MSR I dataset, as shown in Fig.8.7,8.8,8.9,8.10.

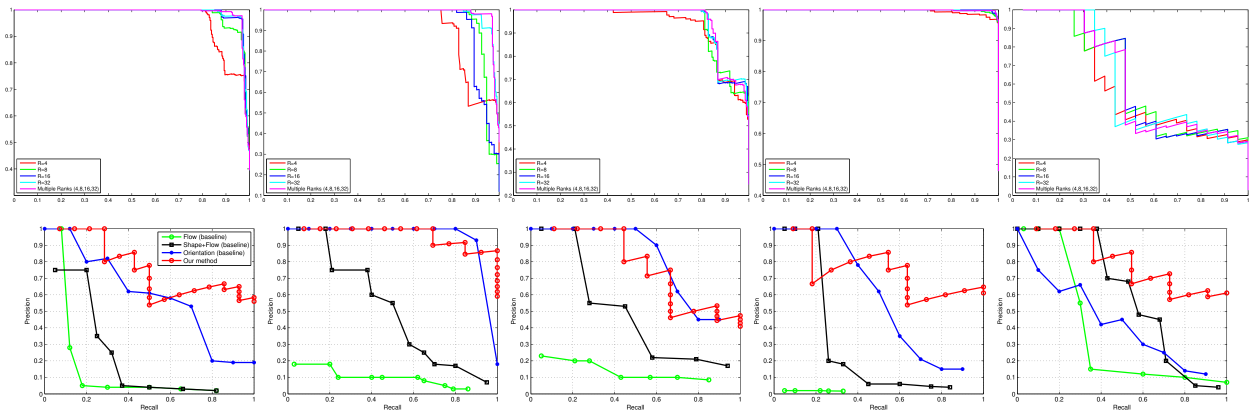


Figure 8.6: The Precision-Recall Curve for CCWebVideo ($1st$ row) and CMU Action dataset ($2nd$ row). For CCWebVideo dataset, each figure denotes the comparison results between using a single rank and multiple ranks. From left to right, the action classes are: *The lion sleep tonight, Evolution of dance, Fold shirt, I will survive Jesus*, and *Little superstar*. Note that our PR curves for *Little superstar* is more jagged than others because there is only 59 ground truths available out of 377 videos (84% are outliers). For CMU Action, we compare our method with 3 baseline methods, namely, the holistic flow [95], the part-based shape plus flow [164], and the spacetime oriented energy measurements [87]. The 5 subfigures correspond to *jumping jacks*, *pickup*, *push button*, *one-handed wave*, and *two-handed wave*, respectively. Red curves correspond to our proposed method (better be viewed by zooming in and in color)

Coarse-to-fine Matching

The trajectory-assisted strategy prunes a considerable amount of search locations. Let the set of survived search locations be $L = \{L_i\}, i = 1..n$. Due to the coarse-to-fine structure in Rank-TCP descriptor, we can further accelerate the matching in an iterative coarse-to-fine manner.

Since the descriptor with lower rank coarsely represent the candidate volume, we first match all locations in L using the lowest rank $R = 4$. Within the resulting 3D score map, we set a loose threshold $\theta_{R=4}$, filtrating all matched candidate cuboids below this threshold. For the survived candidates $C_{R=4}$, we apply the second round of matching under a higher rank $R = 8$. Under a proper second threshold $\theta_{R=8}$, a smaller portion of the survived candidates from last round, denoted by $C_{R=8}$, survives, and participates in the next level where $R = 16$. We iterate this process m times ($m <= 6$). Under lower ranks ($R = \{4, 8\}$), there could be numerous false alarms, because lower ranks correspond to far more cuboids needing match. As rank grows ($R = \{16, 32, 64, 128\}$), many false alarms will be filtered out. To ensure true positives (locations around MOI) are not falsely filtered out, the thresholds under lower ranks, i.e., $\theta_{R=4}$ and $\theta_{R=8}$, will not be set too tight; but as rank grows, more strict thresholds will be enforced to eliminate false positives. In practice, this strategy can reduce $15\% - 30\%$ matching time.

Matching Measure

To define the similarity measure between the query tensor $\mathcal{Q}$ and candidate sub-volume $\mathcal{V}$ in search video $\mathcal{S}$, we extract the Rank-TCP descriptor out of both $\mathcal{Q}$ and $\mathcal{V}$, resulting in two quadruples

$$\begin{cases} \Xi_R^{\mathcal{Q}} = [\![\lambda_R^{\mathcal{Q}}; U_R^{\mathcal{Q}}, V_R^{\mathcal{Q}}, W_R^{\mathcal{Q}}]\!] \\ \Xi_R^{\mathcal{V}} = [\![\lambda_R^{\mathcal{V}}; U_R^{\mathcal{V}}, V_R^{\mathcal{V}}, W_R^{\mathcal{V}}]\!]. \end{cases}$$

We define the matching measure function as

$$D(\Xi_R^{\mathcal{Q}}, \Xi_R^{\mathcal{V}}) = \phi(U_R^{\mathcal{Q}}, U_R^{\mathcal{V}}) + \phi(V_R^{\mathcal{Q}}, V_R^{\mathcal{V}}) + \phi(W_R^{\mathcal{Q}}, W_R^{\mathcal{V}}),$$

where the function ϕ is the Euclidean distance measure function.

We define the similarity score between $\mathcal{Q}$ and $\mathcal{V}$ under rank R as

$$\xi_R = -\log(D(\Xi_R^{\mathcal{Q}}, \Xi_R^{\mathcal{V}})).$$

The score is inversely proportional to the measure function D. A high score means the two tensors are similar while a low score implies dissimilarity. The peaks of the similarity score across the 3D score volume indicates potential match locations.

Experimental Evaluation

We extensively experiment on 5 benchmarks. For CCWebVideo, the retrieval requires no window sliding, since we holistically match query and target. For the remaining 4 datasets, following [164], we match at a single scale instead of multiple scales since actor sizes are stable. We use exhaustive search as our baseline denoted by "RTCP-Exha", and the trajectory-assisted matching is denoted by "RTCP-Traj".

Table 8.1: Average Precision (AP) comparison for CCWebVideo, CMU, MSR I, and MSR II datasets

CCWebVideo	"The lion sleep tonight"	"Evolution of dance"	"Fold shirt"	"I will survive Jesus"	"Little superstar"
Total video #	792	483	436	416	377
Outlier #	458	361	253	29	318
Outlier %	58%	75%	58%	7%	84%
Wu [171]	0.95	0.90	0.86	0.88	0.78
Song [172]	0.94	0.79	0.92	0.94	0.94
RTCP-Traj	**0.97**	**0.94**	**0.93**	**0.98**	**0.81**

Dataset	CMU					MSR I			MSR II		
Actions	jjacks	pickup	pushbutton	1-h wave	2-h wave	clapping	waving	boxing	clapping	waving	boxing
Video instance #	16	20	14	18	34	14	24	25	51	71	81
Deerpanis [87]	0.50	0.95	0.80	0.55	0.48	-	-	-	-	-	-
Ke [164]	0.30	0.45	0.50	0.40	0.60	-	-	-	-	-	-
Yuan [165]	-	-	-	-	0.75	-	-	-	-	-	-
Yu [173]	-	-	-	-	-	-	-	-	0.30	0.84	0.60
Boyraz [174]	-	-	-	-	-	-	-	-	0.35	0.40	0.50
RTCP-Exha (baseline)	0.55	0.59	0.35	0.40	0.57	0.19	0.43	0.25	-	-	-
RTCP-Traj	**0.75**	**0.92**	**0.74**	**0.69**	**0.78**	**0.63**	**0.82**	**0.70**	**0.57**	**0.73**	**0.64**

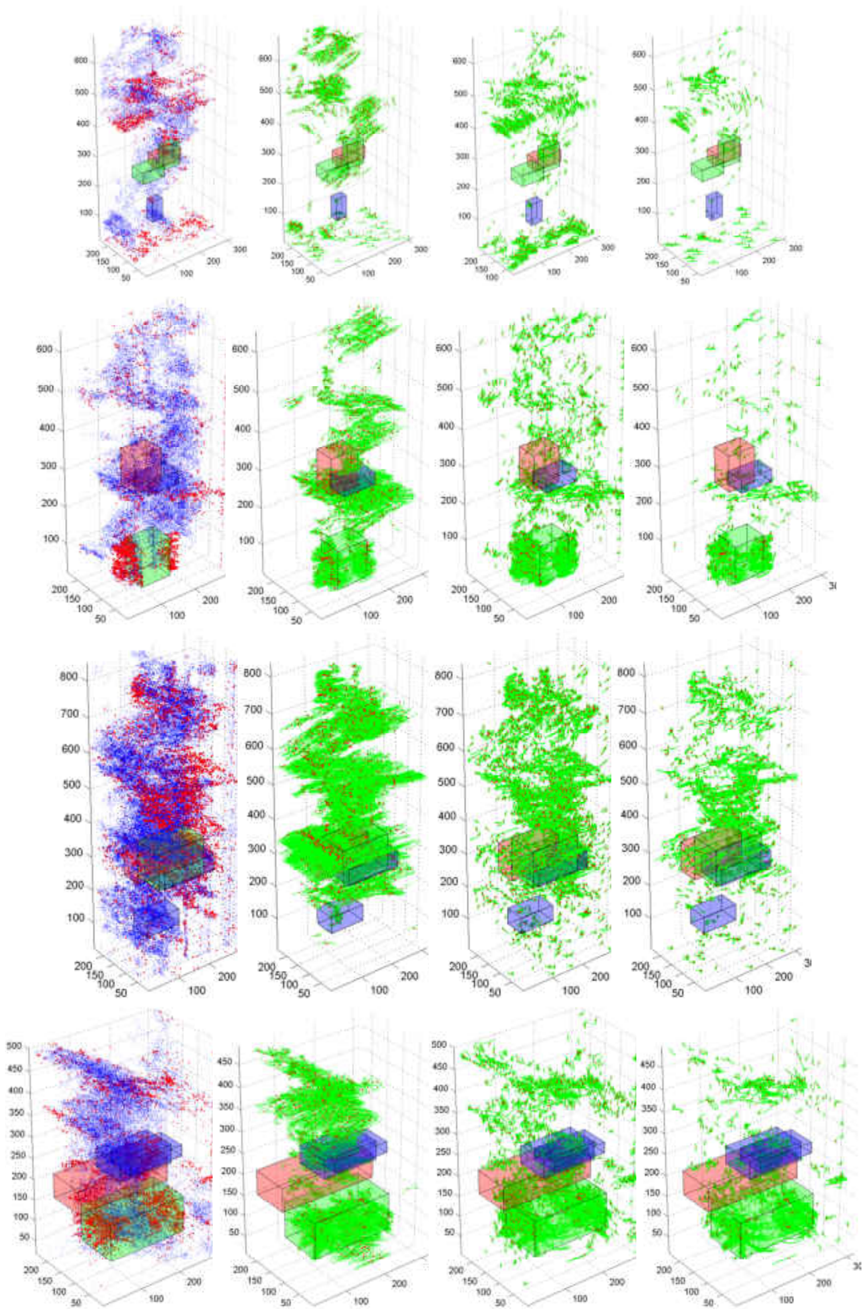


Figure 8.7: Video numbered 1 to 4 of MSR I dataset. ($1st$ column) The point cloud formed by the dense trajectories. Red points map to top trajectories filtered by our metric. ($2nd$ column) Top trajectories thresholded by the mean length of all trajectories. ($3rd$ column) Top trajectories thresholded by the mean unsigned total curvature of all trajectories. ($4th$ column) Our filtered trajectories. The red dots denote the average locations of trajectories. The red, green, and blue boxes denote the ground truth volume for *clapping, waving*, and *boxing*, respectively (zoom in for better view)

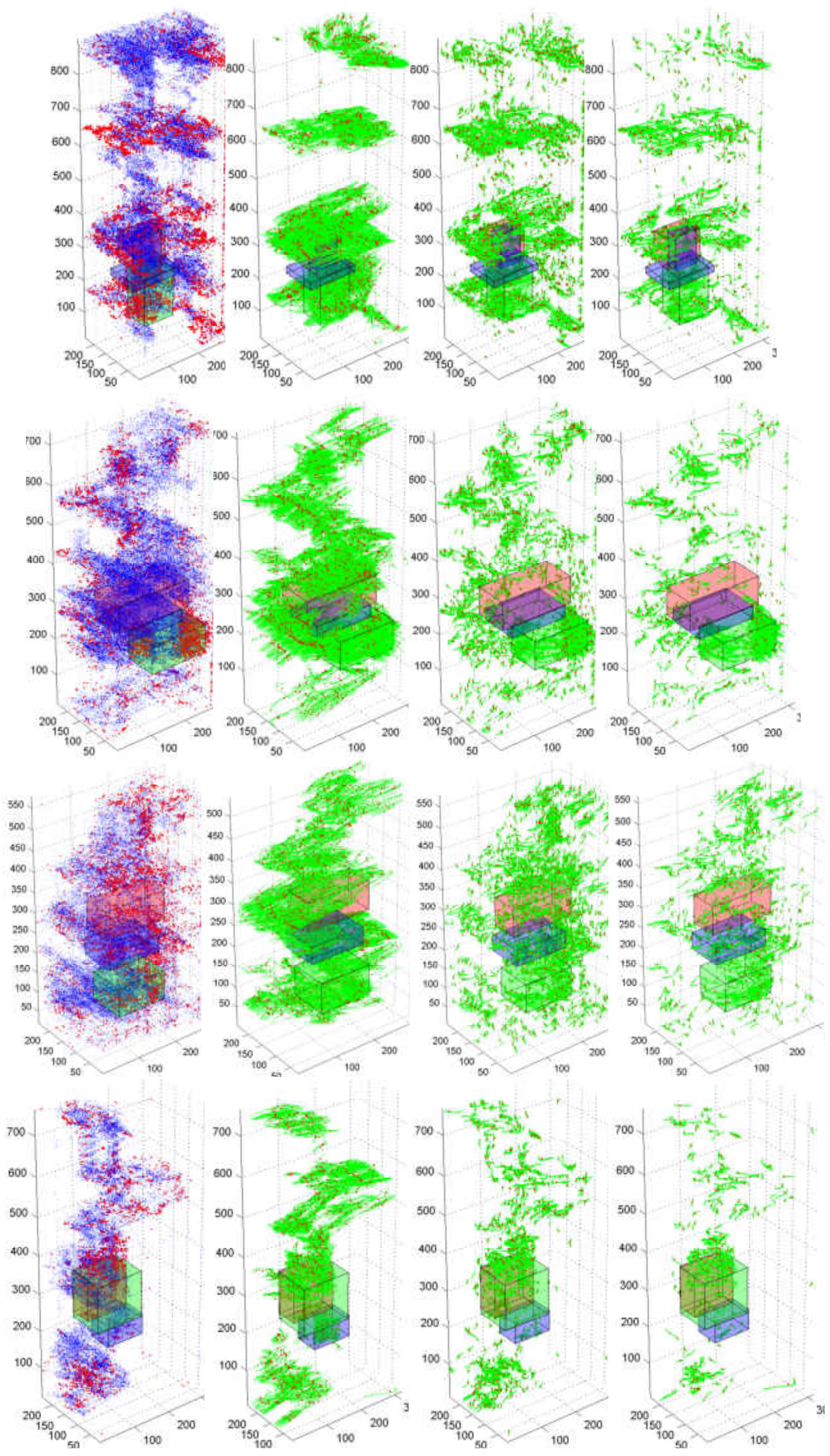


Figure 8.8: (Cont.) Video numbered 5 to 8 of MSR I dataset

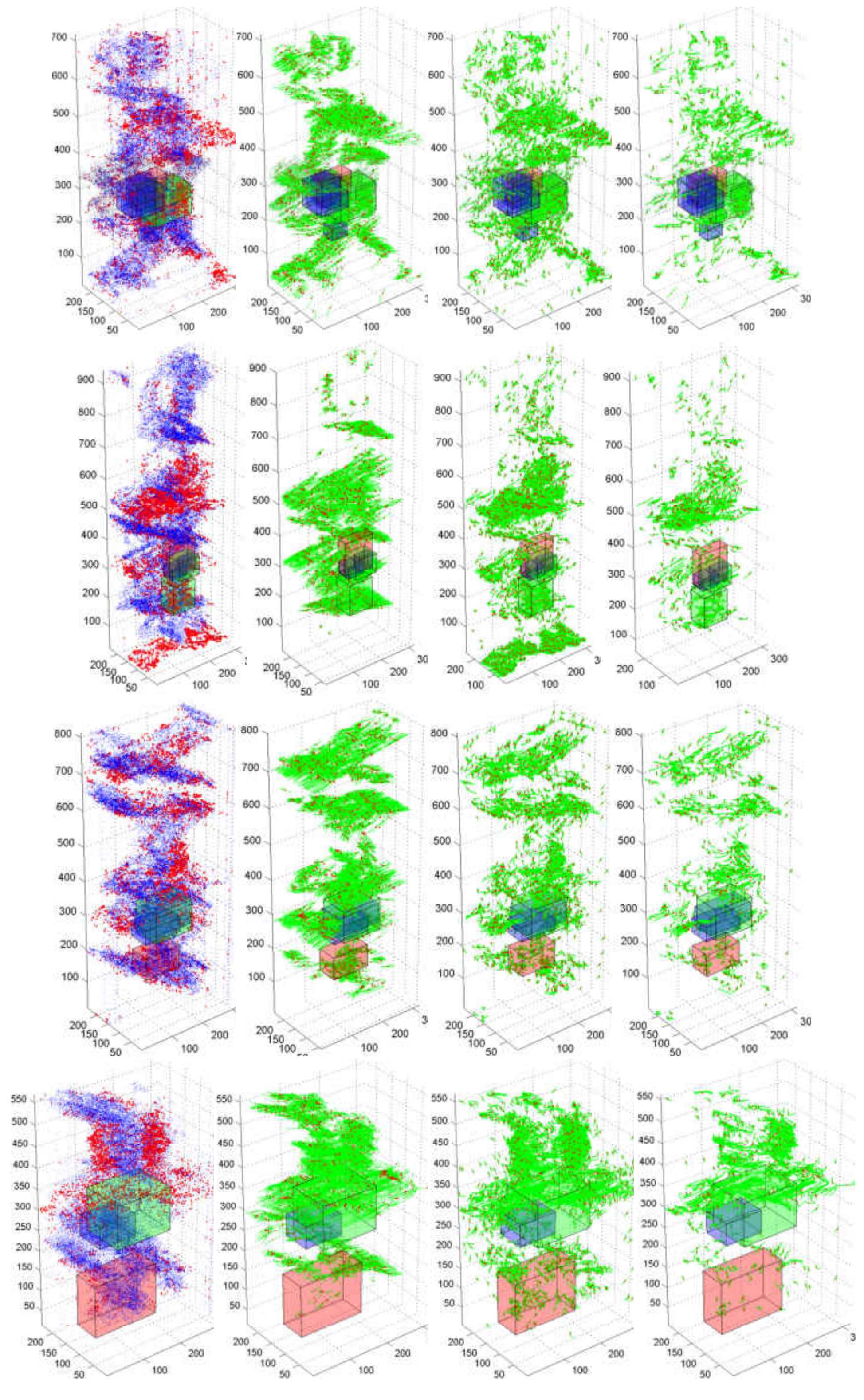


Figure 8.9: (Cont.) Video numbered 9 to 12 of MSR I dataset

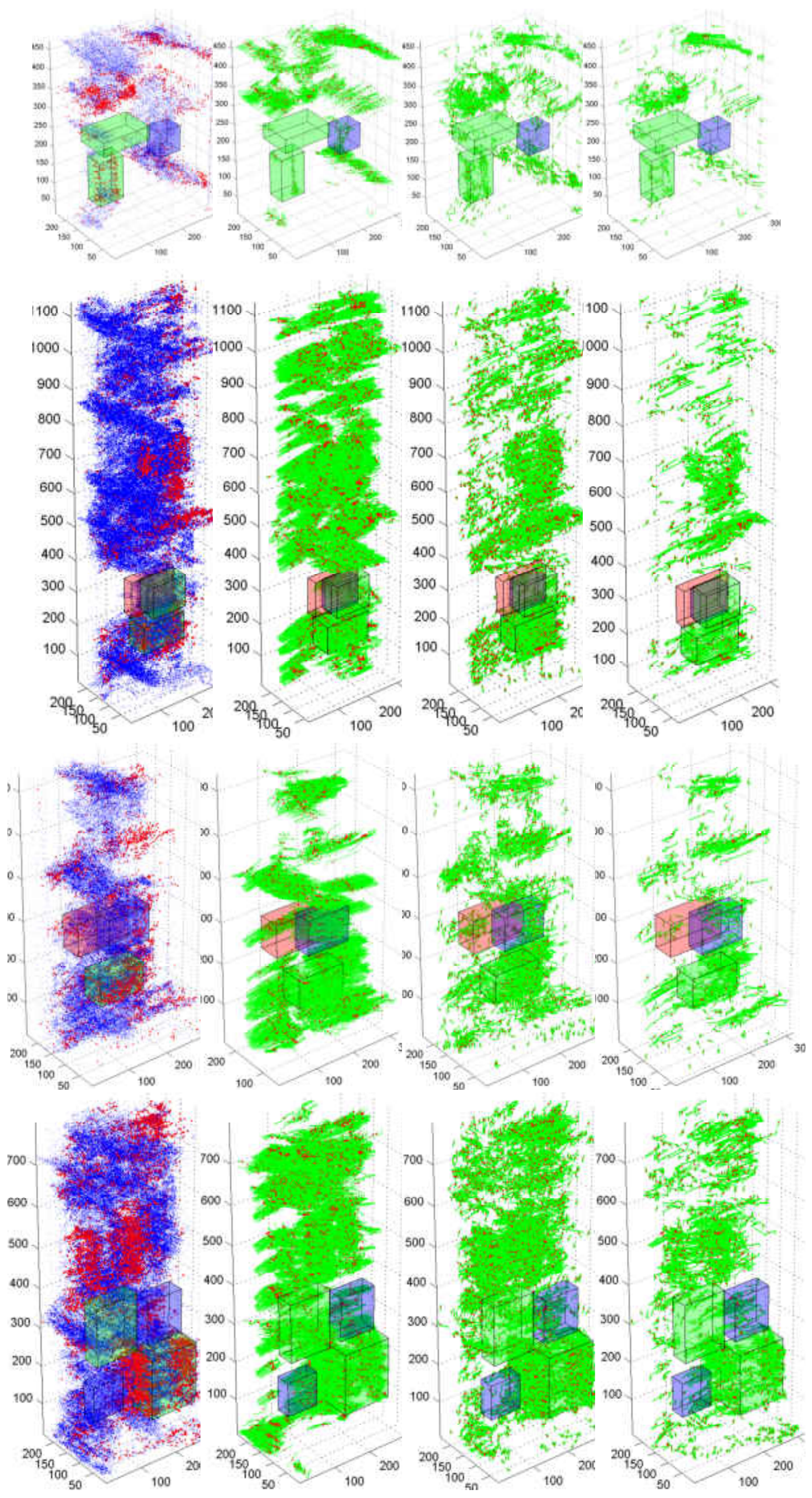


Figure 8.10: (Cont.) Video numbered 13 to 16 of MSR I dataset

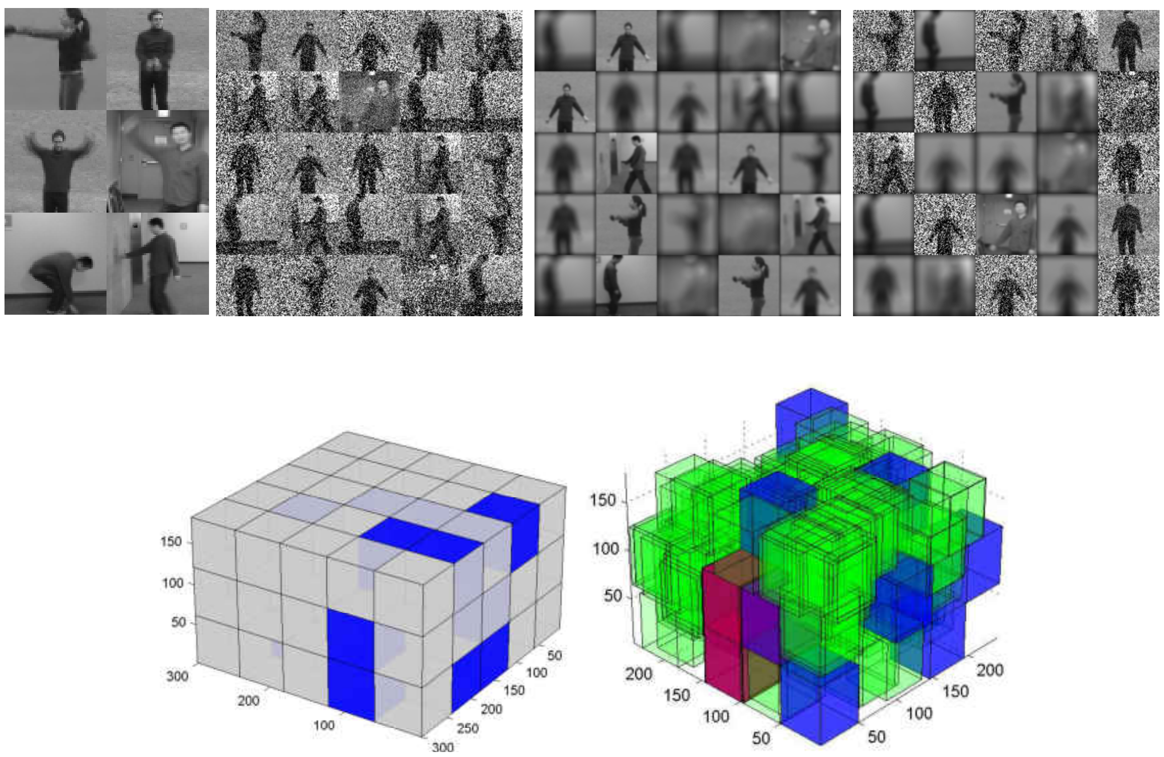


Figure 8.11: Our HPVA dataset. $(1st)$ The 6 seed videos. We use *boxing, clapping, waving* as queries, and the rest as outliers. $(2nd)$ One frame of a video grid perturbed by random heavy noise; $(3rd)$ One frame of a video grid affected by random degree of blurring; $(4th)$ A random mixture of noise and bluring perturbation; $(5th)$ The $3D$ tiled cuboid grid of size $300 \times 300 \times 180$. Each cell is filled with a random seed video. $(6th)$ The exhaustive matching on a randomly generated grid. Blue/red cuboids denote the ground truth position of queries (eg. *boxing, clapping, waving*), while green cuboids denote sliding windows

CCWebVideo Dataset

This huge dataset [171] contains 24 set of 13129 videos. For each seed video, there are hundreds of different versions, with considerable amount of intra-class variation (such as photometric variations, lighting change, unrelated frames, text overlay, etc.). We use this dataset to verify how our Rank-TCP descriptor performs given huge intra-class variance. We select 5 classes, totally 2471 videos. Since action spotting is analogous to video retrieval, we treat the seed as query video, and spot all videos that matche the seed. We test 5 descriptors, of which 4 use a single rank $(4, 8, 16, 32)$, and 1 combines these multiple ranks.

Table 8.2: Average precision for HPVA results

HPVA	Noise	Blur	Mixed
RTCP-Exha (baseline)	0.44	0.37	0.21

Table 8.3: Time (hours) spent in 3 template matching scenarios for 5 benchmarks. "c2f" means using coarse-to-fine matching strategy

	CCWebVideo	CMU	MSR I	MSR II	HPVA
RTCP-Exha (baseline)	23	18	15	n/a	0.8
RTCP-Traj wo/ c2f	n/a	2.0	2.3	7.2	n/a
RTCP-Traj w/ c2f	n/a	1.2	1.5	6.3	n/a

CMU Action Spotting Dataset

The CMU action dataset [164] consists 5 action classes (48 videos, 6 subjects): *jumping jacks*, *pickup*, *push button*, *one-handed wave*, and *two-handed wave*. Videos are recorded in crowded environments such as streets, restaurants, and bus stop. We compare with 3 baseline methods:

the holistic flow [95], the part-based shape plus flow [164], and the space-time oriented energy measurements [87].

MSR Action Dataset I and II

The **MSR Action Dataset I** [165] contains 3 classes (16 video sequences, 10 subjects). The video sequences and has in total 63 actions: 14 hand clapping, 24 hand waving, and 25 boxing. Each sequence contains multiple types of actions. There are both indoor and outdoor scenes with clutter and moving backgrounds. The **MSR Action Dataset II** is an extended version of the MSR I. It contains 3 classes (54 video sequences): hand waving, hand clapping, and boxing. In total 203 action instances. Since instances of a query often do not begin and end at the same time span [164], we preserve only the best in each frame.

Heavily Perturbed Video Arrays (HPVA) Dataset

We created a new challenging dataset to test the robustness of our framework under heavily perturbed situations. We choose 6 seed videos (*boxing, clapping, waving,* 1-*handed wave, pickup, pushbutton*), each of size $60 \times 60 \times 60$. We form a $300 \times 300 \times 180$ 3D *tiled cuboid grid* in space-time, as shown in Fig.8.11 (Each cell contains a random seed). Since down-sampling can be regarded as a variant of Gaussian blurring, we apply one of the two operators, F_{noise} or F_{blur}, on seeds. The degree of F_{noise} and F_{blur} are randomly specified, and which cell maps to which seed is also randomly generated. We created 3 such grids in HPVA, of which 2 are enforced by F_{noise} and F_{blur}, respectively, and the other one by mixing such 2 operations. In other words, our benchmark has three videos ($300 \times 300 \times 180$). The first is under randomly perturbed noise. The second is degraded by random intensive blurring. The last is a random mixture of noise and blurring. We choose *boxing, clapping, waving* as queries and the rest as outliers.

Analysis

Average Precision (AP) For CCWebVideo, despite the large amount of outliers and intra-class variations, the PR curves of our spotted (or retrieved) results show remarkably high precisions. The precision of higher ranks (32) is often higher than lower ranks $(4, 8)$. The combination shows highest AP amongst all 5 cases. For 4 out of 5 seeds, our AP outperform the state-of-the-art (Table 8.1). The results for 3 out of 5 actions in CMU data outperforms previous works. Higher ranks indeed lead to richer representations, and combining multiple ranks is better than a single one.

Using RTCP-Exha, our AP on CMU data outperform 2 out of 5 actions. Using RTCP-Exha, our AP improves for all 5 actions. Especially, we observe that AP of *pickup* action increases 35% compared to baseline. This action involved a large body motion when the person's upper body approaches to the ground. By inspecting the filtered trajectories of *pickup*, we observed that a considerable amount of irrelevant trajectories were filtered out, leading to largely reduced number of false alarms.

The MSR I and MSR II are challenging because of their long durations and dynamic backgrounds. Some previous works did not provide AP per action, so they are unavailable in Table 8.1. Note that, we observe the hands of the *clapping* in some videos often overlap the inner region of human body, and lead to some level of intensity confusions. Thus for *clapping*, our AP is relatively low using exhaustive matching without pruning. Because of the matching time (> 40 hours), the exhaustive matching results for MSR II is unavailable in our test. Our reported trajectory-assisted matching for MSR II took about 4.3 hours with coarse-to-fine boosting (Table 8.3). Overall, our AP is comparable to that of [173]. Across all test scenarios, RTCP-Traj largely outperforms RTCP-Exha, showing that our trajectory-assisted pruning is indeed effective.

We perform only exhaustive matching on HPVA, because the extracted trajectories on HPVA can hardly represent motion dynamics due to heavy contamination in videos. The results are shown in Table 8.2. Due to the robustness of TP-Decomp stated in Section 2, our method still spotted many true positives, regardless of the randomness and contamination introduced in HPVA.

Matching Time In practice, we parallelized template matching on a cluster using multiple cores. We compare the time spent in matching in Table 8.3. Along with the AP analyzed above, our trajectory-assisted space reduction is an effective way to reduce search space and boost the spotting precision. The adaptation of coarse-to-fine (c2f) can reduce around $15\% - 30\%$ of the matching time on average compared to that of without c2f.

CHAPTER 9: CONCLUSIONS AND FUTURE WORK

Conclusions

We use the recurrence plot theory to define a tensor representation of the dynamics of an action in video data, which we refer to as a Joint Self-Similarity Volume. We show that for the purpose of action recognition the Joint SSV is sparse. In other words, it can be for the most part characterized by its rank-1 subspace representation. Therefore, by exploiting this sparseness, we reduce the high-dimensional recognition problem to a linear low-dimensional matching problem in a rank-1 subspace, without compromising our recognition accuracy. A particular feature of our approach is that it leads to a generic solution to this problem in the sense that our solution is independent of the type of input features, i.e. tracked points in a motion capture dataset, manually marked points, automatically extracted silhouettes, Histogram of Gradient (HoG) feature vectors, optical flow, etc.

For reducing the dimensionality of the Joint SSV, we introduced a new rank-1 tensor approximation algorithm that relies on an alternating least squares approach to find the optimal rank-1 decomposition. We demonstrate that in the case of Joint SSV, the proposed decomposition largely preserve the salient characteristics of the scene dynamics. On the other hand, it leads to significant saving in both memory and computational time, since only a collection of rank-1 tensors is needed as the reference database for action class representation and matching. The algorithm also allows one to recognize actions without explicitly aligning the videos in temporal dimension. To validate our method, we devised three types of volume construction schemes, and performed experiments on five different public datasets.

For action recognition in incomplete videos, we study to what extent sparse unlabeled data can affect the action classification problem, and we propose a unified framework for handling the sparsity problem. We experimented with an exhaustive set of possible situations, and recover the complete version of the sparse videos by a CP-based decomposition algorithm. Then, with the help of a semi-supervised learning, we demonstrate the possibility of classifying actions under high or even severe sparsity. Our test results on two benchmarks show that it is feasible to include incomplete videos rather than simply discarding them. We plan to create benchmarks from other challenging datasets to better scrutinize this problem that has a direct application in the popular area of compressive sensing.

For motion retrieval, our technique transforms the initial motion sequences into three vectors that are discriminating the motion manifold based on its dynamics as a whole rather than on frame-by-frame basis. The advantage of this approach lies in the following aspects. *First*, no alignment for individual poses is required when matching the query pose with the one in the reference database. *Second*, for comparing poses, which may be represented by a large matrix when the motion time is large, our generated vectors are merely three low dimensional vectors. This leads to huge saving in motion sequence storage and processing. *Third*, as discussed earlier, our method resolves motion sequences while preserving non-linear characteristics, and avoids confusions due to inter-class similarity, or intra-class dissimilarity. This chapter has thus presented an efficient method for retrieval of motion capture data that in particular solves the problem in cases when other methods typically get confused. One main contribution of this work is a new framework for motion retrieval without explicit time alignment compared to the traditional methods. Also, compared with the traditional DTW-based approaches, our technique only requires storing a collection of compact reference vectors generated by our iterative algorithm rather than complete sequences of motion. Of course, our method can be adopted as an efficient complement to the DTW-based approaches in scenarios where exact frame-wise alignment is also required.

For action spotting, we propose a framework that is feature-independent and does not rely on human localization, segmentation, or frame-wise tracking. We start by treating all involved video cuboids as multilinear tensors, and theoretically and experimentally show that, the internal dynamics of an action can be effectively encoded by our new Two-Phase Decomposition technique. We further verify that combining multiple cores under multiple ranks lead to enhanced performance compared to a single rank. This inspires us to devise hierarchical rank-based descriptors to fully represent action dynamics. We boost the costly template matching by two strategies, which reduce the size of of search space and the matching time. The experimental results on 5 benchmarks, including our newly-created HPVA dataset, show that our framework is very effective in spotting actions under various challenging conditions. We conclude that, (1) Our TP-Decomp method yields compact, discriminative, and robust features. (2) Rank-TCP is effective in yielding richer and reliable representations. (3) Filtering out irrelevant outliers in matching volume, targeting only the best few, indeed leads to largely boosted speed and enhanced precisions. (4) A robust descriptor that preserves action dynamics is critical to spot actions under heavily perturbed situations.

Future Work

In our third major topic, namely, the action spotting, we have three further observations. (1) We observe that, in our action spotting framework, although the trajectory-assisted boosting strategy filters out a large amount of irrelevant search locations, some outliers still remain in the survived locations. (2) For some actions, the motion of interest (MOI) tends to be clustered, while for other actions, they might distribute sparsely in the search volume. (3) By comparing the center of MOI (denoted by C_{center}) with the centroids of the ground truth action volumes (denoted by $C_{centroid}$), we noticed that, sometimes there are a certain amount of displacements between those two. This displacement affects the interaction ratio if we want to setup a threshold between candidate volumes and the ground truths.

We believe that, finding the corresponding explanation or solution for the above three observations, has significance if we want to further enhance our action spotting performance. For this reason, in our future work, we plan to tackle them as follows.

First, we plan to derive more intuitive yet useful cues from the trajectories. For example, the curl (or vorticity) might be a more accurate quantity compared to our “total unsigned curvature”. Indeed, if we can filter out more irrelevant locations while preserving true positives, the precision can be enhanced because the true negatives are reduced.

Second, we will explore the connection between the sparseness and action dynamics in terms of MOI.

Third, we will explore the possibility of using Voronoi Graph (VG) based technique to boost the template matching in action spotting. The reason is that, VG not only has good properties in describing the 3D structure of a point cloud, it also has striking connection with fluid (or particle) flow dynamics in a 3D dynamic system. which shares common characteristics with the problem of video flow and dynamics in action spotting.

APPENDIX A: MULTILINEAR TENSOR FUNDAMENTALS

Multilinear tensor fundamentals

We summarize some necessary preliminaries for multilinear algebra in this section. A tensor is a higher order generalization of a vector and a matrix. An N-mode tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is *rank one* if it can be written as the outer product of N vectors, i.e.,

$$\mathcal{X} = a^{(1)} \circ a^{(1)} \circ \cdots a^{(N)},$$

where $I_1, \cdots, I_N$ are the dimensions of each mode, and the symbol $\circ$ denotes the vector outer product. The scalar product of two tensors $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{I_1 \times \cdots I_N}$ is defined as

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1} \cdots \sum_{i_N} a_{i_1 \cdots i_N} b_{i_1 \cdots i_N},$$

and the Frobenius norm of a tensor $\mathcal{A}$ is $\|\mathcal{A}\| = \sqrt{\langle \mathcal{A}, \mathcal{A} \rangle}$.

The rank of a tensor $\mathcal{X}$, denoted $rank(\mathcal{X})$, is defined as the smallest number of rank-one tensors that generate $\mathcal{X}$ as their sum [143]. The definition of tensor rank is an exact analogue to the definition of the matrix rank. There are different types of tensor rank, namely the *maximum rank*, *typical rank*, and *border rank*. The problem of determining tensor rank is NP-hard [163].

We provide two Lemmas that are used for our Two-Phase Decomposition (TP-Decomp) technique.

Lemma 1: For a 3-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, only the following weak upper bound on its maximum rank is known [159]: $rank(\mathcal{X}) \leq min\{IJ, IK, JK\}$.

Lemma 2: For a 3-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, the CP decomposition is generically unique if the following two conditions hold [175]: (1) $R \leq K$; (2) $R(R-1) \leq I(I-1)J(J-1)/2$, where R is the rank of $\mathcal{X}$.

LIST OF REFERENCES

[1] J.-P. Eckmann, S. O. Kamphorst, D. Ruelle, Recurrence plots of dynamical systems, Europhys. Lett 4 (9) (1987) 973–977.

[2] C. Webber, J. P. Zbilut, Dynamical assessment of physiological systems and states using recurrence plot strategies, Journal of Applied Physiology 76 (2) (1994) 965–973.

[3] L. Cao, Practical method for determining the minimum embedding dimension of a scalar time series, Physica D: Nonlinear Phenomena 110 (1) (1997) 43–50.

[4] C. Webber, M. Schmidt, J. Walsh, Influence of isometric loading on biceps emg dynamics as assessed by linear and nonlinear tools, Journal of Applied Physiology 78 (3) (1995) 814–822.

[5] N. Marwan, M. H. Trauth, M. Vuille, J. Kurths, Comparing modern and pleistocene enso-like influences in nw argentina using nonlinear time series analysis methods, Climate Dynamics 21 (3-4) (2003) 317–326.

[6] C. G. Gilmore, A new approach to testing for chaos, with applications in finance and economics, International Journal of Bifurcation and Chaos 3 (03) (1993) 583–587.

[7] F. Strozzi, J.-M. Zaldı?var, J. P. Zbilut, Application of nonlinear time series analysis techniques to high-frequency currency exchange data, Physica A: Statistical Mechanics and its Applications 312 (3) (2002) 520–538.

[8] P. Faure, H. Korn, A new method to estimate the kolmogorov entropy from recurrence plots: its application to neuronal signals, Physica D: Nonlinear Phenomena 122 (1) (1998) 265–279.

[9] M. C. Romano, M. Thiel, J. Kurths, W. von Bloh, Multivariate recurrence plots, Physics letters A 330 (3) (2004) 214–223.

[10] J. P. Zbilut, A. Giuliani, C. L. Webber, Detecting deterministic signals in exceptionally noisy environments using cross-recurrence quantification, Physics Letters A 246 (1) (1998) 122–128.

[11] N. Marwan, J. Kurths, Nonlinear analysis of bivariate data with cross recurrence plots, Physics Letters A 302 (5) (2002) 299–307.

[12] N. Marwan, N. Wessel, U. Meyerfeldt, A. Schirdewan, J. Kurths, Recurrence-plot-based measures of complexity and their application to heart-rate-variability data, Physical Review E 66 (2) (2002) 026702.

[13] K. Shockley, M. Butwill, J. P. Zbilut, C. L. Webber, Cross recurrence quantification of coupled oscillators, Physics Letters A 305 (1) (2002) 59–69.

[14] J. P. Zbilut, M. Koebbe, H. Loeb, G. Mayer-Kress, Use of recurrence plots in the analysis of heart beat intervals, in: Computers in Cardiology 1990, Proceedings., IEEE, 1990, pp. 263–266.

[15] J. Naschitz, I. Rosner, M. Rozenbaum, M. Fields, H. Isseroff, J. Babich, E. Zuckerman, N. Elias, D. Yeshurun, S. Naschitz, et al., Patterns of cardiovascular reactivity in disease diagnosis, QJM 97 (3) (2004) 141–151.

[16] C. Frontali, E. Pizzi, Similarity in oligonucleotide usage in introns and intergenic regions contributes to long-range correlation in the¡ i¿ caenorhabditis elegans¡/i¿ genome, Gene 232 (1) (1999) 87–95.

[17] C. G. Gilmore, An examination of nonlinear dependence in exchange rates, using recent methods from chaos theory, Global Finance Journal 12 (1) (2001) 139–151.

[18] C. BenAbdelkader, R. Cutler, L. Davis, Gait recognition using image self-similarity, EURASIP Journal on Applied Signal Processing 2004 (2004) 572–585.

[19] R. Cutler, L. Davis, Robust real-time periodic motion detection, analysis, and applications, Pattern Analysis and Machine Intelligence, IEEE Transactions on 22 (8) (2002) 781–796.

[20] S. Carlsson, Recognizing walking people, The International Journal of Robotics Research 22 (6) (2003) 359.

[21] I. N. Junejo, E. Dexter, I. Laptev, P. Perez, View-independent action recognition from temporal self-similarities, IEEE Transactions on Pattern Analysis and Machine Intelligence 99 (PrePrints).

[22] E. Shechtman, M. Irani, Matching local self-similarities across images and videos, in: Proc. CVPR, 2007.

[23] P. Viola, W. Wells III, Alignment by maximization of mutual information, International journal of computer vision 24 (2) (1997) 137–154.

[24] J. Huang, S. You, J. Zhao, Multimodal image matching using self similarity, in: Applied Imagery Pattern Recognition Workshop (AIPR), 2011 IEEE, 2011, pp. 1 –6.

[25] E. Shechtman, M. Irani, Matching local self-similarities across images and videos, in: Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, Ieee, 2007, pp. 1–8.

[26] T. Deselaers, V. Ferrari, Global and efficient self-similarity for object classification and detection, in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE, 2010, pp. 1633–1640.

[27] I. Junejo, Self-similarity based action recognition using conditional random fields, in: Information Retrieval Knowledge Management (CAMP), 2012 International Conference on, 2012, pp. 254 –259.

[28] E. Hörster, R. Lienhart, Deep networks for image retrieval on large-scale databases, in: Proceedings of the 16th ACM international conference on Multimedia, ACM, 2008, pp. 643–646.

[29] C. Lampert, H. Nickisch, S. Harmeling, Learning to detect unseen object classes by between-class attribute transfer, in: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 951–958.

[30] A. Vedaldi, V. Gulshan, M. Varma, A. Zisserman, Multiple kernels for object detection, in: Computer Vision, 2009 IEEE 12th International Conference on, IEEE, 2009, pp. 606–613.

[31] L. De Lathauwer, B. De Moor, From matrix to tensor: Multilinear algebra and signal processing, in: INSTITUTE OF MATHEMATICS AND ITS APPLICATIONS CONFERENCE SERIES, Vol. 67, Citeseer, 1998, pp. 1–16.

[32] N. D. Sidiropoulos, R. Bro, G. B. Giannakis, Parallel factor analysis in sensor array processing, Signal Processing, IEEE Transactions on 48 (8) (2000) 2377–2388.

[33] M. A. O. Vasilescu, D. Terzopoulos, Multilinear analysis of image ensembles: Tensorfaces, in: Computer VisionłECCV 2002, Springer, 2002, pp. 447–460.

[34] M. Vasilescu, D. Terzopoulos, Multilinear subspace analysis of image ensembles, in: Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on, Vol. 2, IEEE, 2003.

[35] B. Savas, Analyses and tests of handwritten digit recognition algorithms, LiTH-MAT-EX-2003-01, Link?ping University, Department of Mathematics.

[36] H. Wang, N. Ahuja, Facial expression decomposition, in: Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on, IEEE, 2003, pp. 958–965.

[37] H. Wang, N. Ahuja, Compact representation of multidimensional data using tensor rank-one decomposition, vectors 1 (2004) 5.

[38] A. Shashua, T. Hazan, Non-negative tensor factorization with applications to statistics and computer vision, in: Proceedings of the 22nd international conference on Machine learning, ACM, 2005, pp. 792–799.

[39] N. Liu, B. Zhang, J. Yan, Z. Chen, W. Liu, F. Bai, L. Chien, Text representation: From vector to tensor, in: Data Mining, Fifth IEEE International Conference on, IEEE, 2005, pp. 4–pp.

[40] J.-T. Sun, H.-J. Zeng, H. Liu, Y. Lu, Z. Chen, Cubesvd: a novel approach to personalized web search, in: Proceedings of the 14th international conference on World Wide Web, ACM, 2005, pp. 382–390.

[41] C. F. Beckmann, S. M. Smith, Tensorial extensions of independent component analysis for multisubject fmri analysis, Neuroimage 25 (1) (2005) 294–311.

[42] E. Martinez-Montes, P. A. Valdes-Sosa, F. Miwakeichi, R. I. Goldman, M. S. Cohen, Concurrent eeg/fmri analysis by multiway partial least squares., NeuroImage 22 (3) (2004) 1023–1034.

[43] F. Miwakeichi, E. Martınez-Montes, P. A. Valdés-Sosa, N. Nishiyama, H. Mizuhara, Y. Yamaguchi, Decomposing eeg data into space–time–frequency components using parallel factor analysis, NeuroImage 22 (3) (2004) 1035–1045.

[44] M. Mørup, L. K. Hansen, C. S. Herrmann, J. Parnas, S. M. Arnfred, Parallel factor analysis as an exploratory tool for wavelet transformed event-related eeg, NeuroImage 29 (3) (2006) 938–947.

[45] L. Zhang, Q. Gao, D. Zhang, Directional independent component analysis with tensor representation, in: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, IEEE, 2008, pp. 1–7.

[46] G. Hua, P. Viola, S. Drucker, Face recognition using discriminatively trained orthogonal rank one tensor projections, in: Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, IEEE, 2007, pp. 1–8.

[47] S. Yan, D. Xu, Q. Yang, L. Zhang, X. Tang, H. Zhang, Multilinear discriminant analysis for face recognition, Image Processing, IEEE Transactions on 16 (1) (2007) 212–220.

[48] X. He, D. Cai, P. Niyogi, Tensor subspace analysis, Advances in Neural Information Processing Systems 18 (2006) 499.

[49] T. Kim, S. Wong, R. Cipolla, Tensor canonical correlation analysis for action classification, in: Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, IEEE, 2007, pp. 1–8.

[50] B. Krausz, C. Bauckhage, Action recognition in videos using nonnegative tensor factorization, in: Pattern Recognition (ICPR), 2010 20th International Conference on, IEEE, 2010, pp. 1763–1766.

[51] Y. Lui, J. Beveridge, Tangent bundle for human action recognition, in: Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on, IEEE, 2011, pp. 97–102.

[52] I. Kotsia, I. Patras, Support tucker machines, in: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE, 2011, pp. 633–640.

[53] D. Weinland, R. Ronfard, E. Boyer, Free viewpoint action recognition using motion history volumes, CVIU 103 (2-3) (2006) 249–257.

[54] M. Abdelkader, W. Abd-Almageed, A. Srivastava, R. Chellappa, Silhouette-based gesture and action recognition via modeling trajectories on riemannian shape manifolds, CVIU 115 (3) (2011) 439–455.

[55] A. Yilmaz, M. Shah, Actions sketch: A novel action representation, in: Proc. CVPR, 2005, pp. I:984–989.

[56] L. Gorelick, M. Blank, E. Shechtman, M. Irani, R. Basri, Actions as space-time shapes, PAMI 29 (12) (2007) 2247–2253.

[57] M. Grundmann, F. Meier, I. Essa, 3d shape context and distance transform for action recognition, in: Proc. ICPR, 2008, pp. 1–4.

[58] Q. Le, W. Zou, S. Yeung, A. Ng, Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis, in: CVPR, IEEE, 2011, pp. 3361–3368.

[59] H. Wang, A. Klaser, C. Schmid, C. Liu, Action recognition by dense trajectories, in: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE, 2011, pp. 3169–3176.

[60] S. Wu, O. Oreifej, M. Shah, Action recognition in videos acquired by a moving camera using motion decomposition of lagrangian particle trajectories, in: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 1419–1426.

[61] K. Huang, Y. Zhang, T. Tan, A discriminative model of motion and cross ratio for view-invariant action recognition, Image Processing, IEEE Transactions on 21 (4) (2012) 2187–2197.

[62] I. Laptev, M. Marszałek, C. Schmid, B. Rozenfeld, Learning realistic human actions from movies, in: Proc. CVPR, 2008.

[63] P. Dollár, V. Rabaud, G. Cottrell, S. Belongie, Behavior recognition via sparse spatio-temporal features, in: VS-PETS, 2005, pp. 65–72.

[64] J. Niebles, H. Wang, F. Li, Unsupervised learning of human action categories using spatial-temporal words, in: Proc. BMVC, 2006.

[65] A. Gilbert, J. Illingworth, R. Bowden, Scale invariant action recognition using compound features mined from dense spatio-temporal corners, in: Proc. ECCV, 2008, pp. I: 222–233.

[66] A. Gilbert, J. Illingworth, R. Bowden, Action recognition using mined hierarchical compound features, Pattern Analysis and Machine Intelligence, IEEE Transactions on (99) (2011) 1–1.

[67] A. A. Efros, A. C. Berg, E. C. Berg, G. Mori, J. Malik, Recognizing action at a distance, in: ICCV, 2003, pp. 726–733.

[68] P. Matikainen, R. Sukthankar, M. Hebert, Feature seeding for action recognition, in: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 1716–1723.

[69] S. Maji, L. Bourdev, J. Malik, Action recognition from a distributed representation of pose and appearance, in: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE, 2011, pp. 3177–3184.

[70] X. Wu, D. Xu, L. Duan, J. Luo, Action recognition using context and appearance distribution features, in: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE, 2011, pp. 489–496.

[71] Z. Lin, Z. Jiang, L. Davis, Recognizing actions by shape-motion prototype trees, in: Computer Vision, 2009 IEEE 12th International Conference on, IEEE, 2010, pp. 444–451.

[72] J. Niebles, H. Wang, L. Fei-Fei, Unsupervised learning of human action categories using spatial-temporal words, International Journal of Computer Vision 79 (3) (2008) 299–318.

[73] J. Wang, Z. Chen, Y. Wu, Action recognition with multiscale spatio-temporal contexts, in: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE, 2011, pp. 3185–3192.

[74] H. Seo, P. Milanfar, Action recognition from one example, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE 33 (5) (2011) 867.

[75] M. Bregonzio, S. Gong, T. Xiang, Recognising action as clouds of space-time interest points, in: Proc. CVPR, 2009, pp. 1948–1955.

[76] M. Muller, T. Roder, M. Clausen, Efficient content-based retrieval of motion capture data, in: ACM Transactions on Graphics (TOG), Vol. 24, ACM, 2005, pp. 677–685.

[77] A. Safonova, J. Hodgins, Construction and optimal search of interpolated motion graphs, in: ACM SIGGRAPH 2007 papers, ACM, 2007, pp. 106–es.

[78] Y. Sakamoto, S. Kuriyama, T. Kaneko, Motion map: image-based retrieval and segmentation of motion data, in: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation, Eurographics Association, 2004, pp. 259–266.

[79] F. Liu, Y. Zhuang, F. Wu, Y. Pan, 3D motion retrieval with motion index tree, Computer Vision and Image Understanding 92 (2-3) (2003) 265–284.

[80] M. Cardle, M. Vlachos, S. Brooks, E. Keogh, D. Gunopulos, Fast motion capture matching with replicated motion editing, in: ACM SIGGRAPH, Citeseer, 2003.

[81] L. Kovar, M. Gleicher, Automated extraction and parameterization of motions in large data sets, in: ACM Transactions on Graphics (TOG), Vol. 23, ACM, 2004, pp. 559–568.

[82] L. Kovar, Automated extraction and parameterization of motions in large data sets, ACM Transactions on Graphics 23 (2004) 559–568.

[83] Z. Deng, Q. Gu, Q. Li, Perceptually consistent example-based human motion retrieval, in: Proceedings of the 2009 symposium on Interactive 3D graphics and games, ACM, 2009, pp. 191–198.

[84] B. Krüger, J. Tautges, A. Weber, A. Zinke, Fast local and global similarity searches in large motion capture databases, in: Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '10, 2010, pp. 1–10.

[85] C. Li, S. Zheng, B. Prabhakaran, Segmentation and recognition of motion streams by similarity search, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP) 3 (3) (2007) 16–es.

[86] Y. Lin, Efficient human motion retrieval in large databases, in: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia, ACM, 2006, pp. 31–37.

[87] K. G. Derpanis, M. Sizintsev, K. Cannons, R. P. Wildes, Efficient action spotting based on a spacetime oriented structure representation, in: CVPR, IEEE, 2010, pp. 1990–1997.

[88] J. Knopp, M. Prasad, G. Willems, R. Timofte, L. Van Gool, Hough transform and 3d surf for robust three dimensional classification, ECCV (2010) 589–602.

[89] S. Ali, A. Basharat, M. Shah, Chaotic invariants for human action recognition, in: ICCV, IEEE, 2007, pp. 1–8.

[90] C. Fanti, L. Zelnik-Manor, P. Perona, Hybrid models for human motion recognition, in: CVPR, Vol. 1, IEEE, 2005, pp. 1166–1173.

[91] Y. Hu, L. Cao, F. Lv, S. Yan, Y. Gong, T. S. Huang, Action detection in complex scenes with spatial and temporal ambiguities, in: ICCV, IEEE, 2009, pp. 128–135.

[92] Z. Lin, Z. Jiang, L. S. Davis, Recognizing actions by shape-motion prototype trees, in: ICCV, IEEE, 2009, pp. 444–451.

[93] S. Ali, M. Shah, Human action recognition in videos using kinematic features and multiple instance learning, PAMI 32 (2) (2010) 288–303.

[94] A. Fathi, G. Mori, Action recognition by learning mid-level motion features, in: CVPR, IEEE, 2008, pp. 1–8.

[95] E. Shechtman, M. Irani, Space-time behavior-based correlation-or-how to tell if two underlying motion fields are similar without computing them?, PAMI 29 (11) (2007) 2045–2056.

[96] P. Dollár, V. Rabaud, G. Cottrell, S. Belongie, Behavior recognition via sparse spatio-temporal features, in: Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on, IEEE, 2005, pp. 65–72.

[97] K. Rapantzikos, Y. Avrithis, S. Kollias, Dense saliency-based spatiotemporal feature points for action recognition, in: CVPR, IEEE, 2009, pp. 1454–1461.

[98] R. Blasco, M. Carmen, Synchronization analysis by means of recurrences in phase space, Ph.D. thesis, Universitat sbibliothek (2004).

[99] R. Gonzalez, R. Woods, Digital image processing, Pearson/Prentice Hall, 2008. URL `http://books.google.com/books?id=8uGOnjRGEzoC`

[100] L. Ye, E. Keogh, Time series shapelets: a new primitive for data mining, in: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2009, pp. 947–956.

[101] M. Hu, Visual pattern recognition by moment invariants, Information Theory, IRE Transactions on 8 (2) (1962) 179–187.

[102] C. Zahn, R. Roskies, Fourier descriptors for plane closed curves, Computers, IEEE Transactions on 100 (3) (1972) 269–281.

[103] S. Belongie, J. Malik, J. Puzicha, Shape matching and object recognition using shape contexts, Pattern Analysis and Machine Intelligence, IEEE Transactions on 24 (4) (2002) 509–522.

[104] D. Weinland, M. Ozuysal, P. Fua, Making Action Recognition Robust to Occlusions and Viewpoint Changes, Computer Vision–ECCV 2010 (2010) 635–648.

[105] A. Klaser, M. Marszałek, C. Schmid, A spatio-temporal descriptor based on 3D-gradients, in: British Machine Vision Conference, Citeseer, 2008, pp. 995–1004.

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

[107] T. Kolda, B. Bader, Tensor decompositions and applications, SIAM review 51 (3) (2009) 455–500.

[108] L. De Lathauwer, B. De Moor, J. Vandewalle, On the Best Rank-1 and Rank-(R˜ 1, R˜ 2,..., R˜ N) Approximation of Higher-Order Tensors, SIAM Journal on Matrix Analysis and Applications 21 (4) (2000) 1324–1342.

[109] P. Kroonenberg, Three-mode principal component analysis: Theory and applications, DSWO press, 1983.

[110] P. Kroonenberg, J. De Leeuw, Principal component analysis of three-mode data by means of alternating least squares algorithms, Psychometrika 45 (1) (1980) 69–97.

[111] T. Zhang, G. Golub, Rank-1 approximation of higher-order tensors, SIAM J. Matrix Anal. Appl 23 (534-550) (2001) 4.

[112] H. Lu, K. N. Plataniotis, A. N. Venetsanopoulos, A survey of multilinear subspace learning for tensor data, Pattern Recognition 44 (7) (2011) 1540–1551.

[113] J. Aggarwal, M. Ryoo, Human activity analysis: A review, ACM Computing Surveys (C-SUR) 43 (3) (2011) 16.

[114] N. Lawrence, Gaussian process latent variable models for visualization of high dimensional data, Advances in neural information processing systems 16 (2004) 329–336.

[115] D. MacKay, Information theory, inference, and learning algorithms, Cambridge Univ Pr, 2003.

[116] C. Rasmussen, C. Williams, Gaussian processes for machine learning. 2006, The MIT Press, Cambridge, MA, USA 38 715–719.

[117] A. Gilbert, J. Illingworth, R. Bowden, Fast realistic multi-action recognition using mined dense spatio-temporal features, in: Computer Vision, 2009 IEEE 12th International Conference on, IEEE, 2010, pp. 925–931.

[118] A. Kovashka, K. Grauman, Learning a hierarchy of discriminative space-time neighborhood features for human action recognition, in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE, 2010, pp. 2046–2053.

[119] Y. Shen, H. Foroosh, View-invariant action recognition from point triplets, IEEE transactions on pattern analysis and machine intelligence (2009) 1898–1905.

[120] K. Schindler, L. Van Gool, Action Snippets: How many frames does human action recognition require?, in: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, IEEE, 2008, pp. 1–8.

[121] Y. Shen, H. Foroosh, View-invariant action recognition using fundamental ratios, in: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, IEEE, 2008, pp. 1–6.

[122] Z. Zhang, Y. Hu, S. Chan, L. Chia, Motion context: A new representation for human action recognition, Computer Vision–ECCV 2008 (2008) 817–829.

[123] A. Laptev, C. Schmid, Will person detection help bag-of-features action recognition?

[124] H. Wang, M. Ullah, A. Klaser, I. Laptev, C. Schmid, Evaluation of local spatio-temporal features for action recognition.

[125] J. Liu, M. Shah, Learning human actions via information maximization, in: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, IEEE, 2008, pp. 1–8.

[126] I. Laptev, On space-time interest points, International Journal of Computer Vision 64 (2) (2005) 107–123.

[127] A. Kläser, M. Marszałek, C. Schmid, A spatio-temporal descriptor based on 3d-gradients, in: British Machine Vision Conference, 2008, pp. 995–1004.
URL `http://lear.inrialpes.fr/pubs/2008/KMS08`

[128] J. Niebles, H. Wang, L. Fei-Fei, Unsupervised learning of human action categories using spatial-temporal words, International Journal of Computer Vision 79 (3) (2008) 299–318.

[129] S. Kwak, W. Nam, B. Han, J. Han, Learning occlusion with likelihoods for visual tracking, in: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 1551–1558.

[130] A. Ayvaci, M. Raptis, S. Soatto, Sparse occlusion detection with optical flow, International Journal of Computer Vision (2011) 1–17.

[131] D. Weinland, M. Özuysal, P. Fua, Making action recognition robust to occlusions and viewpoint changes, Computer Vision–ECCV 2010 (2010) 635–648.

[132] B. Wu, R. Nevatia, Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors, in: Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on, Vol. 1, IEEE, 2005, pp. 90–97.

[133] X. Wang, T. Han, S. Yan, An hog-lbp human detector with partial occlusion handling, in: Computer Vision, 2009 IEEE 12th International Conference on, IEEE, 2009, pp. 32–39.

[134] J. Park, M. Wakin, A multiscale framework for compressive sensing of video, in: Picture Coding Symposium, 2009. PCS 2009, IEEE, 2009, pp. 1–4.

[135] D. L. Donoho, Compressed sensing, Information Theory, IEEE Transactions on 52 (4) (2006) 1289–1306.

[136] E. J. Candès, J. Romberg, T. Tao, Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information, Information Theory, IEEE Transactions on 52 (2) (2006) 489–509.

[137] V. Stankovic, L. Stankovic, C. S., Compressive video sampling, 16th European Signal Processing Conference.

[138] R. Baraniuk, Compressive sensing, IEEE Signal Processing Magazine 24 (4) (2007) 118–212.

[139] J. N. Laska, M. A. Davenport, R. G. Baraniuk, Exact signal recovery from sparsely corrupted measurements through the pursuit of justice, in: Signals, Systems and Computers, 2009 Conference Record of the Forty-Third Asilomar Conference on, IEEE, 2009, pp. 1556–1560.

[140] A. C. Sankaranarayanan, P. K. Turaga, R. G. Baraniuk, R. Chellappa, Compressive acquisition of dynamic scenes, in: Computer Vision–ECCV 2010, Springer, 2010, pp. 129–142.

[141] Y. Wexler, E. Shechtman, M. Irani, Space-time completion of video, Pattern Analysis and Machine Intelligence, IEEE Transactions on 29 (3) (2007) 463–476.

[142] E. Acar, D. Dunlavy, T. Kolda, M. Mørup, Scalable tensor factorizations with missing data, Siam Datamining 2010 (SDM 2010).

[143] J. B. Kruskal, Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics, Linear algebra and its applications 18 (2) (1977) 95–138.

[144] S. Chuan, I. N. Junejo, F. Hassan, Action recognition using rank-1 approximation of joint self-similarity volume., in: ICCV, Computer Vision, 2011 IEEE 13th International Conference on, IEEE, 2011, pp. 1007–1012.

[145] X. Zhu, J. Lafferty, R. Rosenfeld, Semi-supervised learning with graphs, Ph.D. thesis, Carnegie Mellon University, Language Technologies Institute, School of Computer Science (2005).

[146] D. Zhou, O. Bousquet, T. Lal, J. Weston, B. Schölkopf, Learning with local and global consistency, Advances in neural information processing systems 16 (2004) 321–328.

[147] R. Cutler, L. Davis, Robust real-time periodic motion detection, analysis, and applications, Pattern Analysis and Machine Intelligence, IEEE Transactions on 22 (8) (2000) 781–796.

[148] I. Junejo, E. Dexter, I. Laptev, P. Pérez, View-independent action recognition from temporal self-similarities, IEEE transactions on pattern analysis and machine intelligence (2010) 172–185.

[149] K. Forbes, E. Fiume, An efficient search algorithm for motion data using weighted PCA, in: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, ACM, 2005, pp. 67–76.

[150] J. Barbič, A. Safonova, J. Pan, C. Faloutsos, J. Hodgins, N. Pollard, Segmenting motion capture data into distinct behaviors, in: Proceedings of Graphics Interface 2004, Canadian Human-Computer Communications Society, 2004, pp. 185–194.

[151] H. Yasuda, R. Kaihara, S. Saito, M. Nakajima, Motion belts, in: ACM SIGGRAPH, 2007, pp. 05–09.

[152] Y. Hu, S. Wu, S. Xia, J. Fu, W. Chen, Motion track: Visualizing variations of human motion data, in: Pacific Visualization Symposium (PacificVis), 2010 IEEE, IEEE, pp. 153–160.

[153] G. Liu, J. Zhang, W. Wang, L. McMillan, A system for analyzing and indexing human-motion databases, in: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, ACM, 2005, pp. 924–926.

[154] M. Muller, T. Roder, Motion templates for automatic classification and retrieval of motion capture data, in: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation, Eurographics Association, 2006, pp. 137–146.

[155] T.-K. Kim, R. Cipolla, Canonical correlation analysis of video volume tensors for action categorization and detection, PAMI 31 (8) (2009) 1415–1428.

[156] B. W. Bader, M. W. Berry, M. Browne, Discussion tracking in enron email using parafac, Survey of Text Mining II (2008) 147–163.

[157] A. Shashua, A. Levin, Linear image coding for regression and classification using the tensor-rank principle, in: CVPR, Vol. 1, IEEE, 2001, pp. I–42.

[158] L. De Lathauwer, B. De Moor, J. Vandewalle, A multilinear singular value decomposition, SIAM journal on Matrix Analysis and Applications 21 (4) (2000) 1253–1278.

[159] J. B. Kruskal, Rank, decomposition, and uniqueness for 3-way and n-way arrays, Multiway data analysis (1989) 7–18.

[160] C. Eckart, G. Young, The approximation of one matrix by another of lower rank, Psychometrika 1 (3) (1936) 211–218.

[161] P. C. Hansen, The truncatedsvd as a method for regularization, BIT Numerical Mathematics 27 (4) (1987) 534–553.

[162] M. Frank, J. M. Buhmann, Selecting the rank of truncated svd by maximum approximation capacity, in: Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on, IEEE, 2011, pp. 1036–1040.

[163] J. Håstad, Tensor rank is np-complete, Journal of Algorithms 11 (4) (1990) 644–654.

[164] Y. Ke, R. Sukthankar, M. Hebert, Event detection in crowded videos, in: ICCV, IEEE, 2007, pp. 1–8.

[165] J. Yuan, Z. Liu, Y. Wu, Discriminative subvolume search for efficient action detection, in: CVPR, IEEE, 2009, pp. 2442–2449.

[166] A. Yao, J. Gall, L. Van Gool, A hough transform-based voting framework for action recognition, in: CVPR, IEEE, 2010, pp. 2061–2068.

[167] M. Jain, H. Jégou, P. Bouthemy, et al., Better exploiting motion for better action recognition, in: CVPR-International Conference on Computer Vision and Pattern Recognition, 2013.

[168] H. Wang, A. Kläser, C. Schmid, C.-L. Liu, Dense trajectories and motion boundary descriptors for action recognition, IJCV (2013) 1–20.

[169] C. H. Lampert, M. B. Blaschko, T. Hofmann, Beyond sliding windows: Object localization by subwindow search, in: CVPR, IEEE, 2008, pp. 1–8.

[170] J. R. Bennett, J. S. Mac Donald, On the measurement of curvature in a quantized environment, IEEE Trans on computers 24 (8) (1975) 803–820.

[171] X. Wu, A. G. Hauptmann, C.-W. Ngo, Practical elimination of near-duplicates from web video search, in: ACM MM, 2007, pp. 218–227.

[172] J. Song, Y. Yang, Z. Huang, H. T. Shen, R. Hong, Multiple feature hashing for real-time large scale near-duplicate video retrieval, in: ACM MM, 2011, pp. 423–432.

[173] G. Yu, N. A. Goussies, J. Yuan, Z. Liu, Fast action detection via discriminative random forest voting and top-k subvolume search, Multimedia, IEEE Transactions on 13 (3) (2011) 507–517.

[174] H. Boyraz, M. F. Tappen, R. Sukthankar, Localizing actions through sequential 2d video projections, in: Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on, IEEE, 2011, pp. 34–39.

[175] L. De Lathauwer, A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization, SIAM Journal on Matrix Analysis and Applications 28 (3) (2006) 642–666.