Spatial and Temporal Modeling for Human Activity Recognition from Multimodal Sequential Data

2016

Jun Ye

University of Central Florida

SPATIAL AND TEMPORAL MODELING FOR HUMAN ACTIVITY RECOGNITION FROM MULTIMODAL SEQUENTIAL DATA

by

JUN YE
B.S. Huazhong University of Science and Technology, 2007
M.S. Beihang University, 2010

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

Fall Term
2016

Major Professor: Kien A. Hua

ABSTRACT

Human Activity Recognition (HAR) has been an intense research area for more than a decade. Different sensors, ranging from 2D and 3D cameras to accelerometers, gyroscopes, and magnetometers, have been employed to generate multimodal signals to detect various human activities. With the advancement of sensing technology and the popularity of mobile devices, depth cameras and wearable devices, such as Microsoft Kinect and smart wristbands, open a unprecedented opportunity to solve the challenging HAR problem by learning expressive representations from the multimodal signals recording huge amounts of daily activities which comprise a rich set of categories.

Although competitive performance has been reported, existing methods focus on the statistical or spatial representation of the human activity sequence; while the internal temporal dynamics of the human activity sequence are not sufficiently exploited. As a result, they often face the challenge of recognizing visually similar activities composed of dynamic patterns in different temporal order. In addition, many model-driven methods based on sophisticated features and carefully-designed classifiers are computationally demanding and unable to scale to a large dataset. In this dissertation, we propose to address these challenges from three different perspectives; namely, 3D spatial relationship modeling, dynamic temporal quantization, and temporal order encoding.

We propose a novel octree-based algorithm for computing the 3D spatial relationships between objects from a 3D point cloud captured by a Kinect sensor. A set of 26 3D spatial directions are defined to describe the spatial relationship of an object with respect to a reference object. These 3D directions are implemented as a set of spatial operators, such as ”AboveSouthEast” and ”BelowNorthWest,” of an event query language to query human activities in an indoor environment; for example, ”A person walks in the hallway from north to south.” The performance is quantitatively evaluated in a public RGBD object dataset and qualitatively investigated in a live video computing platform.

In order to address the challenge of temporal modeling in human action recognition, we introduce the dynamic temporal quantization, a clustering-like algorithm to quantize human action sequences of varied lengths into fixed-size quantized vectors. A two-step optimization algorithm is proposed to jointly optimize the quantization of the original sequence. In the aggregation step, frames falling into the sample segment are aggregated by max-polling and produce the quantized representation of the segment. During the assignment step, frame-segment assignment is updated according to dynamic time warping, while the temporal order of the entire sequence is preserved. The proposed technique is evaluated on three public 3D human action datasets and achieves state-of-the-art performance.

Finally, we propose a novel temporal order encoding approach that models the temporal dynamics of the sequential data for human activity recognition. The algorithm encodes the temporal order of the latent patterns extracted by the subspace projection and generates a highly compact First-Take-All (FTA) feature vector representing the entire sequential data. An optimization algorithm is further introduced to learn the optimized projections in order to increase the discriminative power of the FTA feature. The compactness of the FTA feature makes it extremely efficient for human activity recognition with nearest neighbor search based on Hamming distance. Experimental results on two public human activity datasets demonstrate the advantages of the FTA feature over state-of-the-art methods in both accuracy and efficiency.

I dedicate this dissertation to my wife, Xing, for being there whenever I need her, and to my parents, Jiaxiang and Huifang, for their love and support through all the years.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Kien A. Hua for his guidance, support and advice throughout my Ph.D. program at University of Central Florida.

I would also like to thank my dissertation committee members, Dr. Cliff C. Zou, Dr. Hassan Foroosh and Dr. Waldemar Karwowski, for their valuable guidance and suggestions on my dissertation.

I would like to specially thank Dr. Guo-Jun Qi for his advise and help on our coauthored publications.

I would like to thank all the past and present DSG members, Kai Li, Kutalmis Akpinar, Omar Nakhila, Sansiri Tarnpradab, Naifan Zhuang, Yusuph Turgun, Hao Hu, Liheng Zhang, Dr. Faisal Amjad, Dr. Alex Aved, Affra Attiah, Fereshteh Jafariakinabad, for being good friends and making DSG such an interesting place to work at.

Finally, I am especially grateful to my wife, Xing, and my parents, Jiaxiang and Huifang, for their love and support and for being the most important people in my life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Problem Statement and Motivation

Human Activity Recognition (HAR) has attracted intense research interest for more than a decade. Many different sensors, ranging from 2D and 3D cameras to miniature inertial sensors, such as accelerometers, gyroscopes and magnetometers, have all been investigated and utilized in HAR systems to capture various human activities. Recent development in commodity depth sensors, such as Microsoft Kinect [3] and Leap Motion [1], have made quality RGBD data accessible at an affordable cost. This new technology has driven the creation of a wide range of applications based on human activity recognition. For example, the original purpose of the development of Kinect depth camera is to support the natural user interface in video games such as dancing [4]. Such a capability is quickly applied to other application area, such as health care where human activity such as the gait information captured by the depth sensor can be used to help early-diagnose diseases or prevent health issues [5, 6]. Other successful applications have also been reported in education [7] and surveillance [8].

Although competitive performance has been reported, existing methods of HAR focus on the spatial representation of the human activity sequence, leaving the temporal dynamics inside the human activity sequence largely under-researched. As a result, they often face the challenge of recognizing visually similar activities composed of dynamic patterns in different temporal order. For instance, “put on a hat” and “take off a hat”, “entering a room” and “exit from a room” are examples of human actions with the same primitive postures but the reverse order. It is very challenging to distinguish these actions without sufficiently exploiting the temporal order information.

[1] `https://developer.leapmotion.com/`

In addition, many model-driven methods based on sophisticated features and carefully-designed classifiers are computationally demanding and unable to scale to a large dataset.

Besides the above camera-based HAR systems, wearable sensors have also been widely utilized as a HAR solution with high availability to track one's daily and fitness activities. For example, smart wristbands and smart phones provide a convenient solution to count steps and monitor heart rate. Some smart wristbands even support the feature of auto-logging of different cycling events,[2] or tracking the number of strokes made in a lap of swimming [3]. The popularity in smart wearable devices have opened a unprecedented opportunity to solve the challenging HAR problem by learning expressive representations from the multimodal sensor signals recording huge amounts of daily activities.

Although many successful products and applications in HAR have been developed, wearable sensor-based HAR remains to be a challenging problem due to the highly noisy nature and the complex temporal dynamics within the sensor signals. Most existing sensor-based HAR approaches focus on extracting statistical and structural features from the time domain and frequency domain, such as *mean, min, max, var, kurtosis, entropy, Fourier Transform coefficients and Discrete Transform coefficients,etc.*, and do no consider the temporal dynamics inside the human activity sequence. Therefore, they suffer from the confusion between activities with similar statistical patterns but different temporal patterns.

Contributions

To address the above challenges of HAR, this dissertation proposes solutions to HAR from three novel perspectives:

[2] `http://cycling.moov.cc/`

[3] `http://sites.garmin.com/en-US/swim/`

1. 3D spatial relationship modeling and 3D spatial query-based human event detection.
2. Dynamic temporal quantization for human action representation.
3. Temporal order encoding for human activity recognition.

The main contributions of this dissertation are:

- A novel octree-based algorithm of computing the 3D spatial relationship between objects is proposed. A set of 26 spatial operators are further defined and implemented to be included in an SQL-style event query language. HAR is then resolved by spatial-event query processing based on the 3D spatial relationship between a human and surrounding objects. Performance is quantitatively evaluated in a public RGBD object dataset and qualitatively investigated in a Live Video Computing Platform.
- A clustering-like algorithm is presented to quantize human action sequences of varied lengths into fixed-size quantized vectors. A two-step optimization algorithm is proposed to jointly optimize the quantization of the human activity sequences. In the aggregation step, frames falling into the sample segment are aggregated by max-polling and produce the quantized representation of the segment. In the assignment step, frame-segment assignment is updated according to dynamic time warping, while the temporal order of the entire sequence is preserved.
- A novel temporal order encoding approach is introduced to model the temporal dynamics of the sequential data for HAR. The algorithm encodes the temporal order of the latent patterns extracted by the subspace projection and generates a highly compact First-Take-All (FTA) feature vector representing the entire sequential data. An optimization algorithm is further presented to learn the optimized projections in order to increase the discriminative power of

the FTA feature. The compactness of the FTA feature makes it extremely efficient for human activity recognition with nearest neighbor search based on Hamming distance.

Organization of the Dissertation

The rest of the dissertation is organized as follows: The literature review is presented in Chapter 2. In Chapter 3, we introduce the 3D spatial relationship modeling from the point cloud data and present the spatial event query-based human activity detection. In Chapter 4, the temporal order-preserving dynamic quantization algorithm for 3D human action recognition is presented. In Chapter 5, A novel temporal order encoding approach and two different First-Take-All feature representations are presented to encode multimodal sensor data into compact feature vectors for human activity recognition, and finally, Chapter 6 concludes the dissertation and discusses the future work.

CHAPTER 2: LITERATURE REVIEW

In this chapter, we review the related work in the literature for spatial relationship modeling, camera-based human action recognition and wearable sensor-based human activity recognition.

3D Spatial Relationship Modeling and Human Event Query Processing

The spatial relationship between objects has a fuzzy nature [9]. It is quite intuitive and common to human understanding when interpreting directions in an approximate framework. Most of the existing methods adopted the fuzzy framework and interpreted the spatial relationship by a number or a range indicating the degree of applicability; and the spatial relationship of two complex objects can be interpreted by a combination of primitive spatial operators. [10] presented a good survey of different methods for computing the spatial relationships.

A great number of methods in the literature are based on angles. Keller and Wang [11] measured the angle between the connection of two points and the x-axis and used the cosine distance to compute the direction "to the right" between two objects. This method can be extended to any 2D directional relationships by simply replacing the x-axis by other directions. Other evaluation functions such as cubic spline functions were also utilized to measure the degree of applicability [12]. In [11], a centroid-based solution was proposed to evaluate the spatial relationship. Objects were represented by their centroid and the angle-based methods were employed to compute the directional relationships. Instead of using only centroid, Miyajima and Ralescu [13] computed the angle between any pair of points in the two objects. Then, a weighted histogram of angles were generated to compute the relationship between two objects. This exhausted comparison-based method can achieve higher accuracy at the cost of more computational overhead. Salamat and Zahzah [14] proposed a method that can combined fuzzy topological relations and directional relations and modeled them simultaneously. Nevertheless, this method can only work with the 2D objects.

Besides angle-based methods, machine learning-based methods were also proposed to address the complexity of the spatial relationships. Artificial Neural Networks [15] were incorporated to train and classify different spatial relationships between several basic shapes. However, each type of shape required a dedicated classifier, making this method not scalable for the real world scenario. In [16], SVM (Support Vector Machine) was employed to detect the contact points of two objects which were further clustered into groups to establish the contact point network. This network served as a skeletonized description of the objects and was finally used to identify the spatial relationships of the objects. The skeleton graph only utilized the contact regions between two objects. As a result, it had limited capability in representing complex spatial relationships. As a matter of fact, this approach only supported two simple spatial relationships, namely “on” and “adjacent”.

Bloch [17] proposed a morphological approach to assess the spatial relationships of multiple objects with respect to the same reference object simultaneously. A fuzzy landscape around the reference object was computed and the target object was then compared against the fuzzy landscape by a fuzzy pattern matching approach. Since the entire space around the reference object was investigated, it was capable of intepreting the spatial relationships between complex objects. Two later versions of this approach were extended to focus on the spatial relationships “between” [18] and “along” [19] in the context of medical imaging. Although these fuzzy-based methods can address very complex objects, each direction of interest must be computed separately, which will inevitably incur huge computational overhead and make it not scalable for more general and realistic scenarios.

Most of the above methods can only compute the 2D spatial relationships from images. Recently, with the prevalance of the commodity depth cameras (e.g. Kinect), depth information and the point cloud data structure have been explored to investigate the spatial relations in the 3D environment. Three supporting relations, "on-top," "partial on-top" and "side," were evaluated in [20] based on the object volumes from the point cloud. An angle-based measurement was then applied to distinguish them. Similar to the idea of the exhuasted comparison in [13], the geometric relations of all pairs of skeleton joints of two people were exploited to detect the interaction between them [21]. Borrmann et. al. [22] addressed the 3D spatial query challenge from the database perspective. They developed a 3D spatial query language from ISO standard SQL to enable the spatial analysis of building information models. More specifically, a slot-tree-based algorithm was proposed to partition the 3D object and compute directions by testing the local coordinates of slot pairs. Since their directional model is straightforward, they only supported six major directions in the 3D space.

Similar idea was also investigated in [23, 24], where a Live Video DataBase Management System (LVDBMS) was introduced and a Live Video Computing (LVC) framework was proposed to achieve the general event detection from a database point of view. A set of 2D spatial operators, along with the temporal operators and logical operators, were introduced and an SQL-like Live Video Query Language (LVQL) was defined to support the continuous query of the spatial-temporal human-centric event.

Different from the database perspective for event detection, a hierarchical event modeling framework was proposed in [25] and [26]. They introduced the definition of three classes of event– transient event, atomic event and compound event, and further evaluated three relationships– temporal, causal and spatial between them. The proposed method mapped an event from the data-level to the domain-level thus achieved a better event representation.

Human Action Recognition From Videos

Human action recognition has been intensively studied in the recent decades. Many of the existing approaches for human action recognition focus on the spatio-temporal feature and local motions. Most of these works adopt histogram-based features to represent the distribution of the spatio-temporal patterns in the human actions. In [27], the space-time interest points (STIP) were proposed as a local feature and used as a compact representation of spatio-temporal events. Dollar *et. al.* [28] applied the sparse spatio-temporal features to characterize the cuboids of spatio-temporally windowed data, and used the dictionary of cuboid prototypes to interpret a variety of human actions. Normally, the Bag-of-Word feature representation is employed as an unsupervised method to learn the dictionary of the spatio-temporal visual words. Supervised learning methods such as SVM is then employed to classify these feature vectors into different action categories.

Beside the Bag-of-Word-style methods, other features such as the motion energy of the action sequences, optical flow and contours have also been investigated for human action recognition. In [29], the motion energy image and motion history image were constructed as a temporal template and the Hue moments were extracted to describe the human activity. [30] introduced a motion descriptor based on optical flow measurements and recognized the activities in a nearest neighbor framework. In [31], a sequence of 2D object contours with respect to time were exploited to generate a spatio-temporal volume (STV), which was later used as the action descriptors to identify human actions. Agarwal and Triggs proposed a shape feature descriptor from image silhouettes and recovered the human pose by using the nonlinear regression [32]. The above global features can interpret the action in a holistic way. However, they are normally dense and less computationally efficient.

Recent advances in depth cameras (e.g. Kinect) provide an affordable solution to access multi-modal data from the indoor environment and attracts huge interest in the research of 3D human action recognition. There are a great number of approaches using only the depth data from the Kinect sensor. [33] proposed a human pose descriptor by computing the average normalized depth value on a dense grid over the 3D point cloud of the human body. In [34], the bag of 3D points from the depth maps was sampled and projected into three orthogonal 2D planes. An action graph was then employed to model the dynamics of the actions. Similar ideas had been applied in [35], where depth motion maps were accumulated over depth sequences and projected into three orthogonal planes, and the HoG (Histogram of Oriented Gradient) feature were computed as the feature descriptor to classify human actions. Oreifej and Liu [36] proposed the 4D normals from the surface of the 3D point cloud and introduced the histogram of oriented 4D normals (HON4D) descriptor. A machine learning-based sampling process was further applied to optimize the weight of each projector to achieve higher discriminative capability. This method has achieved great accuracy on several datasets. However, it is skeptical to datasets containing large spatial location variations.

With the success of skeleton joints estimation from depth images [37], joint-based features [38, 39, 40, 41, 42, 43] gain huge popularity in 3D action recognition. [39] and [38] both proposed view-invariant features from skeleton joints by coordinates transformation to detect human actions. Instead of using joint locations directly, [41] and [42] computed the normalized pair-wise joint distances from all pairs of skeleton joints and demonstrated good discriminative capability. In [40], distances of skeleton joints between the current frame and the previous frame as well as the first frame were computed in order to combine the spatial and the temporal feature. The joint-based features can be further quantized into code words and histogram of 3D joints (HOJ3D) [39] and histogram of visual words [40] were employed to describe the action sequences. Different from those joint-based methods, Vemulapalli [43] proposed a body-part representation of the skeleton and modeled the geometric transformation between different body parts in the 3D space. The temporal dynamics in terms of 3D transformation were captured and projected as a curved manifold in the Lie group. Classifications on the curves eventually determined the labels of the action.

Instead of using single modality data such as depth or skeleton joints, there are also many works leveraging multimodal data to further increase the recognition accuracy. Methods based on the RGB+depth images had been proposed lately. Ni *et.al.* [44] extended the space-time interest point (STIP) feature [27] by combining the color and depth information and proposed their multimodal fusion schemes. Zhang and Parker [45] proposed the 4D local spatio-temporal features that combine both the intensity image and the depth image, and replaced the original cuboid in [28] with the 4D hyper cuboid. Xia and Aggarwal [46] also extended the STIP feature in depth videos and described the local 3D depth cuboid by measuring the Depth Cuboid Similarity. Some other works also consider the combination of Depth+skeleton feature. [41] proposed the Local Occupancy Patterns (LOP) which characterized the distribution of point cloud around joints where the human-object interactions happened.

Most of the above methods adopt the histogram-based representations of local spatio-temporal features. Such features completely remove the global temporal information of the sequence data and therefore may be confused by actions containing similar poses but different temporal order. To address this issue, many methods focus on modeling the temporal structures of the human actions in a holistic way. Graph model-based methods such as Hidden Markov Model (HMM) [47], Max Entropy Markov Model (MEMM) [48], Conditional Random Field (CRF) [49] had been proposed to interpret the dynamics of the human actions. Motion template-based approaches [50, 42] introduced another way of modeling the temporal characteristics of human actions. In the motion template methods, a number of motion templates indicating different action classes were trained. Dynamic Time Warping (DTW) was employed to warp the video sequences of varied length. The labels of the unknown action sequences were then determined by measuring the similarity between the unknown sequences and the motion templates. These methods are able to detect and recognize a subsequence from the entire video and did not require the pre-segmentation of the video sequence. This is a great advantage and can be applied for online human action recognition. Temporal Pyramid [41, 36] was developed to capture the temporal structure of the sequence by uniformly subdividing the action sequence into partitions. Spatio-temporal feature descriptors were then applied to each partition. Since the uniform partition along the temporal axis was employed, the temporal pyramid was less flexible to handle the inter-class similarity and intra-class variation of the temporal patterns of different actions. Adaptive temporal pyramid [51] was proposed to partially overcome the above disadvantage by subdividing the temporal sequence according to the motion energy. A Super Normal Vector feature was then generated from the space-time partition and served as the comprehensive representation of the sequence. However, this method heavily relied on sophisticated features such as 3D surface normals and polynormals which were inapplicable to more general problems. In general, the aforementioned temporal modeling methods suffer from the temporal misalignment due to factors of temporal translation, motion scales and execution rate variations [42]. Temporal modeling still remains to be a challenging problem for human action recognition.

Wearable Sensor-based Human Activity Recognition

Wearable sensor-based HAR has been an active research field for more than a decade. Many different aspects have been investigated to improve the performance of HAR, including sensor and attribute selection, data acquisition, feature extraction, classification and energy consumption. All of the above factors have crucial impacts on the performance of HAR systems. In this section, we review the previous literature on feature extraction and classification for HAR.

Feature Extraction

Acceleration and gyroscopic data from miniature inertial sensors such as the accelerometer and gyroscope have been widely employed to recognize ambulation activities, including walking, climbing stairs, lying down, etc. [52, 53, 54, 55, 56]. Genrally, time series signals are segmented into windows with or without overlapping. Feature extraction is then applied to the windowed data to produce feature vector for recognition. Although sensor signals are oscillatory and prone to high fluctuation, they will still exhibit certain statistical behaviors. These behaviors have led many works to adopt statistical features from the time domain of the sesnor signal. For example, mean, max, min, standard deviation (std), skewness, kurtosis [57], median absolute deviation (mad), root mean square (rms), interquartile range (iqr) [58] and spectral entropy [59] have been utilized to form a statistical representation of the sensor signals. In order to handle the periodic signals in human activities, raw sensor signals are also transformed into the frequency domain by Discrete Fourier Transform or Discrete Cosine Transform, and different statistical features have been extracted, such as peak of DFT coefficients, energy, index of the largest frequency component, and signal power in different frequency bands [55, 56, 54, 57]. Some other techniques have also investigated both time and frequency characteristics of complex signals and employed wavelet coefficients and their energy as features [60]. To further enhance the discriminative power of the statistical features, several HAR systems combine the above time domain and frequency domain features to form a super feature vector. For example, Altun *et. al.* employed a 1170-dimensional feature vector by concatenating a collection of statistical features and further reduced the dimensionality to 30 by Principle Component Analysis [55]. Auguita *et. al.* separated body acceleration and body angular acceleration from the gravity acceleration and implemented 17 features from the time domain and the frequency domain and eventually created a 561-dimensional feature vector representing a human activity sequence [56, 61]. The proposed pFTA feature is significant different from all of the above traditional statistical features and addresses the problem of human activity recognition from a new perspective of temporal dynamics interpretation.

Classification

Most existing methods for HAR adopt off-the-shelf classifiers for the task of prediction. Unsupervised methods such as K Nearest Neighbor(KNN) and Dynamic Time Warping (DTW) [62] have been employed as straightforward solutions to classify feature vectors according to their distance. However, DTW often requires pairwise comparison between temporal order of human activity patterns, which are computationally demanding and unable to scale to a large dataset. Supervised methods, such as Artificial Neural Networks (ANN) [63], Naive Bayes [64], Decision Trees, Random Forests [65], and Support Vector Machine (SVM) [66], have also been widely adopted to train human activity models from labeled training data. The majority of existing classification methods are still shallow and with simple structures. Recurrent Neural Networks (RNN) such as Long Short-Term Memory [67] have proven to be very powerful in capturing the temporal dynamics in time series, and are extremely effective in time series classification inlcuding speech recognition [68] and multimodal translation [69]. More recently, LSTM has also been used in Human action recognition [70, 71, 72]. For example, differential Recurrent Neural Networks (dRNN) has been proposed to model the dynamics of human actions by computing different-orders of derivative of state that are sensitive to the spatio-temporal structure of input sequence [71]. A hierarchical architecture of RNN has been proposed to accumulate the results from multi-layer RNNs and eventually makes the decision by a single-layer perceptron [72]. Besides RNN, Convolutional Neural Networks have also been studied for human activity recognition from sensor streams. In [73], a hybrid end-to-end deep architecture, DeepConvLSTM, has been built from Convolutional Neural Networks (CNN) and LSTM Recurrent Neural Networks to model the temporal dynamics of human activities from wearable sensor signals and achieved state-of-the-art results. In general, LSTM encodes the temporal structures of activity patterns in their memory states, but how the memory states can be mapped to temporal order of patterns is unclear which limits their ability in fully capturing dynamic structures behind human activities.

CHAPTER 3: 3D SPATIAL RELATIONSHIP MODELING FOR HUMAN ACTIVITY DETECTION

This chapter presents the research on 3D spatial relationship modeling for the query-based human activity detection from the data stream captured by the Kinect sensor.

Introduction

Spatial relationship specifies how an object is located in space in relation to the reference object. Effective computation of spatial relationships between objects is an essential problem for many applications including robotics [74, 75, 16], video surveillance [23, 24, 26], and scene understanding [76, 77, 78].

Spatial relationships can be roughly classified into two groups, the topological relationships and the metric relationships [79]. The topological relationships have a large number of instances including *coincide, intersect, touch externally, touch internally, contains, inside, disjoint*, etc. A complete set of topological relationships were proposed in [80]. The metric relationships can be further split into distance relationships and directional relationships. Typical distance relationships include *at, nearby, in the vicinity, far away*, etc. Directional relationships are those operators specifying the relative or absolute spatial directions such as *left, right, in front of, at the back of, north, south, west and east*. Compared with the topological relationships and the metric relationships, the directional relationships remain under-researched [10]. In this chapter, we focus on the research on the directional relationships between two objects and assume one object does not contain or contact the other. In addition, we only deal with the indoor environment considering the limits of the depth camera.

Spatial relationship modeling has been investigated for human activity detectionin [23, 24], where a Live Video DataBase Management System (LVDBMS) was introduced and a Live Video Computing (LVC) framework was proposed to achieve the general event detection from a database point of view. A set of 2D spatial operators, along with the temporal operators and logical operators, were introduced and an SQL-like Live Video Query Language (LVQL) was defined to support the continuous query of the spatial-temporal human-centric event. This research extends the 2D spatial relationship modeling in [23, 24] into the 3D environment by leveraging the RGBD stream from the depth camera. It is very challenging to define and compute the 3D relationship between objects due to its fuzzy nature. In addition, depth data are normally noisy and incomplete with severe self-occlusions, making the 3D object modeling even more challenging. Last but not least, there are significantly more computational overhead for 3D data than 2D data which has a higher demand on the efficiency for real-time applications.

Note that a portion of this work, including the techniques and results, have previously been published by the author in conference [1] proceedings and in journals [2]. The contributions of the proposed research are as follows.

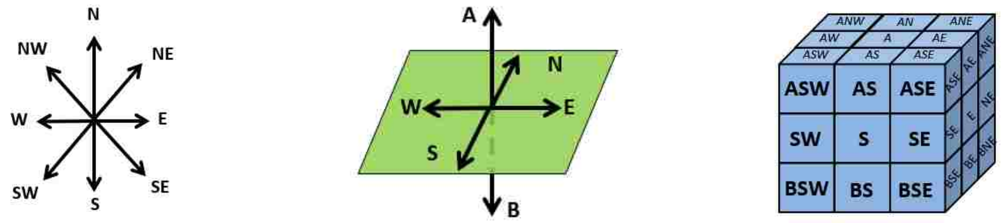
- We define a new set of 3D directional operators that can effectively represent the spatial relationships between objects in the 3D environment;
- We introduce a novel octree-based object modeling technique and propose the Ray-Tracing-like sampling-based technique to compute the 3D spatial relationship between objects of complex structure. A GPU-base implementation is developed to speed up the proposed algorithm for the real-time scenario;
- The performance of the proposed technique is quantitatively evaluated in a public 3D RGBD object dataset and we integrate the proposed technique in the LVDBMS prototype and investigate its performance on human activity detection in a real-time setting.

Proposed Solutions

In this section, we first introduce a set of 3D directional operators that will be used for the 3D spatial relationship modeling. A point cloud based 3D object modeling technique and its extension to the octree-based 3D object modeling algorithm are presented then. A sampling-based method to compute the directional spatial relationship between objects are finally proposed.

Definition of 3D Spatial Relationship

The directional relationship in the 2D space can be represented by either relative direction (*left*, *right*, *front* and *back*, or the global direction (*east*, *west*, *north* and *south*). In this research, we adopt the global term as our directional relationship system.



(a) The Eight Directions in 2D Plane (b) The Six Major Directions in 3D Space (c) 26 3D Directions Fitted in A 3×3 Cube

Figure 3.1: Extending 2D Directions to 3D Directions

As can be seen in Figure 3.1a, 2D directions are defined in a plane without distinguishing whether the objects are above the plane or below the plane. This will result in less accurate description when expressing the directions in the 3D space. To address this issue, an intuitive way is to add two major directions, "*above*" and "*below*" to the original 2D directions by the left hand rule as shown in Figure 3.1b, where A and B denotes the directions "*above*" and "*below*," respectively.

By combining any three orthogonal directions in Figure 3.1b, we can generate a set of 26 3D directional relationships. This set of the 3D directions can be perfectly fitted into a 3×3 cube as illustrated in Figure 3.1c, where the original eight 2D directions are preserved in the middle level of the cube and the extended directions are located in the upper and lower level. Specifically, the 16 new directions in the upper and the lower level are generated by concatenating A or B with the eight original 2D directions. For example, ANW means "*northwest*" in the "*above*" direction. Table 3.1 summarizes the set of 26 3D directions and groups them according to their locations with respect to the 2D horizontal plane. It is obvious that the 3D directions are a super set of the original 2D directions.

Table 3.1: The Set of 26 Primitive 3D Directional Relationships (grouped By Locations)

Above the 2D horizontal Plane	A, AE, ANE, AN, ANW, AW, ASW, AS, ASE
On the 2D horizontal Plane	E, NE, N, NW, W, SW, S, SE
Below the 2D horizontal Plane	B, BE, BNE, BN, BNW, BW, BSW, BS, BSE

3D Object Modeling

[htbp] In this subsection, we introduce the technique of the 3D object modeling by the RGBD data from a single Kinect sensor [81].

Cloud-based 3D Object Modeling

With the advancement of the sensing technology, commodity depth cameras (*e.g.* Microsoft Kinect) are now able to provide access to synchronized and aligned depth and RGB video streams in real-time. Figure 3.2 shows an example of the aligned RGB and color coded depth image from the Kinect. Each pixel in the depth image has a corresponding pixel in the RGB image. Therefore, each point in the point cloud can be represented by a 6-tuple data structure, (i, j, r, g, b, z_c), where i and j denote the column index and the row index of the pixel in the image, r, g and b denote the three channel color information of the pixel and z_c denotes the raw depth reading measured by the Kinect. The coordinate of each pixel (i, j, z_c) in the image can be further converted to a 3D coordinate in the world coordinate system according to the camera model [1]. Additionally, the 3D point cloud is rotated in the 3D space so that the floor plane in the depth image is parallel to the floor plane in the real world [2]. This is to compensate the camera orientation. Figure 3.3 illustrates the point cloud-based 3D model of the computer chair in an office view. The computer chair is segmented from the background according to its depth value.

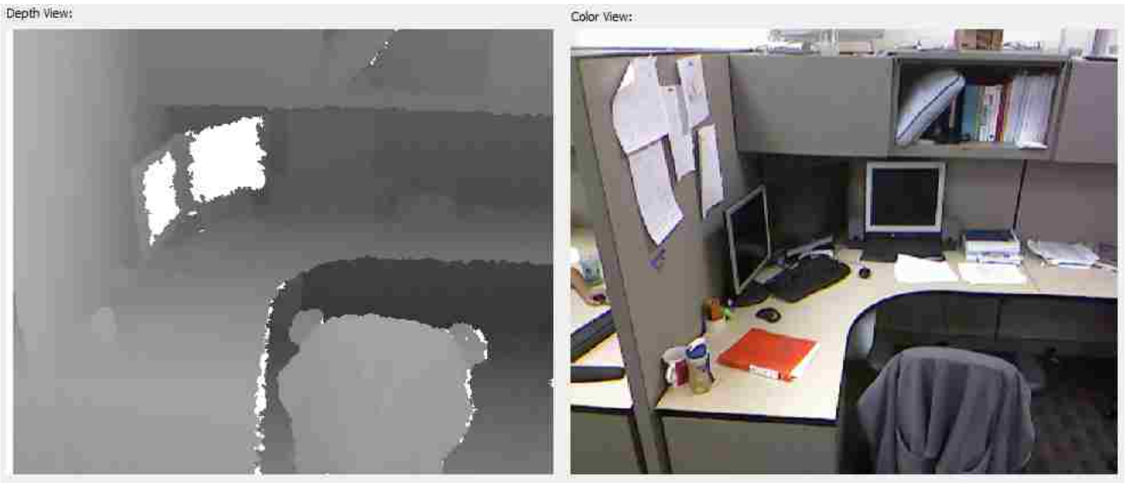
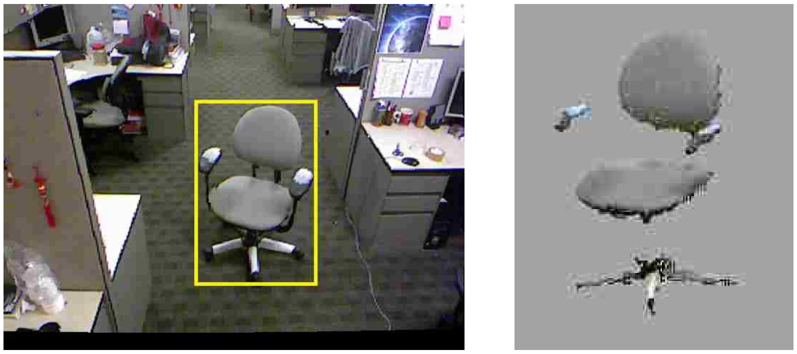


Figure 3.2: Examples of the Aligned RGB Frame and Depth Frame from the Kinect Sensor

Otree-based 3D Object Modeling

The aforementioned object modeling algorithm is based on the dense point cloud data which will inevitably incur heavy computational overhead when computing the spatial relationship between different objects. This is not scalable for large-scale real-time applications. To address this issue, we introduce a more compact data structure by employing the octree decomposition [82] of the original point cloud data. Therefore, the computational overhead is significantly reduced.



(a) Original Image

(b) The Point Cloud of the Extracted Objects after Rotation and Background Removal (The Frontal View)

Figure 3.3: Object Segmentation from the 3D Point Cloud

The octree is a space-dividing hierarchical tree structure for the quantized representation of 3D volumetric geometry. As shown in Figure 3.4, each internal node of the octree has exactly eight children, and it recursively subdivides the 3D space enclosed by the parent node. To construct the octree-based object model in the proposed technique, a bounding box is employed to enclose the object of interest in the RGB image. An axis-aligned bounding box (AABB) is then computed from the point cloud extracted by the 2D bounding box according to [22]. Figure 3.5a illustrates an example of the AABB of a computer chair. Finally, the AABB of the object is used as the root node of the octree and the decomposition process is recursively performed until the number of point in each subcube is less than a threshold which serves as the parameter to control the granularity of the model. It is obvious that, more levels of octree decomposition can produce an octree model of finer quality. However, the trade-off between the accuracy and the computational overhead must be considered for the real-time application scenarios. Those subcubes do not contain any point are removed from the model and the connected components of the remaining subcubes are extracted and clustered to give the octree-based representation of the object.

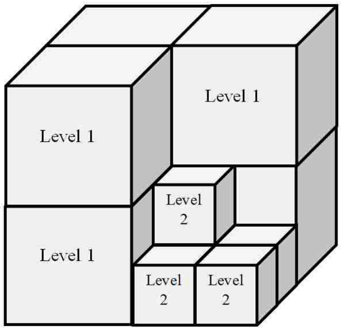


Figure 3.4: A Two-level Octree Model

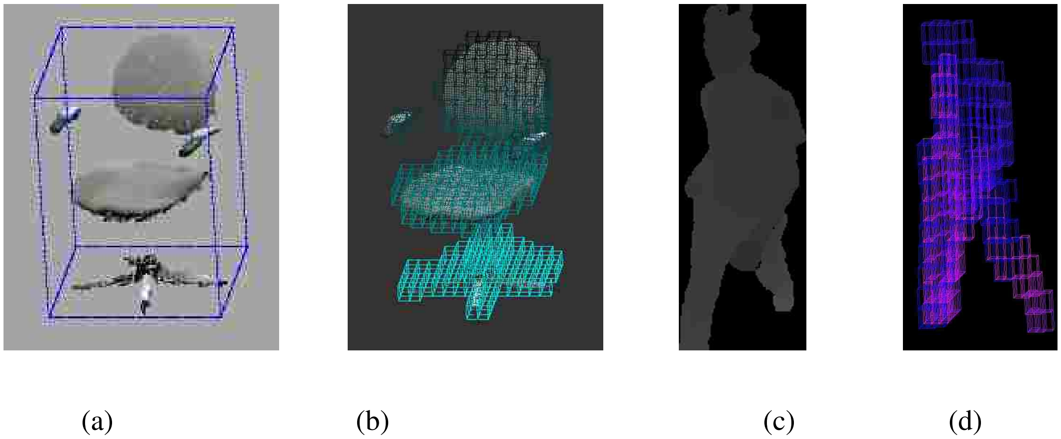


Figure 3.5: Octree-based Modeling. (a) Point Cloud-Based Chair Model, (b) Octree-Based Chair Model, (c) Human in the Depth Image. (d) Octree-Based Human Model.

Since the data is captured by a single Kinect sensor, the back side the objects are self-occluded by the front side of the objects, making the 3D object models incomplete. To deal with the self-occlusion, the backside of the object is filled up by a heuristic proposed in [2]. An example of the final octree models of an static computer chair and a human are illustrated in Figure 3.5. As can be seen, the octree models is able to provide a highly abstract representation of the object while preserve sufficient geometric structure of the object. Essentially, we use the subcubes of the octree modal to replace each individual pixel in the point cloud so that the computational overhead can be significantly reduced.

3D Relationship Computation

In this subsection, the algorithm for computing the spatial relationship between a target object and a reference object is introduced. Most conventional approaches assume object with simple shapes or structures, such as a ball or a box. While this is not always true in reality where objects like human beings can have very complex structure. Inspired by the Ray Tracing rendering technique [83] in the computer graphics, we propose a sampling-based method to compute the 3D directions which considers the sub-structure of the objects.

The general idea of the proposed algorithm is: we sample the entire space around the reference object by generating a number of rays from the reference object. Each ray represents one of the 26 3D directions. If the ray hits any subcube of the target object, the target object is partially in the direction represented by the ray. Finally, results of each ray are aggregated to form a feature vector indicating a global representation of the directional spatial relationships between the target object and the reference object.

Sampling the Space By Rays

Since we use rays origined from the reference object to sample the entire 3D space, it is crutial that the sampling process is uniform and evenly covers the space. To achieve this goal, we adopt the approach in [84], which achieves a uniform point sampling on the spherical surface. Therefore by connecting the origin and the sampling points, we generate rays that can uniformly sample the 3D space.

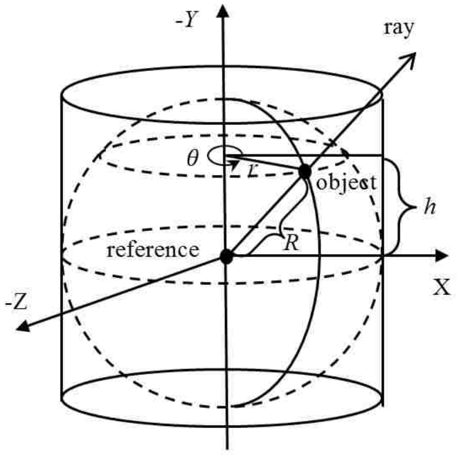


Figure 3.6: Uniform Spherical Surface Sampling

Figure 3.6 shows the sampling process. A random number, $h \in [-R, R]$, (R is the radius of the sphere), is used to create a plane parallel to the xoz plane with a distance of h. This plane will intersect with the sphere surface at a circle. A second random number, $\theta \in [0, 2\pi)$, further determines the position of a random point on the circle. In this way, a random point on the spherical surface is sampled and has the coordinate given by (3.1).

$$(x,\ y,\ z) = (\sqrt{R^2 - h^2}cos\theta,\ h,\ \sqrt{R^2 - h^2}sin\theta). \tag{3.1}$$

The random points generated in the above way have a uniform distribution on the spherical surface according to [84]. Following this way, the rays originated from the sphere center and pointing to the random points on the spherical surface therefore have a uniform distribution over the space enclosed by the sphere. The more rays we use, the higher sampling quality can be achieved. Considering the trade-off between the performance and the computational overhead, we uniformly quantize h and θ into 20 and 39 intervals, respectively and generate 840 rays. We further assign each ray one of the 26 3D directions according to the value of h and θ so that each direction will have approximately the same rays. The detailed assignment can be found in [2].

Ray-Tracing-Based 3D Direction Computation

For each subcube s in the reference octree model, a sphere sampling process is performed and 840 rays are originated from the centroid of the subcube to probe the entire space. A ray-subcube intersection condition is evaluated for each ray. Let's denote p_l^s as intersection result of the l_th ray originated from subcube s. p_l^s is set to 1 if the ray hit any subcubes of the target octree model and 0, otherwise. Results of rays falling into the same directions are aggregated and a 26-dimensional feature vector W_s is developed with each element w_h^s indicating the number of rays intersect with the target octree model in this direction. $W_s = [w_1^s, w_2^s, w_3^s, w_4^s, \cdots, w_{26}^s]$, where w_h^s is computed by (3.2),

$$w_h^s = \sum_{\substack{l=1,\\ map(l)=h}}^{840} p_l^s, \tag{3.2}$$

where $map()$ is a look up table that maps the l_{th} ray to its assigned direction. The above process is performed for each subcube of the reference object and the results at all reference subcubes are further aggregated to give a 26D vector W. W is further normalized by the L1 norm. In addition, a threshold is used to filter out the less salient relationships which are considered to be noise. This threshold is set to 0.1 empirically.

The proposed algorithm has two advantages. Firstly, instead of computing one spatial direction at a time, the proposed method can compute all directions at once, which is significantly more efficient than the conventional methods. Secondly, since each subcube of both the reference and the target objects are considered, such a subcube-to-subcube evaluation strategy can achieve a better global representation of the spatial directions between objects with complex structures.

GPU Implementation

Although the octree-based modeling can greatly reduce the computational overhead, the computation on the ray-subcube intersection on all pairs of subcube between the two objects still requires a significant amount of computation. In order to enable the real-time capability, we implement the the ray-subcube intersection module by the OpenCL [85] programming. In our implementation, the evaluation at each subcube of the reference object is computed in parallel in the OpenCL kernel. As a result, the overall time for computing the results at all subcubes of the refence object is almost equivalent to the time for computing a single subcube, and dramatic speedup is gained. We will show more results on the GPU speedup in the performance study section.

Discussion on the Proposed Solutions

The computational complexity of the proposed algorithm is analyzed and compared with the algorithms in the literature. Comparison results are summarized in Table 3.2. There are three steps in the the proposed algorithm: octree decomposition, ray sampling and result mapping. The complexity of octree decomposition is $O(n_p)$, where n_p the number of pixels in the point cloud of the object. In ray sampling, the complexity of a single ray-subcube intersection is $O(n_o)$. n_o is the number of subcubes of the object octree model. It will be repeated for all rays at each subcube of the reference object and the total complexity is of $O(N_{ray}n_o^2)$ where N_{ray} denotes the number of rays. This is the most time-consuming part. The third step maps the N_{ray} dimension results back to the 26 directions. So the complexity is $O(N_{ray})$. Putting all together, the complexity of the entire algorithm is $O(n_p) + O(n_o^2 N_{ray}) + O(N_{ray})$ which is equivalent to $O(n_o^2 N_{ray})$. Since $n_o \gg N_{ray}$ for most cases, the complexity is essentially quadratic to the number of subcubes, n_o which has a significantly lower complexity than those pixel-based methods.

The polygonal object approximation method [14] has a lower complexity than the proposed method. Nevertheless, it can only handle 2D directions. The morphology-based method [19] has a complexity of only $O(N_d n_c)$, where n_c is the number of points of the 3D contour but it requires extra computation to find the convex hull from the point cloud. As mentioned earlier, our algorithm can compute all the 26 directions at one run. Whereas all the other algorithms listed in Table 3.2 can only compute one direction at a time. That's why they all bear a constant N_d (number of directions) in the complexity.

Table 3.2: Comparison of the Computational Complexity

Methods	Complexity	Note
Proposed method	$O(N_{ray}n_o^2)$	
histogram of angles [13]	$O(N_d n_p^2)$	N_d = 26, N_d is the number of directions
morphology [17]	$O(N_d(1+2n_v)N)$	n_v is the size of neighborhood which is 26 in 3D space
morphology [18]	$O(N_d n_p^2 \sqrt{N})$	N is the number of pixels in the image
morphology [19]	$O(N_d n_c)$	n_c is number of points of the contour, need extra computation to compute contour
Polygonal object approximation [14]	$O(N_d n_v log(n_v))$	n_v is number of points of vertices of polygon

Performance Studies

To have an extensive performance evaluation on the proposed technique, we conduct the experiments in two different settings. Firstly, we evaluate the accuracy of the algorithm in a public RGBD image dataset. Secondly, we implement the proposed technique in the LVDBMS and demonstrate its performance in human activity detection in the live video query processing framework.

Experiments on RGBD Image Dataset

For quantitative performance studies, the proposed spatial relationship modeling algorithm is evaluated in a public RGB-D dataset [86]. This dataset consists of aligned RGB images and depth images captured by a Kinect sensor in eight different indoor scenes such as kitchen and living room. Each image contains several common household objects. We select 43 representing images from the dataset and evaluate the spatial directional relationships between 214 pairs of objects in those images. Note that the RGB-D dataset is originally established for 3D object recognition, therefore no ground truth (GT) of spatial relationship is provided. We manually labeled the GT by ourselves in [1]. It worth mentioned that there can be multiple spatial relationships existing between two objects at the same time. For example, W and NW can coexist between two objects. We adopt recall, precision and F-score as the performance metrics which are defined as follows:

$$\text{recall} = \frac{\text{num of correctly detected relationships}}{\text{total num of relationships in the GT}}, \tag{3.3}$$

$$\text{precision} = \frac{\text{num of correctly detected relationships}}{\text{num of detected relationships}} \tag{3.4}$$

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{3.5}$$

It is clear that increasing recall may result in the drop of precision. Meanwhile, increasing precision may also compromise recall. F_1 score serves as a balanced metric between recall and precision. A higher F_1 score represents higher overall performance.

We compare the performances of the proposed technique with other state-of-the-art solutions based on the above metrics and summarize the results in Table 3.3. From the perspective of spatial query, recall is a more significant metric than precision because normally, it is more interesting to know whether a given spatial relationship exists or not. Our proposed technique achieves 92.6% recall, significantly higher than all the other state-of-the-art methods listed in the Table. We notice the centroid-based method and the landmark-based method in [1] have a higher precision than the proposed methods at the price of a much lower recall. To achieve an overall understanding of the performance, we further compute the F_1 score from the recall and the precision. It shows that the proposed method achieves a significant higher score than the other methods, which demonstrates the superior performance of the proposed method.

Table 3.3: Experimental Results

Method	*Recall*	*Precision*	F_1	*Average Speed*
Centroid [1]	72.8%	**88.6%**	0.798	**41.8** ms
Landmark [1]	82.5%	84.3%	0.834	1370 ms
Morphology [17]	90.4%	75.8%	0.825	406.22 ms
Proposed technique	**92.6%**	79.6%	**0.860**	127.3 ms (non-GPU) 53.6 ms (GPU)

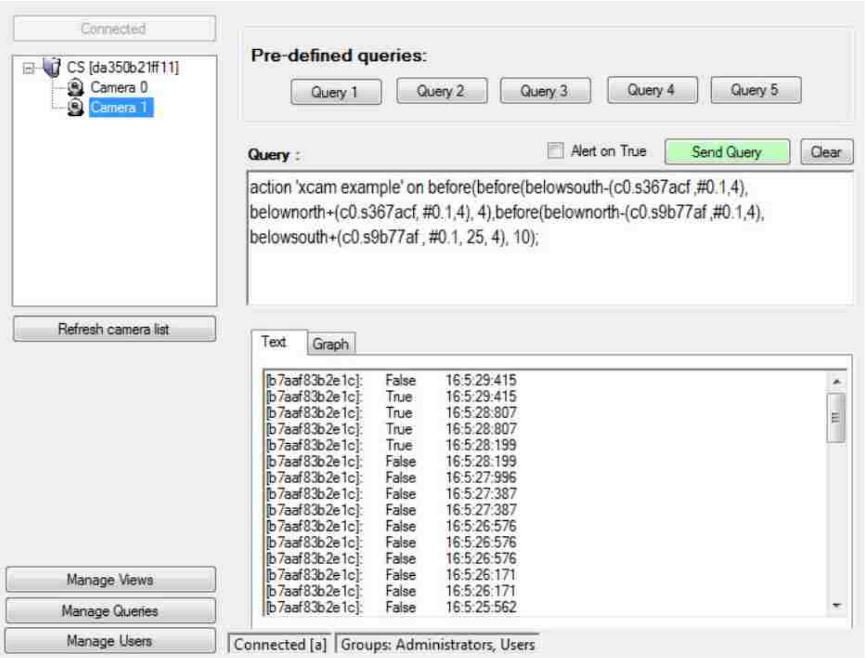
We also compare the performances on speed between the methods listed in Table 3.3. The speed is measured by the average time in millisecond to compute the 26D feature vectors between a pair of objects in one image. Hardware configuration is stated in [1]. Since other methods listed are not implemented with GPU, for fair comparison, we implement the proposed technique in two different versions, a non-GPU version and a GPU-accelerated version. As shown in Table 3.3, the speed of the non-GPU version is approximately 10 times faster than the landmark-based method and 3 times faster than the morphology method, which agrees with the analysis in complexity in the previous subsection. By leveraging the parallelism in the GPU-based programming, we further reduce the processing time from 127.3 ms to 53.6 ms which is almost comparable to the centroid-based method. Technically, the GPU-version is expected to give a much higher speedup due to the fact that the computation at all subcubes are fully in parallel. However, there is a slowdown due to the overhead for OpenCL kernel initialization. We expect a faster speed when processing videos because the resources initialization happens only once for the entire video. This has been demonstrated by the experiments on videos. The GPU version achieves a 25$\sim$27 FPS (frame per second) when processing a video with 640×480 resolution, that is 37ms $\sim$ 40ms for one frame. This justifies the explanation of the slowdown of the GPU version when processing the image data. We belief the speed of the proposed method is sufficient for the general-purpose real-time live video computing.

Validation in the LVDBMS Framework

In this subsection, we investigate the performance of the proposed spatial relationship modeling technique for human activity detection in the LVDBMS. LVDBMS was originally developed by Peng and Aved, *et. al.* in [23, 24] as a general platform for Live Video Computing (LVC) application development. It treats the video feeds from live cameras as a special kind of database and supports the continuous queries to retrieve video segments that satisfy certain query predicates defined in terms of spatial, temporal and logical operators. An SQL-like Live Video Query Language (LVQL) was defined describe various spatial-temporal events in the video stream. More details of the flatform can be found in [23, 24]. In this dissertation, we are particularly interested in its application in human activity detection by querying the spatial and temporal events in the live videos. We extend the original 2D spatial operators in the LVDBMS with the proposed 26 3D spatial spatial relationships.

The spatial operators have the syntax, $Operator(Operand1,\ Operand2,\ Threshold)$. $Operand1$ is the reference object; $Operand2$ is the target object. Both operands can be either dynamic object or static object. A threshold in terms of centimeters is used to specify the minimum distance between the two objects in the direction. As an example, $East(Operand1,\ Operand2,\ 20)$ represents $Operand2$ is to the east of $Operand1$ and their distance in the east direction must be at least 20 cm away. The value of the operator $East$ is evaluated continuously in real-time according to the 3D spatial relationship modeling methods. At each point of time, it output "1" if the direction "*east*" is detected and "0" otherwise. We further extend the 3D spatial operators to support the dynamic trend of the two objects by adding the symbols of "$+$" and "$-$". For example $East-$ means the target object is to the "east" direction of the reference object and it is moving towards the reference object. Similarly, $East+$ shows that the target object is moving away from the reference object in the east direction.

A screen shot of the graphic user interface of LVDBMS is shown in Figure 3.7. To support the continuous live event detection, each query is executed every 250 ms by the query process server in the LVDBMS and outputs either "*True*" (T) or "*False*" (F) with T meaning the event is detected on the current time and F otherwise. In this experiment, we show three different examples to demonstrate the effectiveness of the spatial relationship modeling for human activity detection.



```
action 'xcam example' on before(before(belowsouth-(c0.s367acf ,#0.1,4),
belownorth+(c0.s367acf, #0.1,4), 4),before(belownorth-(c0.s9b77af ,#0.1,4),
belowsouth+(c0.s9b77af , #0.1, 25, 4), 10);
```

```
[b7aaf83b2e1c]:   False   16:5:29:415
[b7aaf83b2e1c]:   True    16:5:29:415
[b7aaf83b2e1c]:   True    16:5:28:807
[b7aaf83b2e1c]:   True    16:5:28:807
[b7aaf83b2e1c]:   True    16:5:28:199
[b7aaf83b2e1c]:   False   16:5:28:199
[b7aaf83b2e1c]:   False   16:5:27:996
[b7aaf83b2e1c]:   False   16:5:27:387
[b7aaf83b2e1c]:   False   16:5:27:387
[b7aaf83b2e1c]:   False   16:5:26:576
[b7aaf83b2e1c]:   False   16:5:26:576
[b7aaf83b2e1c]:   False   16:5:26:576
[b7aaf83b2e1c]:   False   16:5:26:171
[b7aaf83b2e1c]:   False   16:5:26:171
[b7aaf83b2e1c]:   False   16:5:25:562
```

Connected [a] | Groups: Administrators, Users

Figure 3.7: The Graphic User Interface of LVDBMS

Example 1: Event with One Person and One Static Object

In this example, we show the query example of a simple event between a static object and a human being. The event of a person passing by a computer chair (the reference object) is captured by a Kinect, and their spatial relationships in terms of 3D directions are evaluated in real-time. We show some key frames of the above event in Figure 3.8a. Those key frames are manually selected for demonstration purpose only. The following query is used to represent the event.

$$Before(ANW(c1.s6745bb,\ c1.@),\ ASW(c1.s6745bb,\ c1.@),\ 4),$$

where a temporal operator "*before*" and two spatial operators "*ANW*" and "*ASW*" are used. In the LVQL syntax, c1.sxxxxxx denotes the static object (computer chair) in camera 1 which is manually selected by drawing a bounding box around the object in the RGB image. The six-digit number after “c1.s” is a randomly generated ID to uniquely identify the static object in LVDBMS. Symbol “c1.@” refers to the dynamic object automatically tracked by the Kinect SDK in camera 1. During the query processing, the temporal operator “*before*” takes the output of the two spatial operators, ANW and ASW as the input and evaluates the order within a 4-frame window. If “*ANW*” is detected before “*ASW*” in a 4-frame window, the entire query will output true.

The query results are summarized in Table 3.4, where the final query results as well as the intermediate results of individual operators are shown at each point of time. In the 3-second-long event, the query is evaluated for 10 times. “ANW” generates several positive results in the first part of the event while “ASW” outputs positive results at the second half of the event. The event has been detected for three times when “ANW” occurs before “ASW” in a window of 4. The query results perfectly match the human observation.

Table 3.4: Query Result of Example 1

Time ($s : ms$)	58:80	59:10	59:20	59:40	59:80	0:14	0:21	0:41	0:81	1:18
Query result	F	F	F	F	T	T	T	F	F	F
Direction (ANW)	0	0.3811	0.3687	0.3788	0.0670	0	0	0	0	0
Direction (ASW)	0	0	0	0.0077	0.3666	0.4221	0.4032	0	0	0

Example 2: Cross Camera Event

In this example, we show a composite event involving multiple objects across different cameras. Since the LVDBMS utilizes distributed live cameras at different locations, it is common and essential to query events occurring simultaneously or sequentially in different cameras during a period of time. To show the capability to handle the cross camera event, we show an event that a person exits from a room and enters the room next door (See Figure 3.8b). This is almost the most common event in the indoor scene. We use the following query to interpret the event.

$$\begin{aligned}
&Before(\\
&Before(BS^{-}(c1.s367acf,\ \#1.2),\ BN^{+}(c1.s367acf,\ \#1.2),\ 4), \\
&Before(BN^{-}(c2.s9b77af,\ \#1.2),\ BS^{+}(c2.s9b77af,\ \#1.2),\ 4), 10).
\end{aligned}$$

The above nested query has three levels. At the bottom level, spatial operators BS^{-}, BN^{+}, BN^{-} and BS^{+} are computed to generate the spatial relationship between the person and the door at each point of time; at the middle level, the temporal operator "*before*" is employed to compute the sub-event of exiting and entering the doors; Finally at the top level, another temporal operator "*before*" is used to evaluate the temporal order of the two sub-event and eventually determines the entire query result. The symbol "#1.2" denotes the same dynamic object appears in camera 1 and 2. We use a distributed in-memory image retrieval system [24] to perform the cross-camera object tracking.

Table 3.5: Query Result of Example 2

Time $(s : ms)$	0:25	0:65	0:86	1:06	1:46	1:67	1:87	2:07	2:48	2:68	2:88	3:29	3:49	3:69	4:10	4:30	4:51	4:91
Query result															T	T	T	
BS^{-}		T	T	T														
BN^{+}					T	T	T											
Exit					T	T	T											
BN^{-}													T	T				
BS^{+}															T	T	T	
Enter															T	T	T	

Query results at each point of time are shown in Table 3.5. For better data representation, we only show the result of T and omit all the F with blank. In this 5-second-long event, LVDBMS evaluates the query 20 times. In the first half of the event, the sub-event of exiting from the room is detected and three Ts are produced; in the second half, the sub-event of entering another room is detected and another three Ts are generated. The top-level operator $before$ observes that exiting through the door appears earlier than entering the door within a 10-frame window and produces T accordingly. This example demonstrates the capability of the 3D spatial operators in detecting the cross-camera events.

Example 3: Event of Interactions Between Two Persons

All of the above query examples only covers events involving one person and one static object. Spatial relationships between two persons are also common in many events with multiple persons interactions. We give a demonstration of an event involving two persons. Figure 3.8c demonstrates the event of two persons walking towards each other in the hallway. They meet in front of the camera and keep walking out of the view. We use a simple query $South(c1.*,\ c1.*, 20)$ to detect the event, where $c1.*$ denotes any dynamic object in camera 1.

This query captures the moment these two persons are overlapping with each other from the view of camera. At this point of time, one person must be to the "*south*" of the other person and the operator "*south*" can detect the event. Furthermore, we use the threshold "20" to ensure the two people are at least 20 cm away from each other and there is not contact between them. Query results are summarized in Table 3.6. As can been seen, the direction value and the distance reach maximum and minimum, respectively, when the two people meet each other in front of the camera. This example validates the effectiveness of the spatial operators when dealing with two dynamic operands. It also shows that 3D spatial operators can effectively detect whether two visually overlapping objects contact each other or not by exploiting the depth.

Due to the space limit of the dissertation, we only demonstrate three simple events for illustration purposes. The 3D spatial operators, along with the LVQL, is capable of interpreting more complex human activities with more sophisticated spatio-temporal queries.

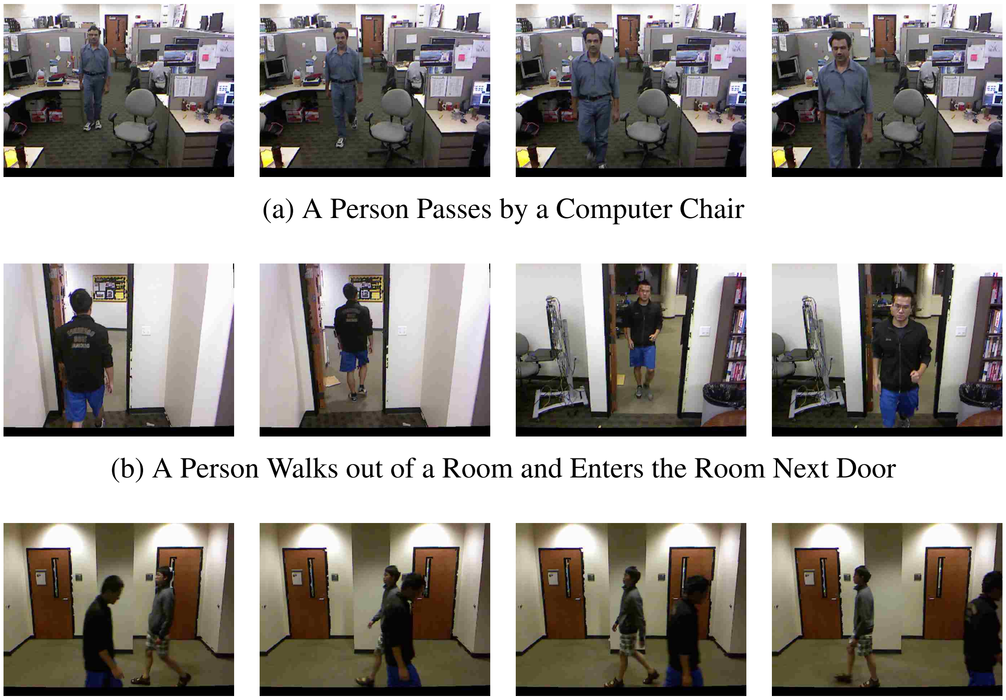
Table 3.6: Query Result of Example 3

Time $(s : ms)$	51:54	51:74	52:14	52:34	52:35	52:55	52:75	52:95	53:35
Query result	F	F	F	F	T	T	F	F	F
Direction (South)	0	0	0	0	0.1994	0.3114	0.0153	0	0
Distance (cm)	0	0	0	0	100.4	91.9	105.3	0	0

Summary

In this chapter, we investigated the problem of human activity detection from the perspective of 3D spatial relationship modeling and query-based event detection. We presented an octree-based 3D spatial modeling for depth cameras, and proposed a uniform sampling-based algorithm for computing the 3D directional spatial relationships between two objects. To achieve the applicability for real-time human activity detection, we also developed a GPU-based implementation of our techniques. Extensive experiments based on a public RGBD dataset as well as demonstrations in the LVDBMS platform have verified the superior performance and efficiency of the proposed algorithms.

The proposed techniques models the human activity from a spatial relationship modeling perspective of view. It can effectively recognize those spatial direction related activities such as "enter a room", "walk and turn right" and "climb upstairs." Nevertheless, it is not capable to detect a vast majority of actions that do not contain any spatial information such as “cooking”, “reading a book” and “making phone calls.” To handle more general categories, we plan to explore the semantic-level human action recognition in the future.



(a) A Person Passes by a Computer Chair

(b) A Person Walks out of a Room and Enters the Room Next Door

(c) Two People Meet in the Hallway

Figure 3.8: Key Frames of the Human-centric Events in the Demonstration

CHAPTER 4: DYNAMIC TEMPORAL QUANTIZATION FOR 3D HUMAN ACTION RECOGNITION

Motivation

Recent commodity depth sensors such as Microsoft Kinect, Leap Motion and Intel RealSense, have been widely used in a variety of applications including video gaming [87], education [88] health [89] and surveillance [23, 24, 2], to name a few. One of the key technologies behind these applications is the human action recognition which directly determines the quality of the user experiences. The depth cameras provide real-time multimedia data streams in terms of the RGB image, depth images and human skeleton joints and shed light on the conventional human action recognition community. With the help of the depth cameras, human actions/gestures can be captured and represented in the 3D environment where more comprehensive spatio-temporal information can be explored for higher recognition accuracy.

Most of the existing methods on 3D human action recognition explore the local spatio-temporal features [27, 44, 45, 46] of the action sequences and represent the video by the bag-of-word feature descriptors [34, 39, 40, 36]. Although some of the above algorithms have achieved superior performances, they are skeptical to actions of similar body postures but different temporal order. As an example, “put on a hat” and “take off a hat” are two actions with exactly the same postures but the reversed order. These methods may easily get confused by such pairs of actions because they don’t leverage the global temporal information of the action sequences. Some other methods like the graph model-based methods [47, 48, 49] and the motion template-based methods [50, 42], have also been proposed from the perspective temporal modeling. Nevertheless, these methods cannot effectively interpret the internal temporal dynamics of human actions. In addition, they normally suffer from the temporal misalignment due to the temporal translation and execution rate variations [42] among different human action instances. As an example, people may perform the action “golf swing” at different speed. Some people are faster, others are slower. All these factors make the temporal modeling for human action recognition very challenging. This research aims to address the challenges of temporal modeling from a quantization point of view.

An Overview of the Proposed Techniques

To address the aforementioned challenges, we formulate the temporal modeling problem from a new perspective which aims to find the optimal temporal quantization of the sequence. To solve the optimization problem, we propose a temporal order-preserving dynamic quantizing algorithm. It is similar to the idea of applying the KMeans clustering on the temporal direction so that similar postures are clustered together. But different from the spatial clustering, the temporal order of the frames must be preserved during the temporal quantization. Figure 5.1 illustrates the general framework of the proposed approach. The human action video is dynamically partitioned by the temporal quantization in a hierarchical way. Each partition summaries a collection of similar postures that are temporally close to each other. Frames in each partition will be aggregated to produce a quantized vector to represent the entire partition. By concatenating the quantized vector of all partitions at all levels, a quantized representation of fixed size can be generated from the original video which can be later used by any classification method. The above quantization process can be applied with different features (e.g. RGB, depth and skeleton joints) and the the final recognition results can be further enhanced by fusing the results of multiple features.

The main contributions of this work are:

1. We present a novel perspective for the temporal modeling of human action sequences from

the quantization point of view and propose a temporal order-preserving dynamic quantization algorithm as the solution;

2. The proposed algorithms are evaluated on three public 3D human action datasets and achieve state-of-the-art performance.

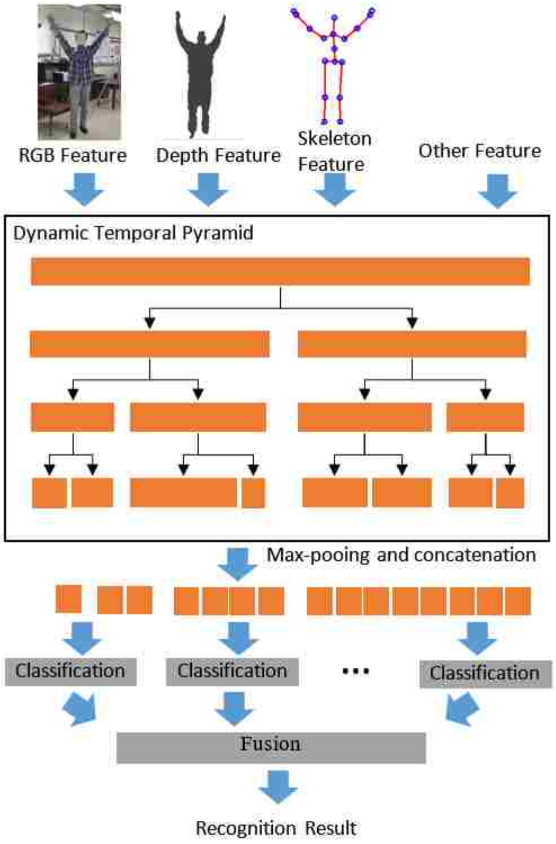


Figure 4.1: The General Framework of the Dynamic Temporal Quantization

Dynamic Temporal Quantization

In this section, we present the details of the temporal-preserving dynamic quantization method for human action recognition.

Problem Formulation

The goal of the Dynamic Quantization is to achieve an optimal partitioning of the video that matches the temporal dynamics of the action sequence. Eventually, we wish to represent the original video by a sequence of quantized vectors extracted from the above partitions. Two restrictions must be followed during the quantization: 1) frames with similar human body postures must be clustered together, 2) the temporal order of the sequence must be preserved.

More formally, we formulate the problem as follows. Denote $S = \{s_1, s_2, \cdots, s_n\}$ as an action sequence with n frames. n can vary across different videos. S can be further represented by a sequence of feature vectors $X = \{\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_n}\}$, where $\mathbf{x_i}$ is the feature vector extracted from frame s_i. The goal is to dynamically quantize X into a new sequence $V = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_m\}$ of fixed size m. Each element of feature vector $\mathbf{v}_j$ represents an unknown hidden stage of a category of human action, whose temporal order is preserved. We use $a_i \in \{1, 2, \cdots, m\}$ to denote a frame $\mathbf{x}_i$ in the feature vectors X is assigned to the quantized vector $\mathbf{v}_{a_i}$ in V. Obviously, the assignments of any two consecutive frames, a_i and a_{i+1}, should satisfy $a_i \leq a_{i+1}$.

Then, a natural way to jointly optimize the assignment vector $\mathbf{a} = \{a_1, a_2, \cdots, a_n\}$ and the quantized sequence V can be achieved by jointly minimizing the following objective,

$$\min_{\mathbf{a},V} \sum_{i=1}^{n} \|\mathbf{v}_{a_i} - \mathbf{x}_i\|^2, \tag{4.1}$$

$$s.t.\ \ \forall i \in [1, n-1],\ \ a_i \leq a_{i+1}, a_i \in [1, m]$$

where $\|\cdot\|$ can be any distance measurement. For simplification, we use Euclidean distance in our proposed solution.

Optimization and Implementation

It is nontrivial to jointly solve the optimal assignment $\mathbf{a}$ that satisfies the temporal order-preserving constraint in each video, along with the optimal quantized sequence V. Our solution is to break down this optimization problem iteratively in a coordinate descent fashion. Two steps, an Aggregation step and an Assignment step, are executed in each iteration until convergence.

Aggregation step: given the assignment $\mathbf{a}$, it is not difficult to show that each element $\mathbf{v}_j$ of the optimal quantized sequence is the mean vector of all the elements $\mathbf{x}_i$ assigned to the j_{th} partition. $\mathbf{v}_j$ provides a highly discriminative representation of the partition j. We have

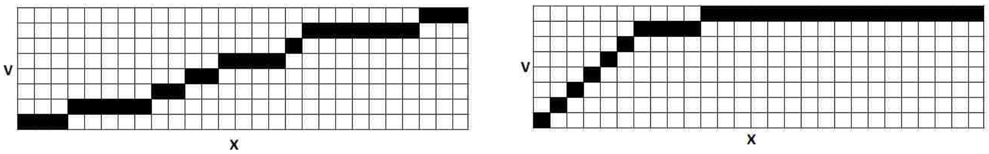
$$\mathbf{v}_j = \frac{1}{|\{a_i = j\}|} \sum_{a_i = j} \mathbf{x}_i \tag{4.2}$$

where $|\cdot|$ is the set cardinality and $\{a_i = j\}$ is the set of elements in $\mathbf{a}$ whose value is j. The above aggregation by average is robust to noise and is theoretically optimal under Euclidean distance. Nevertheless, some salient human postures may be mitigated by the mean pooling and therefore the discriminative power of the quantized vector $\mathbf{v}_j$ may be compromised. To address this issue, other aggregation methods such as min-pooling and max-pooling can also be considered.

Assignment step: when V is fixed, the assignment $\mathbf{a}$ can be updated to minimize the above distance to these quantized vectors in V subject to the temporal order-preserving constraint. Inspired by the dynamic time warping (DTW) algorithm, We are develop a dynamic programming approach to solve this subproblem. Specifically, the minimal distance $D_{l+1,k+1}$ given by the best assignment from $[\mathbf{x}_1 : \mathbf{x}_{l+1}]$ to $[\mathbf{v}_1 : \mathbf{v}_{k+1}]$ can be induced by the following iterative equation

$$D_{l+1,k+1} = \min\{D_{l+1,k}, D_{l,k}, D_{l,k+1}\} + \|\mathbf{x}_{l+1}, \mathbf{v}_{k+1}\|^2, \tag{4.3}$$

where $\|\mathbf{x}_{l+1}, \mathbf{v}_{k+1}\|$ is the distance between the current video frame $\mathbf{x}_{l+1}$ and the current quantized frame $\mathbf{v}_{k+1}$. Then by starting with $D_{1,k} = \min\limits_{k} \|\mathbf{x}_1 - \mathbf{v}_k\|^2$ and $D_{l,1} = \min\limits_{l} \|\mathbf{x}_l - \mathbf{v}_1\|^2$, the best assignment can be found iteratively according to the above equation. The resulting assignment has the same characteristics of DTW [50]. It always measures the closest frame when choosing the next warping step, and the warping path is guaranteed to be non-decreasing in the temporal order.



(a) A Balanced Warping

(b) Warping Going Too Fast on the Quantized Vector V

Figure 4.2: Examples of Good and Bad Warping

We note that m is much smaller than n and the original feature vector X. Therefore the feature vectors X are normally warped to a much shorter quantized vector V. In order to prevent the warping going too fast on the direction of the V, which will result in the effect that most of the frames are assigned to the last portion of V. Figure 4.2b illustrates an example of such sceneario. We further restrict the step size of the warping path in (4.3) to $(1, 0)$ and $(1, 1)$ and developed a more restricted warping as

$$D_{l+1,k+1} = \min\{D_{l,k+1}, D_{l,k}\} + \|\mathbf{x}_{l+1}, \mathbf{v}_{k+1}\|^2. \tag{4.4}$$

(4.4) also avoids the scenario of assigning one frame to multiple quantized vectors at the same time.

Algorithm 1 Iterated Dynamic Quantizing Algorithm

```
Input video length$m$, feature vectors $X = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$,
Output $\mathbf{a} = \{a_1, a_2, \cdots, a_n\}, V = \{\mathbf{v}_1, \cdots, \mathbf{v}_m\}$
procedure IDQA($X, m$)
    Initialize assignment $\mathbf{a}$ by the uniform partition.
    repeat
        $V = Aggregation(X, V)$
        $\mathbf{a} = Assignment(X, V)$
    until Convergence
    return $\mathbf{a}, V$
end procedure
```

The pseudo code of the proposed Iterated Dynamic Quantizing Algorithm is presented in Algorithm 1. In the initialization, S is uniformly split into m partitions, each of which is assigned to an element $\mathbf{v}_j$ of V. An *Aggregation* subprocedure is exploited to update the quantized sequence V according to the assignment $\mathbf{a}$. Then, a *Assignment* subprocedure is used to warp the original feature X to the new quantized sequence V. We propose a modified DTW algorithm for the warping process. The assignment $\mathbf{a}$ can be updated from the warping path. The above two steps iterate until V converges or the maximum number of iteration is reached. The output is the final quantized sequence V and the final frame assignment $\mathbf{a}$. The above iteration algorithm modifies the initial quantization result step by step by the warping and the aggregation and eventually generates a quantization result that not only considers the similarity of human body postures but also preserves the frame order.

Hierarchical Representation

Inspired by the Spatial Pyramid [90] and the Temporal Pyramid [41], we further extend the Dynamic Temporal Quantization by incorporating the hierarchical structure. Figure 4.3 illustrates the hierarchical architecture of the proposed dynamic quantization. The original sequence is recursively partitioned by the Iterated Dynamic Quantizing Algorithm and forms a pyramid structure. The i_{th} level has $2^i - 1$ partitions. As a result of the dynamic quantization, the length of each partition varies. A quantized vector is then aggregated from each partition and eventually the final feature vector is generated by concatenating the quantization vector of all layers. Figure 4.3 shows a pyramid structure of 4 levels. The proposed technique may benefit from a structure with higher levels. Nevertheless, a large number of quantizations may over-segment the action sequence and compromise the generalization capability. We study the relationship between the height of pyramid and performance in the experiments.

The proposed dynamic quantization approach has several benefits. First, by exploiting the dynamic temporal quantizing of the sequence, it can address the challenge of execution rate variation by achieving a dynamic temporal quantization. Second, feature vectors with highly discriminative capability can be extracted by the aggregation procedure of the quantization. Third, the hierarchical description in terms of the multilayer pyramid achieves a comprehensive representation of the temporal dynamics of the action by capturing both the global and local temporal patterns of the action.

Frame Assignment Sharing Between Multiple Features

In the previous section, the Dynamic Temporal Quantizing Algorithm is computed based on the original feature of the action sequence. All of the above four features can be used to compute their own temporal quantization of the video frames. Should the frame assignment of the quantization be computed by one feature and shared by all other features or should it be done independently? Intuitively, the independent quantization strategy may produce the best classification result with respect to individual feature that has been used. Nevertheless, it may encounter the overfitting problem and compromise the generalization capability when multimodal features are fused. Therefore, we compute the frame assignment of the quantization based on the feature with the highest discriminative capability. This assignment is applied to all the other three features to compute their own quantization results. We performed experimental study to justify the benefit of the sharing strategy and will discuss the result in the experiment section.

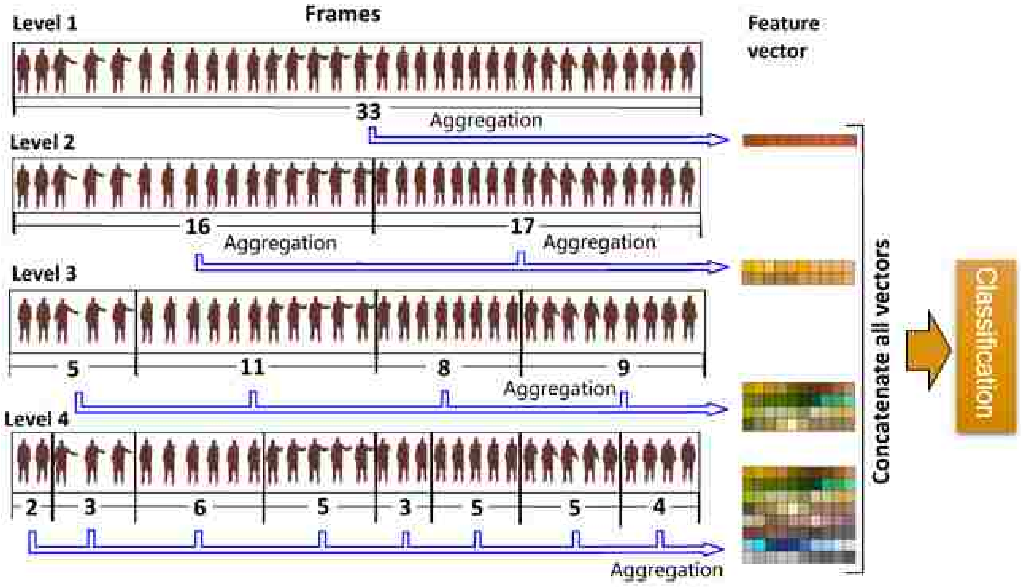


Figure 4.3: Illustration of the Dynamic Temporal Quantization with the Hierarchical Representation

Performance Studies

Experiment Setup

We evaluate the performance of the proposed method on three public datasets: UTKinect-Action [91], MSR-Action3D [34] and MSR-ActionPairs [36]. We choose these datasets because they provide data from at least two modalities and satisfy our multimodal feature fusion framework. In the experiment, the levels of hierarchical quantization structure is set to 4 and 15 partitions are generated to represent the entire action sequence. The PCA dimension reduction is set to preserve 99.5% of the energy of the raw feature. These parameters are tuned according to the performances in the training set. We use the LibSVM [92] with the RBF kernel as the classifier in our experiments. The score-level fusion computes the weighted sum of the predictions of each label from individual features according to the sum rule [93]. The label of the action category with the maximum fused probability are used as the final recognition result.

To leverage the discriminative capability from different data modalities, we use the following features in the proposed approach.

1. **Position:** 3D coordinates of the 20 joints of the skeleton captured by the Kienct sensor [94].
2. **Angle:** normalized pairwise-angle feature [95].
3. **Offset:** offset of the 3D joint positions between the current frame and the previous frame [40].
4. **Velocity:** histogram of the velocity components of the point cloud around the 20 joints [95].

The above four features cover all three modalities of the sensing data (RGB, depth and joints) from the Kinect sensor and provide a rich feature representation of the human action sequences.

MSR-Action3D Dataset

The MSR-Action3D dataset [34] contains 20 action classes and 10 subjects. Each subject performs each action two or three times. The 20 action types are chosen in the context of gaming. They cover a variety of movements related to arms, legs, torso, etc. The noise of the joint locations in the skeleton as well as the high intra-class variations and the inter-class similarities make the dataset very challenging. As an example, the action “Draw x” is easily confused with the action “Draw tick”. We followed exactly the same experiment settings of [41], that all 20 action classes are tested in one group. Half of the subjects are used for training and the rest are for testing. We note there is another experiment setting used in the literature which splits the 20 action types into three subsets and only performs the evaluation within each subset [34]. The experiment setting we followed is more challenging than the subset one because all actions are evaluated together and the chance of confusion is much higher.

Table 4.1: Impact of the height of Quantization Pyramid in the MSR-Action3D Dataset

Levels	5	4	3	2	1	0
Accuracy (%)	77.39	81.61	71.26	67.82	66.28	73.56

To evaluate the impact of the height of the quantization pyramid on the recognition accuracy, we perform experiments on different levels of pyramid on the position feature. Results are summarized in Table 4.1. In this experiment, zero to five levels of pyramid are evaluated. level 0 denotes no hierarchical structure is employed. The entire sequence is quantized into 8 partitions by the proposed Iterated Dynamic Quantizing Algorithm. It can be regarded as the leaf-nodes of the 4-level pyramid. Two observations can be made from the results. First, recognition accuracy increases as the increase of the height. However the performance begins to drop when the height reaches five. This is because the higher levels of the pyramid, the deeper hierarchical structure can be captured to describe the temporal dynamics. Therefore classification rate keeps increasing. Nevertheless, when the sequence is over segmented into too many small partitions, the method becomes less generalized and the turning point occurs. Second, the accuracy of the 4-level pyramid

(81.61%) is higher than that of using only the leaf-nodes (0-level, 73.56%). This has demonstrated that the hierarchical structure has contributed to the dynamic quantization of the sequence. The inclusion of the upper layers has enhanced the generalization capability of the modeling method.

Table 4.2: Impact of the Dynamic Temporal Quantization and the Frame Assignment Sharing in the MSR-Action3D Dataset

Feature	Accuracy		
	Proposed	Deterministic quantization	w/o assignment sharing
position	81.61%	76.24%	81.23%
angle	73.95%	71.65%	72.41%
offset	73.95%	68.20%	64.75%
velocity	80.84%	72.80%	80.08%
fused result	90.42%	83.15%	88.51%

To validate the effectiveness of the Iterated Dynamic Quantizing Algorithm, we compare it against a deterministic quantizing method that always evenly splits the sequence. This is the same method used in the Fourier Temporal Pyramid [41]. Experiments results are summarized in Table 4.2. The second column and the third column show, the proposed algorithm with the dynamic quantization has a higher accuracy than the deterministic quantization method on all individual features as well as the fused result. Such performance increase demonstrates the advantage of the dynamic temporal quantization over the deterministic quantization method.

We also evaluate the performance of the frame assignment sharing strategy and compare it against the independent strategy. In the proposed method, the frame assignment of the quantization is computed with the position feature. The fourth column of Table 4.2 shows the performances of the strategy without the frame assignment sharing. We can see that the performance of the independent strategy is lower than the proposed method with the sharing. Other features which are less discriminative than the position feature benefit from the assignment computed based on the position feature. This has demonstrated the advantage of the frame assignment sharing strategy.

Last but not least, the fused result is significantly higher than the recognition rates of all individual features in all three columns in Table 4.2. One possible explanation is the multimodal features have complimentary discriminative capabilities which can be leveraged by the fusion and therefore yield superior performance over individual features alone. As an example, the position feature, in terms of joint coordinates, is good at describing the human pose from the global point of view. Meanwhile, the velocity feature based on the 3D point cloud is good at capturing the micro movements of the body parts. Therefore these two features from different modalities are complimentary to each other and can generate enhanced performance when fused.

Table 4.3: Comparison with State-of-the-arts in the MSR-Action3D Dataset

Method	Accuracy
Actionlet Ensemble [41]	88.2%
HON4D [36]	88.89%
DCSF [46]	89.3%
Lie Group [43]	89.48%
Super Normal Vector [51]	93.09%
Proposed approach	90.42%

We list state-of-the-art approaches in the recent years in Table 4.3 for comparison. As can be seen, the proposed method achieves a higher recognition rate than most of state-of-the-art results. The only method achieves a higher accuracy than ours is the super normal vector method [51].

Figure 4.4 shows the confusion matrix of the classification results in the MSR-Action3D dataset. Most of the confusions between similar actions have been correctly addressed. The remaining confusions are between actions containing very similar primitive postures such as “Side kick” and “Forward kick” and “Draw x” and “Draw circle”.

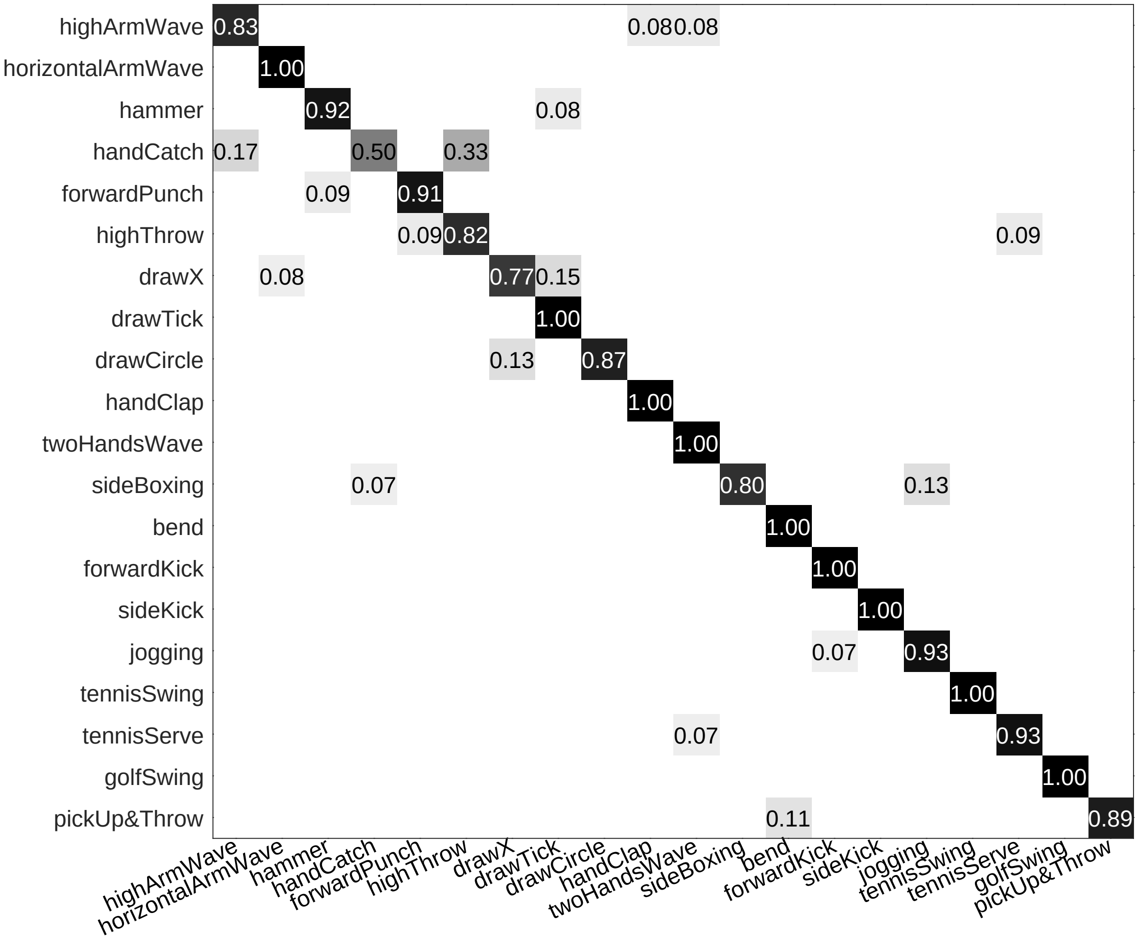


Figure 4.4: Confusion Matrix of the Classification Results in the MSR-Action3D Dataset

MSR-ActionPairs Dataset

The MSR-ActionPairs dataset [36] is composed of 12 actions performed by 10 subjects. Each subject performs all actions three times. Therefore, the dataset contains 360 action sequences in total. Different from the other two datasets, this dataset contains 6 pairs of actions. Each pair of actions have exactly the same primitive postures but the reversed temporal order. As an example, “Pick up” and “Put down”, “Put on a hat” and “Take off a hat”. This dataset is collected to investigate the effects of temporal order on the recognition of the actions. The huge within-pair similarity makes the dataset very challenging. Therefore, histogram-based temporal modeling methods relying on the bag-of-words or visual codes may perform poorly on this dataset if the dynamic patterns are not properly interpreted. We follow the same test setting of [36] that the first 5 subjects are used for testing and the last 5 subjects are used for training. Experimental results are summarized in Table 4.4.

Table 4.4: Impact of the Dynamic Temporal Quantization and the Frame Assignment Sharing in the MSR-ActionPairs Dataset

Feature	Accuracy		
	Proposed	Deterministic quantization	w/o assignment sharing
position	86.28%	86.85%	86.28%
angle	82.86%	82.86%	82.86%
offset	81.71%	81.14%	70.86%
velocity	89.71%	88.57%	90.28%
fused result	93.71%	91.43%	93.14%

Similar to the results from the other two datasets, the position feature and the velocity feature show higher discriminative capability than the other features in all three columns of Table 4.4. The fused results, by leveraging the fusion of multimodal data, all achieve higher performance than any individual feature alone. One interesting result is, the strategy without the assignment sharing has equal or higher accuracy than the proposed method on all feature except the offset feature. However, fused result is still lower than that of the proposed method enabled with the sharing strategy. Such results justify our hypothesis that the dynamic quantization may get overfitted on some individual features and the overall generalization is compromised when multimodal results are fused.

Table 4.5: Comparison with State-of-the-arts in the MSR-ActionPairs Dataset

Method	Accuracy
Skeleton + LOP + Pyramid [41]	82.22%
HON4D [36]	93.33%
HON4D + D_{disc} [36]	96.67%
Super Normal Vector [51]	98.89%
Proposed approach	93.71%

We further compare the accuracy of the proposed method with state-of-the-art methods and report the results in Table 4.5. It can be seen that the proposed method achieves a comparable accuracy to the methods listed. The only methods having a higher classification rate than ours are the histogram of 4D normals with discriminative projection [36] and the super normal vector [51]. These two methods rely on sophisticated features such as 4D surface normals and polynormals which are inapplicable to more general problems. Nevertheless, the proposed method provides a generic solution to the optimal temporal quantization problem and is independent from any features which makes the direct comparison unfair.

Figure 4.5 shows the confusion matrix of the classification results in the dataset. Although the actions within each pair are highly confusing, the proposed method still achieves very good performance by discriminating the actions in each pair. It can be seen that the within-pair confusion only occurs between “Pick up box” and “Put down box”. Most of the actions with similar postures and temporal variations are correctly distinguished by the proposed method. Such performance has demonstrated the advantages of the proposed Dynamic Temporal Quantization method.

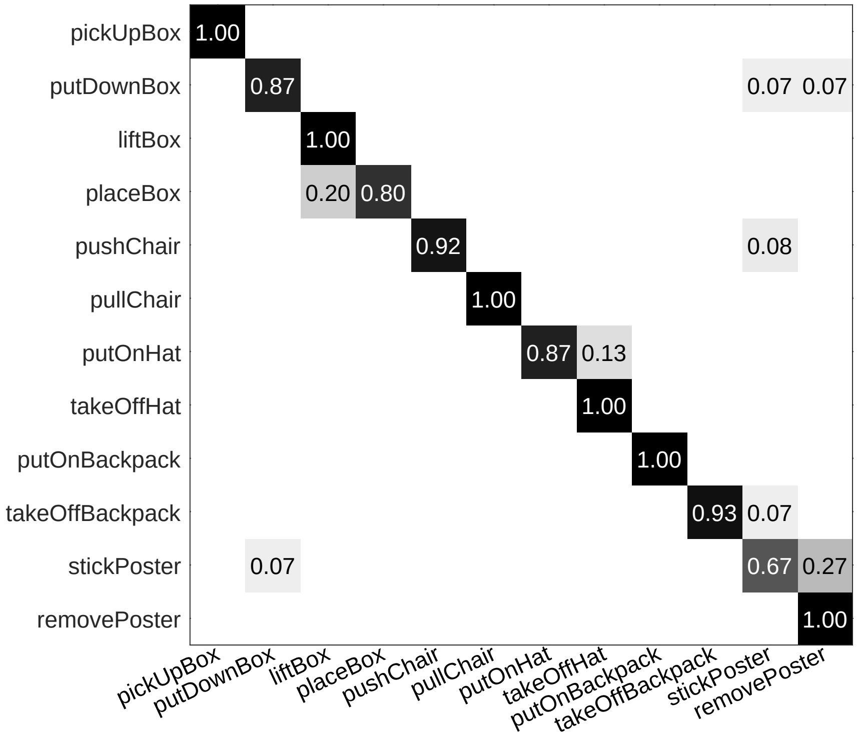


Figure 4.5: Confusion Matrix of the Classification Results in the MSR-ActionPairs Dataset

UTKinect-Action Dataset

The UTKinect-Action dataset [91] consists of 199 action sequences in total. These sequences have 10 action types performed by 10 subjects. All subjects perform each action two times. Different from the previous dataset, subjects perform actions at varied locations in the scene. The huge viewpoint variation and intra-class variance make the dataset very challenging. We follow the same cross-subject test setting as in [43]. Half of the subjects are used for training and the rest are for testing.

We follow the same strategy in the MSR-Action3D dataset that the frame assignment of the quantization is computed based on the position feature and is applied to all other features. The performances of individual features and the fused result are summarized in Table 4.6. Similar results can be found that the proposed dynamic quantization algorithm achieved higher performance than the deterministic quantization on most of the features as well as the fused result. This has again demonstrated the advantage of the dynamic temporal quantizing algorithm. To our surprise, the fused result achieved the 100% accuracy. All samples in the testing set are correctly classified, which strongly demonstrates the performance of the proposed approach.

Table 4.6: Impacts of the Dynamic Temporal Quantization and the Frame Assignment Sharing in the UTKinect-Action Dataset

Feature	Accuracy		
	Proposed	Deterministic quantization	w/o assignment sharing
position	94.95%	88.89%	94.95%
angle	91.92%	87.88%	94.95%
offset	80.81%	74.75%	74.75%
velocity	79.80%	81.82%	78.79%
fused result	100%	96.97%	97.98%

Table 4.7 shows the comparison of the classification accuracy between the proposed algorithm and state-of-the-art methods. Although the Lie Group [43] method achieves very good performance on this dataset (97.08%), the proposed algorithm still outperformed state-of-the-art results by achieving the 100% accuracy.

Summary

In this chapter, we addressed the challenges of temporal modeling for 3D human action recognition from multimodal sensor streams. We studied the problem of optimal temporal quantization of the video sequences and presented a solution of dynamic temporal quantizing. We further introduced a fusion method under this quantization framework to leverage the complementary discriminative capability of multimodal features. Experimental results on three public 3D human action datasets show the proposed algorithms have achieved state-of-the-art performance.

Table 4.7: Comparison with State-of-the-arts in the UTKinect-Action Dataset

Method	Accuracy
Histogram of 3D joints [91]	90.92%
Combined features with random forest [40]	91.9%
Lie Group [43]	97.08%
Proposed approach	100%

CHAPTER 5: TEMPORAL ORDER ENCODING VIA FIRST-TAKE-ALL

In the previous chapters, two different HAR solutions are presented from the perspective of 3D spatial event query processing and temporal dynamic quantization, respectively. However, there are still some challenges remaining, such as the confusion between visually similar activities composed of activity patterns in different temporal order, as well as the computational overhead that prohibits the existing methods from applying to a large scale dataset. To address these challenges, a new perspective of temporal order encoding is investigated and a First-Take-All temporal encoding approach is presented in this chapter.

Introduction

Human Activity Recognition (HAR) has attracted intense research interest in the past decade and continues to be an active area. From the perspective of data acquisition, approaches to HAR can be roughly clssified into two categories: the camera-based methods using 2D or 3D camera as the primary data capturing devices [96, 97, 37, 98] and the wearable sensor-based methods [55, 56, 61]. Compared with camera-based methods, body-worn sensor-based methods employed various miniature inertial sensors, such as accelerometer, gyroscope, magnetometor, etc., and have advantages regarding availability, complexity, and privacy [99]. Firstly, wearable sensors are body-worn and can provide sensing data virtually anytime and anywhere, while surveillance cameras may not have full coverage and human-recorded video may contain occlusions or visibility problems. Secondly, compared with the heavy video data, signals from wearable sensors are lightweight, making them feasible for real-time online human activity detection and recognition on a large scale. Lastly, wearable sensor signals do not reveal the identity of the user and are less vulnerable to privacy issues than the vision-based methods.

Motivated by the above merits, wearable sensor-based HAR has attracted a great deal of interest and become a significant research problem with many real-world applications. For example, in healthcare, wearable sensors can observe daily human activities, including *sitting, standing, lying down, climbing floors, etc.,* and can detect abnormal activities in case of incidence [100]. With the development and increasing popularity of mobile devices, smart wristbands and smart phones provide a convenient solution to keep track of people's daily and fitness activities such as counting steps and monitoring heart rate. Some smart wristbands even support auto-logging of different cycling events,[1] or tracking the number of strokes in swimming [2].

Althought many successful products and applications have been developed, wearable sensor-based HAR remains to be a challenging problem due to the highly noisy nature and the complex temporal dynamics within the sequential data. Most existing sensor-based HAR approaches focus on extracting statistical and structural features from the time domain and frequency domain, such as *mean, standard deviation, interquartile rante (IQR), kurtosis, correlation, entropy and energy* [52, 53, 56, 61], *Fourier Transform coefficients* [55, 56] and *Discrete Cosine Transform coeficients* [54]. Although impressive accuracy has been reported, most of the above handcrafted features do not consider the temporal dynamics within the sequential sensor data and therefore suffer from the confusion between activities with similar statistical patterns but different temporal patterns. Traditional methods like Dynamic Time Warping (DTW) [62] require pairwise similarity comparison between activity sequence, which are computationally demanding and are not scalable to a large scale dataset. More recently, recurrent neural networks like Long Short-Term Memory Machine (LSTM) [67, 71, 72] implicitly encode the temporal structures of activity patterns in the memory states, but it is unclear how the memory states can be mapped to temporal order of patterns, which limits their capability in fully interpreting dynamic structures underlying human activities.

[1] `http://cycling.moov.cc/`

[2] `http://sites.garmin.com/en-US/swim/`

To address these challenges, we propose a novel temporal order encoding approach that explicitly exploits the temporal dynamics inside the human activity sequence and produces a highly compact feature representation. A diagram of the encoding algorithm is illustrated in Figure 5.1. A set of random linear projections are adopted to map sensor signals into subspace representing different latent patterns of human activity sequences. The temporal order of these latent patterns carries useful information that can be exploited to distinguish different human activities. Therefore, we encode the whole sequence by the index of the pattern that comes first. The above process is repeated multiple times and eventually produces a compact feature vector. Since the First-Take-All (FTA) strategy is used to encode the sequence, we term the feature as FTA feature. To further increase the discriminative power of the FTA feature, we also develop an optimization algorithm that learns the most powerful latent patterns to better represent the temporal dynamics of the sequential data. The compactness of the FTA feature makes it extremely efficient for HAR using the nearest neighbor search based on Hamming distance.

The main contributions of this work are summarized as follows:

1. A temporal order encoding approach is proposed to exploit the underlying temporal dynamics in a human activity sequence. A compact FTA feature vector is produced from the above encoding approach, which carries discriminative information for human activity classification.

2. Two different temporal order encoding algorithms are proposed, namely, First-Take-All by expected moment (FTA) and probabilistic First-Take-All (pFTA).

3. We further propose the optimization algorithms for each of the above temporal order encoding algorithms, which can increase the discriminative capability of the FTA feature by learning the most salient linear projections.

4. Extensive performance studies on two public HAR datasets show the proposed FTA and pFTA features can outperform state-of-the-art statistical features on both accuracy and efficiency. pFTA can achieve comparative or even higher classification accuracy than deep recurrent neural networks such as Long Short-Term Memory (LSTM) with significantly higher efficiency.

It is worth noting that although we focus on sensor data from wearable devices in this work, the proposed temporal order encoding approach can potentially be used as a generic solution to other recognition problems on time series that contain rich temporal dynamics. We have made the source code available at `https://github.com/gragonraja/pFTA`.

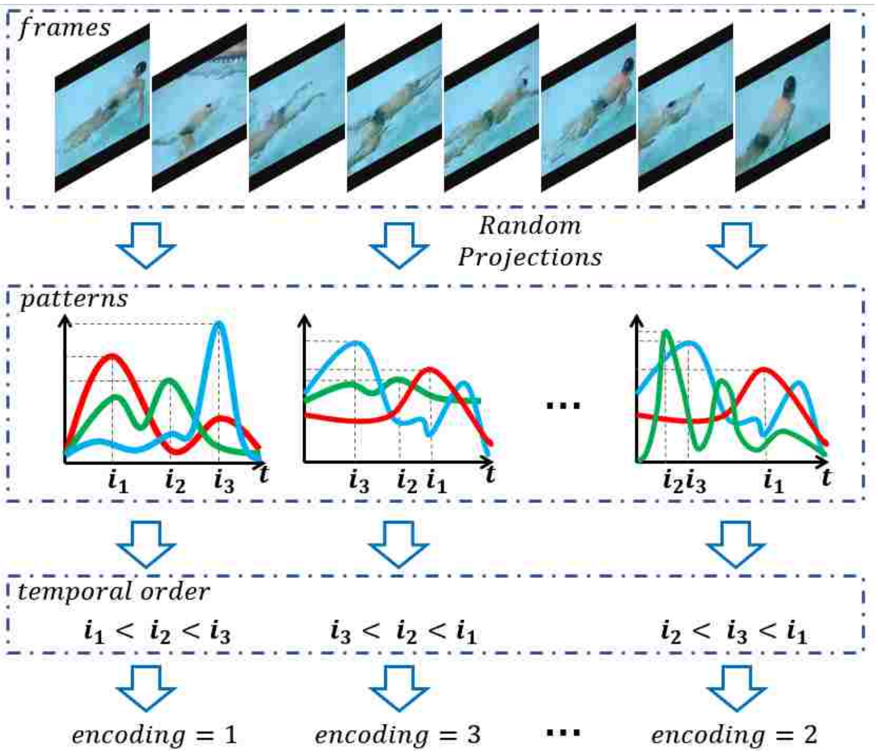


Figure 5.1: Framework of the Temporal Order Encoding Approach

The remainder of this chapter is structured as follows: The FTA by expected moment and the optimization algorithm for the FTA feature representation are introduced in Section 2 and Section 3. The probabilistic FTA encoding algorithm, as well as the pFTA feature, are presented in Section 4. The optimization algorithm to pFTA is introduced and discussed in Section 5. We show the performance evaluation in section 6 and, finally summarize this chapter in Section 7.

Temporal Order Encoding by Expected Moment

Latent Patterns

Suppose a sequential data $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T]$ of length T, where each point of data $\mathbf{x}_t \in \mathbb{R}^D$ is a D dimensional sensor signal collected at time t ($1 \leq t \leq T$) from a sensor unit. Without lost of generality, $\mathbf{X}$ can be any sequential data, such as time series and videos. In order to model the temporal characteristics of $\mathbf{X}$, we use a set of K linear projections $\mathbf{W}$ to map the sensor signal into subspace, where $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K] \in \mathbb{R}^{D \times K}$. $\mathbf{W}$ can be interpreted as forming a linear subspace for latent patterns of the sequential data. Each of the K projections can be regarded as a latent pattern extractor from the multi-dimensional signal $\mathbf{x}_t$. Hence, the inner product of $s_{k,t} = \mathbf{w}_k{}^\intercal \mathbf{x}_t$, ($1 \leq k \leq K$), gives a confident score $s_{k,t}$ of signal $\mathbf{x}_t$ containing the latent pattern represented by $\mathbf{w}_k$. A higher score $s_{k,t}$ means the pattern is more likely to appear at a particular time. A full definition of notations used in the chapter is listed in Table 5.1.

Temporal Order Encoding

We propose to model the temporal characteristics of sequential data by encoding the temporal order of the latent patterns inside them. When comparing the temporal order of the occurrence of different latent patterns, we hope each pattern is salient enough to produce reliable comparison results. Therefore, we further filter the confidence score $s_{k,t}$ by a threshold θ and only retain the

$s_{k,t}$ that is higher than θ. The thresholding process is given by (5.1).

$$s_{k,t} = \begin{cases} -\infty, & \text{if } s_{k,t} < \theta, \\ s_{k,t}, & \text{otherwise.} \end{cases} \tag{5.1}$$

Table 5.1: Definitions of Notations

Notation	Definition
$\mathbf{X} \in \mathbb{R}^{D\times T}$	Sequential data collected from a sensor
T	Length of the sensor data
D	Dimension of the sensor data
$\mathbf{W} \in \mathbb{R}^{D\times K}$	Linear projections
K	Encoding scale for the temporal order encoding
L	Feature dimension of the temporal order encoding
M	Size of mini-batch in each epoch of the optimization
N	Number of samples in the training set
s_{ij}	Pairwise label similarity between $\mathbf{X}_i$ and $\mathbf{X}_j$

To determine the temporal order of the latent patterns, we must first locate the moment they appear. Suppose K latent patterns are used to encode the sensor data. We use the following softmax function to model the probability that pattern k ($1 \leq k \leq K$) appears at time t:

$$p_{k,t} = \frac{e^{\alpha s_{k,t}}}{\sum_{t'=1}^{T} e^{\alpha s_{k,t'}}}, \tag{5.2}$$

where α is a rescale factor that scales the differences between confidence scores over time to avoid a "flat" distribution of $p_{k,t}$. The confidence score from a projection is therefore mapped into a probability as illustrated in Figure 5.2. As can be seen from the figure, different latent patterns shall exhibit different probability density distributions if the projections are properly optimized. For example, the peak of p_2 occurs before that of p_1 in Figure 5.2, which means pattern 2 happens earlier than pattern 1 in this sequential data. It is this intrinsic temporal order relationship that we wish to encode to model the temporal characteristics of the human action data. We further define the expectation of the time pattern k occurs as

$$u_k = \sum_{t=1}^{T} \frac{t}{T} p_{k,t}. \tag{5.3}$$

We call u_k the expected moment of pattern k as it gives the expectation of the normalized time (from 0 to 1) of the occurrence of pattern k given the probability $p_{k,t}$ at each point of time. According to (5.1), it is possible that a pattern fails the thresholding test and keeps silent throughout the time. In this scenario, we set $u_k = 1$ indicating this pattern will appear at the very last moment of the sequence. In this way, any non-silent pattern will appear earlier than this pattern. We then encode $\mathbf{X}$ by the index of the pattern that occurs first among all K patterns.

$$f^* = \arg \min_{1 \leq k \leq K} u_k. \tag{5.4}$$

Take the two latent patterns in Figure 5.2 as an example, since the expectation of t in p_2 is smaller than that in p_1. The temporal order encoding process produces a value of 2, indicating that pattern 2 happens earlier than the other patterns. In a very rare scenario that all K patterns fail the thresholding test and none of them appears in the sequence, then a special index 0 is used to denote this case. As a result, f^* can take $K + 1$ discrete values from 0 to K.

To give an intuitive understanding of the latent temporal patterns for human action recognition, we show the following running examples from the the experiments in the MSRActionPairs Dataset. Each subfigure in Figure 5.3 shows the cuves of probability $p_{k,t}$ of three latent patterns in different videos. The left column are from videos of the action “put up a box”, the videos on the right column are from the action “put down a box”. All six videos use the same set of projections.

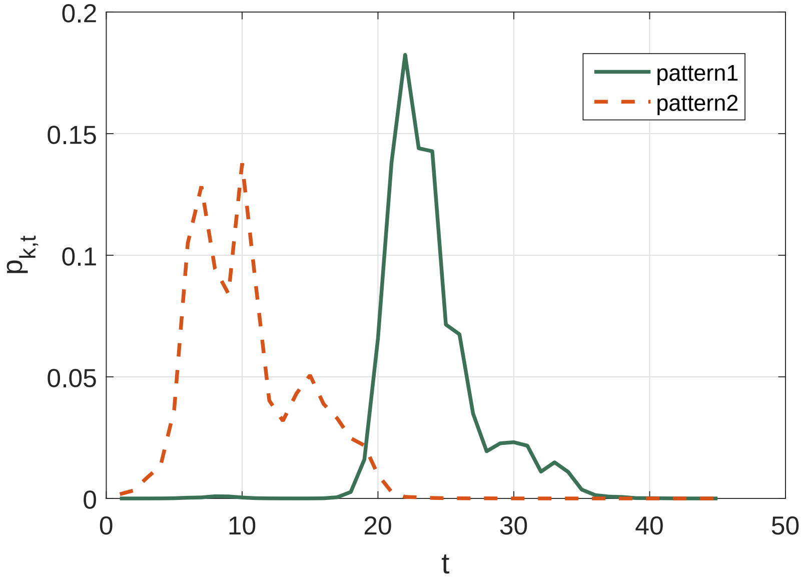


Figure 5.2: Examples of the Probability Distribution of $p_{k,t}$ on Two Different Latent Patterns

It is clear that, pattern 2 (dash line) always appears earlier than the other patterns in the videos of "put up a box" (left column); while pattern 3 (dotted line) always appears earlier than the other patterns in all three videos of "put down a box" in the right column. Therefore, the temporal order of these latent patterns from the same set of projections carries discriminative information which is capable to characterize same actions and distinguish different actions, and the above temporal order encoding aims at extracting such discriminative information for human activity recognition.

As aforementioned, a set of K projections $\mathbf{W} \in \mathbb{R}^{D\times K}$ are randomly selected each time. The above process repeats L times and eventually generates an L-length $K+1$-ary feature vector $\mathbf{f} = [f_1, f_2, \ldots, f_L]$ which is able to extract the temporal characteristics of the original sequential data. Each element in $\mathbf{c}$ represents a temporal order encoding process between K projections. We refer to this feature as the First-Take-All (FTA) feature and call K and L the encoding scale and the feature dimension of the FTA feature vector, respectively.

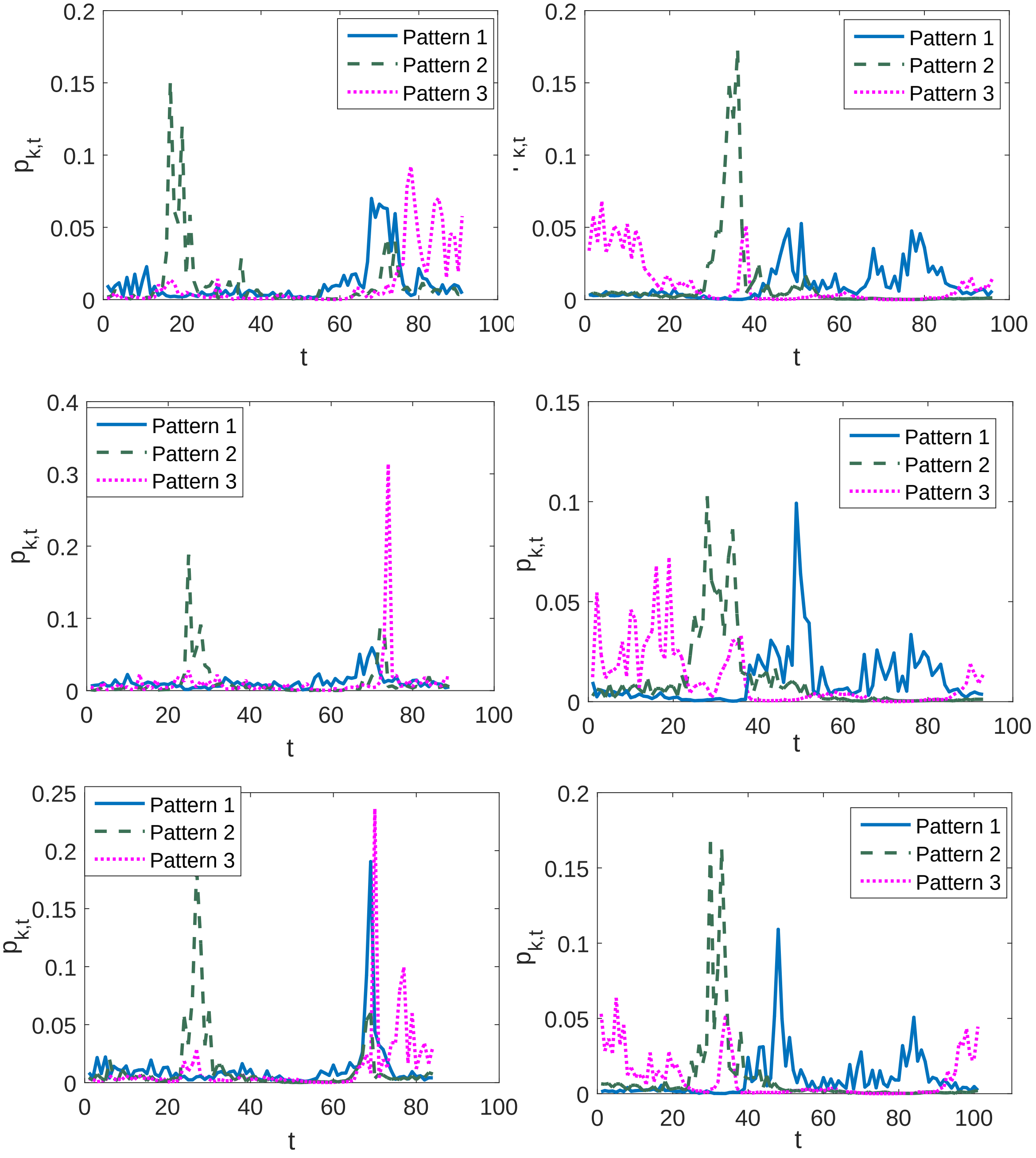


Figure 5.3: Running Examples of Latent Patterns on Two Different Categories of Human Actions. The left column are from the action “Put up a box”, the right column are from action “put down a box.” Results are from the experiments in the MSRActionPairs Dataset.

As can be seen, K and L are two crucial parameters controlling the performance of the above temporal order encoding approach. As an example, when $K = 2$, the pairwise comparison is made between two latent patterns (as illustrated in Figure 5.2). When a larger encoding scale is adopted, a higher order comparison is performed among multiple latent patterns and more discriminative information can potentially be encoded into the FTA feature. Similarly, a larger L may also bring in more information by investigating more latent patterns. However, it may also introduce redundant encoding to the feature vector and increase the computational overhead. More details of the impacts of K and L will be discussed in the experiments.

The above temporal order encoding approach has several nice properties. Firstly, since the temporal order is explored rather than the absolute occurring time of the latent patterns, the above approach is insensitive to temporal translation and the execution rate variation of human action sequences [42]. As an example, the peaks of different patterns in Figure 5.3 appear at different temporal locations of the video due to the execution variations and the temporal translations of human actions. The proposed temporal order encoding approach is insensitive to these variations by exploiting the relatively order information of the latent patterns and produce reliable feature representations of the human action sequences. Secondly, the modeling algorithm produces a highly compact feature vector from the original video. The Hamming distance can then be utilized to compute the similarity between two videos. Thus, action recognition can be efficiently solved by nearest neighbor search based on the TOP feature vector.

Learning Optimized Projections

In this section, we introduce a supervised optimization algorithm that learns the optimized projections and increases the discriminative power of the resultant FTA feature vector. We first formulate the optimization problem and then present the detailed learning algorithm.

Problem Formulation and Objective

The overall goal of optimization is to learn a set of projections $\mathbf{W}$ so that human activity sequences of the same categories can have similar temporal order with respect to the corresponding latent patterns. In other words, their resultant FTA feature vectors should have a smaller Hamming distance. As aforementioned, the temporal order encoding is based on the $\arg\min$, which is discontinuous and non-differentiable. In order to address this challenge, we reformulate the problem as follows. Suppose $\mathbf{u} = [u_1, u_2, \ldots, u_K]$ are the expected moments of K latent patterns from projections $\mathbf{W}$ in a sequential data. In order to investigate the temporal order of the K latent patterns inside the sequence, we use a second softmax function to compute the probability of pattern k coming first among a set of K patterns in a sequence. The definition of the probability h_k is given by (5.5)

$$h_k = \frac{e^{(1-u_k)}}{\sum_{k'=1}^{K} e^{(1-u_{k'})}} \tag{5.5}$$

Given the above probability, we can then define a metric d_{ij} to evaluate whether the two sequences i and j have similar temporal order with respect to the same set of projections $\mathbf{W}$ according to the following sum-product rule,

$$d_{ij} = \sum_{k=1}^{K} h_k^{(i)} \cdot h_k^{(j)}, \tag{5.6}$$

where $h_k^{(i)}$ and $h_k^{(j)}$ denote the probability of pattern k coming first among K patterns in sequence i and sequence j, respectively. It's clear that the product of the two softmax represents the probability that pattern k occurs first in both sequences. Therefore, the summation over all K patterns gives the overall probability of these two sequential data having the same temporal order with respect to $\mathbf{W}$. In this way, the $\arg\min$ is represented by the similarity metric d_{ij}. The higher d_{ij}, the more likely sequences i and j are similar regarding their temporal order. The following $L2$-norm loss function can then be defined to measure the error incurred by the temporal order encoding process,

$$\ell(d_{ij}, s_{ij}) = (d_{ij} - s_{ij})^2, \tag{5.7}$$

where s_{ij} is the pairwise label similarity. s_{ij} equals 1 when sequence i and sequence j have the same label, and 0 otherwise. Thus, the overall learning objective is to find the optimized projections $\mathbf{W}$ by minimizing the total loss over all training pairs

$$\mathcal{L}(\mathbf{W}) = \sum_{i,j} \ell(d_{ij}, s_{ij}) + \lambda \|\mathbf{W}\|_F^2, \tag{5.8}$$

where λ is a non-negative coefficient controlling the weight of the $L2$-regularizer, which is introduced to counter against the overfitting. Note that $\mathbf{W}$ factors into the above objective function because d_{ij} is function of $\mathbf{W}$.

The Optimization Algorithm

We use softmax to model the probability of the first-appearing latent patterns. As a result, the above loss function is not convex. Nevertheless, $\ell(d_{ij}, s_{ij})$ is directly differentiable with respect to $\mathbf{W}$. Therefore, we adopt the stochastic gradient decedent with mini-batch to optimize $\mathbf{W}$ step by step. Specifically, the gradient $\nabla_{\mathbf{W}}$ can be computed by the partial derivative of the total loss $\mathcal{L}$ with respect to each of the K projections in $\mathbf{W}$, $\nabla_{\mathbf{W}} = [\frac{\partial \mathcal{L}}{\partial \mathbf{w}_1}, \frac{\partial \mathcal{L}}{\partial \mathbf{w}_2}, \ldots, \frac{\partial \mathcal{L}}{\partial \mathbf{w}_K}]$, where $\frac{\partial \mathcal{L}}{\partial \mathbf{w}_k}$ can be computed as

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_k} = \sum_{i,j} \frac{\partial \ell}{\partial \mathbf{w}_k} + 2\lambda \mathbf{w_k} = \sum_{i,j} 2(d_{ij} - s_{ij}) \frac{\partial d_{ij}}{\partial \mathbf{w}_k} + 2\lambda \mathbf{w_k}, \tag{5.9}$$

Since d_{ij}, u_k and $p_{k,t}$ are all functions of $\mathbf{w}_k$, it is straightforward to compute $\frac{\partial d_{ij}}{\partial \mathbf{w}_k}$ according to (5.2), (5.3) and (5.6). Thus $\mathbf{W}$ can be updated accordingly by

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \nabla_{\mathbf{W}}, \tag{5.10}$$

where η is the learning rate controlling the step size of each epoch. Figure 5.4 shows an example of the training objective through multiple epochs to learn a set of K projections $\mathbf{W}$ generating one dimension of the L-length FTA feature vector. As can be seen, the learning process converges after 50 epochs.

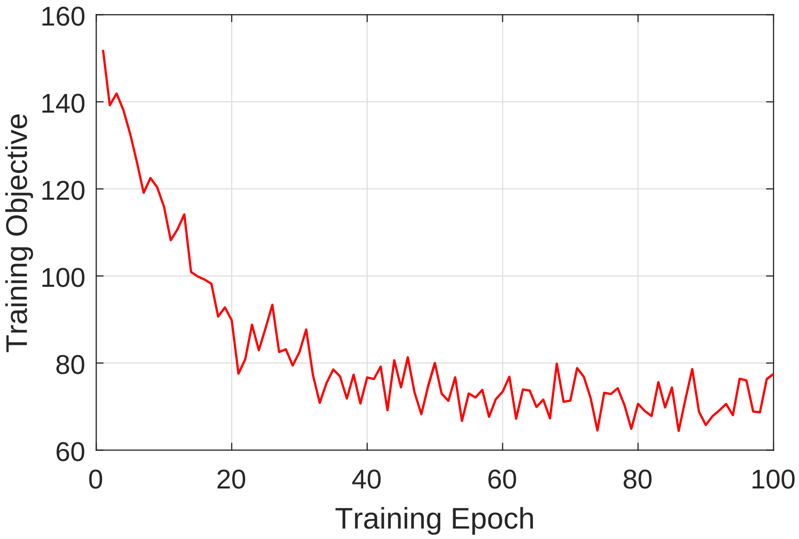


Figure 5.4: Training Loss on a Mini-batch of Pairwise Training Data through Multiple Epochs of the Optimization. Curve is obtained through the optimization of one FTA feature dimension in the MSRActionPairs dataset.

The above learning procedure learns a set of K projections $\mathbf{W}$ to produce one dimension of the FTA feature vector. To generate the entire L-length $K+1$-ary feature vector, we repeat the procedure L times. The convex-concave nature of the objective function suggests there are no global minima, but multiple local minima. Taking advantage of this, the random initialization of $\mathbf{W}$ in each of the L iterations can be optimized to various local minima and eventually produces a complementary representation of the temporal characteristics of the sequential data. The pseudo code of the above learning algorithm is presented in Algorithm 2.

Algorithm 2 Optimization Algorithm for the FTA Modeling

```
Input: training data $\chi = \{X_i\}_{i=1}^{N}$, encoding scale $K$, FTA feature dimension $L$.
for $l = 1$ to $L$ do
    Randomly initialize $\mathbf{W}_l = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K] \in \mathbb{R}^{D \times K}$;
    repeat
        Randomly select a mini-batch of training pairs;
        for $k = 1$ to $K$ do
            Compute $\frac{\partial \mathcal{L}}{\partial \mathbf{w}_k}$ over a mini-batch of $\chi$ by (5.9);
        end for
        Compute the gradient $\nabla_{\mathbf{W}} = [\frac{\partial \mathcal{L}}{\partial \mathbf{w}_1}, \frac{\partial \mathcal{L}}{\partial \mathbf{w}_1}, \ldots, \frac{\partial \mathcal{L}}{\partial \mathbf{w}_K}]$;
        Update $\mathbf{W}_l$ by $\mathbf{W}_l \leftarrow \mathbf{W}_l - \eta \nabla_{\mathbf{W}}$;
    until Convergence
end for
return the optimized projections $[\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_L]$ for an $L$-dimensional FTA feature vector.
```

Complexity Analysis

Given a set of K projections $\mathbf{W}$, the cost of encoding one dimension of the FTA feature is $O(TDK) + O(TK) + O(\log K)$. The first component accounts for the cost of the linear projection as well as the softmax. The second component is the cost of finding the expected moments of the K latent patterns. The third component denotes the cost of finding the first occurring pattern. Considering K is typically small (from 2 to 8), the complexity is then $O(TD)$ which is linear with respect to the size of the input data.

Next, we analyze the complexity of the learning process. Let's consider the complexity of the mini-batch update in one epoch. Suppose the mini-batch contains M training pairs. The complexity at each pair of the training data consists of the following major components: the cost for computing the probability d_{ij} in (5.6) has a complexity of $O(KTD)$, and the cost for computing K partial derivatives $\frac{\partial d_{ij}}{\partial \mathbf{w}_k}$ is $O(K^2TD)$. Putting it all together, the total cost to obtain $\nabla_{\mathbf{W}}$ over the mini-batch is $O(MK^2TD)$. Again, since K is typically a small number, the overall complexity for updating one epoch over the mini-batch is linear with respect to the size of the mini-batch and the size of the input data.

Probabilistic Temporal Order Encoding

The First-Take-All encoding algorithm presented in the last two sections essentially locates the appearing time of a latent pattern by the expectation of the time the patterns appears. The single metric such as the expected moment is prone to the fluctuation and oscillation of the signals, and there may be multiple peaks in a sequence, which makes the expected moment less meaningful in temporal order comparison. To address this issue, a probabilistic First-Take-All encoding algorithm is proposed in this section, which handles the complexity of the temporal dynamics inside the sequence data based on the probability.

Probability and Hypothesis

Following the same notation in the First-Take-All encoding, let's denote $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T]$ as a sequence of length T. Each element $\mathbf{x}_t \in \mathbb{R}^D$ is a D-dimensional signal captured at time $t, (1 \leq t \leq T)$. A set of K linear projections $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K] \in \mathbb{R}^{D \times K}$ are used to map $\mathbf{x}_t$ into a linear subspace for latent patterns of the sequential data. The inner product of $\mathbf{x}_t$ and $\mathbf{w}_k$ $(1 \leq k \leq K)$ gives a confident score $s_{k,t}$ of signal $\mathbf{x}_t$ containing the latent pattern represented by

$\mathbf{w}_k$. Different from the thresholding test in FTA, we adopt a bias term θ and rewrite $s_{k,t}$ as

$$s_{k,t} = [\mathbf{w_k}; \theta]^T [\mathbf{x}_t; -1]. \tag{5.11}$$

so that the linear projections do not need to go through the origin. The softmax function used for FTA encoding requires that the probabilities of a pattern occurring at different locations in a sequence should sum to one. Such a strong constraint enforces the contention between $p_{k,t}$ at different locations. As a result, the peak of a pattern may overwhelm the expected moment, making the temporal order comparison less reliable. Instead of using the softmax function, we relax the constraint and employ the following logistic function to estimate the probability of pattern k appearing at time t,

$$p_{k,t} = \frac{1}{1 + e^{-\alpha s_{k,t}}}, \tag{5.12}$$

where α is a non-negative rescale factor controlling the steepness of the logistic function. It is clear that a high $s_{k,t}$ will yield a high probability $p_{k,t}$. Consequently, $1 - p_{k,t}$ denotes the probability that pattern k didn't appear at time t. A pattern may keep silent throughout the sequential data and never appear. In this case, it is equivalent to say that pattern k occurs at infinite time; we use $p_{k,+\infty}$ to denote the probability that pattern k never appears in sequential data $\mathbf{X}$.

$$p_{k,+\infty} = \prod_{t=1}^{T} (1 - p_{k,t}) \tag{5.13}$$

$p_{k,+\infty}$ is defined as the joint probability of pattern k not appearing at any time t.

Note that sequential data of the same category share similar temporal characteristics. We wish to extract and represent such unique traits by exploiting the aforementioned latent patterns in them. Individual latent patterns may appear at any time in the sequential data. However, the temporal order between a set of K latent patterns in a data sequence of the same category may be consistent.

For example, Figure 5.2 shows two patterns extracted from a 45-point data sequence. We can see that Pattern 2 has high projection scores on the first half of the sequence while pattern 1 mostly appears on the second half of the sequence. It is this temporal order information that we want to encode to represent the temporal dynamics of the sequential data. Following this idea, we introduce the Probabilistic Temporal Order Encoding Algorithm to extract the temporal dynamics by encoding the temporal order of the latent patterns in the sequential data.

To model the temporal order of a set of K latent patterns, we introduce the following hypotheses:

Hypothesis 0: *none of the K patterns ever appears in the sequence.*

Hypothesis k: *pattern k shall appear no later than any other pattern,* ***and*** *no other pattern with a smaller index (i.e., $k' < k$) shall arrive at the same time as pattern k.*

We perform a hypothesis testing by accepting the hypothesis with the highest probability, and use its index to encode the entire sequence.

Note that the second part of *Hypothesis k* assumes that whenever a tie happens (i.e., some patterns arrive at the same time before the other patterns), the pattern that appears first with the smallest index will be used to index the sequence.

We denote h_0 and h_k as the probability of *Hypothesis* 0 and *Hypothesis* k, respectively. h_0 and h_k can be computed by (5.15) and (5.14).

$$h_k = \sum_{t=1}^{T} p_{k,t} \prod_{k'<k} (1 - p_{k',t}) \Big(\prod_{k'=1}^{K} \prod_{t'=1}^{t-1} (1 - p_{k',t'}) \Big) \tag{5.14}$$

h_0 is defined as the joint probability of $p_{k,+\infty}$.

$$h_0 = \prod_{k=1}^{K} p_{k,+\infty} \tag{5.15}$$

Finally, the index of the hypothesis of the highest probability among h_k, $(0 \leq h \leq K)$, is encoded into a discrete feature f to represent the entire sequence.

$$f^* = arg \max_{k=0}^{K} h_k \tag{5.16}$$

f^* can be considered as a feature representation of the original sequential data. Since we model the temporal order by the index of the first-coming latent pattern according to the probability, we term the feature probabilistic FTA (pFTA).

K is the encoding scale of the pFTA feature. It is worth noting that K is a crucial parameter controlling the performance of the pFTA feature. When K is set to 2, a pair of latent patterns are evaluated each time and the full temporal order of these two patterns are encoded. When K is greater than 2, more patterns are investigated each time but only partial information (e.g. the first coming pattern) is encoded by pFTA in a winner-take-all fashion[101]. Therefore, a trade-off between compactness and the amount of information encoded must be considered. More details will be discussed in the performance studies.

Probability First-Take-All and the pFTA feature

The above encoding process repeats L times and eventually produces an L-dimensional K-ary compact feature vector that represents the temporal dynamics of the sequential data. We call L the feature dimension. The pseudo code of the above Probabilistic Temporal Order Encoding Algorithm is listed in Algorithm 3.

In each iteration through step 4 to step 8, a set of K linear projections $\mathbf{W}_l$ are randomly picked. The probability of pattern k occurring at any time point t, as well as the the probability that pattern k never appears, are computed. Next, probabilities of each hypothesis are computed by (5.14) and

(5.15). Finally, one feature dimension of the pFTA feature is then encoded via the index of the hypothesis of the largest probability given by (5.16). The above encoding procedure is repeated L times and eventually produces an L-dimensional pFTA feature vector.

Feature dimension L is another critical parameter to the Probabilistic Temporal Order Encoding Algorithm. A larger L can bring more information to the pFTA feature. However, a longer feature dimension may also bring redundant information and increase the computational cost. We will discuss more details on the impact of L in the performance studies.

Algorithm 3 Probabilistic First-Take-All Encoding

```
Input: Sequential data $\mathbf{X}$, feature dimension $L$, encoding scale $K$
Initialize: $\mathbf{f} \leftarrow$ empty vector
for $l = 1$ to $L$ do
    Randomly initialize a set of K linear projections $\mathbf{W}_l$
    Compute $p_{k,t}$ and $p_{k,+\infty}$ according to (5.12) and (5.13) for each pattern.
    Compute the probability $h_0, \cdots, h_K$ according to (5.15) and (5.14)
    $f^* \leftarrow \arg \min_{0 \leq k \leq K} h_k$.
    $\mathbf{f}^* \leftarrow [\mathbf{f}\ f]$
end for
return $\mathbf{f}$
```

Optimization for pFTA

In this section, we present the optimization algorithm for the probabilistic First-Take-All encoding.

Problem Formulation and Objective

Given a set of K linear projections and two sequential data i and j, we use h_{ij} to represent the probability of data i and j having similar temporal order information on K latent patterns.

h_{ij} is defined as

$$h_{ij} = \sum_{k=1}^{K} h_k^{(i)} h_k^{(j)}, \tag{5.17}$$

where $h_k^{(i)}$ and $h_k^{(j)}$ denote the probability that pattern k appears first in data i and j, respectively. It is clear that if data i and data j are similar to each other, $h_k^{(i)}$ and $h_k^{(j)}$ should also be similar with respect to each latent pattern. As a result, the probability that data i and data j are in the same category should be close to 1. On the contrary, if data i and data j have different temporal dynamics, the corresponding $h_k^{(i)}$ and $h_k^{(j)}$ should be different, and consequently, the probability that data i and data j are in the same class should also be low.

Therefore, the following loss function can be defined using the cross entropy among a pair of training data i and j

$$\mathcal{L}(\mathbf{W}) = -\sum_{i,j} \big((1 - s_{ij}) \log(1 - h_{ij}) + s_{ij} \log h_{ij}\big) + \lambda \|\mathbf{W}\|_F^2, \tag{5.18}$$

where s_{ij} is the pairwise label similarity.

Note that the projection $\mathbf{W}$ factors into the above loss function because h_{ij} is function of $\mathbf{W}$. The training objective is then to minimize the loss over all pairs of training samples.

Optimization Algorithm

The above loss function is not convex, but h_{ij} is differentiable with respect to the projection $\mathbf{W}$. Therefore, we adopt the stochastic gradient descent approach with mini-batch to optimize $\mathbf{W}$ through multiple iterations.

We compute the gradient $\nabla_{\mathbf{W}}^{\mathcal{L}}$ with respect to each of the K projections $\mathbf{w}_k$ in $\mathbf{W}$ by (5.19)

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_k} = \begin{cases} \sum_{i,j} \frac{-1}{h_{ij}} \frac{\partial h_{ij}}{\partial \mathbf{w}_k} + 2\lambda \mathbf{w}_k, & \text{if } s_{ij} = 1, \\ \sum_{i,j} \frac{1}{1-h_{ij}} \frac{\partial h_{ij}}{\partial \mathbf{w}_k} + 2\lambda \mathbf{w}_k, & \text{if } s_{ij} = 0. \end{cases} \tag{5.19}$$

In each iteration, $\mathbf{W}$ can be updated according to $\nabla_{\mathbf{W}}^{\mathcal{L}}$ by $\mathbf{W} \leftarrow \mathbf{W} - \eta \nabla_{\mathbf{W}}^{\mathcal{L}}$, where η is the step size of the update. $\frac{\partial h_{ij}}{\partial \mathbf{w}_k}$ can be computed by

$$\frac{\partial h_{ij}}{\partial \mathbf{w}_k} = \sum_{k'=1}^{K} \frac{\partial h_{k'}^{(i)}}{\partial \mathbf{w}_k} h_{k'}^{(j)} + h_{k'}^{(i)} \frac{\partial h_{k'}^{(j)}}{\partial \mathbf{w}_k} \tag{5.20}$$

Since $h_k^{(i)}$, $h_k^{(j)}$, $p_{k,t}$ and $p_{k,+\infty}$ are all function of $\mathbf{W}$, their partial derivatives with respect to $\mathbf{w}_k$ can also be computed according to (5.14),(5.15),(5.12) and (5.13). Specifically, according to (5.14), h_k consists of three components, $p_{k,t}$, $\prod_{k'<k}(1-p_{k',t})$ and $\prod_{k'=1}^{K}\prod_{t'=1}^{t-1}(1-p_{k',t'})$. Therefore, the following three scenarios with respect to k must be considered when computing $\frac{\partial h_{k'}}{\partial \mathbf{w}_k}$.

When $k' = k$,

$$\frac{\partial h_{k'}}{\partial \mathbf{w}_k} = \sum_{t=1}^{T} \Big(\frac{\partial p_{k,t}}{\partial \mathbf{w}_k} \prod_{t'=1}^{t-1} (1-p_{k,t'}) + p_{k,t} \sum_{t'=1}^{t-1} \big(-\frac{\partial p_{k,t'}}{\partial \mathbf{w}_k} \prod_{\substack{t''=1,\\ t''\neq t'}}^{t-1} (1-p_{k,t''})\big) \Big) \prod_{\substack{k''=1,\\ k''\neq k}}^{K} \prod_{t'=1}^{t-1} (1-p_{k'',t'}) \prod_{k''<k} (1-p_{k'',t}) \tag{5.21}$$

When $k' < k$,

$$\frac{\partial h_{k'}}{\partial \mathbf{w}_k} = \sum_{t=1}^{T} \Big(p_{k',t} \prod_{k''\neq k}^{K} \prod_{t'=1}^{t-1} (1 - p_{k'',t'}) \prod_{k''<k'} (1 - p_{k'',t}) \Big) \sum_{t'=1}^{t-1} - \frac{\partial p_{k,t'}}{\partial \mathbf{w}_k} \prod_{\substack{t''=1,\\ t''\neq t'}}^{t-1} (1 - p_{k,t''}) \tag{5.22}$$

When $k' > k$,

$$\frac{\partial h_{k'}}{\partial \mathbf{w}_k} = \sum_{t=1}^{T} \left(p_{k',t} \prod_{k'' \neq k}^{K} \prod_{t'=1}^{t-1} (1 - p_{k'',t'}) \prod_{\substack{k'' \neq k, \\ k'' < k'}}^{K} (1 - p_{k'',t}) \right) \sum_{t'=1}^{t} - \frac{\partial p_{k,t'}}{\partial \mathbf{w}_k} \prod_{\substack{t''=1, \\ t'' \neq t'}}^{t} (1 - p_{k,t''}) \quad (5.23)$$

h_0 denotes the probability that none of the K patterns appear in the sequence. So according to (5.15), the partial derivative of h_0 with respect to $\mathbf{w}_k$ is computed as

$$\frac{\partial h_0}{\partial \mathbf{w}_k} = \left(\sum_{t=1}^{T} - \frac{\partial p_{k,t}}{\partial \mathbf{w}_k} \prod_{\substack{t'=1, \\ t' \neq t}}^{T} (1 - p_{k,t'}) \right) \prod_{\substack{k'=1, \\ k' \neq k}}^{K} \prod_{t=1}^{T} (1 - p_{k',t}) \quad (5.24)$$

$p_{k,t}$ is the probability of pattern k appears at time t, which is defined by the logistic function as defined in (5.12). Therefore, the partial derivative of $p_{k,t}$ with respect to $\mathbf{w}_k$ can be computed as

$$\frac{\partial p_{k,t}}{\partial \mathbf{w}_k} = \frac{\alpha \mathbf{x}_t e^{-\alpha \mathbf{w}_k^T \mathbf{x}_t}}{(1 + e^{-\alpha \mathbf{w}_k^T \mathbf{x}_t})^2} \quad (5.25)$$

The above procedure produces a set of K optimized linear projections that generate one dimension of the pFTA feature. The process can be repeated L times to generate an L-dimensional pFTA feature vector. Figure 5.5 illustrates an example of the empirical loss on the training set over 200 epochs for one pFTA dimension. As can be seen, the training loss decreases steadily and the training process converges after 50 epochs.

The pseudo code of the above optimization algorithm is given in Algorithm 4. For each dimension of the pFTA feature, the linear projections $\mathbf{W}_l$ are first randomly initialized. During optimization iterations, we randomly pick a mini-batch of training samples and compose the pairwise training pairs. Considering the huge number of possible training pairs, we only use 10% of the training data. In Algorithm 4, each of the L dimensions is learned independently. As a result, the randomly initialized $\mathbf{W}_l$ can be optimized to different local minima and therefore generates a complementary feature representation of the sequential data.

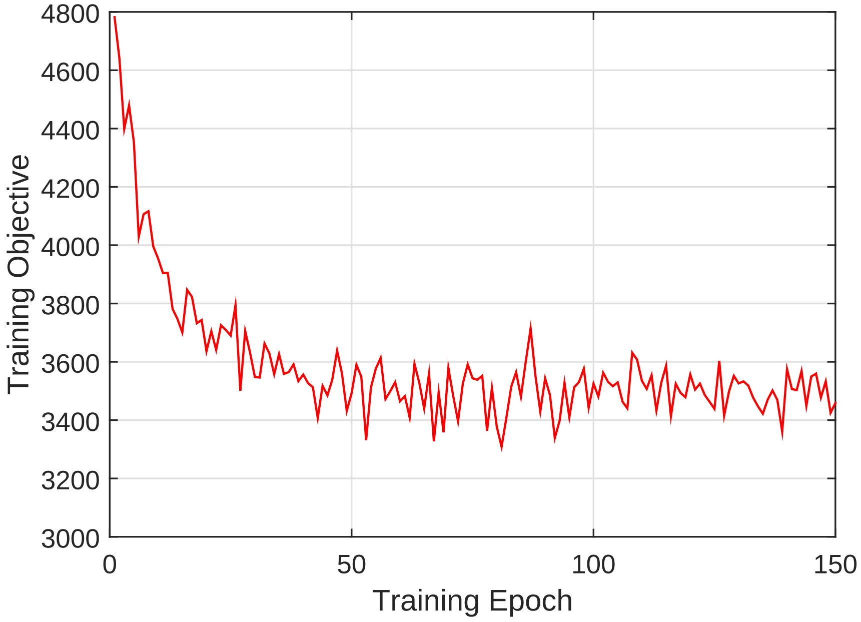


Figure 5.5: Training Loss on One pFTA Dimension through Iterations. Mini-batch size is 256. $K = 3$. Training process converges after 50 epochs.

Complexity Analysis

In this subsection, we analyze the computational complexity of the Probabilistic Temporal Order Encoding Algorithm as well as the Optimization Algorithm.

Let's first consider the complexity for encoding one dimension of the pFTA feature using a set of K linear projections. Given a sequential data of length T, each data point is a D-dimensional signal. The total cost consists of three components, $O(TDK)$ + $O(TK^2)$ + $O(log(K))$. The first component accounts for the cost of applying the linear projection to the raw signal as well as the computation of $p_{k,t}$. The second component denotes the cost of computing the probability h_k for all K latent patterns. The last component is the cost of finding the index of the pattern with the largest probability h_k. Considering that K is normally small (ranging from 2 to 8), and for most scenarios we have $K << T$ and $K << D$, therefore, the overall complexity of encoding one dimension of the pFTA feature is $O(TD)$ which is linear with respect to the size of the sequential data.

Algorithm 4 Optimization Algorithm for Probabilistic FTA

```
Input: training data $\mathbf{X}$, feature dimension $L$, encoding scale $K$.
for $l = 1$ to $L$ do
    Randomly initialize $\mathbf{W}_l = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K] \in \mathbb{R}^{D \times K}$;
    repeat
        Randomly select a mini-batch of training samples
          in $\mathbf{X}$ and compose the training pairs;
        for $k = 1$ to $K$ do
            Compute $\frac{\partial L}{\partial \mathbf{w}_k}$ over the training pairs by (5.19);
        end for
        Compute the gradient: $\nabla^{\mathbf{L}}_{\mathbf{W}} = [\frac{\partial L}{\partial \mathbf{w}_1}, \frac{\partial L}{\partial \mathbf{w}_1}, \ldots, \frac{\partial L}{\partial \mathbf{w}_K}]$;
        Update $\mathbf{W}_l$ by $\mathbf{W}_l \leftarrow \mathbf{W}_l - \eta \nabla^{\mathbf{L}}_{\mathbf{W}}$;
    until Convergence
end for
return the optimized projections $[\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_L]$ for an $L$-dimensional pFTA feature vector.
```

Next we analyze the complexity of the optimization algorithm when optimizing K linear projections that produce one dimension of the pFTA feature. Let's consider the computational cost for the mini-batch in one epoch. Suppose a mini-batch of M training sample is used. The cost for computing $\frac{\partial h_{k'}}{\partial \mathbf{w}_k}$ with respect to one training sample is $O(TDK)$ according to (5.21),(5.22),(5.23),(5.24) and (5.25). Therefore the complexity to compute all K derivatives $\frac{\partial h_{k'}}{\partial \mathbf{w}_k}$ on training samples in the mini-batch is $O(NTDK^2)$. It takes an additional $O(K)$ to compute $\frac{\partial h_{ij}}{\partial \mathbf{w}_k}$ for any pair of training data i and j. Putting all together, the overall complexity is $O(MTDK^3)$ + $O(K^2M^2)$ to compute the gradient $\nabla^{\mathbf{L}}_{\mathbf{W}}$ over the training pairs in the mini-batch. Again, considering that K is typically small, the cost for updating one epoch over a mini-batch is linear with respect to the size of the mini-batch and the dimension of the sequential data.

Performance Evaluation

In this section, we extensively evaluate the performance of the FTA and probabilistic FTA on two public human activity datasets and compare with state-of-the-art results.

Datasets

UCI Daily and Sports Activities: the UCI Daily and Sports Activities Dataset [55] consists of 19 daily and sports human activities, including *sitting, standing, lying down on back, lying down on right side, climbing stairs, descending stairs*, etc. Table 5.2 lists the complete set of the 19 activities. Each of the 19 actions is performed by 8 subjects. Five body-worn miniature inertial and magnetic sensor units are placed on five different body locations. Two of them are placed on the left knee and the right knee, two are placed on the left wrist and the right wrist, and one is placed on the chest. Each sensor unit has a 3-degree of freedom (DOF) triaxial accelerometer, a 3-DOF triaxial gyroscope and a 3-DOF magnetometer. Each unit can produce 9 signals at a time. Therefore, 45 signals are available from the 5 sensor units located over the human body to collect an array of multi-dimensional data, including acceleration, rate of turn and earth-magnetic field data. All sensor units are calibrated to acquire data at 25 Hz sampling frequency. Each subject was asked to perform all the 19 activities for 5 minutes in their own style. The entire activity sequence is further divided into 5-second segments. In total, there are 9120 segments, each of which is a 45-dimensional sensor signal sequence containing 125 data points with human annotated labels. The UCI Daily and Sports Activities Dataset is one of the largest wearable sensor signal datasets for human activities in terms of number of sensor units and activity categories [99].

Table 5.2: The 19 Action Categories in the UCI Daily and Sports Activities Dataset

Action ID	Descriptions
A1	sitting
A2	standing
A3	lying down on back
A4	lying down on right side
A5	ascending stairs
A6	descending stairs
A7	standing in an elevator still
A8	moving around in an elevator
A9	walking in a parking lot
A10	walking on a treadmill in flat
A11	walking on a treadmill with 15^o inclined position
A12	running on a treadmill
A13	exercising on a stepper
A14	exercising on a cross trainer
A15	cycling on an exercise bike in horizontal position
A16	cycling on an exercise bike in vertical position
A17	rowing
A18	jumping
A19	playing basketball

MSRActionPairs: The MSRActionPairs dataset [36] consists of 12 action types performed by 10 subjects. Each subject performs all action three times. The actions are captured by the Microsoft Kinect depth camera [81]. The dataset provides a good variety of multimodal data streams including RGB and depth streams, as well as the coordinates of detected human skeleton joints. The actions are grouped into 6 pairs of similar actions which have exactly the same human body poses but different temporal order. For example, "pick up" and "put down," "push a chair" and "pull a chair." As can be seen, the actions in a pair are easy to confuse with one another, making the dataset very challenging. It is a very suitable dataset to validate the effectiveness of the proposed temporal modeling approach for human action recognition. Although this work is focusing on wearable sensor-based HAR, we would also like to investigate the effectiveness of FTA as a general solution to other time series problem such as human action recognition in depth videos. That's why we include the performance studies on this dataset.

Experiment settings

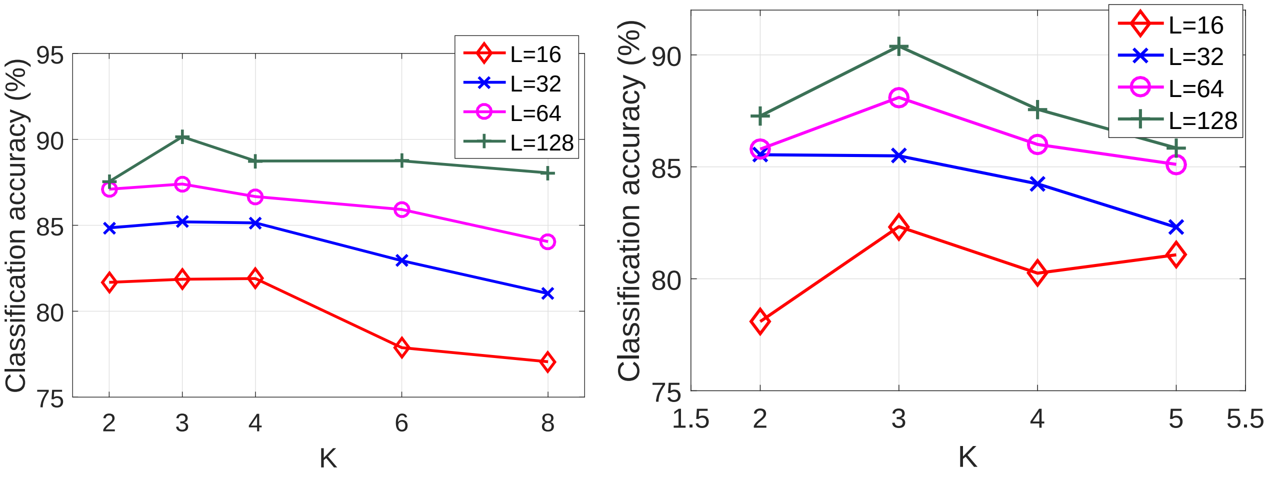
As discussed earlier, FTA and pFTA are compact feature vectors that encode the temporal dynamics of the sequential data. In the experiments, we adopt KNN to classify the sequential data according to similarity based on Hamming distance between the corresponding pFTA feature vectors. Some hyper parameters, such as number of nearest neighbors of KNN, window size and the overlap step of the preprocessing procedure are tuned based on the cross-validation on the training set.

Results on UCI Daily and Sports Activities

In the experiments, we follow the same leave-one-subject-out (L1O) cross-validation technique as in [55] so that the sensor data of 7 of the subjects are used for the training, and the data of the remaining subject are used in turn for validation. This process is repeated 8 times such that the data of each subject is used exactly once for the validation. The final classification accuracy is estimated by the average results over the above subject-based L1O strategy. To address the challenges of the highly noisy sensor signals, we employ a preprocessing step before feeding the signals to the temporal order encoding algorithm. Specifically, a sliding window with overlapping is adopted to filter the raw signals, and the *mean*, *min*, *max* and *variance* of the windowed data are computed as the preprocessed signals for the encoding process. Specifically, each of the 4 preprocessed signals is fed to FTA encoding and yields its own FTA feature vector. During the classification, the Hamming distance based on the above 4 FTA feature vectors are fused by their average, and the final result is estimated via KNN based on the fused Hamming distance. For pFTA, the early fusion strategy is adopted and the 4 preprocessed signals are concatenated to generate a 180 dimensional input vector, which will be used as the input for the probabilistic temporal order encoding process.

Impact of Encoding Scale

As discussed in the previous sections, the encoding scale K and the FTA feature dimension L are two critical parameters to the performance of the temporal order encoding approach. We first evaluate the impact of the encoding scale K on the classification accuracy. In the experiment, we test different combinations of K values and L values. K is set to 2, 3, 4, 6 and 8, and L is set to 16, 32, 64 and 128. Experimental results are shown in Figure 5.6.



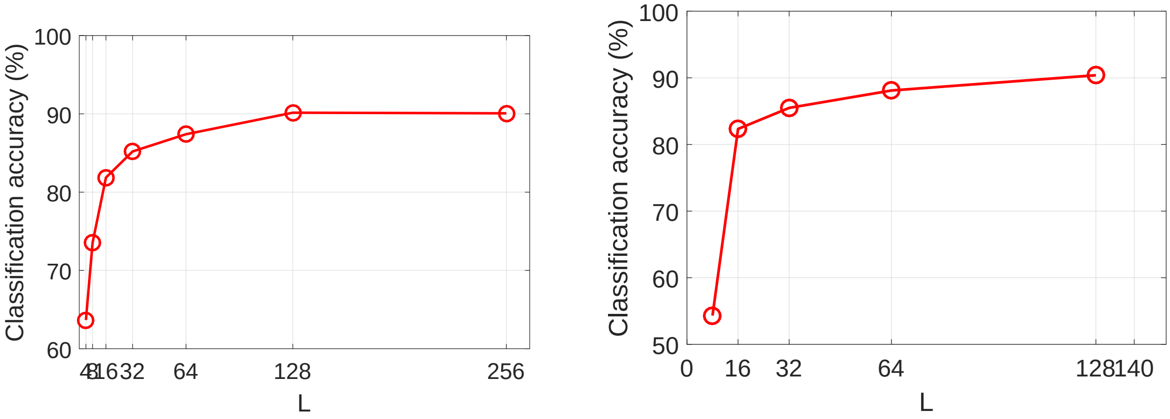
(a) FTA

(b) pFTA

Figure 5.6: Impact of the Encoding Scale of FTA and pFTA on the Classification Accuracy in the UCI Daily and Sports Activities Dataset

We have observed consistent accuracy-K behavior in both FTA feature (shown in Figure 5.6a) and pFTA feature (shown in Figure 5.6b). The classification accuracy increases as K increases and reaches the highest performance when K is set to 3 or 4. After that, the accuracy starts to drop when

K continues to increase. The results are consistent with our discussions in the previous sections. A larger encoding scale may benefit both the FTA encoding and pFTA encoding by investigating more latent patterns. However, the temporal order encoding only exploits partial order information (e.g. the index of the first-coming pattern), and the temporal order of the rest of the patterns are not used. Therefore, the discriminative capability of both the FTA feature and the pFTA feature starts to drop when K reaches the trade-off point.



(a) FTA

(b) pFTA

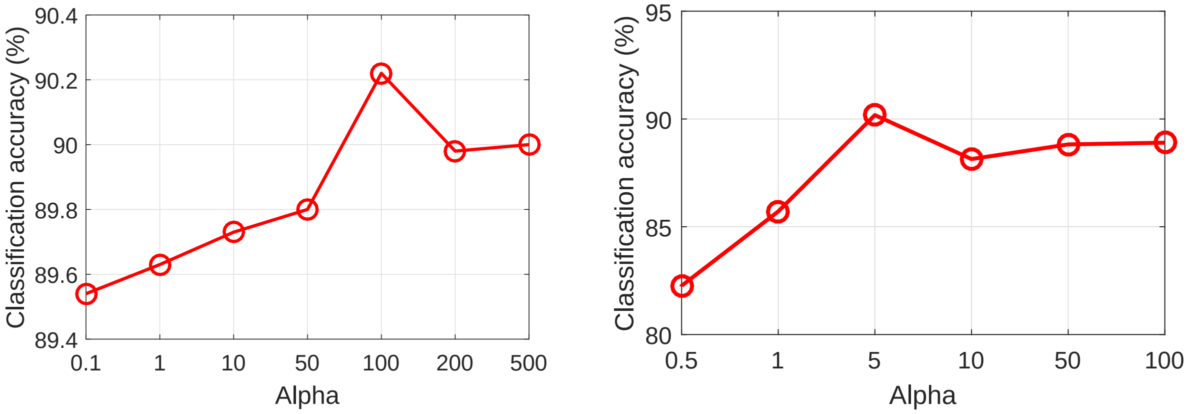
Figure 5.7: Impact of the Feature Dimension of FTA and pFTA on the Classification Accuracy in the UCI Daily and Sports Activities Dataset

Impact of Feature Dimension

Next, we further fix K to the optimal value 3 and investigate the impact of the FTA feature dimension L on the performance. Experimental results are shown in Figure 5.7. As can been seen, for both FTA feature and pFTA feature, the classification accuracy increases as the L increases. It can also be observed that the performance gain becomes increasingly marginal when L is larger than 64. This coincides with our previous discussions that a longer FTA feature dimension contributes to the discriminative power of the feature vector. However, a large L may also bring in redundant information because different randomly initialized linear projections may be optimized to the same local minima and produce redundant temporal order encoding results. As a result, the overall discriminative capability of the feature vector may be compromised. Considering the trade-off between the accuracy and the computational overhead, we set $K = 3$ and $L = 128$ in the following experiments on this dataset.

Impact of Rescale Factor

The rescale factor α in the softmax function (5.2) and the logistic function (5.12) is also an important hyper-parameter. To evaluate the impact of α, we fix K to 3 and L to 128 for FTA and 512 to pFTA, respectively, and run experiments on different α ranging from 0.1 to 500. Figure 5.8a shows that the classification accuracy of FTA feature increase as α increases and reaches the peak when α is 100. When α continues to increase, the accuracy starts to drop. One possible explanation to such a behavior is as follows. Wearable sensor signals are normally fluctuating between a narrow dynamic range. The resultant projection scores appear flat throughout the sequential data. A reasonably large rescale factor α can effectively amplify the disparity of projection scores throughout the sequential data and eventually makes the curve of probability $p_{k,t}$ in Eq.(5.2) less flat, which is beneficial to produce more informative expected moments to encode the temporal pattern inside the sequential data. Therefore, a larger α yields higher classification accuracy. Nevertheless, an excessively large α like 500 may let the peak of $p_{k,t}$ dominate the entire sequence, and always fix the expected moment at the time where the peak occurs. That's why the performance starts to drop when α is too large. Similar results can also be observed in the experiments of probabilistic FTA shown in Figure 5.8b, where the classification accuracy reaches the highest performance when a moderate α is set, e.g. $\alpha = 5$. A reasonably large rescale factor α in the logistic function of the pFTA modeling can effectively amplify the disparity of the confidence score $s_{k,t}$, and is beneficial to the representation of the dynamic characteristics of $p_{k,t}$; whereas an excessively large α makes the logistic function behave like a step function, making different confidence scores in a sequence indistinguishable and eventually decreasing the discriminative power of the resultant pFTA feature.



(a) FTA

(b) pFTA

Figure 5.8: Impact of the Rescale Factor α of FTA and pFTA on the Classification Accuracy in the UCI Daily and Sports Activities Dataset

Confusion Matrix

The confusion matrices of the classification results of FTA and pFTA on the 19 action categories are shown in Figure 5.12 and Figure 5.13, respectively. We can see that both FTA and pFTA can successfully address the confusions between most of the action classes. Even those actions with minimal movements such as “lying on back” (A3) and “lying on the right” (A4) can be accurately distinguished from each other. We do see significant confusions between “walking on a treadmill in flat” and “walking on a treadmill with 15^o inclined position” in FTA and between “standing” and “standing in an elevator” in pFTA. This is because these actions are truly similar by themselves and are extremely difficult to tell apart from one another. Thus there are not very many differences in their temporal patterns to be leveraged by the temporal order encoding algorithms.

Compare with State-of-the-art Results

We compare the performance of the temporal order encoding approach with state-of-the-art results on this dataset. In [55], several state-of-the-art results based on the same subject-based leave-one-out strategy have been reported. As opposed to the proposed method, they extract a collection of 26 engineered features from the time domain and frequency domain of the raw sensor signals, such as *min, max, mean, variance, skewness, kurtosis, auto-correlation sequence, and Discrete Fourier Transformation coefficients.* In total, a 1170-dimensional feature vector is extracted from the original sequential data. Principle Component Analysis (PCA) is then employed to reduce the dimension to 30 for classification. Several representative supervised and unsupervised algorithms, such as Dynamic Time Warping, Least Squares Method, Artificial Neural Networks and Support Vector Machines, have been employed to predict the labels of the testing samples.

Recently, deep recurrent neural networks like the Long Short-Term Memory (LSTM)[67] have been successfully employed in time series classification applications such as speech recognition [68], multimodal translation [69] and action recognition [70]. To have a comprehensive understanding of the performance of the FTA and pFTA algorithms, we also compare our technique with RNN-based methods. Specifically, we collect the source codes of LSTM [67], its improved version, differential Recurrent Neural Networks (dRNN) [71] and DeepConvLSTM model[73] which consists of a seven-layer hybrid architecture of CNN and RNN, and run the experiments following the same subject-based L1O cross-validation policy. Comparison results are summarized in Table 5.3.

FTA and pFTA achieve 88.14% and 90.18% classification accuracy. Both of them significantly outperform state-of-the-art statistical features, which demonstrates the advantage of leveraging the temporal order information for human activity recognition from sensor data. We also show that the pFTA feature with KNN classifier can achieve even higher accuracy than deep RNN-based methods such as LSTM and dRNN. Although LSTM and dRNN are powerful deep models for time series classification and, are extremely capable of interpreting the dynamics of time series, they are not able to properly capture temporal order information from sensor signals; conversely, pFTA is especially designed to effectively encode the temporal order information from the sequential data, which proves to be quite effective for human activity recognition. The hybrid DeepConvLSTM deep model achieves 91.80% classification accuracy and outperforms pFTA as well as all the other state-of-the-art methods. It is worth noting that our FTA and pFTA methods only exploit linear projections and KNN classifier and cannot compare with the nonlinear deep model like DeepConvLSTM in terms of the model complexity and number of parameters. However, our pFTA can still achieve a comparable performance with respect to DeepConvLSTM. In addition, FTA/pFTA can handle sequential data of varied length, whereas DeepConvLSTM only accepts fix-length input sequences due to its fixed 1D convolution structure.

Lastly, pFTA achieves higher classification accuracy than FTA by 2%, which demonstrates the superiority of the probability-based temporal order encoding algorithm over the expected moment-based temporal order encoding algorithm. Performance comparison on these two methods shows that pFTA is more resilient to fluctuation and oscillation in sensor signals and can better capture the temporal dynamics of the human activity sequence.

Comparison of Computational Cost

In addition to the comparison of accuracy, we also show the comparison with state-of-the-art methods on execution time. All the non-deep methods are implemented in MATLAB R2015b; LSTM and dRNN are implemented in Theano and Python. DeepConvLSTM is implemented in Theano

Table 5.3: Comparison with State-of-the-art Methods in the UCI Daily and Sports Activities Dataset. For FTA and pFTA features, L is set to 128 and K is set to 3.

State-of-the-art Methods	*Accuracy*
Statistical Feature [55] + ANN [55]	74.3%
Statistical Feature [55] + Bayesian Decision Making [55]	75.8%
Statistical Feature [55] + DTW [62]	85.2%
Statistical Feature [55] + Least Squares [55]	85.3%
Statistical Feature [55] + KNN [102]	86.0%
Statistical Feature [55] + SVM [92]	87.6%
LSTM [67]	88.88%
dRNN [71]	89.65%
DeepConvLSTM[73]	**91.80%**
Our Methods	
FTA + KNN	88.14%
pFTA + KNN	90.18%

using Lasagne package [103]. Experiments are running in a hardware configuration of Intel Quad Core i7-4770 CPU at 3.4GHz and 32G RAM. We first list the execution time for computing the statistical features [55] and FTA/pFTA feature for all the training samples and testing samples in the dataset. The execution time in Table 5.4 is in units of one second. It takes only 14.64 seconds and 15.31 seconds to compute the pFTA feature and FTA feature from the raw sensor signals in the whole dataset. Both of them are faster than the execution time of the statistical features. Such a performance demonstrates the efficiency of the temporal order encoding Algorithms.

Next, we compare the execution time of training and testing. The testing time accounts for predicting the labels of all samples in the testing set. For stochastic gradient descent-based methods like LSTM, dRNN, DeepConvLSTM, FTA and pFTA, training time accounts for the time elapse in one training epoch. Comparison results are summarized in Table 5.5. We note that pFTA is slower than

SVM in terms of testing time. This is because SVM classifier only needs to check limited number of support vectors [66] and doesn't need to scan the dataset like KNN. Except SVM, pFTA is significantly faster than the other methods by orders of magnitude because of the efficient similarity comparison based on Hamming distance between the compact pFTA feature vectors.

Table 5.4: Comparison of Execution Time for Feature Extraction in UCI Daily and Sports Activities Dataset ($K = 3$, $L = 128$ for FTA and pFTA features). Time is measured in units of one second.

Methods	*Feature Extraction Time*
Statistical Feature [55]	16.70
FTA Feature	15.31
pFTA Feature	14.64

10% of the training samples are used in the training of FTA/pFTA and LSTM/dRNN. All training samples are used for DeepConvLSTM. We note that it is difficult to have a fair comparison on the training time between these methods because many factors, such as learning rate, mini-batch size and implementation, can all impact the training time. In addition, the implementations of LSTM, dRNN and DeepConvLSTM in Theano can leverage the CuDNN library [104] and GPU to speed up the computation. Therefore, for a fair comparison, we run LSTM, dRNN and DeepConvLSTM using only the CPU in the same hardware configuration as pFTA. Results in Table 5.5 show that pFTA is three time faster than LSTM, dRNN and DeepConvLSTM in terms of training time per epoch. In addition, LSTM, dRNN and DeepConvLSTM need 100 epochs to converge while pFTA takes only 30 to 50 epochs to converge in general. Therefore pFTA is more efficient than deep learning-based methods in terms of training time.

Table 5.5: Comparison of Training Time and Testing Time by Learning-based Methods in the UCI Daily and Sports Activities Dataset. Cost is measured in units of one second. For pFTA and FTA features, L is set to 128 and K is set to 3.

Methods	*Training*	*Testing*
Statistical Feature[55] + KNN	-	0.41
Statistical Feature[55] + DTW	-	728.70
Statistical Feature[55] + SVM	0.81	0.23
LSTM [67]	625.61 (per epoch)	350
dRNN [71]	631.56 (per epoch)	354
DeepConvLSTM [73]	776.08 (per epoch)	33.90
FTA + KNN	220.08 (per epoch)	3.24
pFTA + KNN	207.13 (per epoch)	0.52

Results on MSRActionPairs

To assess the feasibility of the Probabilistic Temporal Order Encoding Algorithm as a general solution to camera-based human action recognition tasks, we also perform experiments on the Kinect-based MSRActionPairs Dataset. We follow the same test setting of [36] that the first five subjects are used for testing and the rest are used for training. Considering that visual data such as RGBD streams contain more semantic information compared with the sensor signals, we extract the following commonly used feature for better RGBD image representation.

- **3D Joint Coordinates:** 3D coordinates of the 20 joints of the skeleton.
- **3D Joint Offset:** offset of each of the 20 joints between two consecutive frames [40].
- **Pairwise angle:** angle between a pair of body segments.
- **Histogram of Velocity Component:** histogram of the velocity components of the point

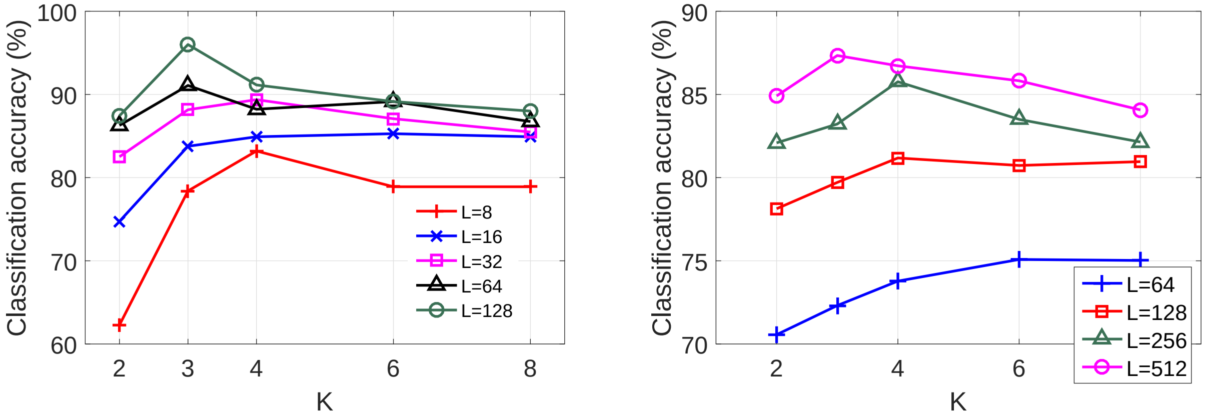
cloud around the 20 joints [95].

The above features are extracted from each frame of the RGBD sequences and are used as the input of the temporal order encoding algorithm to produce compact feature vectors. For FTA, the Hamming distances based on FTA feature vector of individual features are fused, and the KNN classifier is adopted to predict the labels of the testing data based on the fused Hamming distance. pFTA takes an early fusion strategy and concatenates individual features to form a super feature vector. The probabilistic temporal order encoding algorithm is then performed on the concatenated features to produce the pFTA feature vector. Again, KNN is used to classify the testing set according to Hamming distance on the pFTA feature vectors.

Impact of Encoding Scale

Following the same strategy as in the previous experiments, we first evaluate the impact of the encoding scale K on the performance while fixing FTA feature dimension L to different values (e.g. 16, 32, 64, 128).

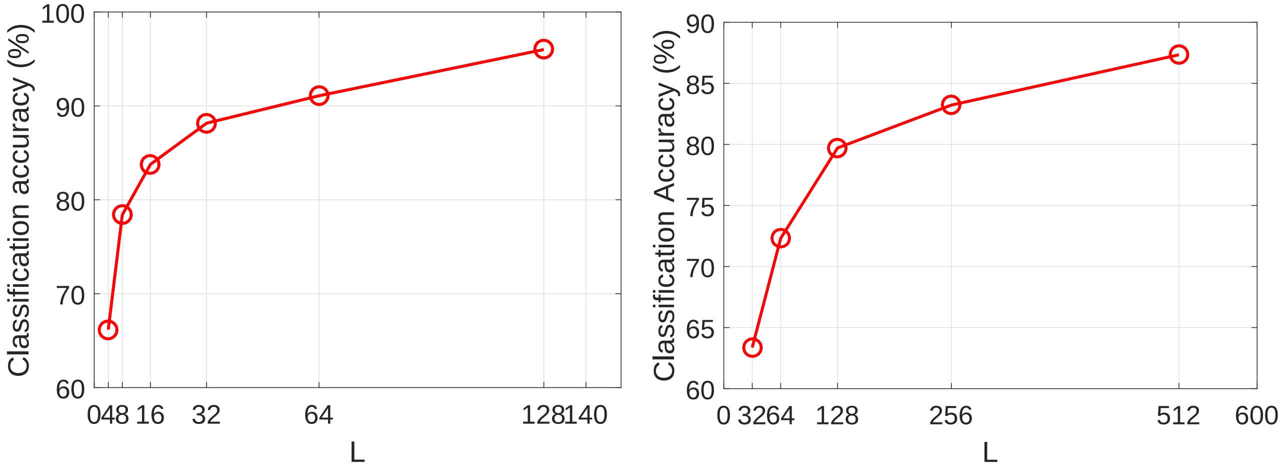
Results are shown in Figure 5.9. Similar to the results on the UCI Dataset, a moderate K (e.g. K=3 or 4) can achieve higher classification accuracy. When K is set to 2, the temporal order is determined between a pair of latent patterns and the complete order information is retained. A large K investigates multiple latent patterns. Nevertheless, to preserve the compactness, only partial information is used for FTA and pFTA encoding. As a result, a trade-off between the information encoded and the compactness of the FTA and pFTA feature vector must be considered. The above results are consistent with the results on the UCI Dataset.



(a) FTA

(b) pFTA

Figure 5.9: Impact of the Encoding Scale of FTA and pFTA on Classification Accuracy in the MSRActionPairs Dataset



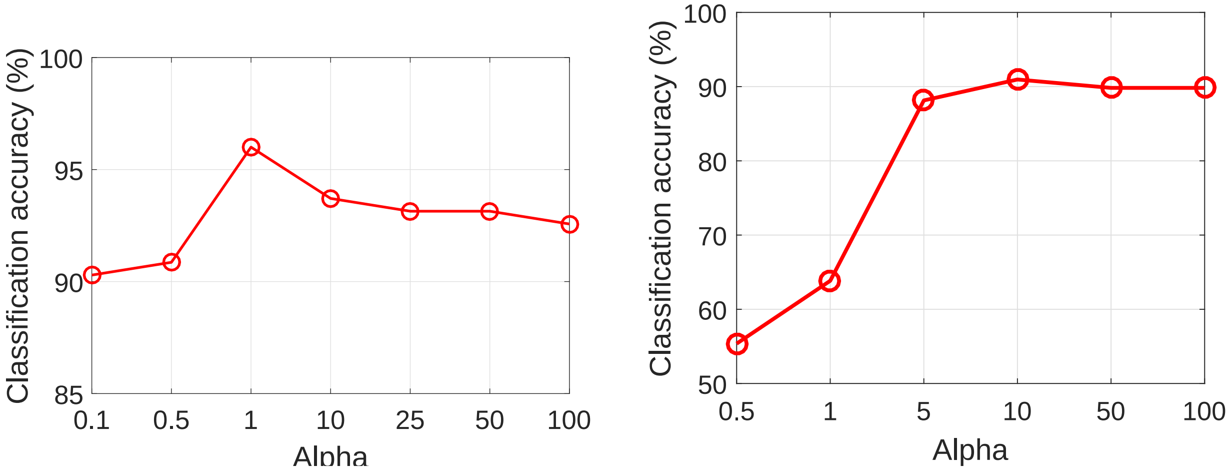
(a) FTA

(b) pFTA

Figure 5.10: Impact of the Feature Dimension of FTA and pFTA on Classification Accuracy in the MSRActionPairs Dataset

Impact of Feature Dimension

We further fix K to the optimal value ($K = 3$) and evaluate the impact of the feature dimension L on the performance. As shown in Figure 5.10, the accuracy increases as L increases. However, the performance gain gets more and more marginal when L is over 32. Again, this is because of the redundant information introduced by the long FTA and pFTA feature vectors. Considering the trade-off between the accuracy and the computational overhead, we fix K to 3 in the following experiments.



(a) FTA

(b) pFTA

Figure 5.11: Impact of the Rescale Factor α of FTA and pFTA on Classification Accuracy in the MSRActionPairs Dataset

Impact of Rescale Factor

We also evaluate the impact of the rescale factor α on the performance and show the results in Figure 5.11. The classification accuracy of both FTA feature and pFTA feature first increases as α increases, and reaches the highest performance when α arrives at 1 (FTA) and 10 (pFTA).

After that, the classification accuracy starts to drop when α continues to increase. Such results are consistent with the experiments in the UCI dataset. A reasonably larger α can help enhance the discriminative power of both FTA feature and pFTA feature by reshaping the curve of $p_{k,t}$ and amplifying the dynamic characteristics underlying the sequential data.

Confusion Matrix

The confusion matrices of the classification results of FTA and pFTA are illustrated in Figure 5.14 and Figure 5.15, respectively. It is clear that both FTA and pFTA can perfectly distinguish the differences between the majority of the action pairs.

Comparison with State-of-the-Art Results

To assess the performance of FTA and pFTA with respect to the existing camera-based methods, we compare our techniques with state-of-the-art results. We first compare FTA and pFTA with two well-known methods, DTW and RNN, which are capable of interpreting the temporal dynamics of time series. For a fair comparison, the same preprocessed 358-dimensional feature, concatenated by 3D joint coordinates, pairwise angle, and histogram of velocity component, are used by DTW, LSTM, dRNN and pFTA. For FTA, a late fusion strategy is adopted to merge the results of individual features. DeepConvLSTM is an end-to-end deep model which doesn't require any feature extraction. Therefore, we concatenate the 3D coordinates of the 20 joints of the skeleton in each frame and form the 60-dimensional input data. Videos in the MSRActionPairs Dataset have different length. Considering that DeepConvLSTM only accepts fixed-size input, we normalized the RGBD videos of varied length into 100-frame-length video by linear interpolation. As shown in Table 5.6, both FTA and pFTA significantly outperform DTW, LSTM and dRNN, which demonstrates the strength of temporal order inside the human action sequence.

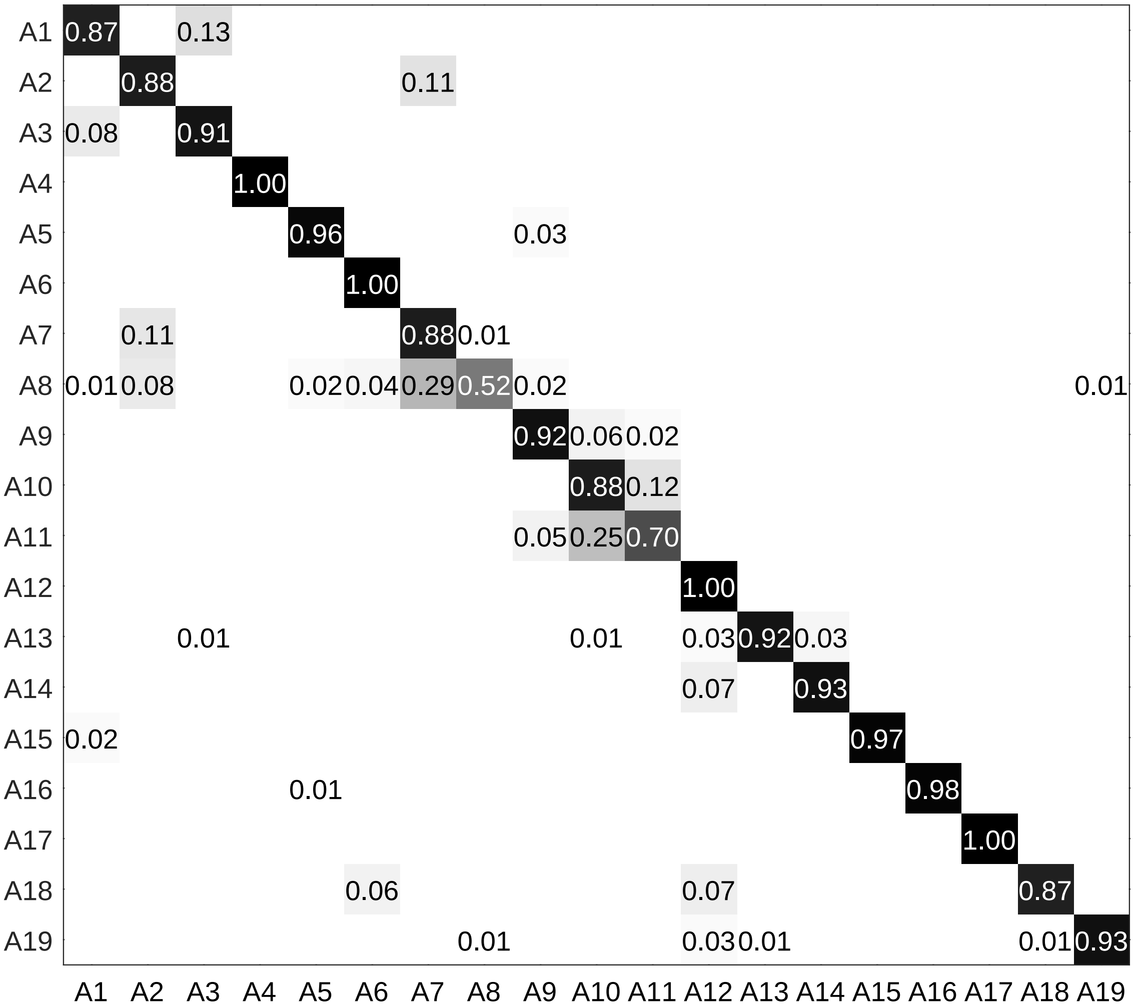


Figure 5.12: Confusion Matrix of the Classification Results by FTA Feature in the UCI Sport and Daily Activity Dataset

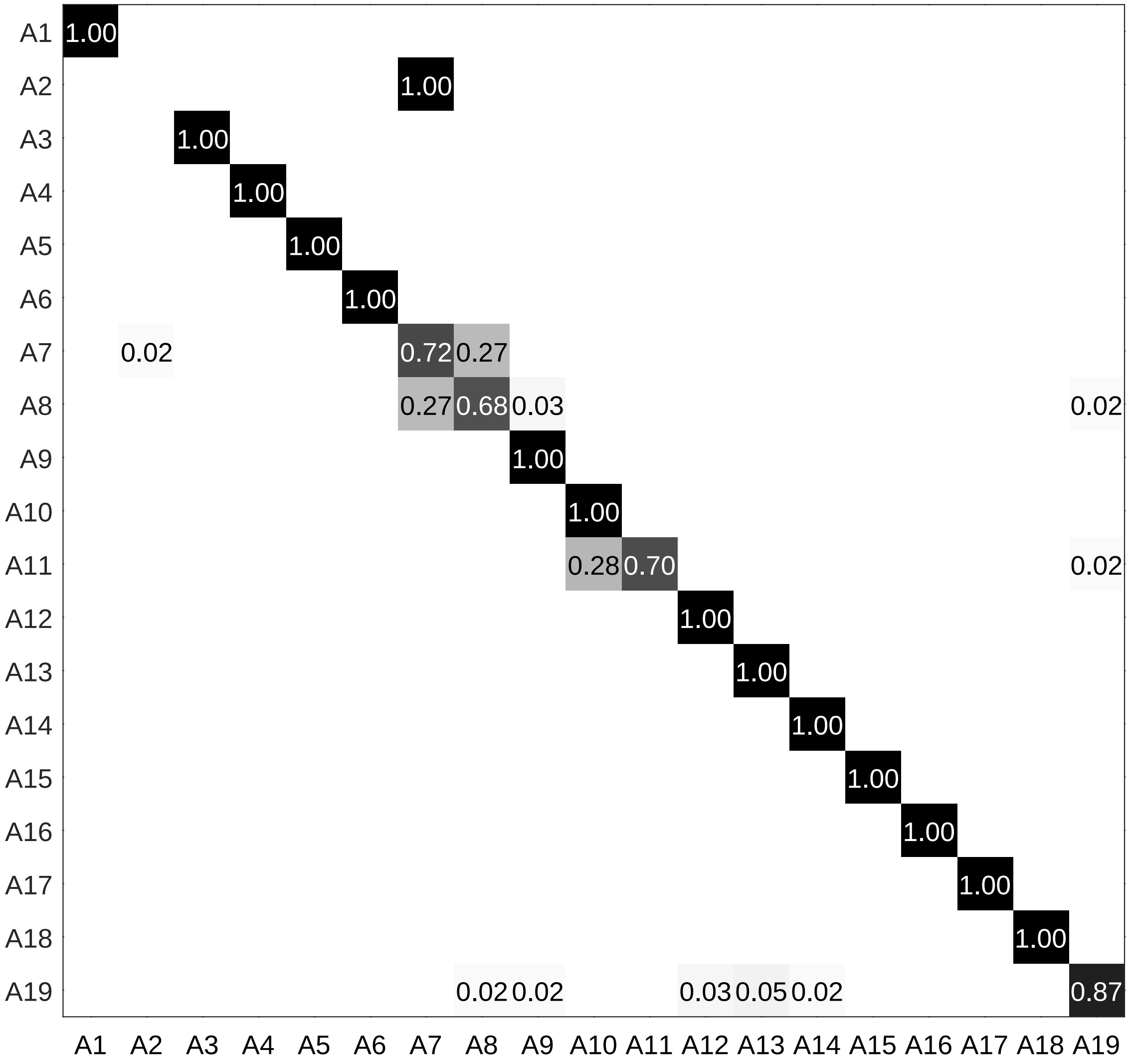
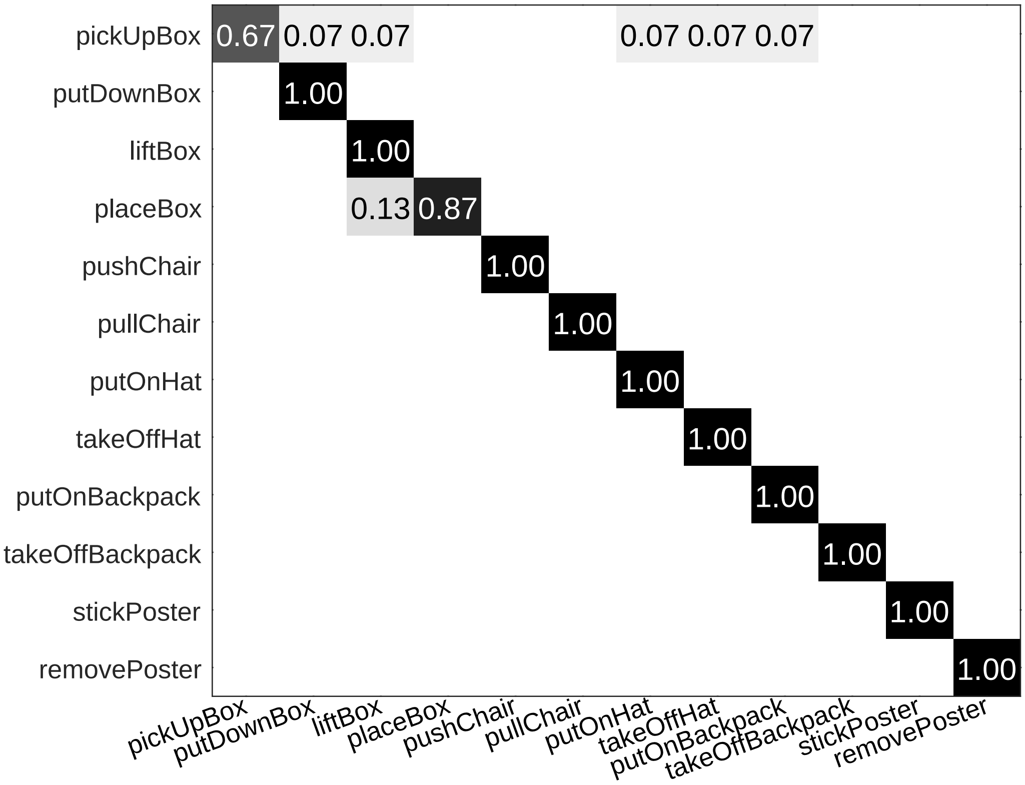


Figure 5.13: Confusion Matrix of the Classification Results by pFTA Feature in the UCI Sport and Daily Activity Dataset



	pickUpBox	putDownBox	liftBox	placeBox	pushChair	pullChair	putOnHat	takeOffHat	putOnBackpack	takeOffBackpack	stickPoster	removePoster
pickUpBox	0.67	0.07	0.07				0.07	0.07	0.07			
putDownBox		1.00										
liftBox			1.00									
placeBox			0.13	0.87								
pushChair					1.00							
pullChair						1.00						
putOnHat							1.00					
takeOffHat								1.00				
putOnBackpack									1.00			
takeOffBackpack										1.00		
stickPoster											1.00	
removePoster												1.00

Figure 5.14: Confusion Matrix of the Classification Results by FTA in the MSRActionPairs Dataset

Next, we compare FTA with state-of-the-art camera-based methods for 3D human action recognition. Table 5.6 shows that FTA and pFTA have achieved comparable results with respect to state-of-the-art camera-based methods in terms of accuracy. It is worth noting that many of those methods adopt handcrated features, such as histogram of 4D normals [36] and 3D hypersurface polynormals of the point cloud [105], which are not able to apply to general problems. Nevertheless, FTA and pFTA are feature independent and can work as a general solution to a wide range of time series classification problems. The results in the MSRActionPairs Dataset demonstrate the effectiveness of the temporal order encoding algorithm in human action recognition in 3D videos.

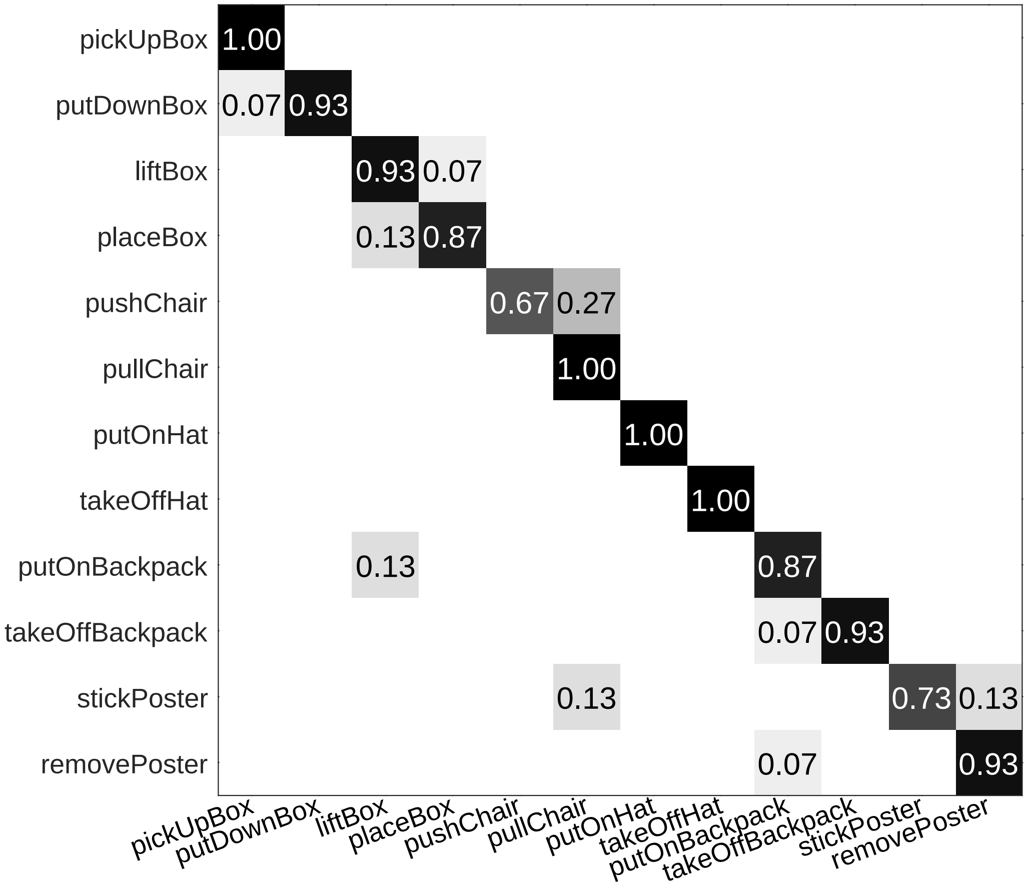


Figure 5.15: Confusion Matrix of the Classification Results by pFTA in the MSRActionPairs Dataset

Lastly, we notice that pFTA achieve lower accuracy than FTA in this dataset which is against our discussion in the previous sections. Such a result may be due to the early fusion strategy adopted by pFTA. Different from the mulitdimensional sensor signals with similar physical meanings, visual features are of different dimensionality and physical meanings. Therefore, feature level fusion may be less effective than score level fusion in this dataset.

Table 5.6: Comparison with State-of-the-art Results in the MSRActionPairs Dataset. For FTA features, L is set to 128 and K is set to 3. For pFTA features, L is set to 512 and K is set to 3.

Existing Methods	*Accuracy*
DTW + KNN	69.49%
Depth Motion Maps [35]	66.11%
Actionlet + LOP + Pyramid [41]	82.2%
LSTM [67]	85.82%
dRNN [71]	87.01%
DeepConvLSTM [73]	93.18%
HON4D [36]	96.67%
SNV [105]	98.89%
Bilinear Heterogeneous Information Machine [106]	**100%**
Our Methods	
FTA + KNN	92.00%
pFTA + KNN	90.96%

Comparison of Computational Cost

We also evaluate the computational cost of the proposed temporal order encoding algorithm and compared the execution time, including feature extraction, training and testing, with that of state-of-the-art approaches. Comparison results are summarized in Table 5.7. For the stochastic gradient descent-based methods such as LSTM, dRNN, DeepConvLSTM, FTA and pFTA, training time listed in the table denotes the time elapse in one epoch. It takes only 2.37 seconds to extract the pFTA feature vector from the preprocessed features for the whole dataset. pFTA takes significant less feature extraction time than camera-based Super Normal Vectors method [105] by an order of magnitude. In testing, it takes only 0.07 second for pFTA + KNN to predict labels of all samples in the testing set, which is also significantly faster than all the other methods listed in Table 5.7.

Table 5.7: Comparison of Execution Time in the MSRActionPairs Dataset. Cost is measured in units of one second. For FTA feature, L is set to 128 and K is set to 3. For pFTA feature, L is set to 512 and K is set to 3.

Methods	*Preprocessing*	*Feature Extraction*	*Training*	*Testing*
DTW	933.18	-	-	9.415
SNV [105]	-	12801	1145	1.64
LSTM [67]	933.18	-	1161.02 (per epoch)	441.12
dRNN [71]	933.18	-	1137.40 (per epoch)	425.86
DeepConvLSTM [73]	1.25	-	22.37 (per epoch)	8.26
FTA	952.52	3.39	7.21 (per epoch)	0.11
pFTA	933.18	2.37	200.1 (per epoch)	0.07

Lastly, we compare the training time between FTA, pFTA, SNV, LSTM, dRNN and DeepConvLSTM. For FTA and pFTA, we sample 50% of the training set to compose the pairwise training data. For LSTM, dRNN, DeepConvLSTM and SNV, all training samples are used. Although FTA and pFTA adopts pairwise-based training, its training time per epoch is still shorter than LSTM and dRNN. In the experiment, it takes around 30 epochs for FTA and pFTA to converge. The other deep learning methods like LSTM, dRNN and DeepConvLSTM take 100 epoch to converge. FTA and pFTA have shown great efficiency in feature extraction, training and testing with respect to other deep and non-deep methods listed in Table 5.7.

Summary

Wearable sensor-based HAR has been an active research field for more than a decade with innumerable applications. With the advancement of mobile technology, smart watches and smart phones with built-in miniature inertial sensors have further increased the demand for HAR in daily and fitness activities. In this chapter, we have addressed the challenge of HAR from the perspective of temporal order encoding and exploited the unique temporal dynamics underlying the sequential data. Extensive performance evaluations on a wearable sensor-based HAR dataset and a RGBD human action dataset have demonstrated that the proposed compact FTA feature and pFTA feature have outperformed start-of-the-art statistical features in both accuracy and efficiency. FTA and pFTA can also achieve comparable results with respect to state-of-the-art camera-based methods with higher efficiency.

CHAPTER 6: CONCLUSION AND FUTURE WORK

Conclusion

With the development of sensing technology and the popularity of mobile devices, new off-the-shelf products, such as Microsoft Kinect and smart wristbands, have opened a unprecedented opportunity to solve the challenging HAR problem and spawned a great number of applications. Although competitive performance has been reported, existing methods for HAR focus on the statistical or spatial representation of the human activity sequence; leaving the internal temporal dynamics of the human activity sequence largely under-researched. As a result, they often face the challenge of recognizing visually similar activities composed of dynamic patterns in different temporal order. Besides, existing model-driven methods based on handcrafted features and sophisticated classifiers are computationally prohibited and are unable to scale to a large dataset. This dissertation have addressed the above challenges from three different perspectives; namely, 3D spatial relationship modeling, dynamic temporal quantization, and temporal order encoding. Specifically, the main contribution of the dissertation are summarized as follows:

- A novel octree-based technique for computing 3D spatial relationship between objects in indoor environment has been proposed. A set of 26 spatial operators are further defined and implemented to be included in an SQL-style event query language. HAR is then resolved by spatial-event query processing based on the 3D spatial relationship between a human and the surrounding objects.
- A clustering-like dynamic quantization algorithm has been introduced to achieve an optimal dynamic quantization of human action videos for 3D human action recognition in sports.
- A novel temporal order encoding approach is developed to encode the temporal order of

latent patterns inside the sequential data for human activity recognition. We have further proposed a First-Take-All by expected moment algorithm and a probabilistic First-Take-All algorithm to generate a compact feature vector to represent the whole sequence. Optimization algorithms have also been proposed for FTA and pFTA to learn the optimal latent patterns and enhance the discriminative power of the feature representation.

We have extensively evaluated the proposed approaches on public 3D human action datasets and experimental results show the advantages of the proposed technique over state-of-the-arts in both accuracy and efficiency.

Future Work

In this dissertation, we have shown the success of employing spatial and temporal modeling for 3D human action recognition in several public 3D human action datasets. Nevertheless, this is not the end of the research, many other interesting questions are still open and demand further investigation. We summarize the future work as follows.

We have focused on the human action recognition on 3D data in this dissertation. One of the major reason is that 3D human video captured from depth sensors can be effectively filtered to produce relatively clean human objects in the fore ground. As a result, pre-segmented 3D human action video has a high signal-to-noise ratio and is robust to the spatial and temporal modeling algorithms. However, the majority data in real-world are 2D videos. It is still challenging to accurately segment human objects from the background in 2D videos. Useful temporal patterns in human action videos may be overwhelmed by the background noise and the proposed modeling algorithms may be compromised. We would like to address these challenges and extend the proposed modeling approach to 2D videos in our future work.

Although we have achieved impressive accuracy and outperformed state-of-the-art approaches in 3D huamn action recognition, the proposed techniques still fall into the category of handcrafted approaches which seek to develop dedicated models for higher classification rate. Inspired by the huge success in deep learning and deep neural networks in general object detection, speech recognition and natural language processing, we would like to exploit deep visual features in individual video frames for better image representation. We would also like to investigate the Recurrent Neural Networks (RNN) and especially the Long Short-Term Memory (LSTM) as well as the temporal pooling techniques for better video representation. Eventually, an end-to-end Convolutional Neural Networks (CNN) and RNN structure will be the ideal model for 2D human action recognition in large-scale datasets.

LIST OF REFERENCES

[1] J. Ye and K. A. Hua, “Exploiting depth camera for 3d spatial relationship interpretation,” in *Proceedings of the 4th ACM Multimedia Systems Conference*, ser. MMSys ’13. ACM, 2013, pp. 151–161.

[2] J. Ye and K. Hua, “Octree-based 3d logic and computation of spatial relationships in live video query processing,” *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 11, no. 2, pp. 28:1–28:23, 2015.

[3] K. Khoshelham and S. O. Elberink, “Accuracy and resolution of kinect depth data for indoor mapping applications,” *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.

[4] M. Raptis, D. Kirovski, and H. Hoppe, “Real-time classification of dance gestures from skeleton animation,” in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation*. ACM, 2011, pp. 147–156.

[5] M. Gabel, R. Gilad-Bachrach, E. Renshaw, and A. Schuster, “Full body gait analysis with kinect,” in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2012, pp. 1964–1967.

[6] E. E. Stone and M. Skubic, “Passive in-home measurement of stride-to-stride gait variability comparing vision and kinect sensing,” in *2011 Annual international conference of the IEEE engineering in medicine and biology society*. IEEE, 2011, pp. 6491–6494.

[7] H.-m. J. Hsu, “The potential of kinect in education,” *International Journal of Information and Education Technology*, vol. 1, no. 5, p. 365, 2011.

[8] J. Hu, R. Hu, Z. Wang, Y. Gong, and M. Duan, "Kinect depth map based enhancement for low light surveillance image," in *2013 IEEE International Conference on Image Processing*. IEEE, 2013, pp. 1090–1094.

[9] J. Freeman, "The modeling of spatial relations," *Computer Graphics and Image Processing*, vol. 4, pp. 156–171, 1975.

[10] I. Bloch, "Fuzzy spatial relationships for image processing and interpretation: a review," *Image and Vision Computing*, vol. 23, no. 2, pp. 89–110, 2005.

[11] J. M. Keller and X. Wang, "Comparison of spatial relation definitions in computer vision," in *Proceedings of The 3rd International Symposium on Uncertainty Modeling and Analysis*. IEEE, Sept 1995, pp. 679–684.

[12] K.-P. Gapp, "From vision to language: A cognitive approach to the computation of spatial relations in 3d space," 1994.

[13] K. Miyajima and A. Ralescu, "Spatial organization in 2d segmented images: representation and recognition of primitive spatial relations," *Fuzzy Sets and Systems*, vol. 65, no. 2-3, pp. 225–236, August 1994.

[14] N. Salamat and E.-H. Zahzah, "On the improvement of combined fuzzy topological and directional relations information," *Pattern Recognition*, vol. 45, no. 4, pp. 1559–1568, 2012.

[15] J. M. Keller and X. Wang, "Learning spatial relationships in computer vision," in *Proceedings of The 5th IEEE International Conference on Fuzzy Systems*. IEEE, Sept 1996, pp. 118–124.

[16] B. Rosman and S. Ramamoorthy, "Learning spatial relationships between objects," *International Journal of Rotbotics Research*, vol. 30, no. 1, pp. 1328–1342, 2011.

[17] I. Bloch, “Fuzzy relative position between objects in image processing: a morphological approach,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 7, pp. 657–664, July 1999.

[18] I. Bloch, O. Colliot, and J. Roberto M. Cesar, “On the ternary spatial relation between,” *IEEE Transaction on System, Man and Cybernetics*, vol. 36, no. 2, pp. 312–327, April 2006.

[19] C. M. Takemura, R. M. Cesar Jr, and I. Bloch, “Modeling and measuring the spatial relation along: regions, contours and fuzzy sets,” *Pattern Recognition*, vol. 45, no. 2, pp. 757–766, 2012.

[20] Z. Jia, A. Gallagher, A. Saxena, and T. Chen, “3d-based reasoning with blocks, support, and stability,” in *Computer Vision and Pattern Recognition, 2013 IEEE Conference on*, 2013, pp. 1–8.

[21] K. Yun, J. Honorio, D. Chattopadhyay, T. L. Berg, and D. Samaras, “Two-person interaction detection using body-pose features and multiple instance learning,” in *Computer Vision and Pattern Recognition Workshops , 2012 IEEE Computer Society Conference on*, 2012, pp. 28–35.

[22] A. Borrmann, S. Schraufstetter, C. Van Treeck, and E. Rank, “An octree-based implementation of directional operators in a 3d spatial query language for building information models,” in *Proc. of the 24th CIB-W78 Conf. on IT in Construction*, 2007.

[23] R. Peng, A. J. Aved, and K. A. Hua, “Real-time query processing on live videos in networks of distributed cameras,” *International Journal of Interdisciplinary Telecommunications and Networking*, vol. 2, no. 1, p. 27, 2010.

[24] A. J. Aved and K. A. Hua, “A general framework for managing and processing live video data with privacy proctection,” *Multimedia Systems*, vol. 18, no. 2, pp. 123–143, Feb 2012.

[25] P. K. Atrey, M. S. Kankanhalli, and R. Jain, “Information assimilation framework for event detection in multimedia surveillance systems,” *Multimedia systems*, vol. 12, no. 3, pp. 239–253, 2006.

[26] P. K. Atrey, “A hierarchical model for representation of events in multimedia observation systems,” in *Proceedings of the 1st ACM international workshop on Events in multimedia*. ACM, 2009, pp. 57–64.

[27] I. Laptev, “On space-time interest points,” *International Journal of Computer Vision*, vol. 64, no. 2-3, pp. 107–123, 2005.

[28] P. Dollár, V. Rabaud, G. Cottrell, and 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.

[29] A. F. Bobick and J. W. Davis, “The recognition of human movement using temporal templates,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 3, pp. 257–267, 2001.

[30] A. A. Efros, A. C. Berg, G. Mori, and J. Malik, “Recognizing action at a distance,” in *Computer Vision, 2003. Proceedings. 9th IEEE International Conference on*. IEEE, 2003, pp. 726–733.

[31] A. Yilmaz and M. Shah, “Actions sketch: A novel action representation,” in *Computer Vision and Pattern Recognition, 2005. IEEE Computer Society Conference on*. IEEE, 2005, pp. 984–989.

[32] A. Agarwal and B. Triggs, “Recovering 3d human pose from monocular images,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 1, pp. 44–58, 2006.

[33] R. Gupta, A. Y.-S. Chia, and D. Rajan, “Human activities recognition using depth images,” in *Proceedings of the 21st ACM international conference on Multimedia*. IEEE, 2013, pp. 283–292.

[34] W. Li, Z. Zhang, and Z. Liu, “Action recognition based on a bag of 3d points,” in *Computer Vision and Pattern Recognition Workshops, 2010 IEEE Computer Society Conference on*. IEEE, 2010, pp. 9–14.

[35] X. Yang, C. Zhang, and Y. Tian, “Recognizing actions using depth motion maps-based histograms of oriented gradients,” in *Proceedings of the 20th ACM international conference on Multimedia*. ACM, 2012, pp. 1057–1060.

[36] O. Oreifej and Z. Liu, “Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences,” in *Computer Vision and Pattern Recognition, 2013 IEEE Conference on*, pp. 716–723.

[37] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, “Real-time human pose recognition in parts from single depth images,” *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.

[38] L. Sun and K. Aizawa, “Action recognition using invariant features under unexampled viewing conditions,” in *Proceedings of the 21st ACM international conference on Multimedia*. ACM, 2013, pp. 389–392.

[39] L. Xia, C.-C. Chen, and J. Aggarwal, “View invariant human action recognition using histograms of 3d joints,” in *Computer Vision and Pattern Recognition Workshops, 2012 IEEE Computer Society Conference on*. IEEE, 2012, pp. 20–27.

[40] Y. Zhu, W. Chen, and G. Guo, "Fusing spatiotemporal features and joints for 3d action recognition," in *Computer Vision and Pattern Recognition workshops, 2013 IEEE Conference on*. IEEE, 2013.

[41] J. Wang, Z. Liu, Y. Wu, and J. Yuan, "Mining actionlet ensemble for action recognition with depth cameras," in *Computer Vision and Pattern Recognition, 2012 IEEE Conference on*, pp. 1290–1297.

[42] X. Zhao, X. Li, C. Pang, X. Zhu, and Q. Z. Sheng, "Online human gesture recognition from motion data streams," in *Proceedings of the 21st ACM international conference on Multimedia*. ACM, 2013, pp. 23–32.

[43] R. Vemulapalli, F. Arrate, and R. Chellappa, "Human action recognition by representing 3d skeletons as points in a lie group," in *Computer Vision and Pattern Recognition, IEEE Conference on*. IEEE, 2014, pp. 588–595.

[44] B. Ni, G. Wang, and P. Moulin, "Rgbd-hudaact: A color-depth video database for human daily activity recognition," in *Consumer Depth Cameras for Computer Vision*. Springer, 2013, pp. 193–208.

[45] H. Zhang and L. E. Parker, "4-dimensional local spatio-temporal features for human activity recognition," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, 2011, pp. 2044–2049.

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

[47] F. Lv and R. Nevatia, "Recognition and segmentation of 3-d human action using hmm and multi-class adaboost," in *Computer Vision–ECCV 2006*, 2006, pp. 359–372.

[48] J. Sung, C. Ponce, B. Selman, and A. Saxena, "Unstructured human activity detection from rgbd images," in *Robotics and Automation, 2012 IEEE International Conference on*. IEEE, 2012, pp. 842–849.

[49] L. Han, X. Wu, W. Liang, G. Hou, and Y. Jia, "Discriminative human action recognition in the learned hierarchical manifold space," *Image and Vision Computing*, vol. 28, no. 5, pp. 836–849, 2010.

[50] M. Müller and T. Röder, "Motion templates for automatic classification and retrieval of motion capture data," in *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2006, pp. 137–146.

[51] X. Yang and Y. Tian, "Super normal vector for activity recognition using depth sequences," in *Computer Vision and Pattern Recognition, IEEE Conference on*. IEEE, 2014, pp. 804–811.

[52] J.-Y. Yang, J.-S. Wang, and Y.-P. Chen, "Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers," *Pattern recognition letters*, vol. 29, no. 16, pp. 2213–2220, 2008.

[53] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, "Activity recognition from accelerometer data," in *AAAI*, vol. 5, 2005, pp. 1541–1546.

[54] Z. He and L. Jin, "Activity recognition from acceleration data based on discrete consine transform and svm," in *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*. IEEE, 2009, pp. 5041–5044.

[55] K. Altun, B. Barshan, and O. Tunçel, "Comparative study on classifying human activities with miniature inertial and magnetic sensors," *Pattern Recognition*, vol. 43, no. 10, pp. 3605–3620, 2010.

[56] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, “A public domain dataset for human activity recognition using smartphones.” in *ESANN*, 2013.

[57] J. Parkka, M. Ermes, P. Korpipaa, J. Mantyjarvi, J. Peltola, and I. Korhonen, “Activity classification using realistic data from wearable sensors,” *IEEE Transactions on information technology in biomedicine*, vol. 10, no. 1, pp. 119–128, 2006.

[58] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher, “Activity recognition and monitoring using multiple sensors on different body positions,” in *International Workshop on Wearable and Implantable Body Sensor Networks (BSN’06)*. IEEE, 2006, pp. 4–pp.

[59] M. Ermes, J. Parkka, and L. Cluitmans, “Advancing from offline to online activity recognition with wearable sensors,” in *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2008, pp. 4451–4454.

[60] O. Politi, I. Mporas, and V. Megalooikonomou, “Human motion detection in daily activity tasks using wearable sensors,” in *2014 22nd European Signal Processing Conference (EUSIPCO)*. IEEE, 2014, pp. 2315–2319.

[61] J.-L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita, “Transition-aware human activity recognition using smartphones,” *Neurocomputing*, vol. 171, pp. 754–767, 2016.

[62] D. J. Berndt and J. Clifford, “Using dynamic time warping to find patterns in time series.” in *KDD workshop*, vol. 10, no. 16. Seattle, WA, 1994, pp. 359–370.

[63] S. I. Gallant, “Perceptron-based learning algorithms,” *IEEE Transactions on neural networks*, vol. 1, no. 2, pp. 179–191, 1990.

[64] H. Zhang, “The optimality of naive bayes,” *AA*, vol. 1, no. 2, p. 3, 2004.

[65] T. K. Ho, “The random subspace method for constructing decision forests,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[66] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[67] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[68] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.

[69] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[70] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Sequential deep learning for human action recognition," in *International Workshop on Human Behavior Understanding*. Springer, 2011, pp. 29–39.

[71] V. Veeriah, N. Zhuang, and G.-J. Qi, "Differential recurrent neural networks for action recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4041–4049.

[72] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1110–1118.

[73] F. J. Ordóñez and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, 2016.

[74] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena, "Labeling 3d scenes for personal assistant robots," *arXiv preprint arXiv:1106.5551*, 2011.

[75] A. Kasper, R. Jakel, and R. Dillmann, “Using spatial relations of objects in real world scenes for scene structuring and scene understanding,” in *Proceedings of The 15th International Conference on Advanced Robotics (ICAR)*. IEEE, June 2011, pp. 20–23.

[76] J. Liu, “A method of spatial reasoning based on qualitative trigonometry,” *Artificial Intelligence*, vol. 98, no. 1, pp. 137–168, January 1998.

[77] V. Hedau, D. Hoiem, and D. Forsyth, “Recovering the spatial layout of cluttered rooms,” in *Proceedings of IEEE The 12th International Conference on Computer Vision (ICCV)*. IEEE, Sept 2009, pp. 1849–1856.

[78] A. Gupta, S. Satkin, A. A. Efros, and M. Hebert, “From 3d scene geometry to human workspace,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2011, pp. 1961–1968.

[79] B. J. Kuipers and T. S. Levitt, “Navigation and mapping in large-scale space,” *AI magazine*, vol. 9, no. 2, p. 25, 1988.

[80] M. Egenhofer and R. Franzosa, “Point-set spatial relations,” *International Journal of Geographical Information Systems*, vol. 5, no. 2, pp. 161–174, 1991.

[81] Z. Zhang, “Microsoft kinect sensor and its effect,” *MultiMedia, IEEE*, vol. 19, no. 2, pp. 4–10, 2012.

[82] D. Meagher, “Geometric modeling using octree encoding,” *Computer Graphics and Image Processing*, vol. 19, no. 2, pp. 129–147, June 1982.

[83] A. S. Glassner, *An introduction to ray tracing*. Elsevier, 1989.

[84] Y. Tashiro, “On methods for generating uniform random points on the surface of a sphere,” *Annals of the Institute of Statistical Mathematics*, vol. 29, no. 1, pp. 295–300, 1977.

[85] Khronos, “OpenCL,” http://www.khronos.org/opencl/, 2013.

[86] K. Lai, L. Bo, X. Ren, and D. Fox, “A large-scale hierarchical multi-view rgbd object dataset,” in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 1817–1824.

[87] M. Raptis, D. Kirovski, and H. Hoppe, “Real-time classification of dance gestures from skeleton animation,” in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation*. ACM, 2011, pp. 147–156.

[88] A. Nagendran, R. Pillat, A. Kavanaugh, G. Welch, and C. Hughes, “Amities: Avatar-mediated interactive training and individualized experience system,” in *Proceedings of the 19th ACM Symposium on Virtual Reality Software and Technology*, ser. VRST ’13, 2013, pp. 143–152.

[89] L. Gallo, A. P. Placitelli, and M. Ciampi, “Controller-free exploration of medical image data: Experiencing the kinect,” in *Computer-based medical systems (CBMS), 2011 24th international symposium on*. IEEE, 2011, pp. 1–6.

[90] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” in *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 2. IEEE, 2006, pp. 2169–2178.

[91] L. Xia, C.-C. Chen, and J. Aggarwal, “View invariant human action recognition using histograms of 3d joints,” in *Computer Vision and Pattern Recognition Workshops, 2012 IEEE Computer Society Conference on*. IEEE, 2012, pp. 20–27.

[92] C.-C. Chang and C.-J. Lin, “Libsvm: a library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[93] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, “On combining classifiers,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 3, pp. 226–239, 1998.

[94] J. Webb and J. Ashley, *Beginning Kinect Programming with the Microsoft Kinect SDK*. Apress, 2012.

[95] J. Ye, K. Li, G.-J. Qi, and K. A. Hua, “Temporal order-preserving dynamic quantization for human action recognition from multimodal sensor streams,” in *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*. ACM, 2015, pp. 99–106.

[96] P. C. Ribeiro and J. Santos-Victor, “Human activity recognition from video: modeling, feature selection and classification architecture,” in *Proceedings of International Workshop on Human Activity Recognition and Modelling*. Citeseer, 2005, pp. 61–78.

[97] N. Robertson and I. Reid, “A general method for human activity recognition in video,” *Computer Vision and Image Understanding*, vol. 104, no. 2, pp. 232–248, 2006.

[98] X. Yang and Y. Tian, “Super normal vector for activity recognition using depth sequences,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 804–811.

[99] O. D. Lara and M. A. Labrador, “A survey on human activity recognition using wearable sensors,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1192–1209, 2013.

[100] J. Yin, Q. Yang, and J. J. Pan, “Sensor-based abnormal human-activity detection,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 8, pp. 1082–1090, 2008.

[101] J. Yagnik, D. Strelow, D. A. Ross, and R.-s. Lin, “The power of comparative reasoning,” in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2431–2438.

[102] A. R. Webb, *Statistical pattern recognition.* John Wiley & Sons, 2003.

[103] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri, D. Maturana, M. Thoma, E. Battenberg, J. Kelly *et al.*, "Lasagne: First release," *Zenodo: Geneva, Switzerland*, 2015.

[104] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," *arXiv preprint arXiv:1410.0759*, 2014.

[105] X. Yang and Y. Tian, "Super normal vector for activity recognition using depth sequences," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on.* IEEE, 2014, pp. 804–811.

[106] Y. Kong and Y. Fu, "Bilinear heterogeneous information machine for rgb-d action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1054–1062.