Spring 4-19-2012

# Vision-based Human Action Recognition: A Sparse Representation Perspective

Zhe Zhang
*University of Nebraska-Lincoln*, zhe.zhang@huskers.unl.edu

# VISION-BASED HUMAN ACTION RECOGNITION: A SPARSE REPRESENTATION PERSPECTIVE

by

Zhe Zhang

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Master of Science

Major: Electrical Engineering

Under the Supervision of Professor Sina Balkir, Senem Velipasalar

Lincoln, Nebraska

May, 2012

VISION-BASED HUMAN ACTION RECOGNITION:
A SPARSE REPRESENTATION PERSPECTIVE

Zhe Zhang, M. S.

University of Nebraska, 2012

Adviser: Sina Balkir, Senem Velipasalar

The objective of vision-based human action recognition is to label the video sequence with its corresponding action category. In this thesis, the human action recognition problem is solved from a novel sparse representation perspective. First, spatial-temporal interest points are extracted in the video sequences. Then, a cuboid is extracted centered at each spatial-temporal interest point. The histogram of oriented gradients (HOG) and histogram of flow (HOF) descriptors for each cuboid are computed and concatenated into a one-dimensional vector. The K-Means clustering algorithm is used to cluster these cuboid feature vectors into a few visual codewords. Finally, each action instance is represented as a histogram of the visual codewords.

We apply sparse representation based classification in the human action recognition problem. Each action instance in the test set is represented approximately as a linear weighted sum of all the action instances in the training set. The $\ell_1$-minimization technique is utilized to derive the sparse result. The residual between the test instance and its corresponding representation using the action instances in each class is calculated. The test action instance falls into the action class with the smallest residual. Our proposed human action recognition system is evaluated on the KTH human action dataset. The experimental results obtained using our method are compared with the results derived using conventional machine learning techniques such as K-Nearest Neighbors (KNN) and Support Vector Machines (SVM) and show that

the proposed framework yields considerable performance improvement in many aspects.

*Keywords:* human action recognition, spatial-temporal interest point, histogram of oriented gradients, histogram of flow, compressed sensing, sparse representation, k-nearest neighbors, k-means clustering, support vector machines, $\ell_1$-minimization, $\ell_1$-regularized least squares.

COPYRIGHT

# ACKNOWLEDGMENTS

First, I would like to thank my adviser, Professor Senem Velipasalar for her encouragement and guidance throughout my graduate program. I also thank her for always being patient and cheering me up no matter how stuck I got or even failed after months of efforts. Although the professors are very busy with their teaching and research work, my adviser can always find time for our meetings in which many valuable discussions are produced. I highly appreciate the environment she fosters in the whole lab, which I believe is very healthy and helpful for creative research. I would like to thank Professor Sina Balkir for serving as my local advisor for the last year of my graduate study, I really appreciate his kind support and help.

I also would like to thank Professor Michael W. Hoffman and Professor Eric Psota for serving on my Master thesis committee and providing many valuable comments and suggestions. I learned a lot from Professor Hoffman's classes and discussion with him. I would like to give my special thanks to Professor Mustafa Cenk Gursoy for his supervision in my research work during my first two years of my graduate study. He helped me a lot on leading me into the world of research and teaching me how to conduct research from scratch. I would like to thank Professor Mehmet Can Vuran in the Department of Computer Science and Engineering for his guidance in my research work about wireless camera networks. I also would like to thank Professor Lance C. Pérez for his kind support and help in my last year as a Master student in Department of Electrical Engineering. It is a great honor for me to have a glimpse of their wisdom.

Throughout my graduate program, I have been surrounded by an amazing group of colleagues at UNL. My countless conversations with them have been invaluable to my research and to learning about other research fields. I especially thank my fellow students from both the the Smart Vision Systems Laboratory (Li He, Youlu

Wang, Mauricio Casares and Alvaro Pinto) as well as the Wireless Communications and Networking Laboratory (Qing Chen, Junwei Zhang, Deli Qiao, Bo Liang and Sami Akin). I also want to deeply thank many of my dear friends here at Lincoln who support me to get through the hard time. They are: Tongqing Liu, Jinya Pu, Lishan Huang, Tiantian Xu, Yuji Mo and Liyuan Zhang. The discussions, chats and laughters with them made my life in Nebraska much more fun and the feeling of home.

Lastly, I would like to thank my parents Wanyou Zhang and Fengrui Zhang for their consistent support to my almost three years' study and life in the United States. They always have their confidence and belief in me, which helps me get through my hardest time. I can always rely on their advices and support. Without them, I could not reach this far.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Human action recognition is the process of labeling video sequences containing human action with corresponding action classes. More specifically, vision-based human action recognition is discussed and studied in this thesis. Vision-based human action recognition is the process of recognizing the human actions in video sequences by utilizing computer vision techniques.

## 1.1 Motivation

Due to the increase of digital video cameras used in everyday life, more and more video content is generated and uploaded to the Internet or stored in large video dataset. Categorizing rich video content based on the actions appearing in the video is a good way to reach the initial goal of organizing these videos. Also, human action recognition is a popular research area due to its potential application in visual surveillance, content-based video retrieval, human-computer interaction and sports annotation, [1][2][3][4][5]. For example, with successful human action recognition, the visual surveillance system in large public area can automatically extract high-level semantic information from

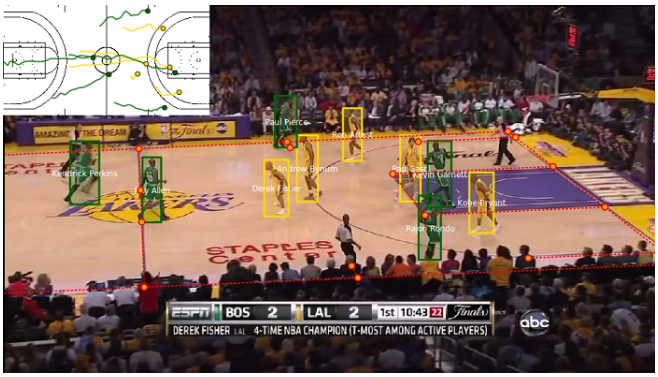
the surveillance video thus making it possible to make alarms to the public when predefined dangerous behaviors occur in the range of surveillance; content-based video retrieval system can search and locate the video content with specific definition fast and precisely; human-computer interaction systems can smoothly interact with human body movement and provide more sophisticated human-computer interface. For instance, Kinect®, a gaming console controller from Microsoft that usually tracks human body movement, is a commercial product that utilizes the power of human action recognition; sports annotation system can perform complicated player motion analysis and extract play strategy information from live video of sport games in real-time. Figure 1.1 shows the different applications introduced above. Figure 1.1a demonstrates an example of commercial surveillance system; Figure 1.1b shows the human-motion based gaming console controller called Kinect® from Microsoft; Figure 1.1c demonstrates a sports annotation system used to intelligently annotate sports games.

## 1.2 Challenges

Although researchers are greatly motivated by the promise of its wide application and have done a lot of research on this topic, vision-based human action recognition is still a very challenging research area due to a few reasons. First, there are great variations of performance for either intra or inter-class action instances. Two action instances in the same action class can demonstrate non-negligible dissimilarity because of the different appearance or movement characteristics between the performers. Different individuals may also perform the same action in different ways. The differences in the speed and stride length of articulation movement can also contribute to variations among intra or inter-class action instances. An ideal human action recognition algorithm should be

(a) Commercial Surveillance System


(b) Microsoft Kinect®: A human computer interface




(c) Sports Annotation System

Figure 1.1: Different Application Scenario for Human Action Recognition (Courtesy image from Google Image Search )

able to adapt the variations within one class and discriminate the variations among different action classes. Second, the video sequences containing the human actions can be captured in different environments with different recording settings. It would be more difficult to correctly recognize the human action in a cluttered or dynamic background. The human body may be occluded in the video sequence which would raise the challenge of correctly recognizing the human action. Change in lighting conditions is also a common source of variation in the recording settings. When using a moving camera, the challenges become even harder since the same action, observed from different viewpoints, can lead to very different image observations. Third, video sequences may be recorded at different rates, which generates great variations in the rate of performance of an action. Stable human action recognition algorithms should eliminate the effect of difference in human action performance rate.

## 1.3 Related Work

In this section, previous research conducted by researchers on vision-based human action recognition is examined. The precedent work can be mainly divided into several general approaches summarized in the following sections.

**Recognition based on cross-correlation** One direction of research is based on calculating the cross-correlation between volumes of video data from different action instances. Higher correlation indicates higher similarity between the two test video sequences. Shechtman and Irani (2005)[6] proposes a method to discriminate whether two different space-time intensity patterns of video sequences are from the same underlying human action category based upon extending the 2-D image correlation to 3-D space-time volume correlation. The advantage of this method is it can detect

complex human action activity with multiple actions included. The disadvantage is that it requires the cross-correlation calculation for one test video sequence against the whole library of video action templates, which results in high computational complexity.

**Recognition based on tracking and trajectory** There is also one approach based on tracking human body parts and utilizing the derived motion trajectories to perform action or behavior recognition. In Yilmaz and Shah[7], landmark points of the human body are first detected in every frame of the video sequences from multiple moving cameras with different viewing angles. The generated corresponding 4D $(x, y, z, t)$ trajectories of these landmark points are used to discriminate different human actions. This approach can be used in action recognition and video retrieval, however, the cost of considerable human annotation is significant. In the work of Fanti et al.[8], a human motion model is represented as a triangulated graph and the model is learned in an unsupervised manner from unlabeled data. Their approach combines multiple cues such as positions, velocities and appearance which are derived by tracking the feature points in a frame-by-frame manner. However, the quality of the point tracking has a great effect on the recognition performance. It highly depends on the photometric conditions, thus making it less robust and applicable.

**Recognition based on global image representation** Another very popular approach to tackle the human action recognition problem is based on global image representations, which encode the ROI (region of interest) of a moving object as a whole. The ROI can be derived by traditional background subtraction and tracking. Bobick and Davis [9] use temporal template to represent human movement. In detail, two different versions of the motion template: motion energy image (MEI) and


Figure 1.2: 3-D space-time volume [10]

motion history image (MHI) are proposed. The calculated MEI and MHI are used to recognize the human action by matching them against the available known human action templates stored in a local action template library. Since this method require robust background subtraction, moving camera and clutter background can reduce its performance and recognition accuracy. Blank et al. [10] treat human actions as three-dimensional shapes generated by accumulating the detected foreground human figures frame by frame in the video sequence. A few different space-time features such as local space-time saliency, action dynamics, shape structure, and orientation are extracted and utilized to perform human action recognition. Similarly, this approach requires robust static background subtraction. Figure 1.2 demonstrates the 3-D space-time volume.

**Recognition based on interest feature points** Alternatively, researchers developed an approach based on spatial-temporal interest points to represent action instance. Laptev et al.[11] extends conventional 2-D Harris corner detector to 3-D scenario with additional consideration of *time* dimension. The proposed algorithm can detect interest points in space-time volumes where the image values vary significantly

in all of the three dimensions. However, their method can only generate a relatively small number of stable interest points which are usually not sufficient to discriminate different complex action sequences. To solve this deficiency, Dollár et al.[12] propose a spatial-temporal interest point detector based on a group of separable linear filters to generate a large number of interest points. The interest points are the local maxima of a response function which is derived by applying the linear filters on the video sequences. These interest points correspond to the local regions where complex motion patterns can be detected.

## 1.4 Overview of Compressed Sensing and Sparse Representation

In recent years, compressed sensing [13] has become a popular research area in the community of signal processing, and sparse representations for classification problems has also drawn some attentions from researchers in the field of computer vision and pattern recognition. Compressed sensing is usually used in acquiring and reconstructing signals known to be sparse or compressible in some specific basis such as Fourier and wavelet. The advantage of compressed sensing is that a signal acquired with a sampling rate under the Nyquist rate can still be reconstructed correctly using techniques in convex optimization. It is possible to find sparse solutions to underdetermined linear systems by $\ell_1$ minimization and its extensions and variations such as $\ell_1$ regularized least squares.

Sparse representation is usually denoted as representing a signal or feature vector as a linear combination of few columns which are called atoms in an over-complete dictionary. The resulting coefficient vector is sparse in terms of the huge collections in

the dictionary, meaning that only a small portion of coefficients in the vector are non-zero. Compared with the original feature vector, this sparse, compact representation is not only more informative and valuable in related signal acquisition and compression problems, but also shows great potential in signal classification and related computer vision tasks.

In this thesis, a sparse representation technique is utilized in a classification framework to recognize the human actions in the video sequences. The theory and applications of compressed sensing and sparse representation are demonstrated and discussed in detail in Chapter 2.

## 1.5 Organization of Thesis

The remaining parts of the thesis are organized as follows: Chapter 2 describes the theory and relevant applications of compressed sensing and sparse representations in the filed of computer vision and pattern recognition. Chapter 3 describes the proposed approach used to solve the problem in this thesis. In particular, spatial-temporal interest point detection, bag of words representation of action instance, and classification framework based on sparse representation are explained in detail. Chapter 4 gives the implementation details of our proposed approach and demonstrates the derived experimental results using our approach. The results are analyzed and compared with the ones derived from other machine learning algorithms. At last, the conclusion is given in Chapter 5.

# Chapter 2

# Compressed Sensing and Sparse Representation for Computer Vision and Pattern Recognition

## 2.1 Background

Traditional signal acquisition procedure follows the classical Nyquist Sampling Theorem, which indicates that a signal can be captured accurately without aliasing if the sampling rate is at least twice the highest frequency of the signal. Nyquist sampling rate is a sufficient, but not necessarily required, condition. Recent research in "Compressed Sensing" or "Compressed Sampling" indicates that it is also possible to recover signals, images, or other data from highly sub-Nyquist-rate sampling.

Consider the development of consumer digital camera industry as an example, the effective number of pixels in the optical image sensors used to sense photons and capture images are increasing all the time since people want to capture images with a higher resolution. However, the next step after sensing and generating the raw image

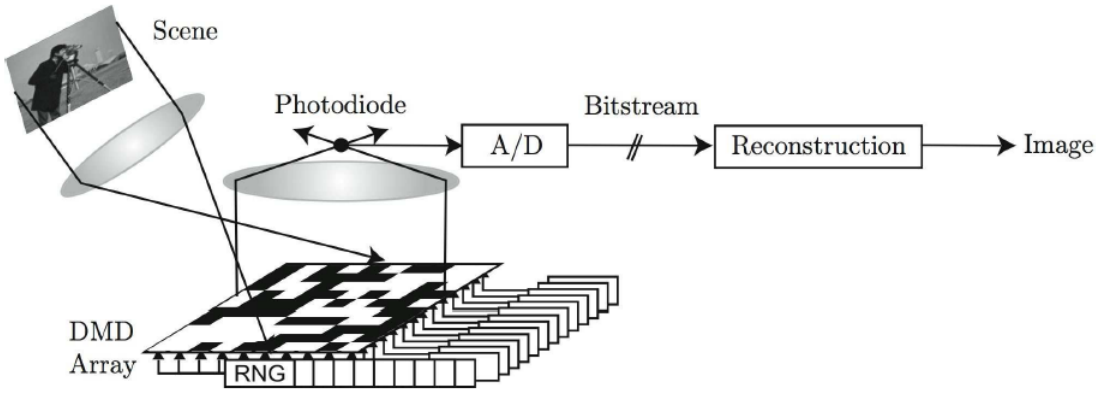



Figure 2.1: Single Pixel Camera Architecture [15]

data is often to compress and discard a great part of the information in the raw image to satisfy the requirements on storage and transmission. This is a considerable waste on the utilization of the image sensor since most of the information acquired at the sensing stage is discarded at the compression stage. It would be better if we could sense the image directly in a compressive way at the sensing stage and then reconstruct the image we want using the compressed measurements. Compressed sensing is aimed to solve the problem and provide a new scheme for signal acquisition and compression at the same time. Therefore, it can cause great innovation in hardware sensor design by following this strategy and changing the way we use sensors to acquire signals from a totally different perspective. For instance, Marco et al. [14] designed a single-pixel camera by applying this compressed sensing technology in the camera hardware design. For light whose wavelength is not in the range of visible spectrum, the traditional CMOS sensors do not work anymore. Specific sensors aimed for special light are much more expensive than the sensors used in consumer digital cameras. Therefore, cost can be greatly reduced by using the single-pixel camera. Figure 2.1 demonstrates the architecture of single-pixel camera.

## 2.2 Compressed Sensing Theory

**Sparse Representation** In this subsection, we use image signals as an example to explain what sparse representation means in compressed sensing theory. For an image $\mathbf{X}$ with resolution $N_1 \times N_2$, we concatenate the column data of the image into a single one-dimensional vector $\mathbf{x}$ with size $N = N_1 \times N_2$. Therefore, $x(n)$ denotes the $n$th element in the vector $\mathbf{x}$, for which $n = 1, 2, ...N$. We express $\mathbf{x}$ in the basis $\mathbf{\Psi} = [\Psi_1, \Psi_2, ...\Psi_N]$ with a $K$-sparse representation:

$$\mathbf{x} = \sum_{n=1}^{N} \Theta(n)\Psi_n = \sum_{l=1}^{K} \Theta(n_l)\Psi_{n_l} \tag{2.1}$$

where $\Theta(n)$ is the coefficient of the $n$th basis vector $\Psi_n$ and the coefficients indexed by $n_l$ are the $K$-nonzero entries of the basis decomposition. Equation 2.1 can be further simplified as:

$$\mathbf{x} = \mathbf{\Psi}\Theta \tag{2.2}$$

where $\Theta$ is an $N \times 1$ column vector with $K$ nonzero elements space.Here $\Theta$ is $K$-sparse, which means that the $\ell_0$ norm of $\Theta$:$\| \Theta \|_0 = K$ which simply counts the non-zero entries of $\Theta$.

**Incoherent Projections** In the compressed sensing framework, the $K$ nonzero entries in $\Theta$ usually cannot be derived directly. On the other hand, a measurement matrix $\mathbf{\Phi}$ is adopted to project the $N$ elements in the image vector $\mathbf{x}$ into $M$ measurements with $M < N$. The relationship can be expressed as:

$$\mathbf{y} = \mathbf{\Phi x} = \mathbf{\Phi\Psi}\Theta \tag{2.3}$$

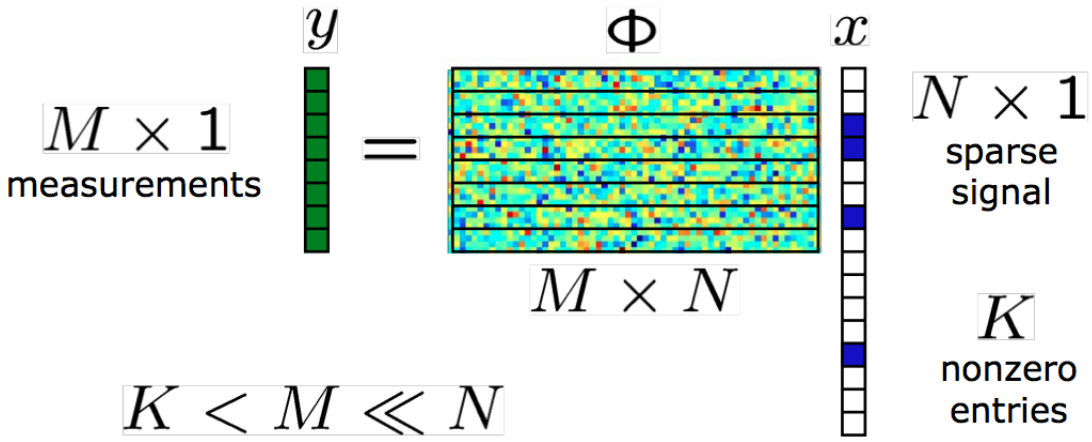



Figure 2.2: Explanation of Compressed Sensing

in which the measurement matrix $\mathbf{\Phi} = [\Phi_1', \Phi_2', , , \Phi_M']'$ with size $M \times N$, the vector $\mathbf{y}$ $(M \times 1)$ contains the compressive sampling measurements. Although the recovering of the image data $\mathbf{x}$ from compressive sampling measurements $\mathbf{y}$ is under-determined since $M < N$, the extra sparsity of the vector $\Theta$ makes it possible to achieve this goal.

There are also some additional requirements added on the compressed sensing theory to make it possible to recover the set of nonzero entries of $\Theta$ from $\mathbf{y}$: the first condition is called incoherence between two bases which means that the sparsity basis $\mathbf{\Psi}$ cannot sparsely represent the rows of the measurement matrix $\mathbf{\Phi}$, the second condition is that the number of measurements $M$ should be larger than $\mathcal{O}(Klog(\frac{N}{K}))$. For the first condition, incoherence holds for many pairs of bases, e.g. delta spikes and the sine waves of the Fourier basis. Also, incoherence surprisingly holds between a randomly generated basis (e.g., i.i.d. Gaussian or Bernoulli/Rademacher $\pm 1$ vectors) and an arbitrary basis. Figure 2.2 shows the procedure of how the compressed sampling measurements are generated.

**Signal Reconstruction via $\ell_1$ Minimization** The most common recovery method for compressed sensing is based on the following $\ell_1$ optimization problem:

$$\hat{\Theta} = arg\ min \parallel \Theta \parallel_1 \quad s.t.\ \mathbf{y} = \mathbf{\Phi\Psi}\Theta \tag{2.4}$$

This convex optimization problem is called *Basis Pursuit*[16] which can be efficiently solved using polynomial time algorithms. There are also other available recovery algorithms such as Orthogonal Matching Pursuit [17](OMP), Lasso and Basis Pursuit with quadratic constraint[18], etc.

## 2.3 $\ell_1$-Regularized Least Squares

In fact, many $\ell_1$ regularization problem can be cast as $\ell_1$-regularized least squares problem. The mathematical expression for $\ell_1$-regularized least squares problem is:

$$minimize \parallel \mathbf{Ax} - \mathbf{y} \parallel_2^2 + \lambda \parallel \mathbf{x} \parallel_1 \tag{2.5}$$

In sparse representation scenario, $\mathbf{A} \in \mathbb{R}^{m \times n}$, which is the over-complete dictionary; $\mathbf{x} \in \mathbb{R}^n$, which is the sparse coefficient vector that needs to be calculated; $\mathbf{y} \in \mathbb{R}^m$, which is the feature vector that can be represented as a linear weighted sum of the column data in the over-complete dictionary $\mathbf{A}$. In this thesis, the sparse representation problem solves the $\ell_1$-regularized least squares. Our implementation solves the $\ell_1$-regularized least squares through an interior-point method, the details of which can be found in [19]. The search direction is calculated by utilizing preconditioned conjugate gradients (PCG) algorithm. The overall computational complexity of the $\ell_1$-regularized least squares is determined by the product of number of PCG steps in all the iterations and the run time for one PCG step.

## 2.4 Applications

A lot of research has been done using the compressed sensing and sparse representation techniques in the field of computer vision and pattern recognition. Some of the most significant research is summarized in the following.

In [20], Cossalter et al. applied compressive sensing theory on background subtraction, which is similar to the idea in [21]. The background subtracted image, in another words, the foreground image is not directly transmitted to the decoder. The random projection of the foreground image is calculated and transmitted. The reconstruction of the foreground image is implemented at the decoder. Then, based on the observed foreground images, the particle filter framework can predict the position of the bounding box for the next frame and continuously track the object. The compressive sensing theory is applied here to enable the privacy of the video sequence. The random projection is derived from a matrix whose entries are sampled from a Gaussian distribution. The matrix, or the seed used to generate matrix, is known at the decoder. Therefore, the decoder can reconstruct the foreground image, while it is impossible to do so without knowing the seed.

In [22], Mei et al. tackle visual tracking by integrating sparse representation in a particle filter framework. The detected foreground moving object in a new frame is represented as a sparse representation of the available target templates and trivial templates. The introduced $\ell_1$-regularized least squares is utilized to solve the sparse representation problem. The new tracking target is identified by locating the template candidate with the smallest projection error. The visual tracking procedure is performed continuously in a Bayesian state inference framework in which particle filtering is adopted.

In [23], Wright et al. proposed robust face recognition via sparse representation.

In this application, there is a data set that contains multiple objects classes. In each object class of the data set, there are several face images in different illumination scenarios. When a test image is presented, the proposed algorithm tries to represent the test face image by the linear combination of the existing face images in the data set. Since the test image is only highly correlated with one or a few images in the data set, therefore, the matrix that represents the linear combination should be a sparse matrix. The relationship can be described in the following formula:

$$x = D\alpha_0 + e_0 \tag{2.6}$$

in which $x$ is the test image, $D$ is the matrix represents the entire training set, $\alpha_0$ is the sparse coefficient vector, $e_0$ is a vector of errors. The searching for the corresponding face object is achieved by solving relevant $\ell_1$-minimization problem.

In [24], Allen et al. proposed distributed recognition of human actions using wearable sensor networks. In their proposed system, each individual sensor node is capable of performing local classification based on the local observation gathered at the sensor node. The sensor will only transmit the data back to the base station when possible occurrence of specific motion are detected. On the base station, a global classifier receives the data from multiple sensor nodes, and further improves the classification accuracy based on the local decisions from the sensor nodes. The distributed recognition system aims to recognize certain body actions, such as sitting, running, going upstairs and downstairs. The proposed solution is based on a classification framework which utilizes the technique of distributed compressed sensing. In this framework, the distribution of multiple human motion classes is modeled as a combination of subspace model, one subspace for each class. Given $C$ classes and a test sample $y$, the objective is to find the sparse linear representation of $y$ in terms of

all the available training data.

In [25], Allen et al. utilize compressive sensing to compress the SIFT histogram which contains the features of the detected object in each camera's scene. The SIFT feature detector is used to extract the viewpoint-invariant features form the corresponding images. The SIFT histogram for each individual object in a single image is sparse compared to the large vocabulary that contains codewords from many object classes. This is the premise that compressive sensing theory can be applied in multiple-view object recognition. The authors also point out high-dimensional SIFT histograms share a joint sparse pattern corresponding to a set of common features in 3-D across the embedded camera network. Such joint sparse patterns can increase the sparsity of the features from all the cameras in the network. All the cameras capture the same object in the scene from different vantage points and the features of the object are extracted in these cameras distributively. The SIFT features are compressed via random projection in each camera and are sent back to the base station. Then the base station would perform object recognition from these features.

## 2.5 Similarity and Differences between Compressed Sensing and Sparse Representation

Although compressive sensing and sparse representation have lots of similarities in many aspects, they have different meanings in the field of computer vision and pattern recognition. The sparse representation is the premise of compressive sensing. Compressive sensing can be applied on certain signal only when the signal has some sparse representation on some basis. Usually, compressive sensing is applied where

there is the demand or preference on the compression of the data or the communication channel is band-limited. In this case, compressive sensing is used to decrease the dimensionality of the data and the privacy is enabled at the same time when random projection is performed. More importantly, compressed sensing is widely applied in new sensor hardware design since there is a sensing stage as the term indicates. Sparse representation is mainly used in recognition or classification. The basic idea is that usually the object that needs to be recognized or classified (test example) can be written in the form of linear combination of all the samples in the training set with a sparse vector. The similarity between the two is that $\ell_1$-minimization or more generally convex optimization is used in both of the two techniques. In compressed sensing, $\ell_1$-minimization is used to reconstruct the sparse signal from the projection measurements. In sparse representation, $\ell_1$-minimization is used to find the sparse linear combination vector. The classification is achieved based on the sparsity of the vector. If the test example is in one of the categories of the data set, the vector should only have a few non-zero values at the corresponding positions. Further classification framework can be developed based on this important property. The goal of this thesis project is to recognize and classify the human actions in video sequences, therefore, the concept and application of sparse representation rather than compressed sensing is exploited and studied extensively.

# Chapter 3

# Human Action Recognition based on Sparse Representation

Our approach adopts the spatial-temporal interest points approach to generate the dimension-reduced feature vectors that are the input to machine learning algorithms. The whole human action recognition system is composed of the following three components: spatial-temporal interest points detection, bag-of-words representation for action instance, and classification framework based on sparse representation. Figure 3.1 demonstrates the system flow diagram. In this chapter, the relevant theories of adopted techniques and algorithms are first explained and then the components in the system model are discussed in the following subsections extensively. The intermediate experimental results in each step are demonstrated to better elaborate the system model.

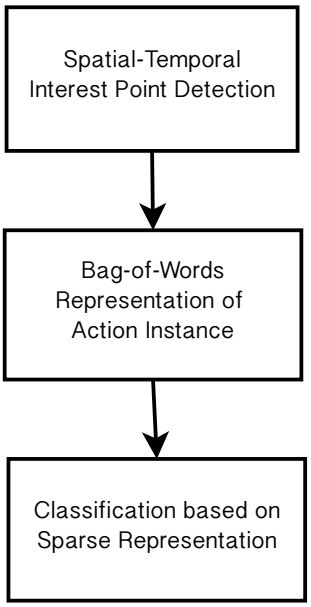



Figure 3.1: Diagram Flow of Human Action Recognition System

## 3.1 Relevant Theory

### 3.1.1 Gabor Filter

The Gabor filter was first introduced by Dennis Gabor[26]. The one-dimensional Gabor filter is defined as the multiplication of a sinusoidal wave with a Gaussian window:

$$g_e(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} cos(2\pi\omega_0 x) \tag{3.1}$$

$$g_o(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} sin(2\pi\omega_0 x) \tag{3.2}$$

in which $\omega_0$ determines the central frequency in which the filter can derive the greatest response and $\sigma$ determines the spread of the Gaussian window. The power spectrum

of the Gabor filter is given by the sum of the two Gaussians center at $\pm\omega_0$:

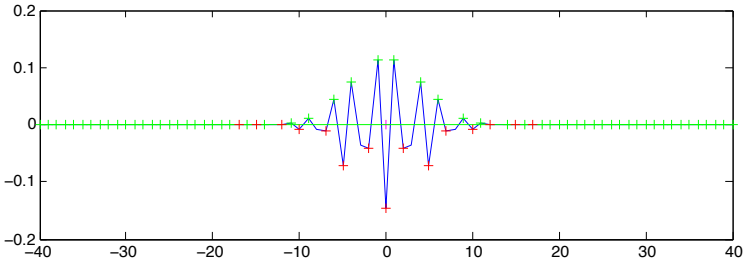
$$\| G(\omega) \| = e^{-2\pi^2\sigma^2(\omega-\omega_0)^2} + e^{-2\pi^2\sigma^2(\omega+\omega_0)^2} \tag{3.3}$$

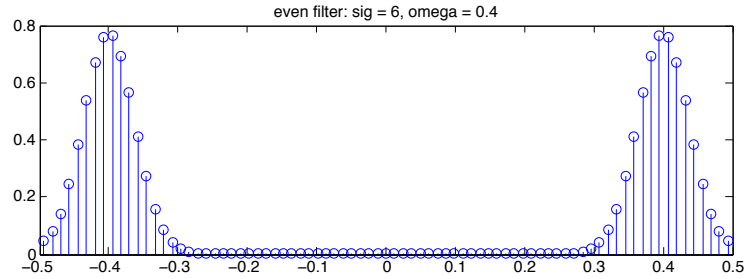
which can be easily explained. The power spectrum of a sine function is two spikes at frequency $\pm\omega_0$ and the power spectrum of a Gaussian is still a Gaussian. Then multiplication in spatial or temporal domain yield convolution in frequency domain. Therefore, there are two Gaussian distribution located at frequency $\pm\omega_0$. The 1-D quadrature of Gabor filter pair is shown in Figure 3.2. Gabor filter has also been extended to 2-D and 3-D version which has widely applied in the field of image processing for edge detection, texture representation and optical flow computation, etc.

### 3.1.2 Image Descriptors

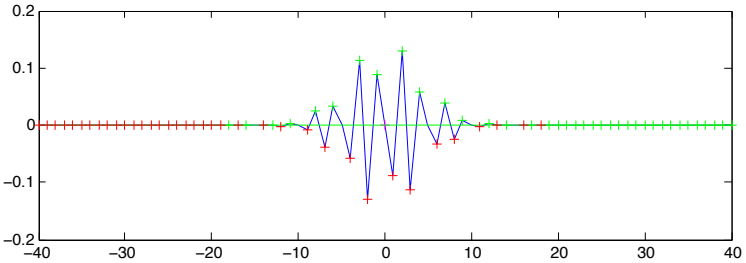
**Histogram of Oriented Gradients** Histogram of Oriented Gradients (HOG) is one of the most popular image descriptors used in human detection in still images. The algorithm of HOG is proposed by Navneet et al. in [27]. There are four different kinds of HOG introduced in Navneet's work: rectangular HOG (R-HOG), circular HOG (C-HOG), bar HOG and centre-surround HOG. The default HOG descriptor adopted in this thesis project is R-HOG. The image gradients are calculated over the $x$ and $y$ direction with simple filters $[-1, 0, 1]$ and $[1, 0, -1]'$ respectively. The gradient orientation at each pixel position is then calculated according to the image gradients at $x$ and $y$ direction.

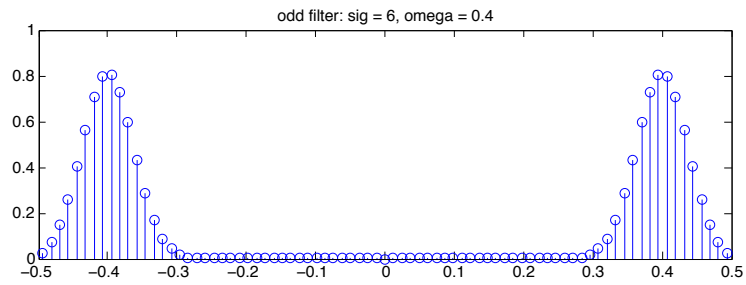
In R-HOG, HOG descriptors are computed over each block, which contains several dense uniformly sampled grids and are usually overlapped with its neighbors. The HOG for each block is normalized independently. Take square R-HOG as an example,






(a) even filter






(b) odd filter

Figure 3.2: One dimensional Gabor Filter and corresponding frequency response

one block contains $\varsigma \times \varsigma$ grids of $\eta \times \eta$ pixel cell, each of which includes $\beta$ orientation bins. These orientation histograms at each cell are concatenated into a one-dimensional feature vector for the interested detection region. The parameters $\varsigma$, $\eta$ and $\beta$ determine the length of the finally generated feature vector.

**Histogram of Flow** The calculation of Histogram of Flow (HOF) is similar with the calculation of HOG. However, HOF operates on differentials of optical flow-either flow orientation or oriented spatial gradients of flow components instead of original image gradients. Compared with HOG, HOF focuses on extracting the motion characteristics of consecutive image pairs in video sequence. HOF is also proposed by Navneet et al. for solving human detection problem[28].

First, optical flow $I^w$ is calculated top-down in a multi-scale approach. The initialization of flow is estimated at a coarse scale, and then the initial flow is propagated from top to down in a pyramid structure to refine the finalized optical flow at the finest scale. The calculation of optical flow is based on the constant brightness equation: $\frac{\partial I}{\partial t} + w\nabla I = 0$. $I$ is the image, $w$ is the infinitesimal motion. After the optical flow $I^w$ is extracted, differentials of optical flow are calculated using a few different mechanisms. The spatial and orientation histogram are calculated on each block of the optical flow differential image in the same way as HOG. The final feature vector is the concatenation of the HOF over all of the blocks in the image window. Readers can refer to Chapter 6 of [29] for further details of the algorithms.

### 3.1.3 Classification Algorithms

**K-Means Clustering** K-Means clustering algorithm is a classical unsupervised machine learning algorithm. Given a dataset contains multiple data instances with the same dimensionality, the K-Means algorithm is aimed to classify these data instances

into a certain number of clusters. As the name K-Means clustering indicates, the output of this algorithm would be $K$ clusters. There is a initialization step before this algorithm runs into loops. First, $K$ centroids are defined, each centroid is for each cluster. Then all the data points in the dataset are associated with the nearest centroids which we just defined in the initialization step. After all the data points are processed, the new $K$ centroids' positions are computed based on the association of clusters for all the data points in the dataset which is determined in the previous step. A new binding has to be done between the same dataset points and the newly computed centroid. Next, the described procedure is repeated and runs in a loop. Although the $K$ centroids' locations change step by step, the loop terminates when there is no change on the positions of the $K$ clusters. This algorithm aims to minimize the following objective function:

$$E = \sum_{j=1}^{K} \sum_{i=1}^{n} \parallel x_i^{(j)} - c_j \parallel^2 \tag{3.4}$$

in which $\parallel x_i^{(j)} - c_j \parallel^2$ is distance measure between a data point $x_i^{(j)}$ and the cluster center $c_j$. This indicates the distance of the $n$ data points from their corresponding cluster centers. The initial selected cluster centers have great effect on the performance of this algorithm. A good initialization should make all these initial cluster centers as far as possible from each other. Usually random selection is common method to generate the initial cluster centers. Running this algorithm several times and using the averaged result is also another strategy to make the result more promising.

**K-Nearest Neighbors** The k-nearest neighbor (KNN) is a machine learning algorithm for classifying objects based on the most similar training samples in the feature space. KNN is a type of lazy learning, in which no training model is built before

recognition. Thus, it is one of the simplest machine learning algorithms. Given a test sample, the distance between the test sample and all the training samples in the training set are calculated based on some distance metric. Euclidean distance and Hamming distance are some common choices adopted. The distances are utilized to represent the similarity between training sample and test sample. The smaller the distance is , the more similarity is shared between the two samples. KNN algorithm selects the k nearest neighbors with the k smallest distances. The class which wins the majority voting on the k nearest neighbors is assigned to the label of the training sample. When $k$ is 1, the KNN algorithm seeks the nearest neighbor and the label of the training sample is recognized as its nearest neighbor's class. The advantage of KNN algorithm is that it is simple and easy to implement. However, when the training set is large, the algorithm requires large memory and the prediction accuracy can quickly degrade when the number of attributes grows.

**Support Vector Machine** Support Vector Machine (SVM) is among the best "off-the-shelf" supervised learning algorithms. SVM is originally a binary classifier. Given a set of feature vectors with labels as $\{-1, +1\}$, SVM aims to discover pattern structure from the training samples and predict the labels of incoming test samples. Mathematically, the theory of SVM is to find a separating hyperplane with the maximal margin to separate the training dataset into two parts. Figure 3.3 demonstrates the example of maximum-margin hyperplane for two sets of data points. The samples on the margin are called the support vectors. SVM is actually solving the following optimization problem:

$$min_{w,b,\varepsilon} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \varepsilon_i \tag{3.5}$$

$$Subject\ to : y_i(w^T \phi(x_i) + b) \geq 1 - \varepsilon_i (\varepsilon_i > 0) \tag{3.6}$$

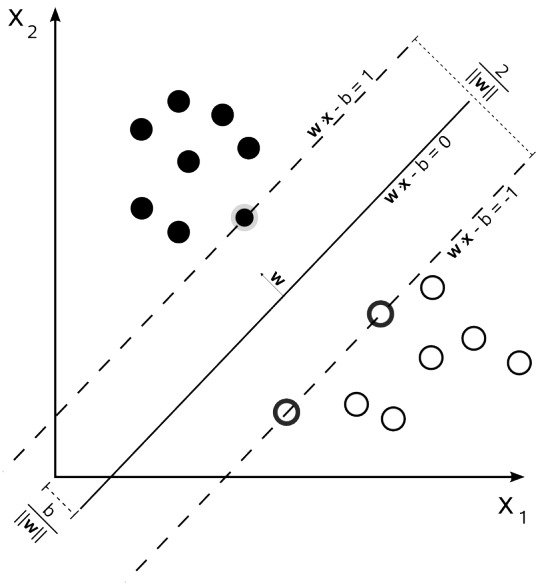



Figure 3.3: Example of hyperplane for categorizing point set

The mathematical details of SVM can be found in [30]. In this thesis project, we utilize a linear SVM model to classify the human actions. To recognize different human actions from multiple action categories, the one-against-all approach is utilized to perform multi-class classification.

## 3.2 System Model

### 3.2.1 Spatial-temporal Interest Points Detection

Features from spatial-temporal interest points have been shown to be effective in human action recognition since they can provide rich descriptors and powerful representations. The interest points for a video sequence are localized in three dimensions: $x$, $y$ for the spatial dimension and $t$ for the temporal dimension. To extract the spatial-temporal

interest points, the response function:

$$R = (I * g * h_{ev})^2 + (I * g * h_{od})^2 \tag{3.7}$$

is used to find strong response to periodic motion as well as the spatial-temporal corners. In the response function, $I$ is the stack of images from the video sequence, $g(x, y; \sigma)$ is the $2D$ Gaussian smoothing kernel, applied along the spatial dimensions, and $h_{ev}$ and $h_{od}$ are a quadrature pair of $1D$ Gabor filters applied along the temporal dimension.

$$h_{ev}(t; \tau, \omega) = -cos(2\pi t\omega)e^{-\frac{t^2}{\tau^2}} \tag{3.8}$$

$$h_{od}(t; \tau, \omega) = -sin(2\pi t\omega)e^{-\frac{t^2}{\tau^2}} \tag{3.9}$$

The two parameters $\sigma$ and $\tau$ correspond to the spatial scale and temporal scale of the detector respectively. They determine the scale at which the detector detects the interest points in all three dimensions. In the implementation, we set $\omega = 4/\tau$ thus making the number of undetermined parameters to be 2. The spatial-temporal interest points are extracted by finding the local maxima of the response function. In Dollár et al.[12], they indicated that any region with spatially distinguishing characteristics undergoing a complex motion can induce a strong response using the above spatial-temporal interest point detector. However, regions undergoing pure translational motion or without spatially distinguishing features will not induce a strong response.

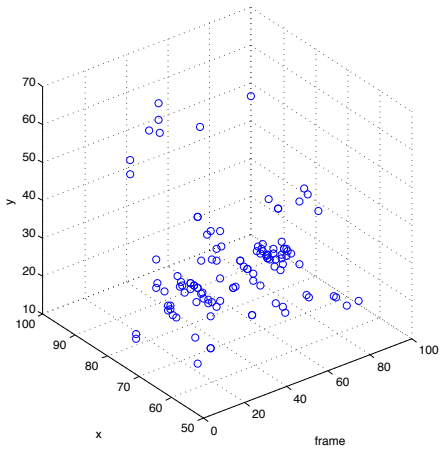
In our implementation, when the spatial scale $\sigma$ and temporal scale $\tau$ are set to be 2 and 3 respectively, the input of the spatial-temporal interest point detector is each sequence from the dataset, in which only one single action is included. After the non-maximal suppression is performed on the response function 3.7, the spatial-temporal interest points detector finds the corresponding spatial-temporal interest points in the

sequence. The number of interest points can be adjusted by changing the window size used in non-maximal suppression. The detected interest points are sufficient for representing the video sequence. There are six human action categories in the adopted dataset: boxing, handclapping, handwaving, jogging, running and walking. To illustrate the distribution of extracted spatial-temporal interest points in the video sequences, Figure 3.4 demonstrates the spatial-temporal interest points in 3-D with the three axis as $x$, $y$ and $frame$. $x$ and $y$ can be regarded as the two-dimensional image plane, $frame$ is the temporal dimension, in which the frames in video sequences are accumulated. There is one example figure for each human action category.

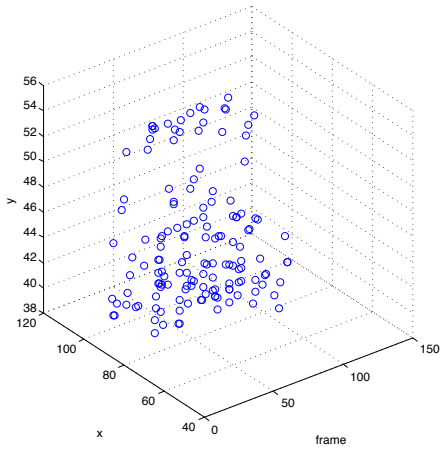
### 3.2.2 Bag of Words Representation of Action Instance

**Extract Space-Time Cuboid** After the spatial-temporal interest points are detected, small video patches can be extracted around these interest points to form the descriptors for each action instance. In this project, we extract the 3D space-time volume instead of $2D$ patches since cuboid contains spatial-temporally windowed pixels values and should provide more discriminative information. The size of cuboid is set to be about six times the scale at which they were detected. Each cuboid is a very high dimensional data vector due to the considerable amount of pixel values inside the window.

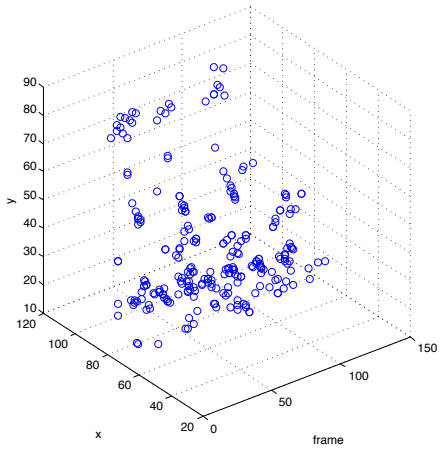
In our implementation, we generate the cuboid around the interest point detected by the spatial-temporal interest point detector. Therefore, each cuboid is a three dimensional array with size $13 \times 13 \times 19$. Figure 3.5 shows the region of generated cuboid in one frame of an *walking* sequence. Figure 3.6 shows a series of continuous frames inside the extracted cuboid corresponding to the spatial-temporal interest point in Figure 3.5. As we can see from Figure 3.6, this cuboid is extracted around



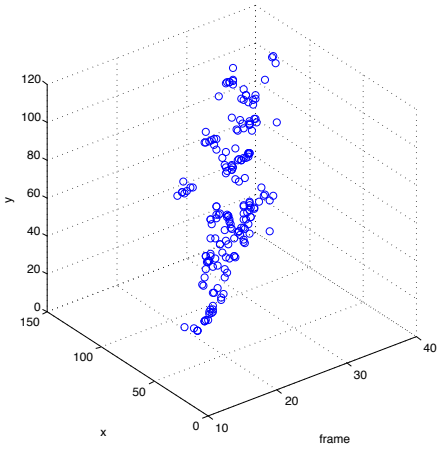

(a) boxing



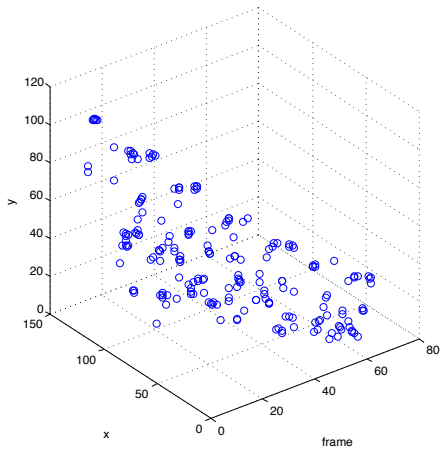

(b) handclapping



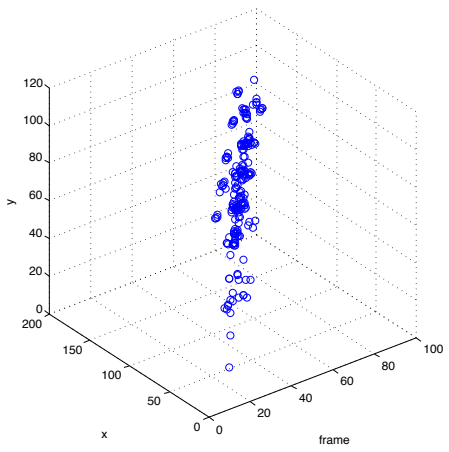

(c) handwaving




(d) jogging




(e) running




(f) walking

Figure 3.4: Extracted spatial-temporal interest points displayed in 3-D space

the articulation of the legs and contains the information of local movement of the legs in a few frames. The other cuboids are extracted around the arms, hands and shoulders etc. All of the extracted cuboids demonstrate the periodic movement of the human body, which is a good representation of the human action sequence. Figure 3.7 demonstrates a few other cuboid examples with different pixel intensity patterns in its original $3D$ form generated from other spatial-temporal interest points.

Figure 3.5: Extracted region of cuboid in one frame of *walking* video sequence

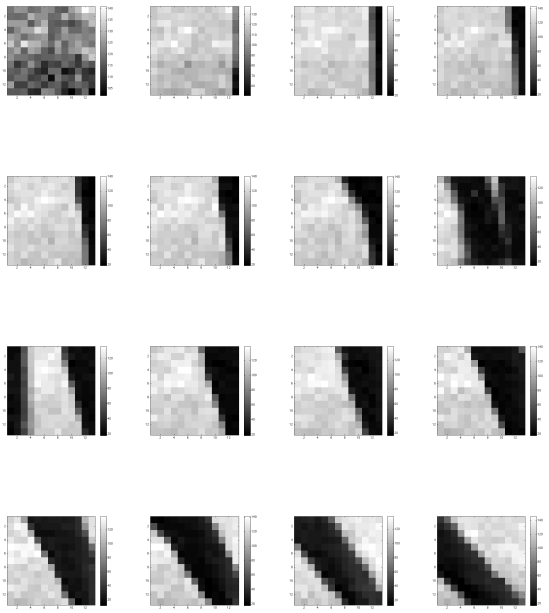


Figure 3.6: Continuous frames of an extracted cuboid

**Calculate HOGHOF Descriptor** After the cuboid is derived, each video sequence is represented as a set of space-time volumes within the cuboid extracted around those spatial-temporal interest points. However, the number of pixels in each cuboid is a few thousands, which is still relatively large. Thus, the original pixel vector in each cuboid is not appropriate to perform as the input feature vector to the classification algorithms. Instead, each cuboid is divided into $3 \times 3 \times 2$ blocks with overlapping. HOG descriptor with 4 bins and HOF descriptor with 5 bins are calculated for each block within a cuboid. The derived HOG and HOF descriptors are normalized and then concatenated into a one-dimensional feature vector with length 162 (HOG: 72, HOF: 90). The dimensionality of each cuboid is now greatly reduced from a few thousands to a few hundreds.

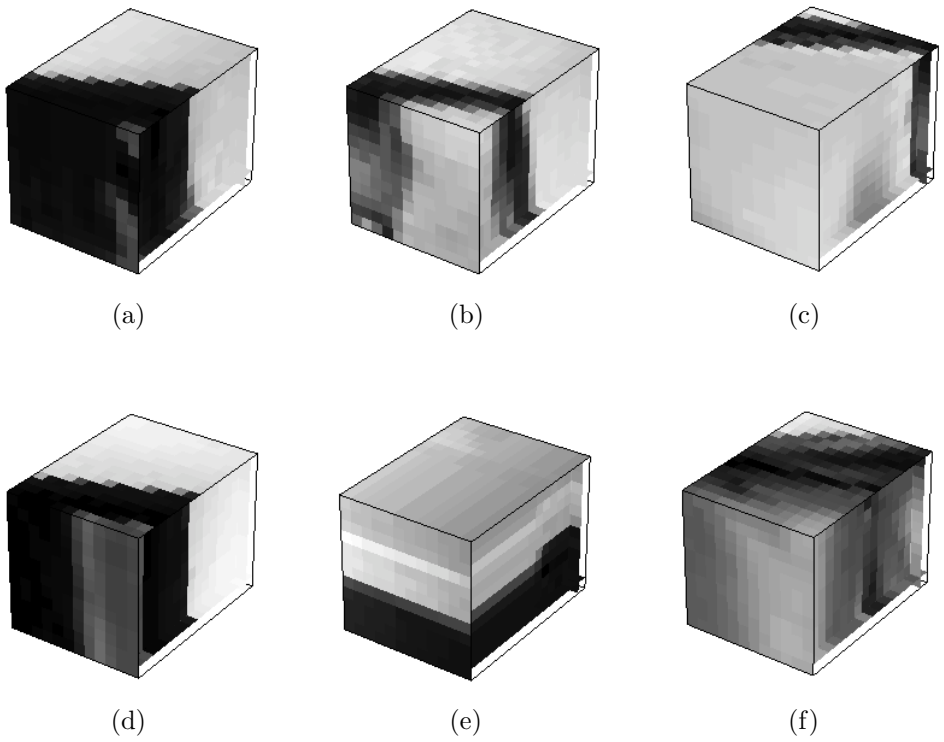



Figure 3.7: Extracted cuboids with different intensity patterns in 3D space

**Generate Histogram of Visual Words** For each action sequence, a number of cuboid can be generated after the cuboid extraction. Theoretically, the number of possible different cuboids is unlimited. However, the number of different types of cuboid should be relatively small since there are some similarities among the spatial-temporal interest points even when these interest points are in different human actions. Therefore, during the process of human action recognition, the exact form of the HOGHOF descriptor for each cuboid is not important any more, its type matters instead. The following notion of cuboid actually denotes its corresponding HOGHOF descriptor.

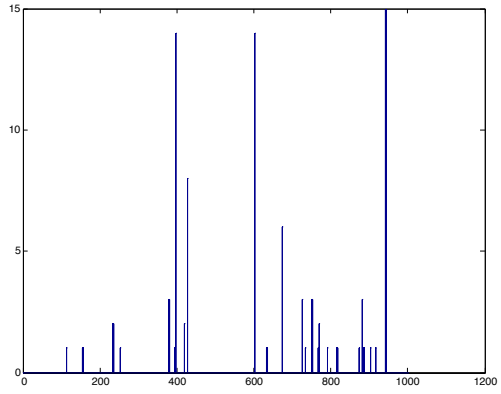
The K-Means clustering algorithm is applied here to cluster a large number of cuboid extracted from the training data into a few cuboid prototypes. These cuboid

prototypes are the visual words in the bag-of-words model. In the clustering phase, when the algorithm requires a random component, the results are averaged over a few runs. The cuboid extracted in the test dataset can also be assigned a type from the cuboid prototype library by finding its nearest neighbor. Now each action sequence can be represented as a histogram of the visual words in the codebook. The histogram of visual words can be used as the feature vector in the machine learning framework. Figure 3.8 shows example of visual words histograms for action instances from different action classes when the number of clusters is set to be 1000. These visual words histograms are the actual inputs to the different classification algorithms.

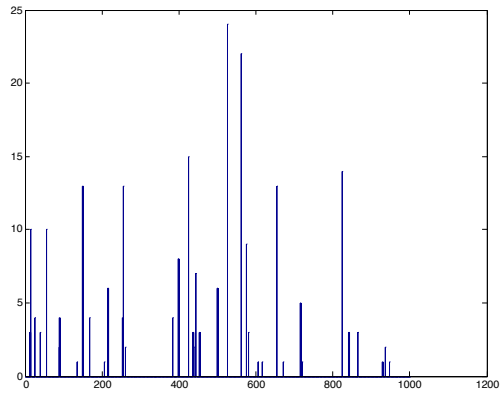
### 3.2.3 Classification Framework based on Sparse Representation

Since the feature vector that is used in the machine learning framework is extracted, the action recognition problem can be simplified as a pure machine learning task. The conventional machine learning algorithms used in the field of human action recognition typically include SVM and KNN. However, inspired by the recent research work of Wright et al.[23], which is about face recognition via sparse representation, we consider recognizing human actions via this sparse representation based classification (SRC) technique.

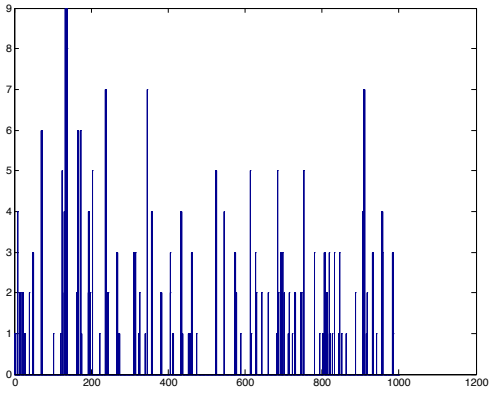
The goal of basic recognition problem is to determine which classes a new test sample belongs to when giving the labeled training sample from all distinct $k$ classes. In SRC, we formalize the given $n_i$ training samples for the $i$th class as the columns data of a matrix $A_i = [\mathbf{v}_{i,1}, \mathbf{v}_{i,2}, ..., \mathbf{v}_{i,n_i}] \in \mathbb{R}^{m \times n_i}$. In our human action recognition scenario, each column in this matrix is the visual words histogram with predefined number of bins which is actually the cluster number determined at the cuboid clustering step.



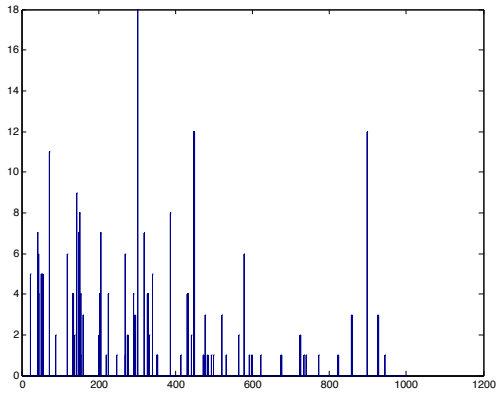

(a) Sample visual words histogram for *boxing*



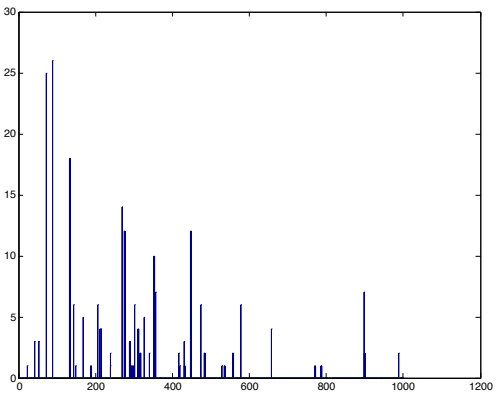

(b) Sample visual words histogram for *hand-clapping*



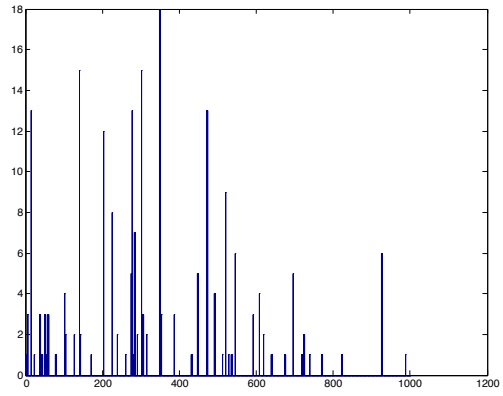

(c) Sample visual words histogram for *hand-waving*




(d) Sample visual words histogram for *jogging*




(e) Sample visual words histogram for *running*




(f) Sample visual words histogram for *walking*

Figure 3.8: Examples of visual words histogram for different human actions

We assume that the training samples from a single human action lie on a subspace since the subspace model is with enough flexibility to capture the variation in the real human action datasets. Therefore, we can try to represent the test sample as a sparse linear combination of training samples. For instance, when enough training samples from the $i$th class in a human action dataset are provided: $A_i = [\mathbf{v}_{i,1}, \mathbf{v}_{i,2}, ..., \mathbf{v}_{i,n_i}] \in \mathbb{R}^{m \times n_i}$, any new test sample $y \in \mathbb{R}^m$ from the same action class would be approximately represented as a linear weighted sum of the training samples:

$$\mathbf{y} = \alpha_{i,1}\mathbf{v}_{i,1} + \alpha_{i,2}\mathbf{v}_{i,2} + ... + \alpha_{i,n}\mathbf{v}_{i,n_i}, \tag{3.10}$$

for some coefficients, $\alpha_{i,j} \in \mathbb{R}, j = 1, 2, ..., n_i$. Since we cannot determine the actual action class of one test sample $\mathbf{y}$ at the beginning of recognition, the matrix $A$ has to be rewritten to incorporate the $n$ training samples from all the $k$ action classes:

$$A = [A_1, A_2, ..., A_k] = [\mathbf{v}_{1,1}, \mathbf{v}_{1,2}, ..., \mathbf{v}_{k,n_k}]. \tag{3.11}$$

Therefore, the linear representation of the test sample $\mathbf{y}$ can be rewritten in terms of all the training samples which perform as a over-complete dictionary:

$$\mathbf{y} = A\mathbf{x}_0 \in \mathbb{R}^m, \tag{3.12}$$

where $\mathbf{x}_0 = [0, ..., 0, \alpha_{i,1}, \alpha_{i,2}, \alpha_{i,n_i}, 0, ..., 0]^T \in \mathbb{R}^m$ have zeros entries except those coefficients associated with the $i$th class. Therefore, $\mathbf{x}$ can be derived by solving the linear equations $\mathbf{y} = A\mathbf{x}_0$. Obviously, if $m > n$, the system of equation $\mathbf{y} = A\mathbf{x}_0$ is overdetermined thus leading $\mathbf{x}_0$ to its unique correct solution. However, in our action recognition scenario, the dimension of the feature vectors is usually less than 1000, and the number of training samples are relatively large (more than 1500) due to the large

size of the human action dataset, making the linear equation system under-determined. Therefore, the solutions will not be unique. Conventionally, minimum $\ell_2$-norm solution is selected to resolve the non-uniqueness:

$$(\ell_2): \quad \hat{x}_2 = arg\,min \parallel \mathbf{x} \parallel_2 \quad subject\ \ to\ \ A\mathbf{x} = \mathbf{y} \tag{3.13}$$

Although this optimization can be easily solved, the solution $\hat{x}_2$ is not discriminative enough to recognize the test sample $\mathbf{y}$ since the non-zero entries of the derived coefficients vector are dense. The ideal coefficients vector should be sparse since the test sample only has high correlation with a few training samples in the same class. Therefore, only a small amount of coefficients are large, the rest of the coefficients are zero or approximately are zero. This motivates us to find the sparsest solution which means the coefficients vector with smallest number of non-zero entries is the optimal solution. The following optimization problem:

$$(\ell_0): \quad \hat{x}_0 = arg\,min \parallel \mathbf{x} \parallel_0 \quad subject\ \ to\ \ A\mathbf{x} = \mathbf{y} \tag{3.14}$$

is the mathematical expression for finding the minimum $\ell_0$-norm solution. However, this problem of finding the sparsest solution of under-determined system of linear equation is NP-hard and even very difficult to approximate. Alternatively, we solve the following $\ell_1$-norm minimization problem:

$$(\ell_1): \quad \hat{x}_1 = arg\,min \parallel \mathbf{x} \parallel_1 \quad subject\ \ to\ \ A\mathbf{x} = \mathbf{y} \tag{3.15}$$

This problem can be solved via standard linear programming methods in polynomial time. In practice, noise cannot be avoided in the real data. The equation 3.12 does

not hold exactly. Equation 3.12 can be modified as:

$$\mathbf{y} = A\mathbf{x}_0 + \mathbf{z} \tag{3.16}$$

to incorporate the effect of dense noise. $\mathbf{z} \in \mathbb{R}^m$ is a noise term with bounded energy $\| \mathbf{z} \| < \epsilon$. The sparse solution can be approximately derived by solving the following *stable* $\ell_1$-minimization problem:

$$(\ell_s^1): \quad \hat{x}_1 = arg\,min \| \mathbf{x} \|_1 \quad subject\ \ to \quad \| A\mathbf{x} - \mathbf{y} \|_2 \leq \epsilon \tag{3.17}$$

This problem can be solved via second-core programming[16]. However, since the above optimization problem is usually solved by converting the primal problem to a Lagrangian dual problem, the actually algorithm adopted in this thesis project is the one that solves a $\ell_1$-regularized least squares problem as indicated in Equation 2.5: $minimize \| \mathbf{Ax} - \mathbf{y} \|_2^2 + \lambda \| \mathbf{x} \|_1$.

After the sparse representation $\hat{x}_1$ is recovered via Equation 2.5, we perform classification on the corresponding test sample $\mathbf{y}$ in terms of the closeness between $\mathbf{y}$ and each reproduction of $\mathbf{y}$ that is derived by the coefficients associated with all the training samples in each action class. Here we define $\delta_i : \mathbb{R}^n \to \mathbb{R}^n$ as the function to select the coefficients only associate with the class $i$. For $\mathbf{x} \in \mathbb{R}^n, \delta_i(\mathbf{x}) \in \mathbb{R}^n$ represents the vector whose non-zero entries are the entries associated with the $i$th class. Therefore, $\hat{\mathbf{y}}_i = A\delta_i(\hat{\mathbf{x}}_1)$ is the approximation of $\mathbf{y}$ only using the coefficients associated with the $i$th class. $\mathbf{y}$ is classified as an action instance in the class that has the minimal residual between $\mathbf{y}$ and $\hat{\mathbf{y}}_i$:

$$\min_i r_i(\mathbf{y}) = \| \mathbf{y} - A\delta_i(\hat{\mathbf{x}}_1) \|_2 \tag{3.18}$$

Algorithm 1 summarizes the complete SRC algorithm.

**Algorithm 1** Sparse Representation-based Classification (SRC)

1: **Input:** a matrix of training samples $A = [A_1, A_2, ..., A_k] \in \mathbb{R}^{m \times n}$ for $k$ classes, a test sample $\mathbf{y} \in \mathbb{R}^m$, (and an optional error tolerance $\epsilon > 0$.)
2: Normalize the columns of $A$ to have unit $\ell_2$-norm.
3: Solve the $\ell_1$-minimization problem:

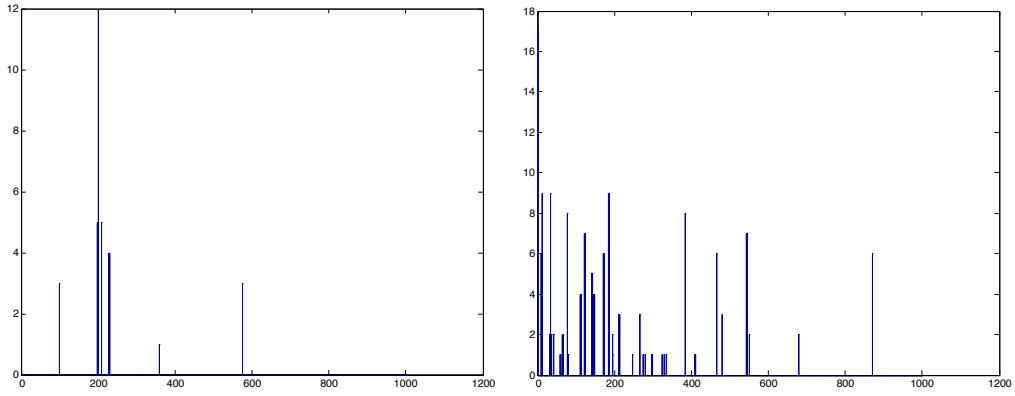
$$\hat{x}_1 = arg\,min \parallel \mathbf{x} \parallel_1 \quad subject \;\; to \;\; A\mathbf{x} = \mathbf{y} \tag{3.19}$$

Or alternatively, solve

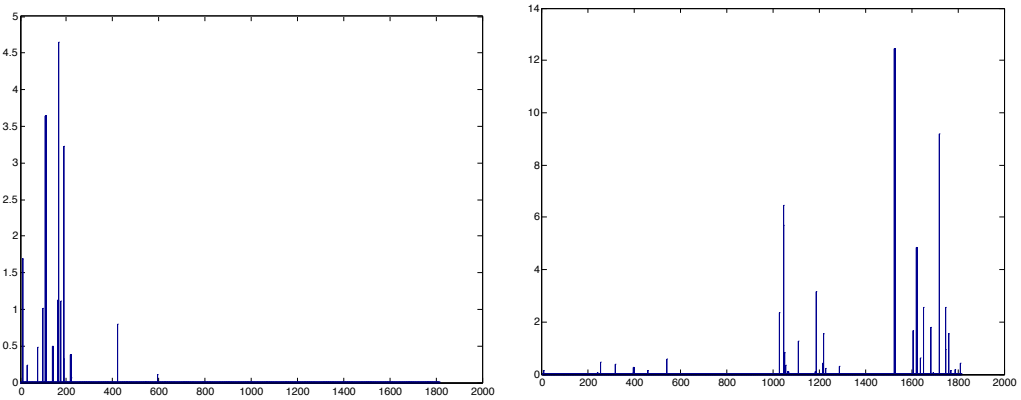
$$\hat{x}_1 = arg\,min \parallel \mathbf{x} \parallel_1 \quad subject \;\; to \quad \parallel A\mathbf{x} - \mathbf{y} \parallel_2 \leq \epsilon \tag{3.20}$$

4: Compute the residuals $r_i(\mathbf{y}) = \parallel \mathbf{y} - A\delta_i(\hat{\mathbf{x}}_1) \parallel_2$ for $i = 1, ..., k$.
5: **Output:** identify$(\mathbf{y}) = argmin_i r_i(\mathbf{y})$.

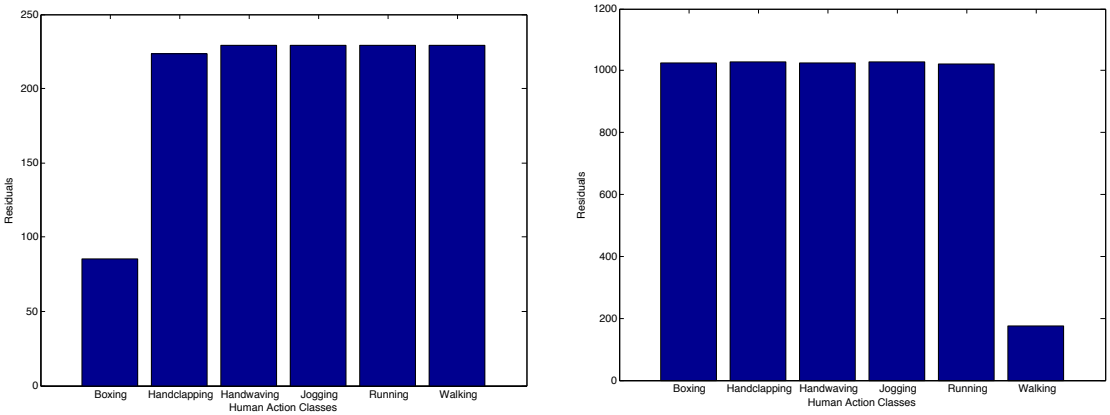
Figure 3.9 demonstrates the derived sparse coefficient vector after performing SRC and the calculated residuals in terms of each action class for a *boxing* action instance and a *walking* action instance. As can be observed from Figure 3.9, the reconstructed two coefficient vectors are quite sparse in terms of the number of the training samples. Only a small amount of entries in these coefficient vectors are non-zero. The calculated residuals clearly indicate the correct recognition of corresponding human action instances.



(a) Test visual words histogram from *boxing*(b) Test visual words histogram from *walking*



(c) Derived sparse coefficients vector for the test sample from *boxing*(d) Derived sparse coefficients vector for the test sample from *walking*




(e) Calculated residuals for all the action classes -*boxing*(f) Calculated residuals for all the action classes -*walking*

Figure 3.9: Human action recognition based on sparse representation

# Chapter 4

# System Implementation and Experiment Results

## 4.1 KTH Human Action Dataset

The objective of this project is to recognize the categories of the human actions shown in the video files of the public test dataset. The dataset that is used in the thesis project is the KTH human motion dataset[31] (Figure 4.1) and can be obtained from the website http://www.nada.kth.se/cvap/actions/. The KTH human motion dataset contains six actions (walking, jogging, running, boxing, hand waving and hand clapping), performed by 25 different actors. There are four different scenarios for all the sequences in the dataset: outdoors, outdoors with variations, outdoors with different clothing and indoors. The dataset contains 2391 sequences and all the sequences were taken over homogeneous backgrounds with a static camera with 25fps frame rate. Apart from the zooming scenario, the backgrounds are relatively static with sometimes slight camera movement. The sequences have the spatial resolution of $160 \times 120$ and have a length of four seconds in average. All sequences are stored using

AVI file format. There are 600 video files in total corresponding to all the combination of 25 performers, 6 action categories and 4 scenarios. Each file contains about four subsequences, each of which is used as a *sequence* in the experiments. The subdivision of each file into *sequences* in terms of start frame and end frame as well as the list of all sequences can also be obtained from the website mentioned above.

## 4.2 Evaluation Methodology

The proposed human action recognition system is evaluated in two manners: Leave One Out Cross Validation (LOOCV) and Leave Part Out Cross Validation (LPOCV). In LOOCV, each action instance in the dataset is treated as the test sample, the rest of the whole action dataset is the training set. The overall accuracy is calculated as the percentage of right recognition for all the action instances in the dataset. In LPOCV, the KTH dataset are divided into four groups according to the performing individuals; Table. 4.1 shows the division of groups. One of the four groups is used as test set and the rest three are used as training set. For each test set, the recognition accuracy is calculated; then the overall accuracy is the result combined over all the test sets. For conditions where there exist random components such as clustering, the accuracy is the average value over several runs. The accuracies are then put into a confusion matrix as the final results. To examine the effectiveness of SRC algorithm in the human action recognition problem, we also perform the experiments with KNN and SVM in the same conditions. In addition, the computational complexity of SRC, the effects of the training set size and regularization parameter $\lambda$ on the overall recognition accuracy are also investigated.

| Group | Performer |
|---|---|
| 1 | 1 - 6 |
| 2 | 7 - 13 |
| 3 | 14 - 19 |
| 4 | 20 - 25 |

Table 4.1: The groups for evaluation

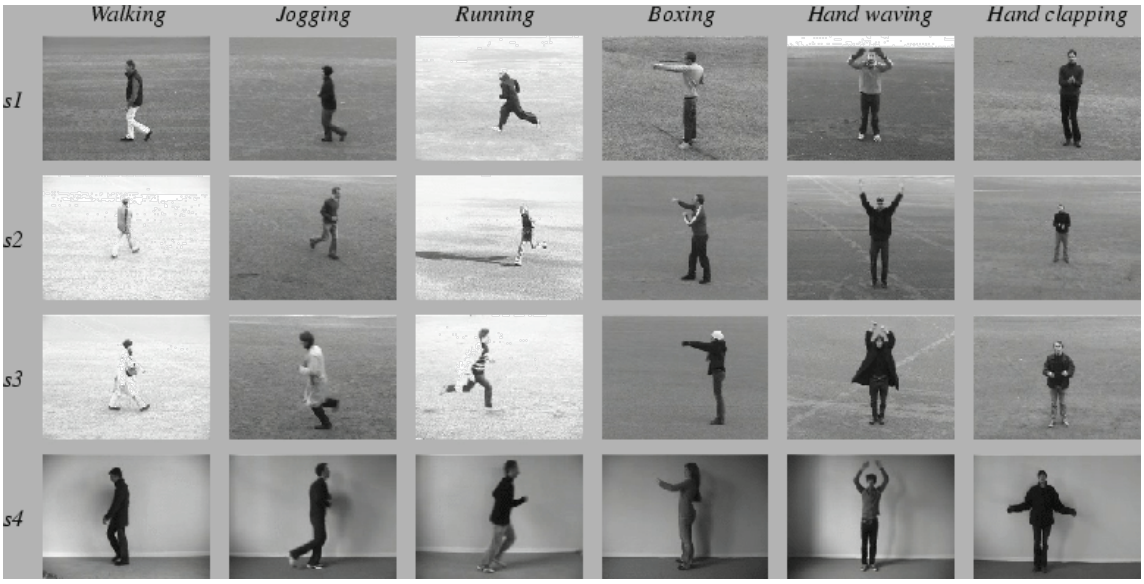



Figure 4.1: KTH human action dataset

## 4.3 Experimental Results and Analysis

### 4.3.1 Leave One Out Cross Validation

In the first experiment, the LOOCV strategy is adopted to evaluate the effectiveness of SRC in human action recognition problem. In LOOCV, each action instance is chosen as the test sample, and the rest action instances are chosen as the test set.

For SRC, the highest recognition accuracy is 96.65% which is achieved when codebook size is 1000. The corresponding confusion matrix is shown in 4.2.

For KNN, the highest recognition accuracy is 96.16% which is achieved when the

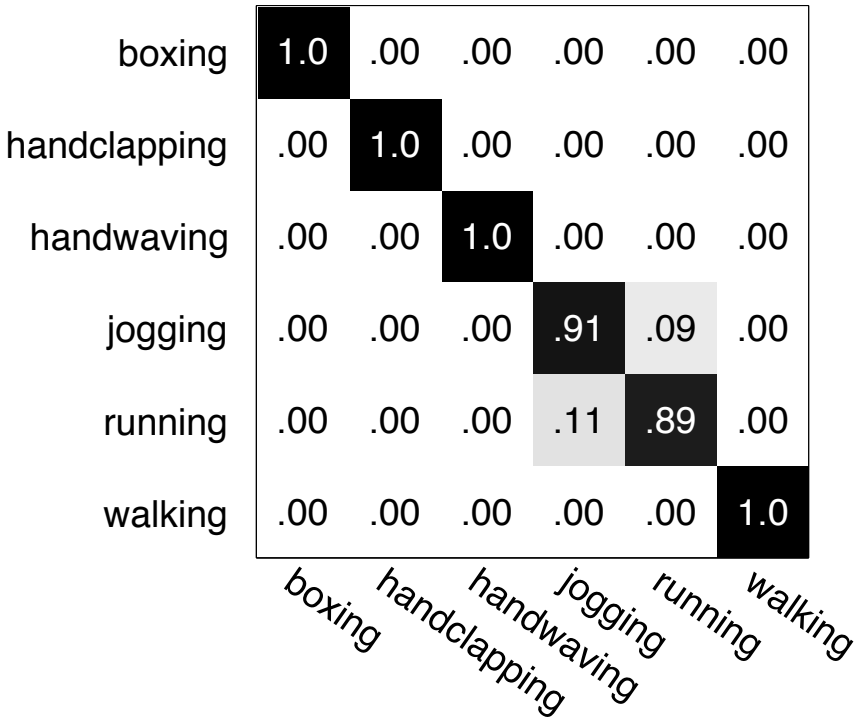



Figure 4.2: LOOCV recognition accuracy for SRC when codebook size is 1000

codebook size is 750. The corresponding confusion matrix is shown in 4.3.

Figure 4.4 demonstrates the changes of average recognition accuracy in LOOCV with the variation of codebook size. The average recognition accuracy is examined for the codebook size varying from 50 to 1000 with the interval set to be 50. As shown in Figure 4.4, SRC outperforms KNN for most of codebook size. One thing worth noting is that performing SVM algorithm on the human action recognition problem in LOOCV is enormously time-consuming because one specific training model of SVM has to be generated for each different combination of the training samples. The number of different training models have to be generated is the same as the number of actions instances in the human action dataset. Therefore, we do not evaluate the performance of SVM in LOOCV.

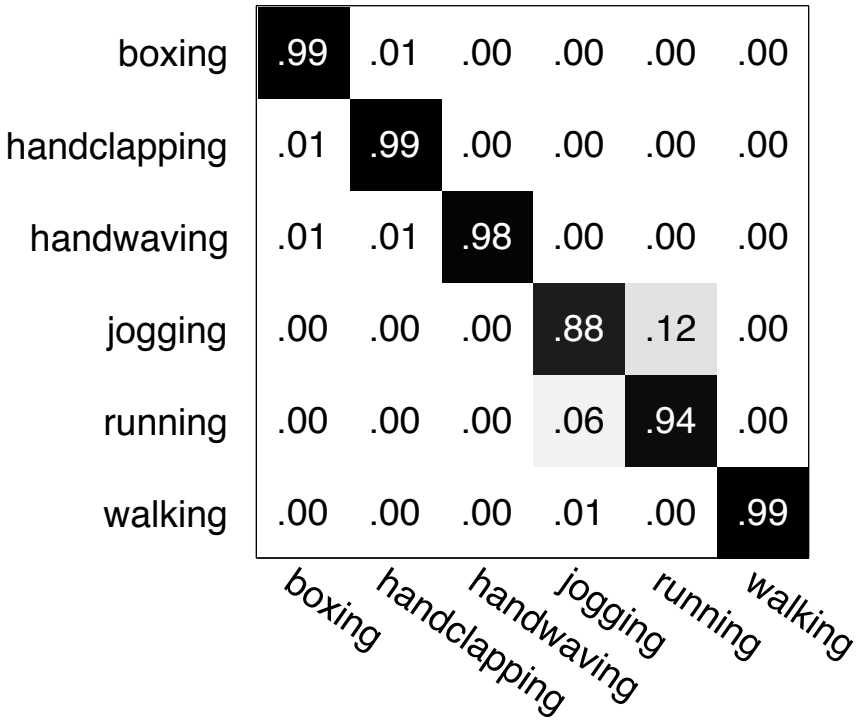



Figure 4.3: LOOCV recognition accuracy for KNN when codebook size is 750

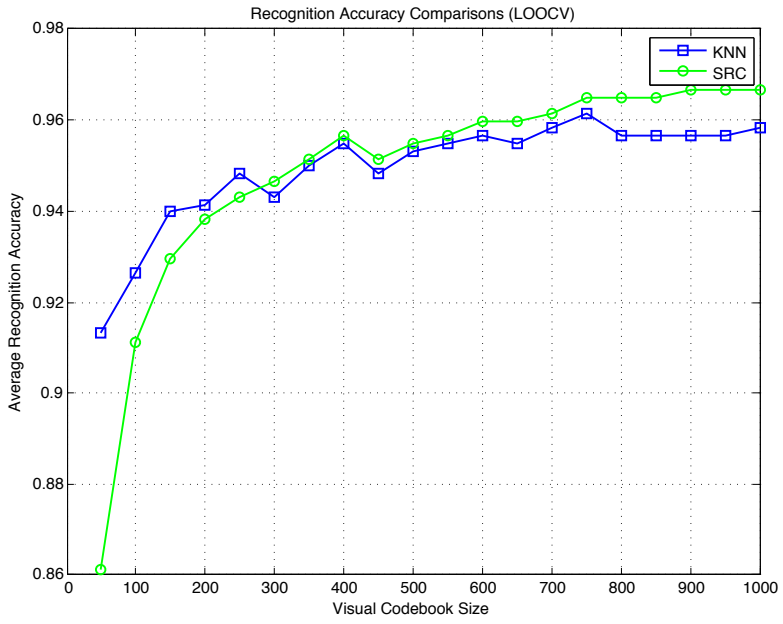



Figure 4.4: Comparison of LOOCV recognition accuracy for KNN and SRC with variable codebook size

### 4.3.2 Leave Part Out Cross Validation

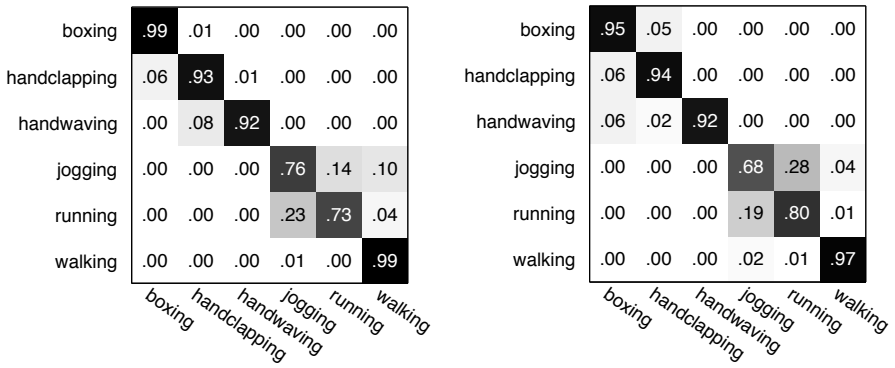
In this experiment, the performance of LPOCV is examined. Each group is regarded as the test set, the remaining three groups are regarded as the training set. The overall recognition accuracy is derived by combining the results from the four different scenarios. Still, the average recognition accuracy is examined for the codebook size varying from 50 to 1000 with the interval set to be 50. We also compare the performance among SRC, SVM and KNN.

The highest recognition accuracy for SRC is 89.01% which is achieved when the codebook size is 1000. The corresponding matrices for the four scenarios are demonstrated in Figure 4.5.

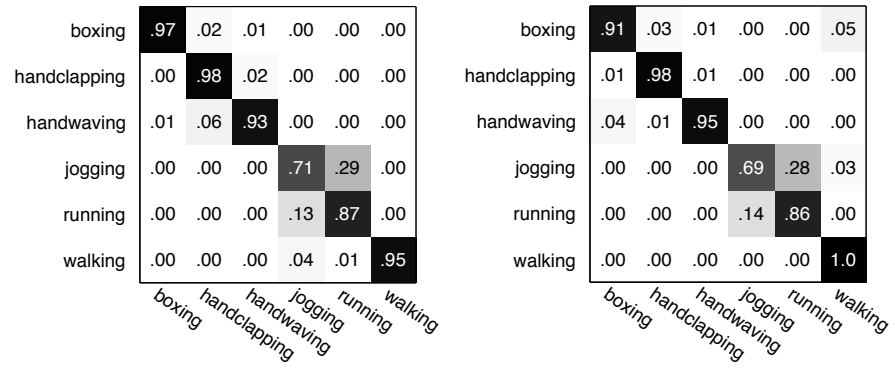
The highest recognition accuracy for SVM is 87.98% which is achieved when the codebook size is 950. The corresponding matrices for the four scenarios are demonstrated in Figure 4.6.

The highest recognition accuracy for KNN is 81.91% which is achieved when the codebook size is 300. The corresponding matrices for the four scenarios are demonstrated in Figure 4.7.

The direct comparisons of average recognition accuracy derived from SRC, SVM, KNN along with the variations of codebook size is shown in Figure 4.8.

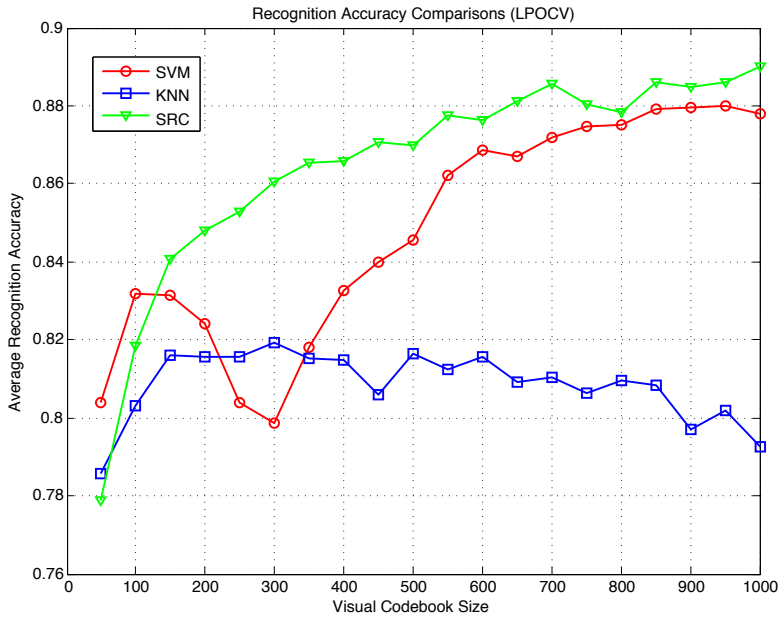


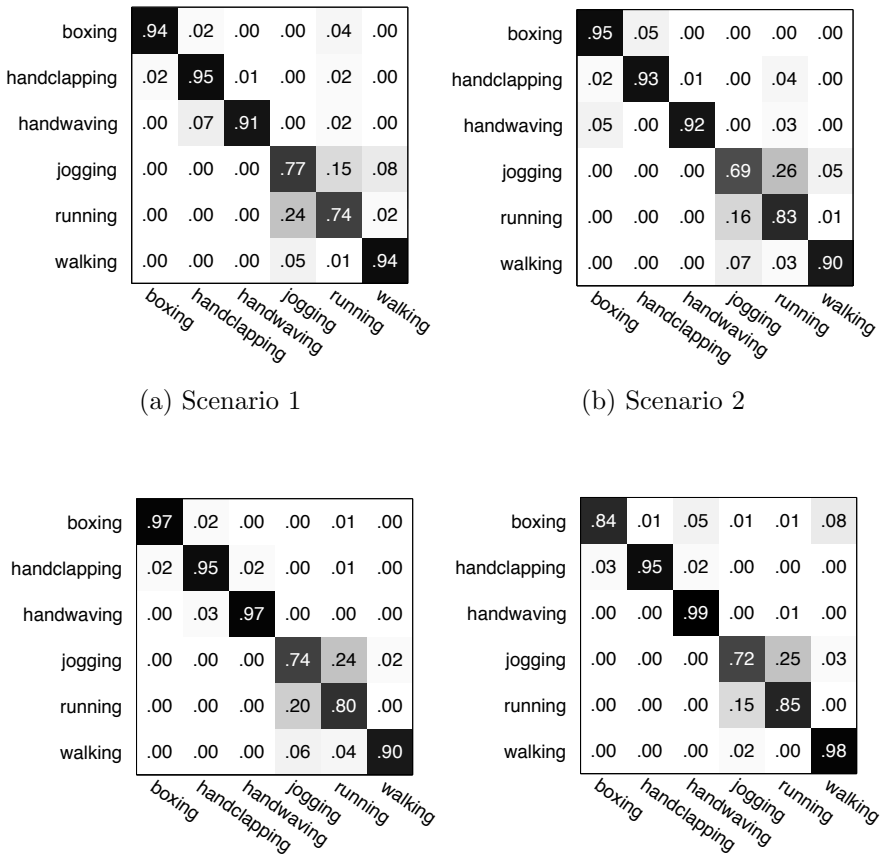

(a) Scenario 1 (b) Scenario 2




(c) Scenario 3 (d) Scenario 4

Figure 4.5: Confusion matrices derived from SRC on four different scenarios



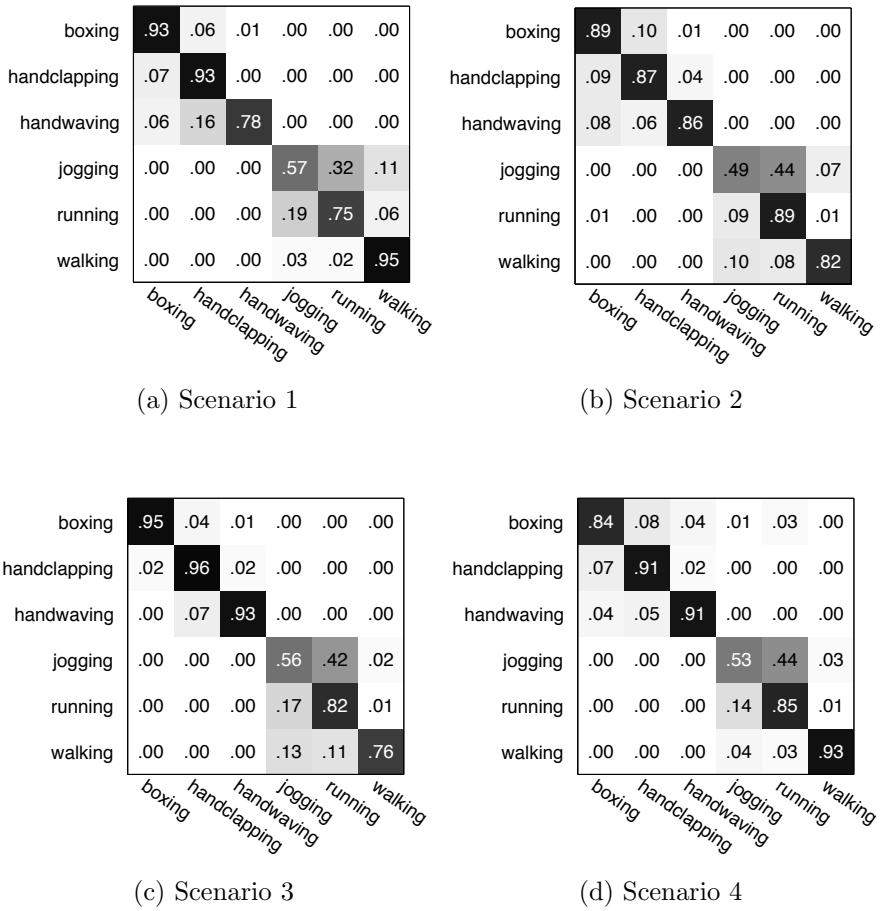



(a) Scenario 1 (b) Scenario 2

(c) Scenario 3 (d) Scenario 4

Figure 4.6: Confusion matrices derived from SVM on four different scenarios

As shown in Figure 4.8, although SRC achieves lower recognition rate when the codebook size is very small, SRC demonstrates overall higher recognition rate than SVM and KNN for most codebook sizes. SRC demonstrates an overall increasing trend of recognition accuracy with the increase of codebook size. There is obvious fluctuation of recognition accuracy for SVM when the codebook size increases. For KNN, the average recognition accuracy decreases a little bit when the codebook size increase. Therefore, SRC demonstrates better robustness with the variation of codebook size compared with SVM and KNN. By observing the confusion matrix of




(a) Scenario 1 (b) Scenario 2

(c) Scenario 3 (d) Scenario 4

Figure 4.7: Confusion matrices derived from KNN on four different scenarios

SRC, the performance of SRC algorithm on different human action classes can be analyzed. SRC derive very impressive recognition accuracy on the action class *boxing*, *handclapping*, *handwaving* and *walking* with average recognition accuracy over 95%. However, the recognition accuracy on action classes *jogging* and *running* are relative low. There are more miss classifications between these two action categories. This is mainly due to the great similarities shared between these two human actions classes. Sometimes, it is even difficult for human individuals to discriminate one from the other since the definitions of *jogging* and *running* are not quite clear to some extent.

### 4.3.3 Computational complexity analysis with different visual codebook sizes

The computational complexity of SRC is investigated in terms of the variation of visual codebook size. The average running time for recognizing one test sample is calculated based on the total running time for recognizing the whole test dataset. The human action recognition system is implemented in Matlab and runs on a PC with Dual Core 3.33GHz CPU and 4GB RAM. Figure 4.9 demonstrates the corresponding result. The average running time shown in Figure 4.9 does not take the computational cost of video sequence preprocessing into consideration. It is all about the computational complexity of the classification algorithms. Based on the results shown in Figure 4.9, we can tell that the running time for SRC increases when the visual codebook size grows. It is approximately a linear relationship between the running time and the codebook size. When the codebook size is 1000, SRC achieves its maximal computational complexity, which requires about 3.5 seconds in average to recognize a test sample. In addition, the direct comparison of the computational complexity for different algorithms is also demonstrated in Figure 4.9. The KNN and SVM algorithms indicate similar relationships between the running time and the codebook size as SRC. However, these two algorithms have much smaller computational complexity than SRC since SRC needs to do iterations containing the PCG steps many times to finish the optimization problem which is quite time consuming. Therefore, SRC derives overall better performance in recognition rate at the cost of increasing computational complexity.

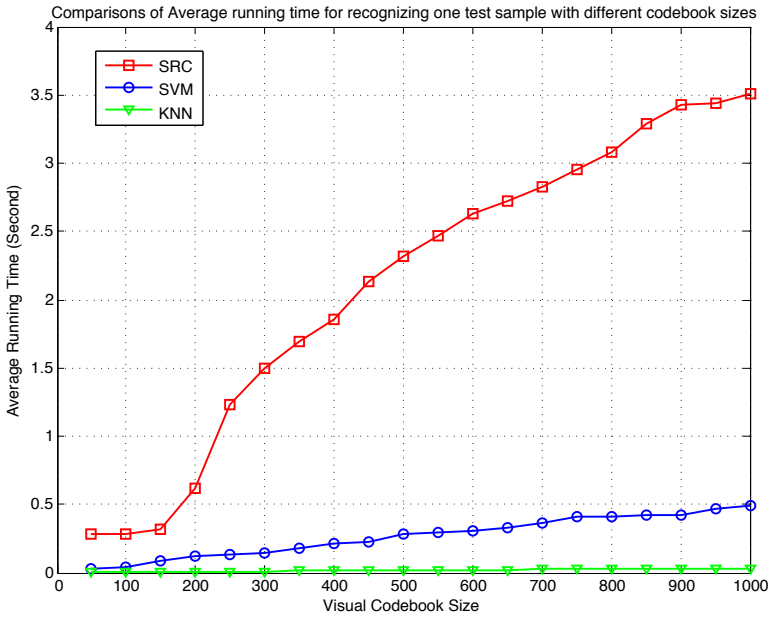



Figure 4.9: Comparisons of Average running time for recognizing one test sample with different codebook sizes

### 4.3.4 The effect of training sample size on recognition accuracy

The effect of training set size on average recognition accuracy is also investigated for SRC and SVM. Since SRC and SVM outperform KNN a lot in LPOCV according to the experimental results presented in the previous subsection, we do not compare SRC and SVM with KNN in this experiment. In this experiment, Group 3 is adopted as the test set. All the possible combination of the remaining three groups are chosen as the training set. Therefore, there are 7 possible training set combinations which are shown in Table 4.2.

| Test Set | Training Set |
|---|---|
| Group 3 | Group 1 |
| Group 3 | Group 2 |
| Group 3 | Group 4 |
| Group 3 | Group 1, 2 |
| Group 3 | Group 1, 4 |
| Group 3 | Group 2, 4 |
| Group 3 | Group 1, 2, 4 |

Table 4.2: Different combinations of training set

The codebook size is set to be 500. The average recognition accuracy for SRC and SVM is shown in Table 4.3.

| Test Set | Training Set | SRC | SVM |
|---|---|---|---|
| Group 3 | Group 1 | 84.15% | 82.68% |
| Group 3 | Group 2 | 85.99% | 82.54% |
| Group 3 | Group 4 | 85.85% | 79.23% |
| Group 3 | Group 1, 2 | 88.68% | 86.69% |
| Group 3 | Group 1, 4 | 86.17% | 85.57% |
| Group 3 | Group 2, 4 | 90.24% | 87.46% |
| Group 3 | Group 1, 2, 4 | 89.34% | 87.60% |

Table 4.3: Comparison of recognition accuracy for SRC and SVM with different combination of training samples

To compare the performance of SRC and SVM in terms of the variation of training set, the mean and standard deviation are calculated for the two groups of recognition accuracy. For SRC, the mean recognition accuracy is 87.2%, the standard deviation is 2.22%; for SVM, the mean recognition accuracy is 84.54%, the standard deviation is 3.14%. Thus, SRC does not only demonstrate higher average recognition accuracy than SVM, but also show more robustness with regard to variation of training set than SVM.

### 4.3.5 The effect of regularization parameter on recognition accuracy

The sparse representation based classification (SRC) result is actually derived by solving an $\ell_1$-regularized least squares (an extension of $\ell_1$-minimization problem), whose mathematical representation is denoted in Equation 2.5.

The term $\lambda$ in Equation 2.5 is the regularization parameter. The relative tolerance of the above problem is set to be 0.001, which means that the derived solution would never be worse than 0.1% suboptimal. We want to investigate the effect of the value change in regularization parameter on the average recognition accuracy. The average recognition accuracy is evaluated using the same methodology introduced above, however, with regularization parameter $\lambda$ being set to a series of different values from 0.001 to 10. Figure 4.10 demonstrates the corresponding result.

Table 4.4 demonstrates the detailed recognition accuracy with different regularization parameters. Through the observation from Figure 4.10 and Table 4.4, we can draw the conclusion that the overall recognition accuracy does not vary a lot across a vast range of regularization parameter values. In other words, the performance of SRC is close to parameter-independent and demonstrates great robustness on the variation

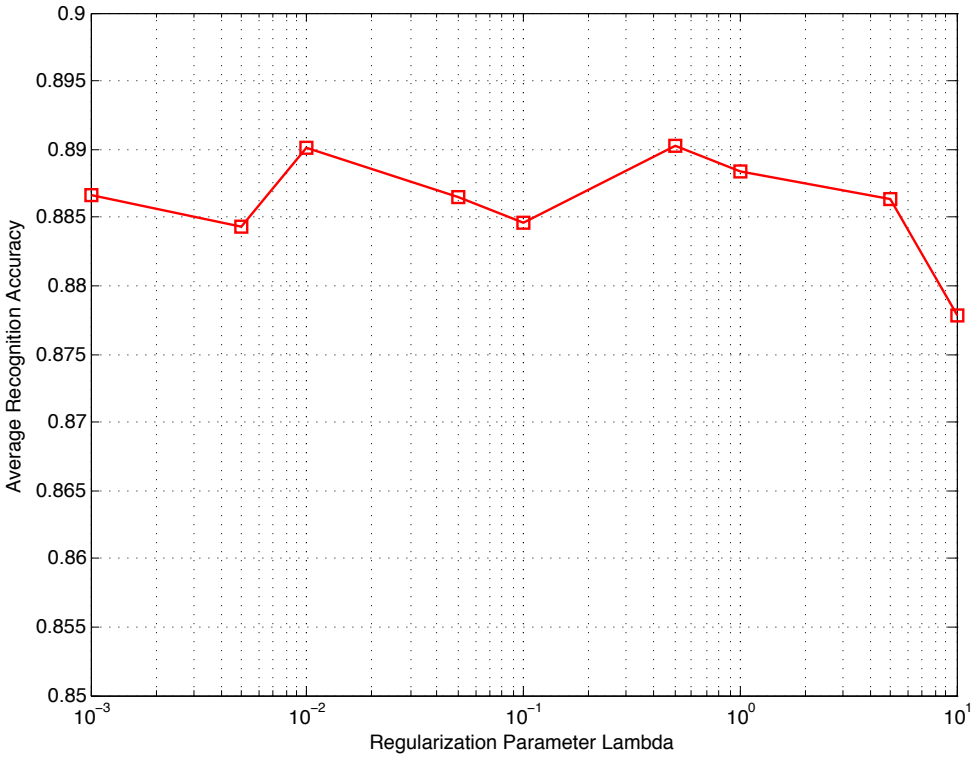



Figure 4.10: Average recognition accuracy with different regularization parameters

of regularization parameters. Smaller $\lambda$ results in longer execution time in solving the $\ell_1$-regularized least squares. Therefore, larger $\lambda$ can achieve accelerated running time and acceptable recognition accuracy without obvious compromise.

| $\lambda$ | 0.001 | 0.005 | 0.01 | 0.05 | 0.1 | 0.5 | 1 | 5 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| Result | 88.67% | 88.43% | 89.01% | 88.65% | 88.46% | 89.02% | 88.84% | 88.63% | 87.78% |

Table 4.4: Average recognition accuracy with different regularization parameters

# Chapter 5

# Conclusion and Future Work

In this thesis, the human action recognition problem is first examined by investigating the previously established solutions and then solved from a novel sparse representation perspective. In this sparse representation framework, each action instance is represented as a collection of spatial-temporal words. First a large number of spatial-temporal interest points are extracted in the video sequence. Then, a cuboid is extracted centered at each spatial-temporal interest points. The histogram of oriented gradients (HOG) and histogram of flow (HOF) descriptor for the cuboid are computed and concatenated into a one-dimensional vector. The K-Means clustering algorithm is used to cluster these cuboid feature vectors into only a few cuboid prototypes which are called visual codewords. After all the processing, each action instance is represented as a histogram of the visual codewords. Sparse representation can achieve great efficiency and performance in classification problems and can be embedded into machine learning frameworks. We apply this technique in the human action recognition problem. Each action instance in the test set is represented approximately as a linear weighted sum of all the action instances in the training set. Since the linear weight vector is sparse, we can use the convex optimization technique called $\ell_1$-minimization

which is widely used in compressed sensing and sparse representation problems to derive the result. The action category of the test instance is recognized by calculating the residual between the test instance itself and its corresponding representation using the action instances in each action class. The test action instance falls into the action class with the smallest residual. Our proposed human action recognition system is tested and evaluated on the famous and challenging KTH human motion dataset. The derived experimental results using our method are compared with the results derived using conventional machine learning techniques such as K-Nearest Neighbor (KNN) and Support Vector Machines (SVM) and show that the proposed framework yield considerable performance improvement in many aspects. In addition to comparison with SVM and KNN, the computational complexity of SRC, the effects of training set size and regularization parameters on the performance of SRC are also investigated thoroughly.

Some future work could be performed to further improve the performance and results. For instance, better results may be obtained by further tuning the parameters. Although the HOGHOF descriptor is adopted in the thesis project, the algorithm gives a framework of action recognition thus making it possible to incorporate other image descriptors into the framework to investigate the performance of SRC with regard to different image descriptors. It is also possible to combine more than one descriptor at the same time and see how they perform. In this project we used the $k$-means algorithm to build the visual codebook. It is a relative simple and straightforward clustering algorithm. More sophisticated clustering algorithms such as spectral clustering could be utilized to perform the visual codewords clustering. Also, other extensions of $\ell_1$-minimization algorithms can be adopted to improve the overall performance of the human action recognition system.

# Bibliography

[1] R. Poppe, "A survey on vision-based human action recognition," *Image and Vision Computing*, pp. 976–990, 2010. 1.1

[2] T. B. Moeslund, A. Hilton, and V. Kruger, "A survey of advances in vision-based human motion capture and analysis," *Computer Vision and Image Understanding*, vol. 104(2-3), pp. 90–126, 2006. 1.1

[3] D. M. Gavrila, "The visual analysis of human movement: a survey," *Computer Vision and Image Understanding*, vol. 73(1), pp. 82–98, 1999. 1.1

[4] R. Poppe, "Vision-based human motion analysis: an overview," *Computer Vision and Image Understanding*, vol. 108(1-2), pp. 4–18, 2007. 1.1

[5] C. Cedras and M. Shah, "Motion-based recognition: a survey," *Image and Vision Computing*, vol. 13(2), pp. 129–155, 1995. 1.1

[6] E. Shechtman and M. Irani, "Space-time behavioral correlation," in *Proc. of the Conference on Computer Vision and Pattern Recognition*, 2005. 1.3

[7] A. Yilmaz and M. Shah, "Recognizing human actions in videos acquired by uncalibrated moving cameras. images," in *Proc. of the tenth IEEE internatinal conference on computer vision*, 2005. 1.3

[8] C. Fanti, L. Zelnik-Manor, and P. Perona, "Hybrid models for human motion recognition," in *Proc. of the tenth IEEE internatinal conference on computer vision*, 2005. 1.3

[9] A. F. Bobick and J. Davis, "The recognition of human movement using temporal templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23(3), pp. 257–267, 2001. 1.3

[10] L. Gorelick, M. Blank, E. Schechtman, M. Irani, and R. Basri, "Actions as space-time shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29 (12), pp. 2247–2253, 2007. 1.2, 1.3

[11] I. Laptev, "On space-time interest points," *International Journal of Computer Vision*, vol. 64 (2-3), pp. 107–123, 2005. 1.3

[12] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *Proc. of the International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005. 1.3, 3.2.1

[13] E. Candes and M. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25 (2), pp. 21–30, 2008. 1.4

[14] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE Signal Processing Magazine*, pp. 83–91, March 2008. 2.1

[15] D. Takhar, J. N. Laska, M. B. Wakin, M. F. Duarte, D. Baron, S. Sarvotham, K. F. Kelly, and R. G. Baraniuk, "A new compressive imaging camera architecture

using optical-domain compression," in *Proceedings of Computational Imaging IV at SPIE Electronic Imaging*, pp. 43–52, January 2006. 2.1

[16] S. S. Chen, D. L. Domoho, and M. A. Saunders, "Atomic decomposition by bais pursuit," *SIAM Review*, vol. 43 (1), pp. 129–159, 2001. 2.2, 3.2.3

[17] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 53, pp. 4655–4666, December 2007. 2.2

[18] E. Candes, "Compressed sampling," in *Proceedings of the International Congress of Mathematicians*, 2006. 2.2

[19] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior-point method for large-scale $\ell_1$-regularized least squares," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 606–617, 2007. 2.3

[20] M. Cossalter, M. Tagliasacchi, and G. Valenzise, "Privacy-enabled object tracking in video sequences using compressive sensing," in *Proc. of the Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 436–441, 2009. 2.4

[21] V. Cevher, A. C. Sankaranarayanan, M. F. Duarte, D. Reddy, R. G. Baraniuk, and R. Chellappa, "Compressive sensing for background subtraction," in *Proceedings of the European Conference on Computer Vision*, pp. 155–168, 2008. 2.4

[22] X. Mei and H. Ling, "Robust visual tracking using $\ell 1$ minimization," in *Proceedings of the International Conference on Computer Vision*, pp. 1436–1443, 2009. 2.4

[23] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31 (2), pp. 210–227, 2009. 2.4, 3.2.3

[24] A. Y. Yang, R. Jafari, S. S. Sastry, and R. Bajcsy, "Distributed recognition of human actions using wearable motion sensor networks," *Journal of Ambient Intelligence and Smart Environments*, pp. 1–5, 2009. 2.4

[25] A. Yang, S. Maji, C. Christoudias, T. Darrell, J. Malik, and S. Sastry, "Multiple-view object recognition in band-limited distributed camera networks," in *Third ACM/IEEE International Conference on Distributed Smart Cameras*, pp. 1–8, 2009. 2.4

[26] D. Gabor, "Theory of communication," *Journal of the Institute of Electrical Engineers*, vol. 93, pp. 429–457, 1946. 3.1.1

[27] N. Dalal and B. Triggs, "Histogram of oriented gradients for human detection," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886–893, 2005. 3.1.2

[28] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histogram of flow and appearance," in *Proc. of the European Conference on Computer Vision*, pp. 7–13, 2006. 3.1.2

[29] N. Dalal, *Finding People in Images and Videos*. PhD thesis, Institut National Polytechnique de Grenoble / INRIA Grenoble, Grenoble, July 2006. 3.1.2

[30] J. Shawe-Taylor and N. Cristianini, *Support Vector Machines*. Cambridge University Press, 2000. 3.1.3

[31] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: a local svm approach," in *Proc. of the International Conference on Pattern Recognition*, 2004. 4.1