5-2014

Feature Extraction and Recognition for Human Action Recognition

Jiajia Luo
University of Tennessee - Knoxville, jluo9@utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Jiajia Luo entitled "Feature Extraction and Recognition for Human Action Recognition." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Engineering.

Hairong Qi, Major Professor

We have read this dissertation and recommend its acceptance:

Mongi A. Abidi, Peter K. Liaw, Husheng Li, Qing Cao

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

5-2014

Feature Extraction and Recognition for Human Action Recognition

Jiajia Luo
University of Tennessee - Knoxville, jluo9@utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Jiajia Luo entitled "Feature Extraction and Recognition for Human Action Recognition." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Engineering.

Hairong Qi, Major Professor

We have read this dissertation and recommend its acceptance:

Mongi A. Abidi, Peter K. Liaw, Husheng Li, Qing Cao

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Feature Extraction and Recognition for Human Action Recognition

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Jiajia Luo

May 2014

Dedication

This dissertation is dedicated to my parents, my husband, and my brother.

Acknowledgements

I am grateful to many people that made this work possible. First, I would like to thank my advisor, Dr. Hairong Qi. Her patient guidance through my studies, strong passion for scientific research, broad knowledge in different areas, and rigorous attitudes to research, always enlightened my path and supported me to solve bottlenecks. I am also indebted that she gave me all the freedom and resources to develop my skills as a researcher in a supportive team environment. More importantly, I learned a lot from her on how to become a good researcher, wife and mother. Also, I would like to thank Dr. Mongi A. Abidi, Dr. Peter K. Liaw, Dr. Husheng Li, and Dr. Qing Cao. Their advice and counsel have been of equal importance. I greatly appreciate their time and input to this dissertation.

Within the AICIP Lab, I owe many thanks to my fellow graduate students. I enjoyed the many conversations and discussions that have had a tremendous impact on my research and myself as a person. Yang Bai, Sangwoo Moon, Mahmut KaraKaya, Dayu Yang, Wei Wang, Zhibo Wang, Shuangjiang Li, Li He, Rui Guo, Bryan Bodkin, Shravani Kanjarkar, Liu Liu, Song Yang, Alireza Rahimpour and Cristian Capdevila, thank you very much.

Last but not least, I would like to express my greatest gratitude to my parents, my brother and my husband. I feel very lucky and privileged to have them.

Abstract

How to automatically label videos containing human motions is the task of human action recognition. Traditional human action recognition algorithms use the RGB videos as input, and it is a challenging task because of the large intra-class variations of actions, cluttered background, possible camera movement, and illumination variations. Recently, the introduction of cost-effective depth cameras provides a new possibility to address difficult issues. However, it also brings new challenges such as noisy depth maps and time alignment. In this dissertation, effective and computationally efficient feature extraction and recognition algorithms are proposed for human action recognition.

At the feature extraction step, two novel spatial-temporal feature descriptors are proposed which can be combined with local feature detectors. The first proposed descriptor is the Shape and Motion Local Ternary Pattern (SMltp) descriptor which can dramatically reduced the number of features generated by dense sampling without sacrificing the accuracy. In addition, the Center-Symmetric Motion Local Ternary Pattern (CS-Mltp) descriptor is proposed, which describes the spatial and temporal gradients-like features. Both descriptors (SMltp and CS-Mltp) take advantage of the Local Binary Pattern (LBP) texture operator in terms of tolerance to illumination change, robustness in homogeneous region and computational efficiency.

For better feature representation, this dissertation presents a new Dictionary Learning (DL) method to learn an overcomplete set of representative vectors (atoms) so that any input feature can be approximated by a linear combination of these atoms with minimum reconstruction error. Instead of simultaneously learning one overcomplete dictionary for all classes, we learn class-specific sub-dictionaries to increase the discrimination. In addition, the group sparsity and the geometry constraint are added to the learning process to further increase the discriminative power, so that features are well reconstructed by atoms from the same class and features from the same class with high similarity will be forced to have similar coefficients.

To evaluate the proposed algorithms, three applications including single view action recognition, distributed multi-view action recognition, and RGB-D action recognition have been explored. Experimental results on benchmark datasets and comparative analyses with the state-of-the-art methods show the effectiveness and merits of the proposed algorithms.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Human Action Recognition

Human action recognition has been researched since early 1980s, due to its promise in many application domains, including visual surveillance, video indexing, gesture recognition, video retrieval and human-computer interaction. The goal of human action recognition is to recognize ongoing actions from unknown video consisting of a sequence of images. As shown in Figure 1.1, the input sources are videos (RGB or Depth videos) and the output are class labels which are automatically generated by designed algorithms.

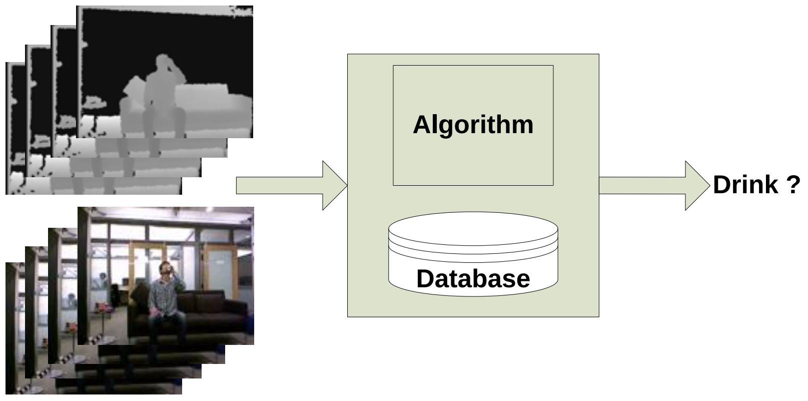
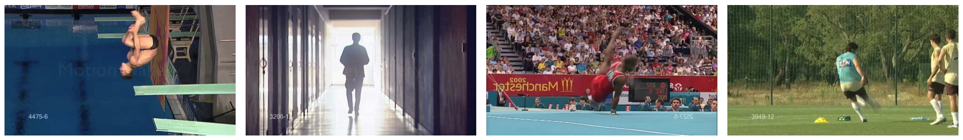
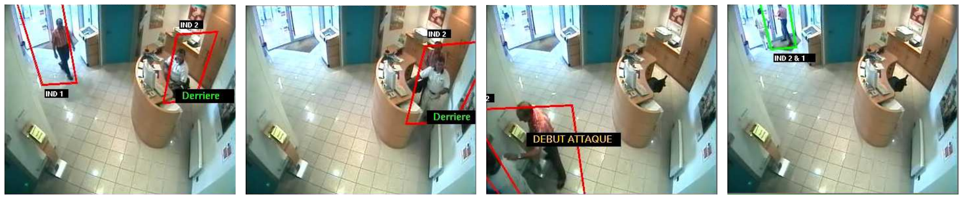


Figure 1.1: Task of human action recognition.



(a) Actions



(b) Activity

Figure 1.2: Sample images for actions and activities. (a) Four actions from the UCF action dataset Rodriguez et al. (2008): diving, walking, swinging and kicking. (b) Example video sequence of a simulated bank attack Georis et al. (2004).

Based on the complexity of motions, human body movements can be divided into different levels. Human action recognition is to classify motions that belong to the "action" level. According to the definition in the literature Turaga et al. (2008); Aggarwal and Ryoo (2011), we conceptually categorized the motions into three levels including: *gesture*, *action* and *activity*.

- *Gestures* are the atomic components that describe the meaning of the motions. "raising an arm" and "waving arms"are motions belong to this category.

- *Actions* are single person activities that may be composed of multiple gestures organized temporally. 'Walking', 'waving' and 'drinking' are examples of simple actions.

- *Activities* are complex sequences of actions performed by several persons and/or objects. They are typically characterized by interacting with each other in a constrained manner. "Playing basketball" is an example of activity consisting of actions, such as running, shooting, dribbling, and objects such as basketball.

Figure 1.2 shows the sample images for actions and activities. Actions such as diving, walking, swinging and kicking may contain complex background as shown in

Figure 1.2a. The task of human action recognition is to generate class labels for query video that may contain complex background information. Images in Figure 1.2b shows the sample sequences of a simulated bank attack, which is an example of activity. As shown in Figure 1.2b, two persons are evolved in this activity including a series of actions, i.e walking, standing up, opening the door and existing the door. Before fully exploring the characteristics of human activity, how to design a system that can automatically recognize what actions are being or have been performed is the initial step and of great importance.

In this dissertation, we focus on actions and do not explicitly consider context such as background, interactions between persons or objects. Especially, we explore the spatial-temporal information for human action recognition and proposed novel algorithms that can be applied on different tasks. To generate the recognition result, there are three important and necessary steps:

- *Feature extraction:* Raw video sequence consists of massive spatio-temporal pixel intensity variations that contribute nothing to the action itself, such as pixels related to the color of clothes and cluttered background. Feature extraction is a process that detects and extracts most representative information from raw data as features.

- *Feature representation:* Any video sequence will generate a specific number of features, and different video sequences will have distinctive number of features. Feature representation is a process to give a unique representation for every video sequence based on the extracted features. The final representation should be of the same dimension among different videos.

- *Classification:* Based on the calculated representation, classification is a process to assign class labels to any unknown video sequence according to the learned classifier. Parameters for the classifier are learned from training samples.

1.2 Motivations

The solution to automatically recognize actions from videos is more difficult than the question itself. During the past 20 years, many approaches have been pursued to address this problem. However, there is still a long distance from practical application of these proposed algorithms due to challenges in this task.

One main challenge is how to efficiently extract representative features from raw data. These detected features should be distinctive for various actions, and be similar among actions from the same class. However, subjects are free to choose their styles to perform actions which could cause large intra-class variations. In addition, cluttered background and camera motion will further introduce noise to the process. In the literature, feature detection and feature description are two main steps in extracting local spatio-temporal features. The feature detector aims to detect locations of representative interest points with various scales. The shape and motion characteristics of the detected 3D patches (or interest regions surrounding the detected interest points) can be further described by feature descriptors. Many feature detectors Dollar et al. (2005); Huang et al. (2007); Laptev and Lindeberg (2005); Oikonomopoulos et al. (2006); Willems et al. (2008b); Wong and Cipolla (2007) and descriptors I.Laptev et al. (2008); Willems et al. (2008b); Klaser et al. (2008); Laptev and Lindeberg (2004); Scovanner et al. (2007) have been proposed in the past years. However, the performance of various descriptors depends on the selection of detectors and action datasets Wang et al. (2009). In addition, the computational cost is large for some algorithms, which hinders the real world application.

Another main challenge is how to represent the extracted features for any video sequence. The number of extracted features can be large and different for given videos. Moreover, how to incorporate the spatial and temporal information during the representation process for better performance should be taken into consideration. The most popular framework at this step is the Bag-of-Words (BoW) model, where the “visual words” are referred to as extracted features. One of the notorious disadvantages of BoW is that it ignores the spatial relationships among the extracted features, which is very important in video representation. Furthermore, the BoW model has not been extensively tested yet for view point invariance and scale invariance.

Recently, the release of the Microsoft Kinect provides a new possibility to address these issues. The Kinect device can provide both high resolution depth maps and RGB images at low cost, which opens up new opportunities to the research in computer vision, gaming, gesture-based control and virtual reality Xia et al. (2012). Figure 1.3 shows the sample images captured by the RGB camera and depth camera. Pixel intensities in the depth maps show the distances of pixel locations so that all the texture information is neglected.

However, action recognition based on depth maps also have new challenges. For example, the direct usage of the 3D joint positions extracted by the skeleton tracker Shotton et al. (2011) is not optimal due to the possible failure caused by noisy depth maps or occlusions. In addition, different lengths of sequences require algorithms to be tolerant to the time misalignment. Moreover, to fully utilize the existing sources for better performance, the problem of whether or how to combine the features or classifiers from both the depth maps and the color images remains unanswered.

Most existing algorithms for human action recognition focus on solving challenging issues without considering the real world application requirements. In addition, how to design a framework that is effective under variant scenarios has not been well explored yet. For example, algorithms designed for single view action recognition may not be able to solve the problem of multi-view action recognition, or feature extractions for color images may not be suitable for depth maps.

In this dissertation, we explore different characteristics of both sources, RGB images and depth maps for the task of human action recognition and present efficient feature extraction and representation algorithms. Especially, the designed algorithms

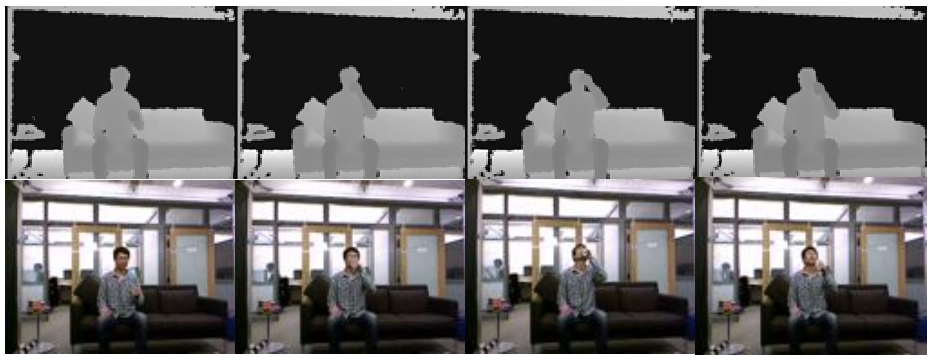


Figure 1.3: Sample images from RGB and depth cameras Wang et al. (2012b). The upper row shows the images captured by depth camera and the lower row shows the images captured by RGB camera.

take the real world application into consideration and require low computational cost to achieve state-of-the-art performance.

1.3 Contributions

In this dissertation, algorithms relate to the feature extraction and representation steps are proposed since both steps are most important to the recognition accuracy. The designed algorithms aim to characterize the intrinsic spatial-temporal properties of human actions such that high accuracy can be achieved at low computational cost. We briefly emphasize our research contributions as follows.

From the feature extraction perspective, our contribution lies in the design of two computationally efficient feature descriptors.

- *Shape and Motion Local Ternary Pattern (SMltp)*: As a feature descriptor, SMltp captures both the appearance and motion information efficiently and takes advantage of the Local Binary Pattern (LBP) Ojala et al. (2002) texture operator in terms of tolerance to illumination change, robustness in homogeneous region and computational efficiency. In addition, SMltp can be easily combined with various feature detectors to perform action recognition at low computational cost.

- *Center Sysmetric Motion Local Ternary Pattern (CS-Mltp)*: CS-Mltp is proposed to describe gradient-like characteristics of RGB sequences at both the spatial and temporal directions. Compared to the SMltp, CS-Mltp is more sensitive to noise, but better for describing motions with small body movements.

From the feature representation perspective, our contribution lies in the design of a temporal pyramid matching approach based on *sparse coding* of the extracted features to represent the temporal patterns. Especially, a discriminative class-specific dictionary learning algorithm is proposed for sparse coding. By adding the group sparsity and geometry constraints, features can be well reconstructed by the sub-dictionary belonging to the same class, and the geometry relationships among features are also kept in the calculated coefficients. Different from the BoW model that assigns each feature to its nearest "visual word", the proposed algorithm will choose a sparse and linear combination of "visual words" to approximate the representation, such that less quantization error will be introduced during feature representation step.

We further apply the proposed framework on the tasks of human action recognition under different scenarios to show the robustness of the proposed algorithms and discuss the problems of how to fuse information from different sources. Single view human action, distributed human action and RGB-D human action recognition are discussed:

- *Single view human action recognition:* In the single view recognition task, the RGB videos are used as input source and there are only one video for each action performed by a specific person. However, background can be very complicated and different among these videos. We evaluate the proposed algorithms on challenging datasets and propose a patch selection scheme that makes the average number of features 20 times smaller than dense sampling, so that the computational load for the bag-of-features representation is reduced. Parameters existing in the proposed algorithms are well evaluated and adjusted.

- *Distributed human action recognition:* In the distributed camera networks (DCNs), there are several local smart cameras surrounding the actors and simultaneously capturing the actions with different viewing angles. In addition, there is a base station connected with local cameras through wireless connection. Due to the resources limitation, communication among local cameras are limited, and information transmission between local cameras and base station should satisfy the bandwidth limitation. Since it is not practical to transmit original video sequences to the base station, lightweight feature extraction algorithms should be explored. In addition, how to fuse the information from local cameras should be designed as well. We proposed a new framework to realize action recognition in DCNs and apply the proposed descriptors to reduce bandwidth consumption and computational cost.

- *RGB-D human action recognition:* In the RGB-D human action recognition, both RGB videos and depth maps are available for processing, which requires both RGB and depth feature extractions. We first proposed a new feature extraction and representation algorithm based on the 3D skeleton tracker to solve the problem of time alignment in depth maps. The proposed dictionary learning algorithm is applied on depth feature representation and compared with the state-of-the-art work. In addition, different fusion schemes to combine both RGB and depth information for better performance are explored.

1.4 Dissertation Organization

The dissertation is organized as follows: In Chapter 2, a literature review on feature extraction and representation is provided. In Chapter 3, two feature descriptors are proposed to describe the shape and motion patterns at local structure. In addition, a key frame detection scheme is presented to detect important frames from depth sequence. In Chapter 4, a discriminative dictionary learning algorithm is presented for feature representation. In Chapter 5, the proposed descriptors are applied on the single view human action recognition. In Chapter 6, a distributed human action recognition framework is evaluated based on the proposed motion descriptor. In Chapter 7, both RGB images and depth sequence are available for RGB-D human action recognition. Conclusions and future work are discussed in Chapter 8.

Chapter 2

Literature Review

In this chapter, literature work in the field of human action recognition is explored and discussed. We start with an review on feature extraction methods, which is the first step for action recognition. Instead of working on the raw video sequences which contain many pixels (e.g. for a 100-frame sequence video, the number of pixels will be $30,720,000$ with image size as 480×640) and redundant information (e.g. the temporal variations can be very small between two consecutive frames), it is necessary to extract a set of features which are considered as more compact representation of input data. This is the process referred to as *feature extraction*. We then describes existing algorithms for feature representation from RGB videos and depth maps. Since different videos will generate different number of features, we need to generate a vector representation for the video sequence and the dimension of such vector should be the same for all the videos for classification. The process to generate a unique-length vector representation based on extracted features is *feature representation*.

2.1 Feature Extraction from RGB Videos

In the literature, feature extraction from RGB videos can be divided into two categories: global feature extraction and local feature extraction. The former category usually involves two steps: a person is localized first in the image using background subtraction or tracking. Then, the region of interest is encoded as a whole, which results in the image descriptor. The latter category usually detects the local spatio-temporal interest points first, and then local patches are calculated around these points. Local spatio-temporal features describe the observed video sequence as a collection of these detected local patches. Compared to the global feature extraction, local feature extraction is less sensitive to noise and partial occlusion, and does not strictly require background subtraction or tracking Poppe (2010). Therefore, local spatial-temporal features have been heavily used for human action recognition.

In this dissertation, we focus on the local spatio-temporal feature extraction algorithms that include feature detection and feature description.

2.1.1 Feature Detectors

Feature detection is a process to examine every pixel to see if there is a feature present at that pixel. For human action recognition, regions (patches) centered at the detected pixels can be cropped with calculated spatial and temporal scales and orientations. Algorithms used to generated features and patches can be referred to as feature detectors. Feature detectors usually select spatio-temporal locations, scales and orientations in video by maximizing specific saliency functions. There are a variety of detectors briefly reviewed in the following.

Harris3D Detector

The Harris3D detector Laptev and Lindeberg (2005) is a space-time extension of the Harris detector Harris and J. (1988) and detect local structures in space-time where the image values have significant local variations in both space and time. A spatio-temporal second-moment matrix at each video point is computed as $\mu(\cdot, \sigma, \tau) = g(\cdot, s\sigma, s\tau) * (\Delta L(\cdot, \sigma, \tau)(\Delta L(\cdot, \sigma, \tau))^T)$. Here, σ, τ are independent spatial and temporal scale values respectively, g is a separable Gaussian smoothing function, and ΔL is space-time gradients. The final locations of space-time interest points are given by local maxima of $H = det(\mu) - ktrace^3(\mu)$, $H > 0$.

Cuboid Detector

The Cuboid detector was proposed by Dollar Dollar et al. (2005) to detect the spatio-temporal interest points. Instead of using a 3D filter on the spatio-temporal domain, it applies two separate linear filters respectively to both the spatial and temporal dimensions. The response function at pixel location $I(x, y, t)$ is of the form $R = (I(x, y, t) * g_\sigma(x, y) * h_{ev}(t))^2 + (I(x, y, t) * g_\sigma(x, y) * h_{od}(t))^2$, where $g_\sigma(x, y)$ is the 2D Gaussian smoothing function that applied only in the spatial domain. The h_{ev} and h_{od} are a quadrature pair of 1D Gabor filters that applied in the temporal direction. The 1D Gabor filters are defined as $h_{ev}(t; \tau, \omega) = -\cos(2\pi t\omega)exp^{-t^2/\tau^2}$ and $h_{od}(t; \tau, \omega) = -\sin(2\pi t\omega)exp^{-t^2/\tau^2}$, where $\omega = 4/\tau$. The parameters σ and τ roughly correspond to the spatial and temporal scale. The interest points are detected at locations where the response is local maximum. The spatio-temporal patches around the points are extracted for feature description. The size of the patches is defined by the value of σ and τ.

Hessian Detector

The Hessian detector was proposed by Willems Willems et al. (2008b) as a spatio-temporal extension of the Hessian saliency measure for blob detection in images. The determinant of the 3D Hessian matrix is used to measure the saliency. Computations are speeded-up further through the use of approximative box-filter operations on an integral video structure. Since the scale invariance and good scale selection are achieved simultaneously with the scale-normalized determinant of Hessian matrix, no iterative procedure is needed.

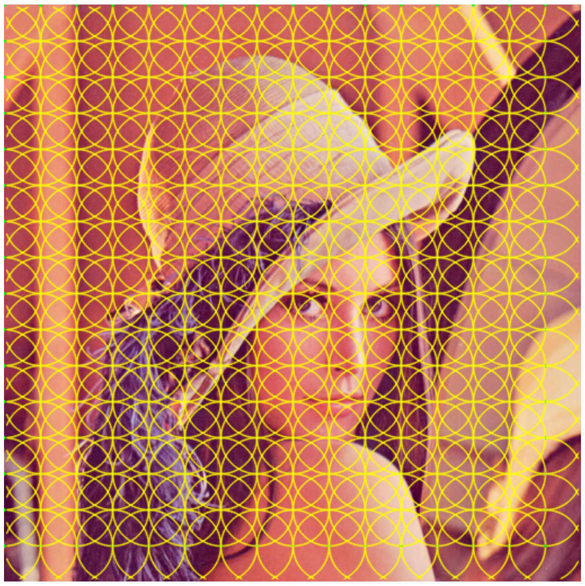


Figure 2.1: Dense sampling.

Dense Sampling

The dense sampling extracts video blocks at regular positions and scales in space and time Wang et al. (2009). There are 5 dimensions to sample from: (x, y, t, σ, τ), where σ and τ are the spatial and temporal scale, respectively. Multi-scale patches can be obtained by selecting different values of σ and τ. Figure 2.1 shows the example of dense sampling in spatial direction. Given the predefined scales corresponding to the patch sizes, overlapped regions can be defined from the entire image. Different from the other feature detectors, dense sampling cover the whole image patch and will generate large number of features.

Dense Trajectory

Dense trajectories Wang et al. (2011) are extracted for multiple spatial scales. Feature points are sampled on a grid spaced by N pixels and tracked in each scale separately. Then 8 spatial scales spaced by a factor of $\frac{1}{\sqrt{2}}$ are used to keep scale invariance. Each point $\mathbf{P}_t = (x_t, y_t)$ at frame t is tracked to the next frame $t+1$ by median filtering in a dense optical flow field $\omega = (u_t, v_t)$. Once the dense optical flow field is computed, points can be tracked very densely without additional cost. Points of subsequent frames are concatenated to form a trajectory.

2.1.2 Feature Descriptors

Feature descriptors are used to describe the detected regions in a representation that is ideally invariant to background clutter, appearance and occlusions, and possibly to rotation and scale Poppe (2010). The spatial and temporal size of a patch is usually determined by the scale of the interest point. For example, after feature detectors detected the features centered at (x, y, t) with the spatial size defined by σ and temporal size defined by τ, the corresponding 3D patch can be cropped, and the feature descriptor is used to summarize the information within that patch.

Cuboid

The Cuboid descriptor Dollar et al. (2005) is used to describe the 3D patches detected by Cuboid detector. The size of the descriptor is determined by $\Delta x(\sigma) = \Delta y(\sigma) = 2{\times}3\sigma{+}1$ and $\Delta t(\tau) = 2{\times}3\tau{+}1$. The gradients for all pixels within the extracted patch are calculated and concatenated into a vector. The principle component analysis (PCA) is applied on the vector for dimension reduction.

HOG/HOF

The HOG/HOF descriptors (Laptev and Lindeberg, 2005) characterize local appearance and motion by computing histograms of spatial gradient and optic flow accumulated in space-time neighborhoods of the detected interest points. The descriptor size is defined by $\Delta x(\sigma) = \Delta y(\tau) = 18\sigma, \Delta t(\tau) = 8\tau$. To keep the spatio-temporal relationship, each volume is subdivided into a $n_x \times n_y \times n_t$ grid of cells. For each cell, 4-bin histograms of gradient orientations (HOG) and 5-bin histograms of optic flow (HOF) are computed. Normalized histograms are concatenated into HOG, HOF as well as HOG/HOF descriptor vectors.

HOG3D

The HOG3D descriptor Scovanner et al. (2007) is based on histograms of 3D gradient orientations and can be seen as an extension of the popular SIFT descriptor Lowe (2004) to video sequences. Gradients are computed using an integral video representation. Regular polyhedrons are used to uniformly quantize the orientation of spatio-temporal gradients. Similar to the other feature descriptors, a given 3D patch is divided into $n_x \times n_y \times n_t$ cells. The corresponding descriptor concatenates gradient histograms of all cells and is then normalized.

Extend SURF

Willems et al. (Willems et al., 2008a) proposed the extended SURF (ESURF) descriptor which extends the image SURF descriptor (Bay et al., 2006) to videos. The size of extracted 3D patches is defined as $\Delta x(\sigma) = \Delta y(\sigma) = 3\sigma, \Delta t(\tau) = 3\tau$. After dividing the 3D patches into $n_x \times n_y \times n_t$ cells, a vector of weighted sums $v = (\sum dx, \sum dy, \sum dt)$ can be calculated on each cell. Note that dx, dy, dt are uniformly sampled responses of Haar-wavelets along the three axes.

Motion Boundary Histogram

The motion boundary histogram (MBH) Dalal et al. (2006) is a descriptor for human detection, where derivatives are computed separately for the horizontal and vertical components of the optical flow. The MBH descriptor separates the optical flow field $I_\omega = (I_x, I_y)$ into its x and y component. Spatial derivatives are computed for each of them and orientation information is quantized into histograms. Wang et al. Wang et al. (2011) combined the MBH descriptor and dense trajectories for human action recognition to achieve state-of-the-art performance.

Local Ternary Pattern

The Local Binary Pattern (LBP) texture operator Ojala et al. (2002), based on the relative order of neighboring pixels, has been very successful in solving various problems, such as texture classification Ojala et al. (2002), face recognition Ahonen et al. (2004); Tan and Triggs (2007) and feature description Heikkila et al. (2009); Gupta et al. (2010). Binary patterns are created for each pixel by comparing the pixel value with its neighboring pixel intensities. Thus, LBP is robust to monotonic gray-scale changes and is computationally efficient.

Local Ternary Pattern (LTP) Tan and Triggs (2007), as a variant of LBP, is a 3-valued coding that includes a threshold around zero for improved resistance to noise. Different from the LBP descriptor that assigns positive or negative pattern to each comparison, LTP includes a smooth pattern when the intensity differences are within a small region. Therefore, LTP will be more robust to noise.

Although both LBP and LTP descriptors are originally applied on 2D images, the various flavors of LBP have also been used for human action recognition. Kellokumpu et al. Kellokumpu et al. (2008) used a dynamic LBP description from two orthogonal planes: xt and yt to detect human bounding volumes and to describe human movements. The Hidden Markov Modelling was adopted to perform activity classification. However, the detection part in this algorithm is not adaptive to the background change, which limits its application in more challenging dataset. Yeffet and Wolf Yeffet and Wolf (2009) presented a local trinary pattern for human action recognition, which combined the self-similarity with LTP. It sliced the video sequence in space and time, and concatenated the histograms from different regions to get a global representation. However, there are two limitations with this algorithm: one is the long length of the extracted feature vector; the other is the absence of appearance information, which is also of great importance for human action recognition Wang et al. (2009); Klaser et al. (2008).

2.2 Feature Extraction from Depth Videos

The introduction of the low-cost RGB-D sensors (Kinect) largely benefits the practice of human action recognition by providing both depth maps and color images simultaneously. Although the issues of feature extraction and representation have been widely explored for color images, related research on depth maps has been very limited.

As for feature extraction from the depth sequence, some algorithms focus on 3D point cloud such as the bag of 3D points Li et al. (2010) and Random Occupied Patterns Wang et al. (2012a), while others take advantage of the 3D joint positions detected by the skeleton tracker Shotton et al. (2011) as robust and compact representation. For example, a view-invariant 3D joint feature Xia et al. (2012) was extracted based on the 3D joint positions. The work of Wang et al. (2012b); Yang and Tian (2012) used the relative 3D joint positions as the discriminative and robust features.

Since 3D joint features are frame-based, the different numbers of frames in each sequence requires algorithms to provide solutions for "temporal alignment". Most existing algorithms solve this problem through temporal modeling that models the temporal evolutions of different actions. For example, the Hidden Markov Model (HMM) is widely used to model the temporal evolutions Xia et al. (2012); Gu et al. (2010). The conditional random field (CRF) Han et al. (2010) predicts the motion patterns in the manifold subspace. The dynamic temporal warping (DTW) Muller and Roder (2006) tries to compute the optimal alignments of the motion templates composed by 3D joints. However, the noisy joint positions extracted by the skeleton tracker Shotton et al. (2011) may undermine the performance of these models and the limited number of training samples makes these algorithms easily suffer from the overfitting problem.

Recently, a Fourier Temporal Pyramid Wang et al. (2012b) technique is proposed to solve the temporal alignment problem. This method partitions every sequence into a pyramid structure and calculates the low-frequency Fourier coefficients for all the segments. Since each segment is actually a discrete signal with short length ($8 \sim 300$ frames), frequency calculations of such discrete short signal will be sensitive to disturbances caused by different speed of movements.

2.3 Feature Representation

Since different number of features are collected for different video sequences, a process that can generate a unique-length vector representation for each video is necessary for classification purpose. Feature representation treats each video as a collection of extracted features, and generate a vector representation for each video which can be good for classification purpose. In the literature of human action recognition, Bag-of-Words (BoW) and sparse coding (SC) are widely used feature representation schemes.

2.3.1 Bag-of-Words

The BoW model uses k-means clustering to learn the codebook and every cluster center corresponds to a "visual word". This method treats each video as a collection of features and quantize them into the nearest "visual words" in terms of the Euclidean distance, as shown in Figure 2.2.

To keep the temporal order of extracted features, existing methods based on BoW framework will use a *pyramid matching* scheme which divide the features into different segments according to spatial and temporal locations of interest points. However, BoW model will generate large quantization error into the representation since each feature is assigned to the nearest visual word only. In addition, computational cost is expensive since the BoW model performs well when combined with a particular type of nonlinear Mercer kernels, e.g. the intersection kernel or the Chi-square kernel.

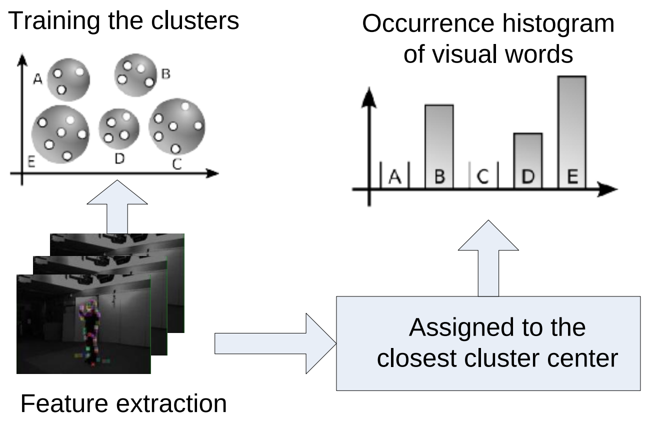

Training the clusters
A
B
E
D
C
Occurrence histogram of visual words
A
B
C
D
E
Assigned to the closest cluster center
Feature extraction

Figure 2.2: Bag-of-Words framework used for action recognition.

2.3.2 Sparse Coding

Compared to the codebook used in BoW modeling, the learned overcomplete basis set in sparse coding is referred to as the "dictionary", consisting of a set of representative vectors learned from a large number of features. These representative vectors are referred to as the "codewords" in BoW and "atoms" otherwise. Considerable amount of quantization error is incurred by the approximation process that each sample vector is assigned to the nearest codeword of BoW. This can be largely improved by sparse coding that allows a linear and sparse combination of atoms to be used in the approximation process. The calculated sparse codes for one feature corresponds to the responses of that feature to all the atoms in the dictionary. Or put it another way, the sparse codes represent the coefficients in the linear combination of atoms. Let $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]^T \in \mathbb{R}^{N \times f}$ be a set of extracted features in a f-dimensional feature space, and $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^T \in \mathbb{R}^{N \times K}$ be the sparse codes for these feature vectors. The approximation process can be represented by sparse coding (Yang et al., 2009a):

$$\begin{aligned} \min_{D,X} = \quad & \sum_{m=1}^{N} \|\mathbf{y}_m - \mathbf{D}\mathbf{x}_m\|^2 + \lambda|\mathbf{x}_m| \\ \text{subject to} \quad & \|\mathbf{d}_k\| \leq 1, \qquad \forall k = 1, 2, \ldots, K \end{aligned} \tag{2.1}$$

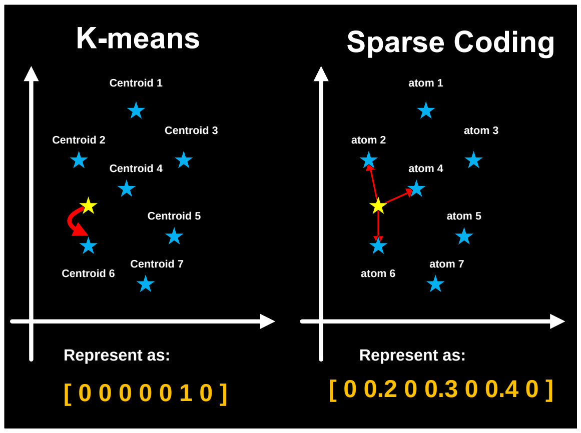


Figure 2.3: Comparison of K-means and the sparse coding for feature quantization.

where $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_K]^T \in \mathbb{R}^{K \times f}$ is the learned *overcomplete* dictionary with K atoms, and $\mathbf{x}_m$ is the calculated sparse codes for feature vector $\mathbf{y}_m$. The unite L_2-norm constraint on $\mathbf{d}_k$ is typically applied to avoid trivial solutions (Yang et al., 2009a). Note that the L_1 regularization on $\mathbf{x}_m$ enforces $\mathbf{x}_m$ to have a small number of nonzero elements. In other words, instead of assigning the nearest codeword as does in the BoW modeling, each feature vector is approximated by a linear and sparse combination of atoms in the learned dictionary.

Figure 2.3 shows the difference of the two quantization strategies: BoW and SC. Given the feature (yellow start) to be quantized, K-means will find the nearest centroid (blue stars) and thus be sensitive to the variation of the locations of the feature. For example, if we move the location of yellow star to slightly closer to centroid 4, the feature will be quantized to centroid 4 instead of centroid 6. SC method will find the sparse and linear combination of atoms (blue stars) to approximate the feature, and thus be stable to the variations and generate less approximation error. For example, if we move the location of feature slightly, it will still select the same atoms to approximate it but with a small variations in the coefficients. Compared to the BoW model, the SC will generate a representation that is more close the original representation.

Chapter 3

RGB-D Feature Extractions

3.1 Introduction

Traditional human action recognition is based on RGB videos, and spatio-temporal feature extraction algorithms Dollar et al. (2005); Klaser et al. (2008); Scovanner et al. (2007); Laptev and Lindeberg (2005); Willems et al. (2008b); Wang et al. (2011) have been widely used. Recently, the release of Kinect provides depth information (3D motion information) and makes it available to use RGB-D information for human action recognition. In this chapter, we explore new feature extraction algorithms for both RGB images and depth maps.

For the RGB videos, we propose two efficient spatio-temporal descriptors: the Shape and Motion Local Ternary Pattern (SMltp) descriptor and the Center-Symmetric Motion Local Ternary Pattern (CS-Mltp) descriptor, which can be combined with existing feature detectors efficiently for action recognition.

SMltp contains both shape (Sltp) and motion (Mltp) descriptors. For shape information, Sltp encodes each frame as a short string of ternary digits by a revised Center-Symmetric Local Ternary Pattern (CS-LTP) Gupta et al. (2010), which extracts gradient-like features along the diagonal directions. For motion information, Mltp compares each pixel at every frame with its vertical and horizontal neighboring pixels in the previous and the next frames, respectively. A ternary pattern can then be generated to describe the motion directions based on the comparisons. Given a 3D patch, SMltp describes both the shape and motion information by various local ternary patterns.

CS-Mltp is proposed to describe gradient-like characteristics of image sequences at both the spatial and temporal directions. The proposed CS-Mltp first compares the diagonal difference at each frame, and then a ternary pattern will be assigned to the difference based on the center pixel's intensity. Compared to the SMltp, CS-Mltp is more sensitive to motions since it compares the difference at the same spatial locations with different temporal information.

Compared to other feature descriptors, there are several advantages of the proposed algorithms: 1) The proposed descriptors can be combined with any feature detectors, such as Harris3D Laptev and Lindeberg (2005) and Cuboids Dollar et al. (2005), with state-of-the-art performance for action recognition. 2) Without sacrificing the recognition accuracy, the proposed descriptors are much more computationally efficient than the traditional Histogram of Gradient (HOG) and Histogram of Flow (HOF) descriptors. 3) Both descriptors can reduce the noise generated from background to some extend and capture the most important motion characteristics.

For the depth-based feature extraction, the 3D joint skeleton Shotton et al. (2011) will generate stable 3D joint positions. Compared to the original depth maps, the 3D joints information is more compact and discriminative. Therefore, 3D joints information can be used as depth features. In this chapter, we introduce a pre-processing step where a simple key frame detection method based on "joint angles" is developed to remove frames without motion information from the whole depth sequence, so that the signal-to-noise ratio is increased, providing more reliable inputs to the subsequent feature representation step.

3.2 Feature Extraction from RGB Videos

3.2.1 The SMltp Descriptor

The proposed Shape and Motion Local Ternary Pattern (SMltp) is the integration of two feature descriptors: the shape local ternary pattern (Sltp) that describes gradient-like features and the motion local ternary pattern (Mltp) that captures the effect of motion on local structure.

Shape Descriptor - Sltp

The Center-Symmetric Ternary Pattern (CS-LTP) is a variant of the Center-Symmetric Local Binary Pattern (CS-LBP) and is designed to be more robust to Gaussian noise than previous descriptors in the LBP family. CS-LTP uses ternary codes to represent the local gradient information by two intensity comparisons between diagonal pixels. Mathematically, CS-LTP at the center point (n_c) with a neighboring distance (h) and threshold (T) can be represented as Gupta et al. (2010):

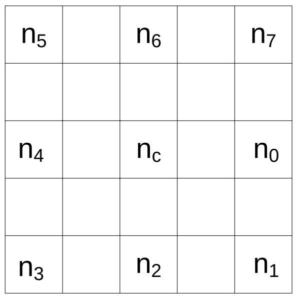
$$CS - LTP(n_c, h, T) = f(n_1 - n_5) + f(n_3 - n_7) \times 3 \tag{3.1}$$

where

$$f(x) = \begin{cases} 0 & x < -T \\ 2 & x > T \\ 1 & else \end{cases} \tag{3.2}$$

and n_1, n_3, n_5, n_7 are neighboring pixels of point n_c in the two diagonal directions, as illustrated in Fig. 3.1, where $h = 2$.

We adopt CS-LTP as the foundation for Sltp due to its success in feature description for 2D images. Compared to other LBP-like feature descriptors, CS-LTP has a shorter histogram representation with 9 bins. To make the CS-LTP suitable for the action recognition task, we make modifications in three aspects. The revised CS-LTP is referred to as the Shape Local Ternary Pattern (Sltp).



$$CS-LTP = f(n_1 - n_5)\times 1 + f(n_3 - n_7)\times 3$$

Figure 3.1: The CS-LTP operator for 8-neighborhood definition where n_c is the center pixel and n_0-n_7 are 8 neighboring pixels at distance $h = 2$.

First, Eq. 3.2 is revised as:

$$f(x) = \begin{cases} 1 & x < -\max\{T_s \times I_{n_c}, t\} \\ 2 & x > \max\{T_s \times I_{n_c}, t\} \\ 0 & else \end{cases} \tag{3.3}$$

such that the homogeneous regions would have zero Sltp values, and the threshold will be dynamically depending on the gray scale value of center point. The reason for using a dynamic threshold depending on the center pixel's intensity (I_{n_c}) is to ensure its sensitivity at dark regions. The value of t is assigned to be 5 (for any 256 gray scale image) and used for noise compression. We refer to the image generated by applying Eq. 3.1 as the texture image. Figure 3.2 shows results of using different sign functions $f(x)$ on the original image ($h = 2$, i.e. a 5×5 patch). The texture images 3.2c and 3.2d, calculated by Sltp, have intuitive representations of zero values (i.e., background), compared to the texture images 3.2b calculated by CS-LTP.

Second, Sltp uses a large threshold $T_s \times I_{n_c}$ to keep only the strongest gradient information, compared to a small and fixed value of T used in the literature. For Sltp, only the shape information is kept for further analysis, and the locations of shape outlines always possess large gradient values. The results of choosing various values of T_s are shown in Figure 3.2. The large threshold $T_s = 0.12$ produces a

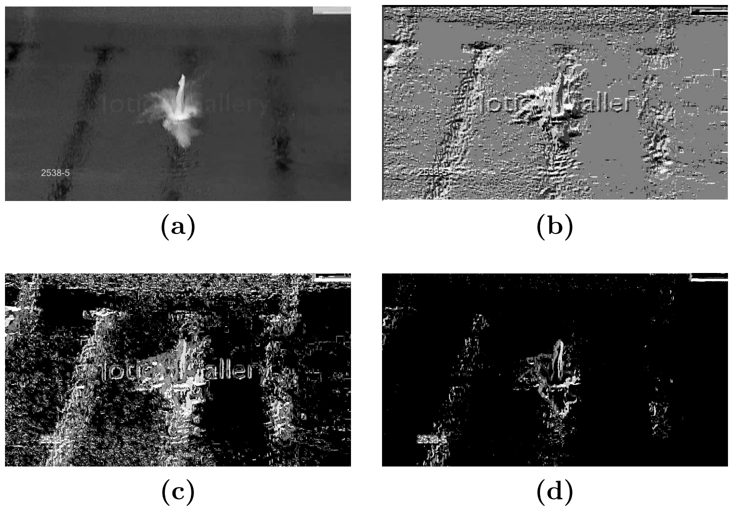
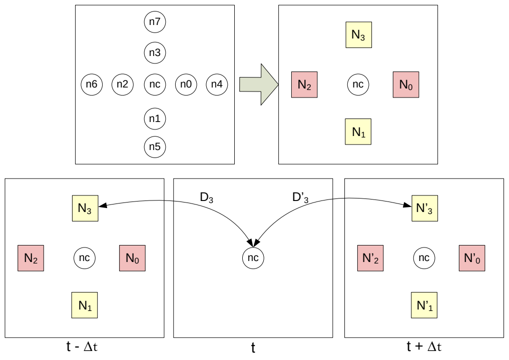


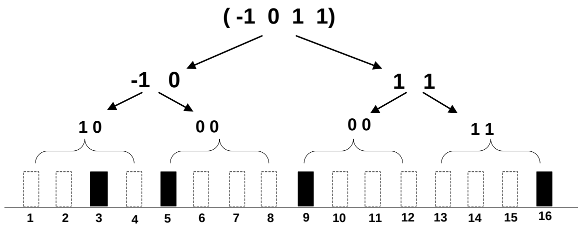
Figure 3.2: Texture images ($h = 2$). (a) The original image. (b) The texture image calculated by Eqs. 3.1 and 3.2 ($T = 3$, the same as used in Gupta et al. (2010)). (c) The texture image calculated by Eqs. 3.1 and 3.3 ($T_s = 0.03$). (d) Texture image calculated by Eq. 3.1 and 3.3 ($T_s = 0.12$).

clean image, as shown in Figure 3.2d, while the small threshold $T_s = 0.03$ or $T = 3$ introduces many noisy points which makes the foreground pixels less distinctive, as shown in Figures 3.2b and 3.2c.

Third, at the step of histogram representation of Sltp, it gives equal weight to every code generated by Eq. 3.1, and thus 9 bins in total are used as the unique feature vector. The histogram of original CS-LTP removes the bin of code (1,1) calculated by Eq. 3.2 (corresponding to the code (0,0) calculated by Eq. 3.3), claiming that this code is less reliable than the other codes Gupta et al. (2010). However, our experimental results indicate that equal weight to every code actually generates the highest accuracy. Since Sltp uses a large and dynamic threshold, the homogeneous region is less sensitive to noise and much more reliable than Gupta et al. (2010). Furthermore, by keeping the zero value bin (corresponding to code (0,0) calculated by Eq. 3.3), a patch can be defined as non-informative if the counts of this bin is high.



(a) Mltp Calculation



(b) Mltp Histogram

Figure 3.3: Encoding process of Mltp. N_i and N_i' are the calculated mean neighboring pixel values at the i-th location centered at n_c of the previous frame and the next frame, respectively. D_3 and D_3' are distances between N_i (or N_i) and the center pixel intensity of the current frame (I_{n_c}).

In order to give a short representation of extracted features, the Sltp descriptor counts the occurrence of the 9 codes within the extracted 3D patches and gets a 9-bin histogram representation. To keep the spatial and temporal information to some extend, every 3D patch detected from the video sequence is further divided into a grid of $n_\sigma \times n_\sigma \times n_\tau$ cells. For each cell, a 9-bin histogram of Sltp can be computed. The normalized histograms are then concatenated into Sltp with length of $9n_\sigma^2 n_\tau$.

Motion Descriptor-Mltp

Motion Local Ternary Pattern (Mltp) is a new feature descriptor used for capturing motion information by combining the effective description properties of LTP with the intensity similarity of the neighboring pixels among adjacent frames. For any pixel (x, y) at frame t, motion will cause the intensity change of its neighboring pixels at the previous frame $t - \Delta t$ and the next frame $t + \Delta t$. Mltp is designed to capture the effect of motion at local structure.

Details of the process of Mltp is illustrated in Figure 3.3(a). The first step of Mltp is to calculate the *gradient* information among frames in 8 directions. Given a pixel n_c, its neighboring pixels can be defined as $n_0 \sim n_3$ and $n_4 \sim n_7$, referred to as the inner and outer neighbors, receptively. To reduce the effect of noise, the mean values of the four inner and outer neighbors, $N_i = 0.55 \times I_{n_i} + 0.45 \times I_{n_{i+4}}$, $i = 0, 1, 2, 3$, are calculated and used as new intensities for the neighborhood. Note that parameters 0.55 and 0.45 are empirically selected, and various ratios between them do not affect much the recognition result. The gradient information are simply defined as $D_i = N_i - I_{n_c}$ for the previous frame $(t - \Delta t)$ and $D'_i = N'_i - I_{n_c}$ for the next frame $(t + \Delta t)$. In total, there are 8 comparisons that correspond to 8 directions. In other words, the intensity differences between the neighborhood pixels and the center pixel at frame (t) are used as the spatio-temporal gradient information. The second step of Mltp is to assign a ternary pattern to the comparison results:

$$Mltp = \sum_{i=0}^{3} f(|D_i| - |D_i|')3^i \tag{3.4}$$

$$f(x) = \begin{cases} 1 & x < -\max\{T_m \times I_{n_c}, T_0\} \\ 2 & x > \max\{T_m \times I_{n_c}, T_0\} \\ 0 & else \end{cases} \tag{3.5}$$

where T_m is the threshold and need to be selected. Similar to Eq. 3.3, the dynamic threshold is used for sensitivity purpose. The value of T_0 is assigned to be 5, the same as that in Sltp.

Each pixel within a video can be assigned its Mltp value by Eq. 3.4. To intuitively show the result of Mltp, we use the action "hand waving" as an example to apply Mltp on nearby frames as shown in Figure 3.4. For the action "hand waving", it contains two basic movements: *hands up* and *hands down* as shown in Figures 3.4(a) and 3.4(c) respectively. After getting the Mltp values for the current frame (the center images in Figure 3.4(a) and 3.4(c)), we define three motion status for illustration purpose. Here, *black* or value 0 is used to represent pixels with no motion, *gray* or value 125 is used to represent locations where motion is caused by the intensity difference between the current frame and the next frame, and *white* or value 255 is used to represent pixels where motion is caused by the difference between the current frame and the previous frame. As shown in Figure 3.4(c) and 3.4(d), the two opposite movements are distinctively described since their *gray* and *white* pixels distributed differently. Note that pixels with no motions are well captured and assigned with zero values, such as all the background pixels, as well as body parts with no contribution to the motion.

Although Mltp is designed to reflect the directions of a motion among nearby frames, it is also possible to reveal the magnitude of a motion to some extend . For example, a value $40 = 1 \times 3^0 + 1 \times 3^1 + 1 \times 3^2 + 1 \times 3^3$ calculated from Eq. 3.4, means $f(x)$ is equal to 1 at every bit. In that case, the center pixel is more similar to the four neighboring pixels ($N_0 \sim N_3$) in the previous frame, which means a motion exists between the current frame and the next frame at that pixel location. Similarly, a value $2 = 2 \times 3^0 + 0 \times 3^1 + 0 \times 3^2 + 0 \times 3^3$ indicates the center pixel is more similar to the pixel N_0' at the next frame and no motion detected from the rest bits, which reveals a slight motion already occurred.

We further construct a histogram of Mltp as a motion feature descriptor. Various values of Mltp are mapped onto bins in an unconventional manner, which reduces the number of possible bins from $3^4 = 81$ to 16. Figure 3.3(b) gives an example where code 2011 is calculated by Mltp. We first divide this four digits into two strings, 20 and 11, respectively. For each string, it can be further separated into two parts: one part indicates positions of "2" and the other part indicates positions of "1", i.e., $20 = (10, 00)$ and $11 = (00, 11)$. For each part, 4 bins are enough to uniquely represent the possible values. In total, histogram with 8 bins is used for each short string. Finally, concatenating the two histograms generated by short strings, a full-length histogram of $2 \times 8 = 16$ bins is constructed. For each code generated by Mltp, it can be represented by 4 bins on the 16-bin histogram, i.e., the bins $(3, 5, 9, 16)$ in the Figure 3.3(b).

Similar to the Sltp descriptor, the Mltp descriptor divides the extracted 3D patches into a grid of $n_\sigma \times n_\sigma \times n_\tau$ cells as well. For each cell, a 16-bin histogram of Mltp can be computed. Then for each 3D patch, a normalized histogram with length of

$16n_\sigma^2 n_\tau$ are generated as Mltp descriptor vector. Note that the SMltp descriptor is a normalized histogram with length of $25n_\sigma^2 n_\tau$ formed by concatenating the Sltp descriptor and Mltp descriptor.

Although Mltp is inspired by the work of Yeffet and Wolf Yeffet and Wolf (2009), it outperforms the previous work in the field of action recognition in three aspects. 1) The pixel-wise comparison, as compared to the patch-wise comparison in Yeffet and Wolf (2009), generates clear motion image with thin and smooth edges, making the detected/sampled motion patches distinctive even at smaller scales. 2) Mltp generates a 16-bin histogram, much shorter than the 512-bin histogram in Yeffet and Wolf (2009), and thus can be used for both local and global representations. 3) Mltp uses a dynamic threshold that ensures the capture of motion information even in dark regions. Due to these advantages, Mltp achieves much higher recognition rates on all tested datasets than Yeffet and Wolf (2009) does.

3.2.2 The CS-Mltp Descriptor

CS-Mltp is a new feature descriptor used for describing motion by combining the effective description properties of Local Binary Pattern Heikkila et al. (2009) with the intensity similarity of the neighboring pixels among adjacent frames. Figure **??** shows the details of CS-Mltp using an 8-neighbor definition. For an image patch centered at n_c, pixels $n_0 \sim n_7$ are the 8 neighbors. The radius of the neighborhood R is defined as the distance between the center pixel and n_0, i.e., 2, in Figure **??**. At frame t, the CS-Mltp feature of the center pixel n_c is calculated by two steps.

First, at each frame t, the difference of two pixels at opposite sides around the center pixel n_c is calculated as $N_i(t) = I_{n_i}(t) - I_{n_{i+L/2}}(t)$ $(i = 0, \ldots, L/2 - 1)$, where $I_{n_i}(t)$ and $I_{n_{i+L/2}}(t)$ correspond to the gray-scale values of the center-symmetric pairs of $L(L = 8)$ equally spaced pixels on a circle of radius R Heikkila et al. (2009) at frame t. This step is to obtain the gradient-like features of every frame, i.e., the gradient at four directions are evaluated. Compared to the original pixel intensity,

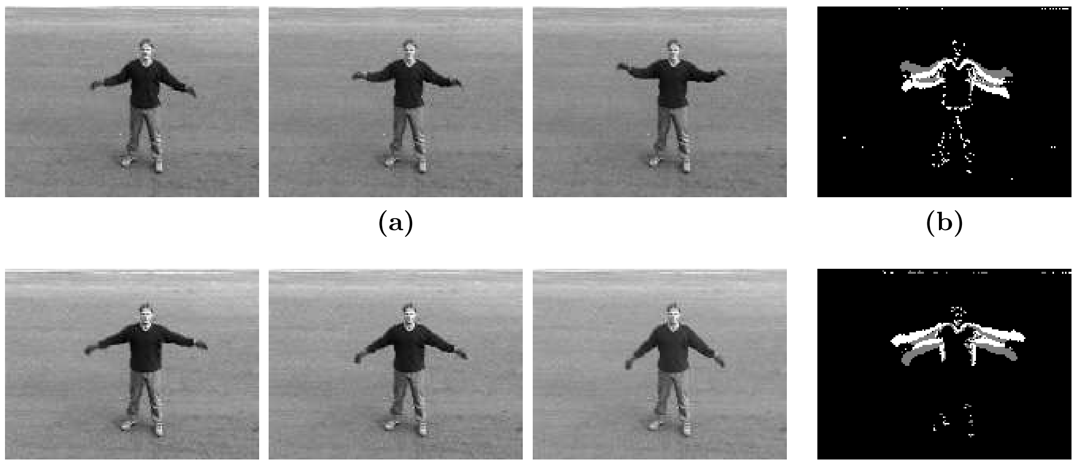


Figure 3.4: Mltp results on nearby frames from the KTH action dataset Schuldt et al. (2004). (a) Three subsequent frames from the beginning of hand waving motion. (b) Motion image calculated by Mltp on (a). The gray pixels (with intensity value 125) that have more similar intensity values as pixels in the previous frame than pixels in the next frame in (a), and thus motion is caused by the difference between the current frame and the next frame. The white pixels (with intensity value 255) indicate locations where motion is caused by the difference between the current frame and the previous frame, and black pixels indicate locations with no motion. (c) Three subsequent frames from the end of hand waving motion. (d) Motion image calculated by Mltp on (c).

the gradient-like features are more discriminative and less sensitive to illumination changes.

Second, in order to further extract the motion characteristics, CS-Mltp compares the gradient-like features at the temporal direction by local ternary pattern. The differences between the previous frame $t - \Delta t$ and the current frame t are calculated as D_is, and the differences between the next frame $t + \Delta t$ and the current frame t are calculated as D'_i. Then a ternary pattern is assigned to each comparison given by:

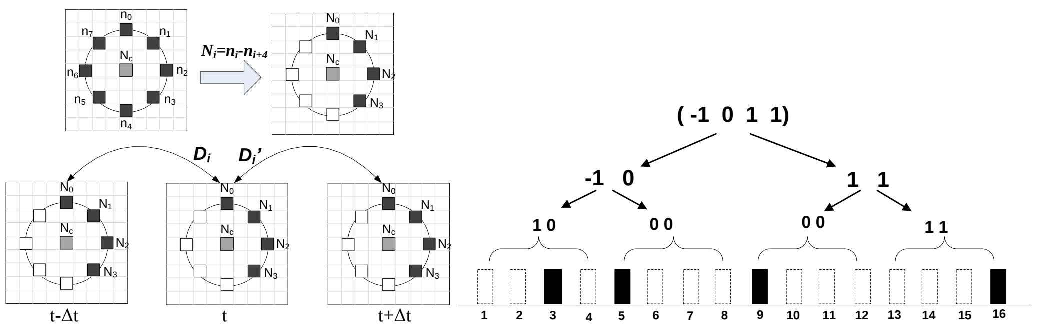
$$\begin{aligned} \text{CS-Mltp}(i) &= f(|N_i(t - \Delta t) - N_i(t)| - |N_i(t) - N_i(t + \Delta t|) \\ &= f(D_i - D'_i) \qquad \forall i = 0, \ldots, L/2 - 1 \end{aligned} \tag{3.6}$$

$$f(x) = \begin{cases} -1 & x < \min\{-I_{n_c} \times T_{cs}, -T_0\} \\ 1 & x > \max\{I_{n_c} \times T_{cs}, T_0\} \\ 0 & else \end{cases} \tag{3.7}$$

where T_{cs} is the threshold and the operator $|m|$ is to calculate the absolute value of m. Since the threshold is adaptively related to the pixel intensity, CS-Mltp is capable of describing motion information with wide intensity range. To reduce the sensitivity on dark region with small I_{n_c}, a fixed threshold T_0 is also set up. The parameter T_0 is assigned empirically to be 5 (for any 256-level gray scale images) obtained under extensive experimental study. Therefore, intensity changes larger than 5 will be considered for motion patterns and assigned as 1 or -1 by Eq. 3.7.

Similar to the SMltp descriptor, we further construct a histogram of CS-Mltp as a motion feature descriptor. Various values of CS-Mltp are mapped onto bins in an unconventional manner, which reduces the number of possible bins from $3^4 = 81$ to 16. Figure 3.5b gives an example where $(-1)011$ is calculated by Eq. 3.6. For each code generated by CS-Mltp, it can be represented by 4 bins on the 16-bin histogram, i.e., the bins $(3, 5, 9, 16)$ in Figure 3.5b.

For pixels within an extracted 3D patch from feature detection, the CS-Mltp values can be calculated according to Eq. 3.6. For any 3D patch, we then divide it



(a) CS-Mltp Calculation **(b)** CS-Mltp Histogram

Figure 3.5: The Center-Symmetric Motion Local Ternary Pattern. The first step of CS-Mltp is to calculate the center symmetric intensity variations at four directions within local region for every frame. The second step of CS-Mltp is to describe the motion information among frames by local ternary patterns.

into $n_\sigma \times n_\sigma \times n_\tau$ cells, each of which can be represented by a 16-bin histogram by CS-Mltp. We then concatenate the normalized histograms into a feature vector at the length of $16n_\sigma^2 n_\tau$. This histogram representation is referred to as the *CS-Mltp feature*.

3.3 Feature Extraction from Depth Maps

Given a depth image, 20 joints of the human body can be tracked by the skeleton tracker Shotton et al. (2011). The names of the 20 joints are marked in Figure 3.6a. Samples of depth maps from the MSR-Action 3D dataset Li et al. (2010) and their corresponding skeletons are shown in Figure 3.7. Each joint is marked as a red dot, and the skeleton is formed by blue lines that connect two joints. At frame t, the position of each joint i is uniquely defined by three coordinates $\mathbf{p}_i(t) = (x_i(t), y_i(t), z_i(t))$ and can be represented as a 3-element vector.

Compared to the original depth maps, the 3D joint positions are much more compact since each frame can be represented by the 20 joints while originally there are large number of pixels. However, the 3D joint positions are not as stable as the relative joint positions Wang et al. (2012b). Instead of using the positions of joints

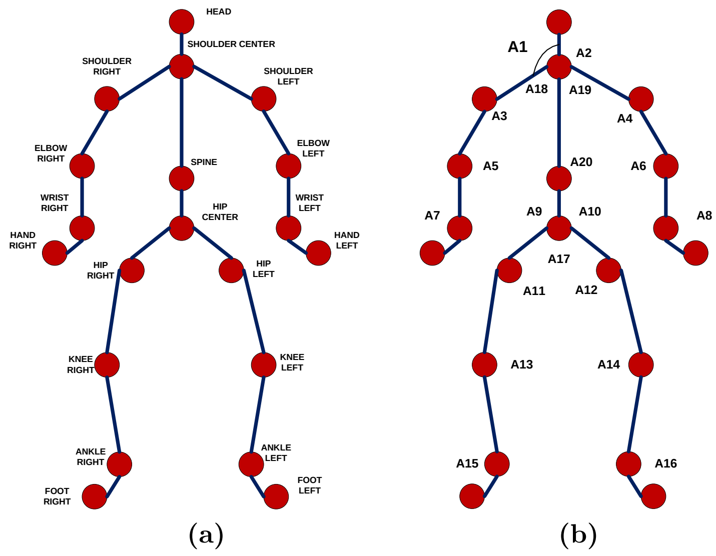


Figure 3.6: Locations and names of the 20 joints and angles formed by concatenated joints. (a) The names of the 20 joints used in the skeleton tracker Shotton et al. (2011). (b) The joint angles (from $A_1 \sim A_{20}$) used for key frames estimation where three joints form one angle, e.g., angle A_1 is formed by joints "head", "shoulder center", and "shoulder right".

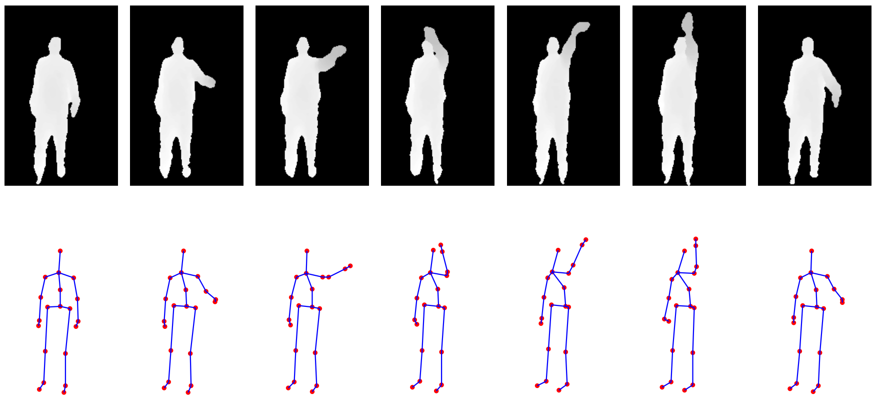


Figure 3.7: Depth maps with 20 joints. Locations of joints are marked as red dots. Connected joints are marked by blue lines.

directly, we adopt the pairwise relative position Wang et al. (2012b); Yang and Tian (2012) as more discriminative and intuitive 3D joint features. At frame t, the 3D joint feature for joint $i, (i = 1, \ldots, 20)$ is defined as:

$$\mathbf{P}_i(t) = \{\mathbf{p}_i(t) - \mathbf{p}_j(t) | \forall j \neq i\} \tag{3.8}$$

Note that both $\mathbf{P}_i$ and $\mathbf{p}_i$ are functions of time, and $\mathbf{P}_i$ is a 57-element vector. For any depth map sequence, there will be 20 joint features from $\mathbf{P}_1$ to $\mathbf{P}_{20}$.

In reality, the recording time of cameras usually last longer than the duration of actions. In other words, frames captured at the beginning or end usually contain almost no motion information. Figure 3.8 shows the extracted 20 joints for a sequence of action "draw tick". In this figure, the first 8 frames together with the last 5 frames contain no obvious motion postures and contribute nothing to the representation of action "draw tick". Yet, these frames cover more than 1/3 of the sequence length, and thus the significance of frames with representative postures is weakened. Or put it another way, the signal-to-noise ratio of the entire sequence is low since the idle frames do not carry any information (i.e., signal).

We propose a simple key frame detection method as a pre-processing step that takes advantage of the so-called "joint angles" (angles formed by concatenated joints) to remove frames with limited motion postures. Here the key frames are defined as frames containing important postures that relate to the representation of actions. Locations of the 20 joint angles are shown in Figure 3.6b. For example, A_1 is the angle formed by three joints: "head", "shoulder center", and "shoulder left". Taking the first frame as a reference, degrees of body movements for the subsequent frames can be measured by the angle difference. For the angle formed by joints i, j, and k, its cosine value can be determined as:

$$\cos(A) = \frac{\mathbf{p}_{ij} \cdot \mathbf{p}_{jk}}{\|\mathbf{p}_{ij}\| \|\mathbf{p}_{jk}\|} \tag{3.9}$$

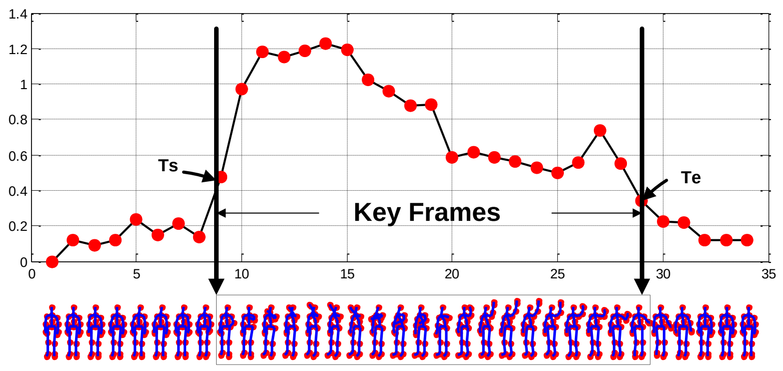


Figure 3.8: Key frame extraction by setting up the thresholds. For every red dot, its x value shows the current frame and the y value shows the calculated L_2 norm value. Key frames that contain large motion information are detected.

where $\mathbf{p}_{mn} = \mathbf{p}_m - \mathbf{p}_n$ $(m, n = 1, \ldots, 20)$. At each frame t, 20 cosine values corresponding to the 20 angles can be calculated by Eq. 3.9 and form a vector as $\mathbf{A}(t) = [\cos(A_1(t)), \ldots, \cos(A_{20}(t))]$. By calculating the L_2 norm of $A(t) - A(1)$, the motion variation between the current frame t and the reference frame can be evaluated. Note that the first frame is always considered as the reference frame. Detected frames with relatively large motion variations are considered as key frames. For a sequence with T frames in total, the set of key frames from T_s to T_e is calculated by two predefined thresholds t_s and t_e, as illustrated in Eq. 3.10. An example of the key frames extraction is shown in Figure 3.8, which plots the values of the calculated L_2 norm of $A(t) - A(1)$ at every frame. The bigger the value, the larger the body movements in that frame. By removing the frames with low values of the L_2 norm, the key frames can be determined.

$$\begin{aligned} T_s &= \min\{t\} \quad \forall t \in (1, T] : \|A(t) - A(1)\|_2 > t_s \\ T_e &= T - \max\{t\} \quad \forall t \in [0, T_s) : \|A(T-t) - A(1)\|_2 < t_e \end{aligned} \tag{3.10}$$

Feature extraction based on key frames instead of the whole sequence has exhibited some benefits. First, extracted features are more discriminative and intuitive since idle frames that present no useful information are removed. Second, extracted features are less sensitive to the time misalignment. For example, time shift of 7 frames of the sequence in Figure 3.8 would not influence the result no matter how the feature is represented. Finally, storage space of features is much reduced. Hereinafter, the extracted relative 3D joint positions by Eq. 3.8 on detected key frames are referred to as the *3D joint features*.

Chapter 4

Feature Representation for Depth Maps

4.1 Introduction

For the RGB videos, each frame contain many texture information and spatio-temporal feature detectors and descriptors can be well applied. In addition, due to the large number of features, the classic bag-of-words representation is widely used for feature representation. Different from the RGB sequences, there are no texture information in depth maps which record the depth information only. Therefore, local spatio-temporal feature extraction algorithms are not suitable for depth maps. As described in Chapter 3, the 3D joint skeletons Wang et al. (2012b) are frequently used as features. Due to the possible noise, the extracted 3D joint positions are not stable, which require a strong representation scheme for better performance. Although the Bag-of-Words representation based on K-means clustering can serve the purpose, it discards all the temporal information and large vector quantization error can be introduced by assigning each 3D joint feature to its nearest "visual word".

Recently, Yang et al. Yang et al. (2009a) showed that classification accuracies benefit from generalizing vector quantization to sparse coding. However, discrimination of the representation can be compromised due to the possible randomly distributed coefficients solved by sparse coding Fang et al. (2012). Therefore, it is necessary to design new sparse coding algorithms which can increase the discrimination power of the feature representation so that good classification accuracy can be expected. In this chapter, we propose a new *Dictionary Learning* (DL) method which aims to learn a discriminative dictionary for feature representation. The proposed DL method aims to learn an overcomplete set of representative vectors (atoms) so that any input feature can be approximated by linear combination of these atoms. The coefficients for the linear combination are referred to as the "sparse codes".

Recent trend on sparse coding is to develop "discriminative" dictionaries to solve classification problems. For example, Zhang and Li Zhang and Li (2010) proposed a discriminative K-SVD method by incorporating classification error into the objective function and learned the classifier together with the dictionary. Jiang et al. Jiang et al. (2011) further increased the discrimination by adding a label consistent term. Yang et al. Yang et al. (2011) proposed to add the Fisher discrimination criterion into the dictionary learning. For these methods, labels of inputs should be known before training. However, this requirement cannot be satisfied in our problem. Since different actions contain shared local features, assigning labels to these local features would not be proper.

The process that assigns each feature with coefficients according to a learned dictionary can be defined as "quantization", following the similar definition in the field of image classification. As shown in Figure 4.1, different quantization methods will generate different representations. Atoms from two classes are marked as "circles" (class A) and "triangles" (class B), respectively. We use two similar features to be quantized (both from class A) as an example to illustrate the coefficient distribution from various quantization methods. In k-means, features are assigned to the nearest atoms, which is sensitive to the variations of features. In the sparse coding with l_1 norm, features are assigned to atoms with lowest reconstruction error, but the distributions of selected atoms can be random and from different classes Fang et al.

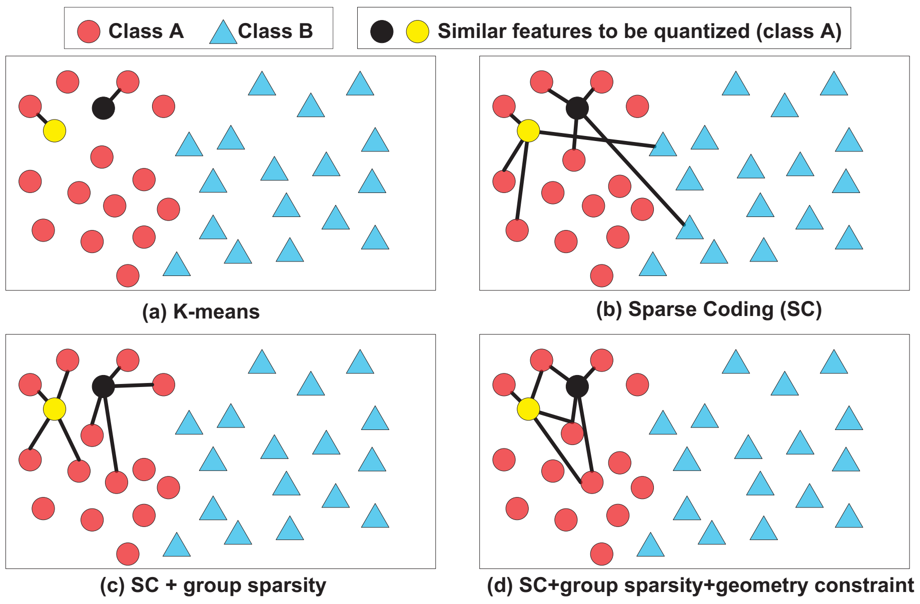

Class A
Class B
Similar features to be quantized (class A)
(a) K-means
(b) Sparse Coding (SC)
(c) SC + group sparsity
(d) SC+group sparsity+geometry constraint

Figure 4.1: Illustration of different feature quantization strategies. (a) K-means. (b) Sparse coding. (c) Sparse coding with group sparsity constraint. (d) Proposed method (sparse coding with group sparsity and geometry constraint).

(2012). In the spare coding with group sparsity, features will choose atoms from the same group (class), but similar features may not choose the same atoms within the group. In our method, features from the same class will be forced to choose atoms within the same group, and the selections of atoms also relate to the similarity of features.

Instead of simultaneously learning one overcomplete dictionary for all classes, we learn class-specific sub-dictionaries to increase the discrimination. In addition, the $l_{1,2}$**-mixed norm** and **geometry constraint** are added to the learning process to further increase the discriminative power. Existing class-specific dictionary learning methods Ramirez et al. (2010); Kong and Wang (2012) are based on l_1 norm which may result in randomly distributed coefficients Fang et al. (2012). In this chapter, we add the group sparsity regularizer Zou and Hastie (2005), which is a combination of l_1- and l_2- norms to ensure features are well reconstructed by atoms from the same class. Moreover, the geometry relationship among local features are incorporated during the process of dictionary learning, so that features from the same class with high similarity will be forced to have similar coefficients.

4.2 Background

As discussed in Chapter 2, given a dataset $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]$, sparse coding is a process to solve the optimization problem as:

$$\min_{\mathbf{D},\mathbf{X}} \left\{ \sum_{i=1}^{N} \|\mathbf{y}_i - \mathbf{D}\mathbf{x}_i\|_2^2 + \lambda |\mathbf{x}_i|_1 \right\} \tag{4.1}$$

where matrix $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_K]$ is the dictionary with K atoms and elements in matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ are coefficients. Different from the K-means clustering that assigns every data with its nearest cluster center, sparse coding uses a linear combination of atoms in the dictionary $\mathbf{D}$ to reconstruct the data, and only a sparse number of atoms have nonzero coefficients.

To increase the discriminative power of dictionary, class-specific dictionary learning methods have been proposed that learn a sub-dictionary for each class Ramirez et al. (2010); Kong and Wang (2012). For example, Eq. 4.1 can be rewritten as:

$$\min_{\mathbf{D},\mathbf{X}} \sum_{i=1}^{C} \left\{ \|\mathbf{Y}_i - \mathbf{D}_i\mathbf{X}_i\|_F^2 + \lambda \sum_{j=1}^{N_i} |\mathbf{x}_j^i|_1 \right\} \tag{4.2}$$

where $\mathbf{Y}_i = [\mathbf{y}_1^i, \ldots, \mathbf{y}_{N_i}^i]$ and $\mathbf{X}_i = [\mathbf{x}_1^i, \ldots, \mathbf{x}_{N_i}^i]$ are the dataset and coefficients for class i, respectively. Matrix $\mathbf{D}_i$ is the learned sub-dictionary for class i.

Since the sub-dictionaries are trained independently, it is possible that related atoms among those sub-dictionaries are generated. In this case, the sparse representation will be sensitive to the variations among features. Even though an incoherence promoting term $\sum_{i \neq j} \|\mathbf{D}_i^T \mathbf{D}_j\|_F^2$ can be added to the dictionary learning, correlated atoms still exist Ramirez et al. (2010).

4.3 Group Sparsity and Geometry Constrained Dictionary Learning

The process that generates a vector representation for any depth sequence with a specific number of extracted 3D joint features is referred to as "feature representation". In this chapter, a class specific dictionary learning method based on group sparsity and geometry constraint is proposed, referred to as **DL-GSGC**.

Group sparsity encourages the sparse coefficients in the same group to be zero or nonzero simultaneously Zou and Hastie (2005); Fang et al. (2012); Bondell and Reich (2008). Adding the group sparsity constraint to the class-specific dictionary learning has three advantages. First, the intra-class variations among features can be compressed since features from the same class tend to select atoms within the same group (sub-dictionary). Second, influence of correlated atoms from different sub-dictionaries can be compromised since their coefficients will tend to be zero or nonzero simultaneously. Third, possible randomness in coefficients distribution can be removed since coefficients have group-clustered sparse characteristics. For the proposed DL algorithm, the **Elastic net regularizer** Zou and Hastie (2005) is added as the group sparsity constraint since it has automatic group effect. The Elastic net regularizer is a combination of the l_1- and l_2 norms. Specifically, the l_1 penalty promotes sparsity, while the l_2 norm encourages the grouping effect Zou and Hastie (2005).

Given a learned dictionary that consists of all the sub-dictionaries and an input feature from class i, it is ideal to use atoms from the i-th class to reconstruct it. In addition, similar features should have similar coefficients. Inspired by the work of Gao et al. Gao et al. (2010), we propose to add geometry constraint to the class-specific dictionary learning process.

Let $\mathbf{Y} = [\mathbf{Y}_1, \ldots, \mathbf{Y}_C]$ be the dataset with N features for C classes, where $\mathbf{Y}_i \in \mathbb{R}^{f \times N_i}$ is the f-dimensional dataset from class i. **DL-GSGC** is designed to learn a discriminative dictionary $\mathbf{D} = [\mathbf{D}_1, \ldots, \mathbf{D}_C]$ with K atoms in total ($K = \sum_{i=1}^{C} K_i$), where $\mathbf{D}_i \in \mathbb{R}^{f \times K_i}$ is the class-specified sub-dictionary associated with class i. The objective function of DL-GSGC is:

$$\min_{\mathbf{D},\mathbf{X}} \left\{ \begin{array}{l} \sum_{i=1}^{C} \{\|\mathbf{Y}_i - \mathbf{D}\mathbf{X}_i\|_F^2 + \|\mathbf{Y}_i - \mathbf{D}_{\in i}\mathbf{X}_i\|_F^2 + \\ \|\mathbf{D}_{\notin i}\mathbf{X}_i\|_F^2 + \lambda_1 \sum_{j=1}^{N_i} |\mathbf{x}_j^i|_1 + \lambda_2 \|\mathbf{X}_i\|_F^2\} \\ + \lambda_3 \sum_{i=1}^{J} \sum_{j=1}^{N} \|\alpha_i - \mathbf{x}_j\|_2^2 w_{ij} \end{array} \right\} \tag{4.3}$$
$$\text{subject to} \quad \|\mathbf{d}_k\|_2^2 = 1, \qquad \forall k = 1, 2, \ldots, K$$

where $\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_C]$ represents the coefficients matrix and coefficients vector for the j-th feature in class i is $\mathbf{x}_j^i$. The value $\mathbf{D}_{\in i}$ is set to be $[\mathbf{0}, \ldots, \mathbf{D}_i, \ldots, \mathbf{0}]$ with K columns and the value $\mathbf{D}_{\notin i}$ is calculated as $\mathbf{D} - \mathbf{D}_{\in i}$. Term $\|\mathbf{Y}_i - \mathbf{D}\mathbf{X}_i\|_F^2$ represents the minimization of reconstruction error using dictionary $\mathbf{D}$. The terms $\|\mathbf{Y}_i - \mathbf{D}_{\in i}\mathbf{X}_i\|_F^2$ and $\|\mathbf{D}_{\notin i}\mathbf{X}_i\|_F^2$ are added to ensure that features from class i can be well reconstructed by atoms in the sub-dictionary $\mathbf{D}_i$ but not by other atoms belonging to different classes.

The group sparsity constraint is represented as $\lambda_1 |\mathbf{x}_j^i|_1 + \lambda_2 \|\mathbf{x}_j^i\|_2^2$, and the geometry constraint is represented as $\lambda_3 \sum_{i=1}^{J} \sum_{j=1}^{N} \|\alpha_i - \mathbf{x}_j\|_2^2 w_{ij}$. In the geometry constraint, elements in vector α_i are calculated coefficients for "template" feature $\mathbf{y}_i$. Here, templates are small sets of features randomly selected from all classes. In total, there are J templates used for similarity measure. Especially, coefficients α_i for the template $\mathbf{y}_i$ belonging to class m can be calculated by Eqs. 4.4 and 4.5:

$$\beta = \min_{\beta} \|\mathbf{y}_i - \mathbf{D}_m \beta\|_2^2 + \lambda_1 |\beta|_1 + \lambda_2 \|\beta\|_2^2 \tag{4.4}$$

$$\alpha_i = [\underbrace{\mathbf{0}}_{K_1}, \ldots, \underbrace{\mathbf{0}}_{K_{m-1}}, \beta, \underbrace{\mathbf{0}}_{K_{m+1}}, \ldots, \underbrace{\mathbf{0}}_{K_C}] \tag{4.5}$$

In α_i, only coefficients corresponding to the atoms from the same class m are nonzero. The weight w_{ij} between the query feature $\mathbf{y}_j$ and template feature $\mathbf{y}_i$ is defined as:

$$w_{ij} = exp(-\|\mathbf{y}_i - \mathbf{y}_j)\|_2^2/\sigma) \tag{4.6}$$

4.3.1 Optimization Step - Coefficients

The optimization problem in Eq. 4.3 can be iteratively solved by optimizing over $\mathbf{D}$ or $\mathbf{X}$ while fixing the other.

The Eq. 4.3 can be rewritten as:

$$\min_{\mathbf{D},X} \left\{ \begin{array}{l} \sum_{i=1}^{C} \{ \left\| \begin{array}{l} \mathbf{Y}_i - \mathbf{D}\mathbf{X}_i \\ \mathbf{Y}_i - \mathbf{D}_{\in i}\mathbf{X}_i \\ \mathbf{D}_{\notin i}\mathbf{X}_i \end{array} \right\|_F^2 + \lambda_1 \sum_{j=1}^{N_i} |\mathbf{x}_j^i|_1 \\ + \lambda_2 \|\mathbf{X}_i\|_F^2 \} + \lambda_3 \sum_{i=1}^{J} \sum_{j=1}^{N} \|\boldsymbol{\alpha}_i - \mathbf{x}_j\|_2^2 w_{ij} \end{array} \right\} \tag{4.7}$$

After fixing the dictionary $\mathbf{D}$, the coefficients for the j-th feature in class i, $\mathbf{x}_j^i$, can be calculated by solving the following convex problem:

$$\min_{x_j^i} \left\{ \begin{array}{l} \left\| \begin{array}{l} \mathbf{y}_j^i - \mathbf{D}\mathbf{x}_j^i \\ \mathbf{y}_j^i - \mathbf{D}_{\in i}\mathbf{x}_j^i \\ \mathbf{0} - \mathbf{D}_{\notin i}\mathbf{x}_j^i \\ \mathbf{0} - \sqrt{\lambda_2}\mathbf{I}\mathbf{x}_j^i \end{array} \right\|_2^2 + \lambda_1 |\mathbf{x}_j^i|_1 \\ + \lambda_3 \sum_{m=1}^{J} \|\boldsymbol{\alpha}_m - \mathbf{x}_j^i\|_2^2 w_{mj} \end{array} \right\} \tag{4.8}$$

where $\mathbf{I} \in \mathbb{R}^{K \times K}$ is an identity matrix. To remove the influence of shared features among classes, we use templates belonging to the same class as the input feature for similarity measure at this stage. Therefore, the objective function becomes:

$$\min_{\mathbf{x}_j^i} \left\{ \left\| \mathbf{s}_j^i - \mathbf{D}_i' \mathbf{x}_j^i \right\|_2^2 + \lambda_1 |\mathbf{x}_j^i|_1 + \lambda_3 L(\mathbf{x}_j^i) \right\} \tag{4.9}$$

where

$$\mathbf{s}_j^i = [\mathbf{y}_j^i; \mathbf{y}_j^i; \underbrace{0; \ldots; 0}_{f+K}] \tag{4.10}$$

$$\mathbf{D}_i' = [\mathbf{D}; \mathbf{D}_{\in i}; \mathbf{D}_{\notin i}; \sqrt{\lambda_2}\mathbf{I}] \tag{4.11}$$

$$L(\mathbf{x}_j^i) = \sum_{m=1}^{A_i} \|\boldsymbol{\alpha}_m - \mathbf{x}_j^i\|_2^2 w_{mj} \tag{4.12}$$

According to Eqs. 4.5 and 4.6, we know that term $L(\mathbf{x}_j^i)$ encourages the calculated coefficients to have zeros at atoms not from the same class as the input feature. In total, there are A_i templates used to calculate the unknown coefficient $\mathbf{x}_j^i$.

Since the analytical solution can be calculated for Eq. 4.9 if the sign of each element in $\mathbf{x}_j^i$ is known, the *feature-sign search method* Lee et al. (2007) can be used to obtain the coefficients. However, the augmented matrix $\mathbf{D}_i'$ needs to be normalized before using the feature-sign search method. Let $\overline{\mathbf{D}_i'}$ be the l_2 column-wise normalized version of $\mathbf{D}_i'$. By simple derivations, we know that $\mathbf{D}_i' = (\sqrt{2+\lambda_2})\overline{\mathbf{D}_i'}$. Therefore, Eq. 4.9 can be rewritten as:

$$\min_{\overline{\mathbf{x}}_j^i} \left\{ \begin{array}{l} \left\| \mathbf{s}_j^i - \overline{\mathbf{D}_i'} \overline{\mathbf{x}}_j^i \right\|_2^2 + \dfrac{\lambda_1}{\sqrt{2+\lambda_2}} |\overline{\mathbf{x}}_j^i|_1 + \\ \dfrac{\lambda_3}{2+\lambda_2} \displaystyle\sum_{m=1}^{A_i} \|\sqrt{2+\lambda_2}\alpha_m - \overline{\mathbf{x}}_j^i\|_2^2 w_{mj} \end{array} \right\} \tag{4.13}$$

where $\overline{\mathbf{x}}_j^i = \sqrt{2+\lambda_2}\mathbf{x}_j^i$. Therefore, the feature-sign search method can be applied to Eq. 4.13 to obtain $\overline{\mathbf{x}}_j^i$, and the coefficients for input feature $\mathbf{y}_j^i$ should be $\frac{1}{\sqrt{2+\lambda_2}}\overline{\mathbf{x}}_j^i$.

To simplify notation, we present the following equivalent optimization problem:

$$\begin{aligned} &\min_{\mathbf{x}} f(x) \equiv h(\mathbf{x}) + \lambda|\mathbf{x}|_1 \\ &h(\mathbf{x}) = \|\mathbf{s} - \mathbf{B}\mathbf{x}\|_2^2 + \gamma \sum_{m=1}^{A_i} \|\boldsymbol{\beta}_m - \mathbf{x}\|_2^2 w_m \end{aligned} \tag{4.14}$$

where $\mathbf{B} = \overline{\mathbf{D}'_i}$, $\lambda = \frac{\lambda_1}{\sqrt{2+\lambda_2}}$, $\gamma = \frac{\lambda_3}{2+\lambda_2}$ and $\boldsymbol{\beta}_m = \sqrt{2+\lambda_2}\boldsymbol{\alpha}_m$. Then the first derivative of $h(\mathbf{x})$ over $\mathbf{x}$ can be represented as:

$$\begin{aligned} \nabla h(\mathbf{x}) = &- 2\mathbf{B}^T\mathbf{s} - 2\gamma \sum_{m=1}^{A_i} w_m \boldsymbol{\beta}_m \\ &+ 2(\mathbf{B}^T\mathbf{B} + \gamma \sum_{m=1}^{A_i} w_m \mathbf{I})\mathbf{x} \end{aligned} \tag{4.15}$$

The feature-sign search algorithm Lee et al. (2007) is designed to search the "signs" of the coefficients x_i, and then the results can be obtained by solving unconstrained quadratic optimization problem (QP). In feature-sign search, an "active set" containing potentially nonzero coefficients and their signs is maintained and the solution is updated using an efficient discrete line search.

Details of coefficients calculation using the feature-sign search method Lee et al. (2007) are provided in Algorithm 1.

According to the Algorithm 1, solutions for Eq. 4.13 can be obtained as $\overline{\mathbf{x}}_j^i$. The coefficients should be $\mathbf{x}_j^i = \frac{1}{\sqrt{2+\lambda_2}}\overline{\mathbf{x}}_j^i$.

4.3.2 Optimization Step - Dictionary

Fixing the coefficients, atoms in the dictionary can be updated. In this chapter, the sub-dictionaries are updated class by class. In other words, while updating the sub-dictionary $\mathbf{D}_i$, all the other sub-dictionaries will be fixed. Terms that are independent of the current sub-dictionary can then be omitted from optimization, and the objective

Algorithm 1 Coefficients Calculation

```
Input: $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{A_i}]$, $\mathbf{W} = [w_1, \ldots, w_{A_i}]$, $\lambda$, $\gamma$, $\mathbf{B}$, $\mathbf{s}$, and $h(\mathbf{x})$
Output: $\mathbf{x}^T = [x_1, \ldots, x_K]$
 1: Step 1: Initialization
 2: $\mathbf{x} = \mathbf{0}$, $\boldsymbol{\theta} = \mathbf{0}$, and active set $\mathcal{H} = \{\}$, where $\theta_i = \{1, 0, -1\}$ denotes $sign(x_i)$.
 3: Setp 2: Active Set Update
 4: From zero coefficients of $\mathbf{x}$, choose the one with the maximum absolute value of the first derivative: $i = \arg\max_i |\nabla h(x_i)|$
 5: if $\nabla h(x_i) > \lambda$ then
 6:     set $\theta_i = -1$, $\mathcal{H} = \{i\} \cup \mathcal{H}$
 7: end if
 8: if $\nabla h(x_i) < -\lambda$ then
 9:     set $\theta_i = 1$, $\mathcal{H} = \{i\} \cup \mathcal{H}$
10: end if
11: Setp 3: Feature-sign step
12: Let $\hat{\mathbf{B}}$ be a submatrix of $\mathbf{B}$ that contains only the columns corresponding to the active set $\mathcal{H}$. Let $\hat{\mathbf{x}}$, $\hat{\boldsymbol{\beta}}_m$, and $\hat{\theta}$ be subvectors of $\mathbf{x}$, $\boldsymbol{\beta}_m$, and $\boldsymbol{\theta}$ corresponding to the active set $\mathcal{H}$.
13: Compute the analytical solution according to Eqs. 4.14 and 4.15: $\hat{\mathbf{x}}^{new} = (\hat{\mathbf{B}}^T\hat{\mathbf{B}} + \gamma \sum_{m=1}^{A_i} w_m \mathbf{I})^{-1}(\hat{\mathbf{B}}^T\mathbf{s} + \gamma \sum_{m=1}^{A_i} w_m \hat{\boldsymbol{\beta}}_m - \frac{\lambda\hat{\theta}}{2})$
14: Perform a discrete line search on the closed line segment from $\hat{\mathbf{x}}$ to $\hat{\mathbf{x}}^{new}$: check the objective value at $\hat{\mathbf{x}}^{new}$ and all points where any coefficients changes sign; and then update $\hat{\mathbf{x}}$ to the point with the lowest objective value.
15: Remove zero coefficients of $\hat{\mathbf{x}}$ from the active set $\mathcal{H}$ and update $\boldsymbol{\theta} = sign(\mathbf{x})$.
16: Step 4: Check the Optimality Conditions
17: if $\nabla h(x_j) + \lambda sign(x_j) = 0, \forall x_j \neq 0$ then
18:     if $|\nabla h(x_j)| \leq \lambda, \forall x_j = 0$ then
19:         Return $\mathbf{x}$ as solution
20:     else
21:         Go to Step 2
22:     end if
23: else
24:     Go to Step 3
25: end if
```

function when updating the sub-dictionary $\mathbf{D}_i$ can be given as:

$$\min_{\mathbf{D}_i} \left\{ \|\mathbf{Y}_i - \mathbf{D}\mathbf{X}_i\|_F^2 + \|\mathbf{Y}_i - \mathbf{D}_{\in i}\mathbf{X}_i\|_F^2 \right\} \tag{4.16}$$

To solve Eq. 4.16, atoms in the sub-dictionary $\mathbf{D}_i$ are updated one by one. Let $\mathbf{d}_k^i$ be the k-th atom in the sub-dictionary $\mathbf{D}_i$. When updating atom $\mathbf{d}_k^i$, all the rest atoms in $\mathbf{D}$ are fixed, and the first derivative of Eq. 4.16 over $\mathbf{d}_k^i$ can be represented as:

$$\nabla(f(\mathbf{d}_k^i)) = (-4\mathbf{Y}_i + 2\mathbf{M}\mathbf{X}_i + 4\mathbf{d}_k^i\mathbf{x}_{i(k)})\mathbf{x}_{i(k)}^T \tag{4.17}$$

where $\mathbf{x}_{i(k)}$ is the r-th row ($r = \sum_{j=1}^{i-1} K_j + k$) in matrix $\mathbf{X}_i \in \mathbb{R}^{K \times N_i}$, and it is corresponding to the coefficients contributed by the atom $\mathbf{d}_k^i$. Matrix $\mathbf{M}$ is of the same size as $\mathbf{D}$ and is equal to $\mathbf{M}_1 + \mathbf{M}_2$. Here $\mathbf{M}_1$ is the matrix after replacing the r-th column in $\mathbf{D}$ with zeros, and $\mathbf{M}_2$ is the matrix after replacing the r-th column with zeros in $\mathbf{D}_{\in i}$. The updated atom $\mathbf{d}_k^i$ can be calculated by setting Eq. 4.17 to zero, which is:

$$\mathbf{d}_k^i = (\mathbf{Y}_i - 0.5\mathbf{M}\mathbf{X}_i)\,\mathbf{x}_{i(k)}^T / \|\mathbf{x}_{i(k)}\|_2^2 \tag{4.18}$$

4.3.3 Representation

After constructing the discriminative dictionary $\mathbf{D}$, the coefficients for a given feature $\mathbf{y}$ can be calculated by solving the following optimization problem:

$$\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2 + \lambda_1|\mathbf{x}|_1 + \lambda_2\|\mathbf{x}\|_2^2 + \lambda_3 \sum_{i=1}^{J} \|\alpha_i - \mathbf{x}\|_2^2 w_i \tag{4.19}$$

Similar to the derivation in Sec. 4.3.1, the feature-sign search method Lee et al. (2007) can be used to obtain the coefficients.

Chapter 5

Application 1: Single View Human Action Recognition

5.1 Introduction

The vision-based human action recognition has become an important research topic due to its promise in many application domains, including visual surveillance, video indexing, human-computer interaction, etc. In human action recognition, the common approach is to extract image features from the video, based on which a corresponding action class label can be assigned Poppe (2010). However, the large variations in performing actions, different environmental settings, and possible camera movements make the human action recognition problem very challenging.

In the literature, local spatio-temporal features have been successfully used for human action recognition I.Laptev et al. (2008); Schuldt et al. (2010). Such features describe the observed video sequence as a collection of local patches, and are less sensitive to noise and partial occlusion. Different from global features that require pre-processing methods such as background subtraction and tracking, local spatio-temporal features are extracted directly from videos and therefore do not require pre-processing methods. Feature detection and feature description are two main steps in extracting local spatio-temporal features. Many feature detectors Dollar et al. (2005); Huang et al. (2007); Laptev and Lindeberg (2005); Oikonomopoulos et al. (2006); Willems et al. (2008b); Wong and Cipolla (2007) and descriptors I.Laptev et al. (2008); Willems et al. (2008b); Klaser et al. (2008); Laptev and Lindeberg (2004); Scovanner et al. (2007) have been proposed in the past years. The feature detector aims to detect the representative interest points and scales by solving an objective function. The shape and motion characteristics of the detected 3D patches (i.e., local regions surrounding interest points) can be further described by feature descriptors.

However, limitations exist in current local spatio-temporal detectors and descriptors. For example, their performances are sensitive to the selection of datasets. Some local spatio-temporal features perform well on simple dataset but poor on some challenging realistic datasets Wang et al. (2009). In addition, the computational cost is expensive for some local spatio-temporal features. Especially, when using dense sampling Wang et al. (2009) to extract features, a very large number of features (15-20 times more than features generated by feature detectors such as Cuboid Dollar et al. (2005) and Harris3D Laptev and Lindeberg (2005)) will be generated.

In this chapter, we apply the proposed Shape and Motion Local Ternary Pattern (SMltp) descriptor to overcome the above-mentioned limitations and can be easily combined with local feature detectors for human action recognition. SMltp contains both shape (Sltp) and motion (Mltp) descriptors. For shape information, Sltp encodes each frame as a short string of ternary digits by a revised Center-Symmetric Local Ternary Pattern (CS-LTP) Gupta et al. (2010), which extracts gradient-like features along the diagonal directions. For motion information, Mltp compares each pixel at every frame with its vertical and horizontal neighboring pixels in the previous and the next frames, which reflect both the directions and magnitude of a motion among nearby frames.

Compared with current local spatio-temporal descriptors, the proposed SMltp descriptor has three advantages. First, it is computationally efficient by assigning

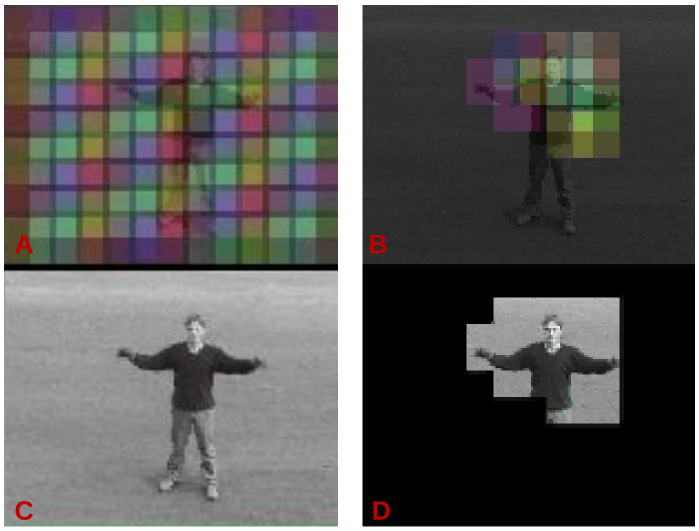


Figure 5.1: Example of patch removal for dense sampling. (a) The original image with detected patches by dense sampling marked by different colors. (b) The original image with detected patches after using patch removal. (c) Image generated by adding all the detected patches in (a) together. (d) Image generated by adding all the selected patches in (b).

ternary patterns to the local comparison results among frames. Second, it helps to dramatically remove the non-informative features that in turn further improves the computational efficiency without sacrificing the recognition accuracy. Third, it consistently performs superior to the current spatio-temporal descriptors on the evaluated public datasets, and even outperforms other more complicated algorithms for action recognition.

5.1.1 Non-informative Patch Removal

As shown in Figures 3.4b and 3.4b, regions with little motion information contain large number of (0000)-valued pixels. Patches from these regions are determined as "non-informative" since they are useless for action recognition. Let $r = N_d/N$ be the ratio between the number of zero-valued code (N_d) and the total number of pixels in the extracted 3D patches (N). The non-informative patches are the ones with $r > 0.9$. These patches are deleted for further analysis, reducing the number of patches generated by feature detectors to a great extent. Figure 5.1 shows the results before and after the non-informative patches are removed. Though dense sampling generate many useless patches, the proposed patch removal scheme can dramatically reduce the number of detected patches, and all the kept patches are more related to the motions.

It should be noted that, for local interest point detection, around 10% extracted patches will be removed as non-informative, and for dense sampling, around 60% extracted patches will be selected as non-informative.

For dense sampling, the number of extracted patches also depend on how many spatial and temporal scales adopted. For example, in Wang et al. (2009), 8 spatial scales and 2 temporal scales with 50% overlapping is used for dense sampling. The minimum size of a 3D patch is 18×18 pixels and 10 frames. Multiple scales are spaced by a factor of $\sqrt{2}$. For SMltp, the same temporal scales at 10 and 14 frames are used. However, for low resolution videos (such as KTH dataset Schuldt et al. (2004) and Weizmann dataset Gorelick et al. (2007)), only 3 spatial scales starting from 24×24 with a scale factor of $\sqrt{2}$ are used. For high resolution videos (such as the UCF dataset Rodriguez et al. (2008)), 3 spatial scales starting from 48×48 with the same factor as $\sqrt{2}$ are used. In this manner, the number of extracted patches by dense sampling will be reduced dramatically and be close to the number of patches (or interest regions) generated using feature detectors.

Table 5.1 compares the number of extracted features (after patch removal) by various feature detectors on the KTH dataset and the UCF dataset respectively. In addition, the number of patches generated by the original dense sampling used in Wang et al. (2009) is listed in the table for comparison purpose. For both the evaluated datasets, the patch removal scheme makes the detected number of features close to $20 \sim 40$ times less than the original dense sampling, and this number is even less than the number of features generated by feature detectors on the UCF dataset. It should be pointed out that the number of features extracted by the Cuboid detector is fixed to be 600 for the KTH dataset and 2,000 for the UCF dataset respectively. In

Table 5.1: The average number of generated features (features/frame) by dense sampling, Cuboid and Harris3D, combined with SMltp descriptor after patch removal scheme is used.

Detectors \ Datasets	KTH Features/frame	UCF Features/frame
Dense Wang et al. (2009)	148	1,840
Dense (SMltp)	7.8	49
Cuboid	1.8	42.3
Harris3D	1.1	82.9

addition, since we use the default parameter settings Laptev and Lindeberg (2005) for Harris3D, the spatial scales and temporal scales are different from the dense sampling.

5.1.2 Recognition

Given a video, detectors will extract a specific number of 3D patches as features. For any extracted 3D patch, a histogram representation of SMltp can be obtained.

Next, the bag-of-words representation is adopted to describe the whole video sequence. For vocabularies construction, the K-means clustering is used for all the experiments in this chapter. The number of vocabularies is set to be 3,000. Due to the memory limitation, we randomly select 120,000 patches from the whole training features for clustering. Every video is further represented as a histogram of visual words occurrences using Euclidean distance.

Finally, we use the Support Vector Machine Chang and Lin (2001) with the χ^2-kernel I.Laptev et al. (2008) for classification:

$$K(H_i, H_j) = exp\left(-\frac{1}{2A}\sum_{n=1}^{V}\frac{(h_{in}-h_{jn})^2}{h_{in}+h_{jn}}\right) \tag{5.1}$$

where $H_i = \{h_{in}\}$ and $H_j = \{h_{jn}\}$ are the frequency histograms of visual words occurrences, and V is the number of visual words or the number of clusters. A is the mean value of distances between all training samples Zhang et al. (2007).

5.2 Experiments and Results

In this section, four sets of experiments are conducted to evaluate the performance of the proposed SMltp descriptor. First, the influence of different parameters on recognition accuracy is evaluated. Secondly, the computational time of SMltp descriptor is tested. Thirdly, the comparison between the proposed SMltp descriptor and existing popular descriptors is thoroughly conducted when different feature detectors are involved. Fourthly, the performance of SMltp is compared with other state-of-the-art algorithms. We have used five popular action datasets for evaluation: the KTH dataset Schuldt et al. (2004), the Weizmann action dataset Gorelick et al. (2007), the Weizmann robustness dataset Gorelick et al. (2007), the UCF sports dataset Rodriguez et al. (2008) and the Hollywood2 Marszałek et al. (2009) dataset. Figure 5.2 shows the sample frames from the five datasets.

The **KTH action dataset** Schuldt et al. (2004) contains six types of human actions: walking, jogging, running, boxing, hand waving and hand clapping. There are 25 subjects in four different scenarios: outdoors, outdoors with scale variation, outdoors with different clothes and indoors. In total, the database contains 2,391 sequences. We choose 16 persons for training and the rest 9 persons for testing, the same as Wang et al. (2009), and the leave-one-out scheme is also investigated.

The **Weizmann action dataset** Gorelick et al. (2007) consists of 90 low-resolution (180×140) video sequences of 9 different people, each performing 10 natural actions such as running, walking, skipping, jumping jack, jumping forward, jumping in place, galloping sideways, waving both hands, waving one hand and bending. Fixed camera setting and a simple background is used in this database. The simple leave-one-out strategy is adopted for the evaluation.

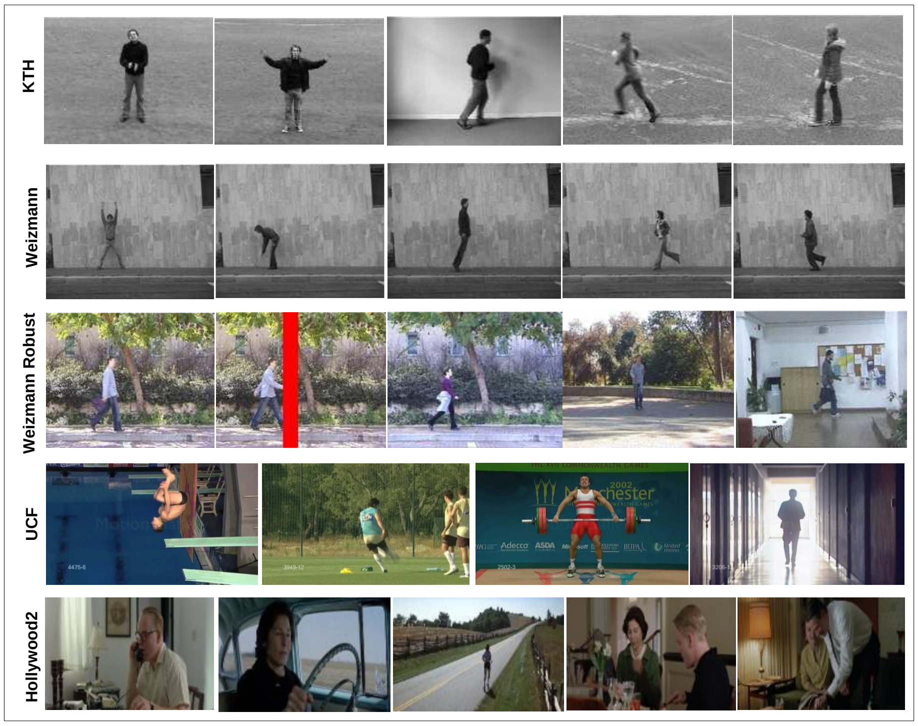


Figure 5.2: From top to down: the KTH action dataset, the Weizmann action dataset, the Weizmann robustness action dataset, the UCF sports action dataset and the Hollywood2 action dataset.

The **Weizmann robustness dataset** Gorelick et al. (2007) are 20 additional video sequences used to test the robustness of various methods to high irregularities in the performance of an action. Ten of the sequences are collected for viewpoint changes and the rest contains occlusion, clothes changes and unusual walking styles. The classifiers are trained from the Weizmann action dataset.

The **UCF action dataset** Rodriguez et al. (2008) consists of a set of actions collected from various sports which are typically featured on broadcast television channels. This dataset contains 150 video samples representing ten types of human actions: diving, golf swinging, kicking, lifting, horseback riding, running, skating, swinging (two different kinds) and walking. The sequences in this data set have large intra-class variance, due to the background change, camera motion and illumination variance. A leave-one-out setup is again used to train a multi-class classifier and get the average accuracy.

The **Hollywood2 action dataset** Marszałek et al. (2009) contains 12 actions collected from 69 different Hollywood movies. The 12 classes of actions are: answering phone, driving car, eating, fighting, getting out of the car, hand shaking, hugging, kissing, running, sitting down, sitting up and standing up. The *clean training subset* with action labels manually verified to be correct is used for training. We first calculate the average precision (AP) for every action class, and then compute the mean AP (mAP) over all classes.

5.2.1 Parameter Setup

There are several parameters need to be decided to extract the SMltp descriptor. For Sltp, the radius of the neighborhood is fixed to be 2 as commonly used in the literature. The only parameter needs to be selected is the threshold T_s in Eq. 3.3. For Mltp, the value of Δt is fixed to be 2 since it consistently generates best results. Note that the parameter t in both Eqs. 3.3 and 3.5 is fixed to be 5 to eliminate possible too small values of dynamic threshold. Parameters need to be further adjusted are:

the neighborhood radius h and the threshold T_m in Eq. 3.5. Note that h is defined as the distance between the center pixel and any outer pixel in Figure 3.3. For histogram representation, every extracted 3D patch is furthered divided into the grid of $n_\sigma \times n_\sigma \times n_\tau$ cells, and parameters n_σ and n_τ also need to be estimated. When evaluating one parameter, the other parameters are fixed at the default values, i.e., $h = 1$, $T_s = 0.12$ (Sltp), $T_m = 0.1$ (Mltp), and $n_\sigma = 3$, $n_\tau = 2$.

The KTH dataset is used to select the parameters for SMltp, since this dataset contains simple background, various scales and illumination changes. Note that the parameters are not sensitive to the testing datasets. In addition, the dense sampling is used as feature detection in order to eliminate the influence of various feature detectors. Here, 3 spatial scales and 2 temporal scales with 50% overlapping is used for dense sampling. The minimum size of a 3D patch is 24×24 pixels and 10 frames.

We first evaluate the influence of radius h in Mltp, and values 1, 2, and 3 are tested, with accuracies of 90.4%, 88.2% and 86.4% respectively. Thus, we fix the radius value to be 1 in this chapter.

We then test the results of using various thresholds in Sltp and Mltp, as shown in Figure 5.3. It is clear to see that threshold $T_s = 0.12$ for Sltp and $T_m = 0.1$ for Mltp achieve the highest recognition rates of 88.06% and 90.38% respectively. The mean accuracies of SMltp, Sltp and Mltp descriptors with various cell grid structure are shown in Figure 5.4. We observe the highest accuracy with a $3 \times 3 \times 2$ cell grid. Further increasing the number of cells does not improve the results.

As a local spatio-temporal descriptor, SMltp descriptor can be combined with any detectors, such as Cuboid, Harris3D and dense sampling. In order to make a fair comparison with the literature, we choose the similar parameter setting as Wang et al. (2009). For the Cuboid detector, we detect the features using the original code[*] and the standard scale values $\sigma = 2, \tau = 4$, and the number of extracted 3D patches are fixed as 500 for the Weizmann dataset, 600 for the KTH dataset, 2,000 for the UCF dataset and 3,000 for the Hollywood2 dataset. For the Harris3D detector, we

[*]http://vision.ucsd.edu/ pdollar/research.html

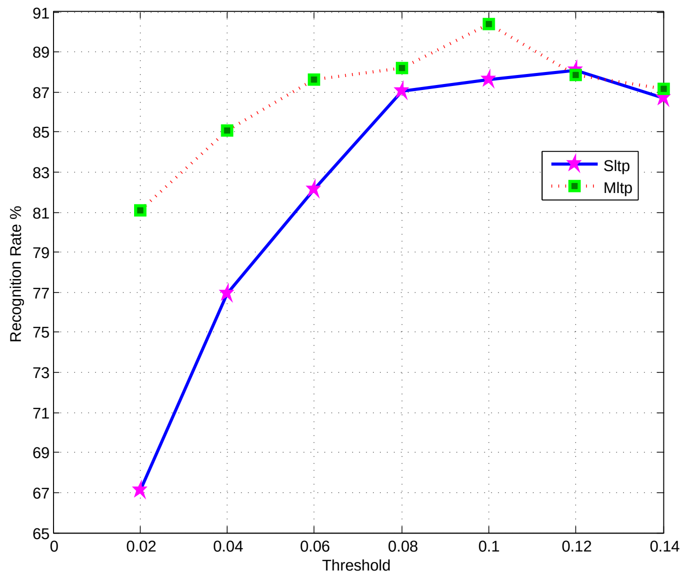


Figure 5.3: Threshold selection for SMltp descriptor. The KTH dataset Schuldt et al. (2004) is used for threshold selection. Both Sltp and Mltp adopt dynamic threshold, and the values in x axis indicate the values of T_s in Eq. 3.3 for Sltp and T_m in Eq. 3.5 for Mltp respectively.

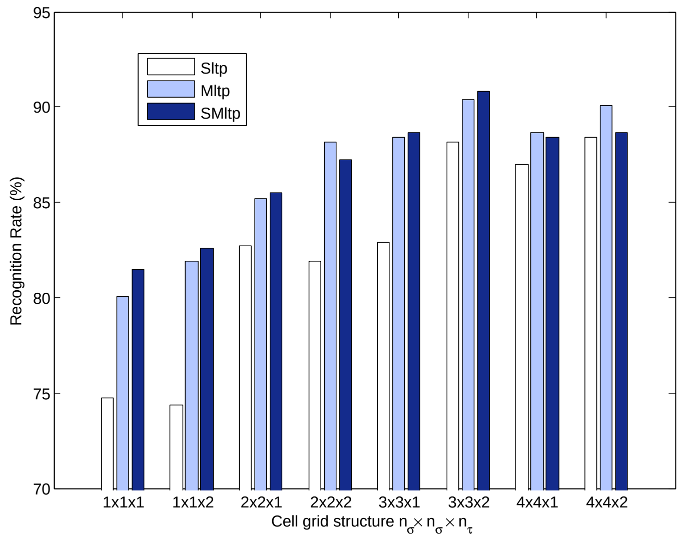


Figure 5.4: Selection of various number of cells. Performance of SMltp combined with dense sampling with various cell grid structure on the KTH dataset Schuldt et al. (2004). The cell grid structure is represented as $n_\sigma \times n_\sigma \times n_\tau$.

Table 5.2: Mean accuracies when different minimum spatial sizes are used. The SMltp descriptor is combined with dense sampling with different minimum spatial sizes on the KTH dataset Schuldt et al. (2004).

Size	SMltp	Sltp	Mltp
18×18	**91.3%**	**90.7%**	**91.3%**
24×24	**90.8%**	88.1%	**90.4%**
36×36	88.2%	82.2%	88.8%
48×48	87.0%	79.2%	88.8%

use the original implementation available on-line[†] and standard parameter settings $k = 0.0005, \sigma^2 = 4, 8, 16, 32, 64, 128, \tau^2 = 2, 4$. For dense sampling, we use 3 spatial scales compared to the 8 spatial scales used in Wang et al. (2009), and 2 temporal scales. Multiple scales are spaced by a factor of $\sqrt{2}$. Table 5.2 shows the results by using different minimal spatial sizes. The bigger the spatial sizes selected, the lower the recognition accuracy and higher the computational cost. For low resolution videos (such as the KTH dataset Schuldt et al. (2004) and Weizmann dataset Gorelick et al. (2007)), only 3 spatial scales starting from 24×24 with a scale factor of $\sqrt{2}$ are used. For high resolution videos (such as the UCF dataset Rodriguez et al. (2008) and the Hollywood2 dataset Marszałek et al. (2009)), 3 spatial scales starting from 48×48 with the same factor as $\sqrt{2}$ are used.

5.2.2 Computational Time

SMltp takes advantage of the LBP-like features in terms of computational efficiency. From Eqs. 3.1 and 3.4, we see that the calculation of SMltp descriptor focuses on simple pixel comparisons within local regions. It can be considered as having the same complexity as gradient calculation. Furthermore, the histogram representation of SMltp does not need any angle calculation as most local descriptors require. We compare the computational cost of SMltp with HOG/HOF, combined with dense

[†]http://www.di.ens.fr/ laptev/download.html

Table 5.3: Computational time (features/frame) of SMltp and HOG/HOF combined with dense sampling.

	HOG/HOF	SMltp
video size	$720 \times 576 \times 101$	$720 \times 576 \times 101$
spatial scales	3	3
temporal scales	2	2
run time (frames/second)	1.8	5.9

sampling on 3 spatial scales and 2 temporal scales, on a $720 \times 526 \times 101$ video. The run-time estimates are obtained on a PC with Intel Xeon Processor X5550 (Nahalem) 2.66 GHz Quad Core and 12 GB memory. The coding language is C++. Table 5.3 lists the calculation time of SMltp compared with the HOG/HOF descriptor, combined with dense sampling. From Table 5.3, we can see that SMltp is almost three times as fast as HOG/HOF.

5.2.3 Comparison with Other Descriptors

To test the robustness of the proposed descriptors, the results of SMltp descriptor on the KTH, UCF and Hollywood2 datasets are compared with other popular local spatio-temporal descriptors. The KTH dataset is well established with simple background, containing illumination, environment and scale variation, while UCF and Hollywood2 datasets are more challenging datasets with complicated background and camera motions. We follow the frequently used experimental setup in the literature to evaluate the performance on these datasets. Table 5.4 lists the recognition rates of the proposed descriptors combined with Harris3D, Cuboids and dense sampling for action recognition, and the results of other descriptors (HOG3D, HOG/HOF and ESURF) are provided for comparison.

Experimental results show that the proposed SMltp descriptor significantly outperforms the other shape and motion descriptors no matter what feature detectors are used. Especially, the Sltp descriptor performs much better than HOG as a

Table 5.4: Comparison with local spatio-temporal descriptors. Recognition rates (%) of different local spatio-temporal descriptors are provided by Wang et al. (2009). The *ESURF* descriptor is combined with Hessian detector Willems et al. (2008a).

Datasets	Detectors	Descriptors					**Proposed**		
		HOG3D	HOG/HOF	HOG	HOF	ESURF	SMltp	Sltp	Mltp
KTH	Harris3D	89.0	91.8	80.9	92.1	81.4	**94.0**	92.1	93.7
	Cuboids	90.0	88.7	82.3	88.2		**93.7**	91.3	92.1
	Dense	85.3	86.1	79.4	88.4		**90.8**	88.1	90.4
UCF	Harris3D	79.7	78.1	71.4	75.4	77.3	**85.3**	82.0	83.3
	Cuboids	82.9	77.7	72.7	76.7		**83.3**	78.7	80.7
	Dense	85.6	81.6	77.4	82.6		**95.3**	93.3	94.0
Holly2	Harris3D	43.7	45.2	32.8	43.3	38.2	**48.5**	42.3	47.6
	Cuboids	45.7	46.2	39.4	42.9		**47.8**	41.4	46.2
	Dense	45.3	47.4	39.4	45.5		**48.9**	42.5	47.2

shape descriptor, and the Mltp descriptor outperforms HOF as a motion descriptor. Specifically, the SMltp descriptor improves the performance by 3%∼5% when combined with the Harris3D detector, by 2%∼5% when combined with the Cuboids detector, and by 2%∼10% when combined with dense sampling.

For the simple action dataset such as the KTH dataset, the local spatio-temporal detectors perform better than the dense sampling. For the challenging datasets, such as UCF and Hollywood2 datasets, dense sampling generates higher accuracies when combined with different descriptors. Note that dense sampling will generate many more features that make the process of BoW representation for HOG3D and HOG/HOF very time consuming. However, the proposed SMltp descriptor can efficiently remove non-informative patches/features to ensure the number of features generated by dense sampling close to the other local spatio-temporal detectors as illustrated in Table 5.1. In addition, the computational speed of the proposed descriptor is 3 times faster than the HOG/HOF descriptor which performs better than HOG3D and ESURF. Therefore, not only the recognition rates are improved by using SMltp, the computational time is also significantly reduced.

It is interesting to note that the improvements of performance using dense sampling on the UCF dataset is more significant than that on the Hollywood2 dataset.

Although SMltp helps to remove non-informative patches generated from background by dense sampling, there are still patches from background if camera motions or scene changes exist. Since the UCF dataset contains actions from sportscast on television channels, backgrounds for the same actions are similar. In this case, if the descriptor can well describe the detected regions, the recognition rate on the UCF dataset can be satisfied. However, videos in the Hollywood2 dataset are from movies with different background and large intra-class variations. In this case, the patches generated from the background will obscure the importance of patches generated by actions. Even if the descriptor can well describe each patch, the performance is still limited on this dataset.

5.2.4 Comparison with State-of-the-art Algorithms

We evaluate the performance of the proposed algorithm with state-of-the-art methods on the five public action datasets: Weizmann action dataset, Weizmann robustness dataset, KTH dataset, UCF dataset and Hollywood2 dataset.

The Weizmann action dataset is well established with simple and fixed background, which makes it suitable for performance evaluation. Since this dataset does not contain occlusion, viewpoint, spatial and temporal scales changes, only the silhouettes extracted by background subtraction are used for the proposed descriptors, same as Gorelick et al. (2007); Guha and Ward (2011). Table 5.5 shows the action recognition rates by various approaches, all of which use the leave-one-out scheme for evaluation. The highest recognition rate of 100% is achieved by the SMltp descriptor combined with dense sampling or Cuboid. To the best of our knowledge, this accuracy is the highest in the literature when all the 10 classes are used. It should be pointed out that the work of Yeffet and Wolf Yeffet and Wolf (2009) achieves the highest accuracy 100% on the old version of this dataset with 9 actions only, while the proposed algorithm is tested on the new version of this dataset with 10 actions.

Table 5.5: Results on Weizmann dataset.

Method	No. of actions	Accuracy
Guha and Ward (2011)	10	98.9%
Gorelick et al. (2007)	10	97.8%
Riemenschneider and Bischof (2009)	10	96.7%
Ali and Shah (2010)	10	95.7%
Thurau and Hlavac (2008)	10	94.4%
Zhang et al. (2008)	10	92.8%
Niebles et al. (2008)	10	90.0%
Scovanner et al. (2007)	10	84.2%
Yeffet and Wolf (2009)	9	**100%**
Wang and Mori (2009)	9	**100%**
Junejo et al. (2009)	9	95.3%
Proposed		
Harris3D+SMltp	10	97.8%
Cuboid+SMltp	10	**100%**
Dense+SMltp	10	**100%**

The Weizmann robustness dataset contains a non-uniform background with various difficult scenarios such as occlusion, various walking styles, clothing changes and viewpoints difference. We train the proposed algorithms on the Weizmann action dataset that contains no occlusion or viewpoint change, and evaluate the performance on the Weizmann robustness dataset. Table 5.6 presents the results of the SMltp descriptor under occlusion and other difficult scenarios, compared with that reported in Gorelick et al. (2007); Guha and Ward (2011). Note that the work of Gorelick et al. (2007) uses global features, while both Guha and Ward (2011) and the proposed algorithms use local features. The SMltp descriptor combined with the dense sampling outperforms the work of Guha and Ward (2011) that uses sparse coding for representation. Table 5.7 shows the results under viewpoint changes, varying between $0°$ and $81°$. Better than the sparse representation in Guha and Ward (2011), 100% accuracy is achieved by SMltp combined with Harris3D or Cuboids

Table 5.6: Performance evaluation on the Weizmann robustness dataset with real occlusion and other difficult scenarios.

Query sequence	Algorithms		**Proposed**		
	Gorelick et al. (2007)	Guha and Ward (2011)	Harris3D	Cuboid	Dense
Normal walk	walk	walk	walk	walk	walk
Walking in a skirt	walk	walk	walk	walk	walk
Carrying briefcase	walk	walk	walk	walk	walk
Limping man	walk	walk	**skip**	**skip**	walk
Occluded Legs	walk	walk	walk	walk	walk
Knees Up	walk	**run**	walk	walk	walk
Walking with a dog	walk	walk	walk	walk	walk
Sleepwalking	walk	walk	walk	walk	walk
Swinging a bag	walk	walk	walk	walk	walk
Occluded by a "pole"	walk	walk	walk	walk	walk

detector. Therefore, SMltp descriptor is robust to large intra-class variations and viewpoint changes.

The KTH action dataset contains illumination, environment and scale variation, which make it more challenging than the Weizmann action dataset. In the literature, various algorithms require different experimental setups. There are two frequently used experimental setups: one is the same as the original one Schuldt et al. (2010) (16 people for training and 9 people for testing), and the other is to use leave-one-out cross validation setting. Table 5.8 shows the comparison of our algorithm with other approaches in the literature that use 16 people for training and 9 for testing. The SMltp descriptor performs competitively to the state-of-the-art algorithms and achieves 94.0% accuracy. Note that Wang et al. (2011) used combined features which is computationally more demanding as compared to the features we propose. Also, Kovashka and Grauman (2010) and Gilbert et al. (2011) do not use the traditional bag-of-words representation as we do. Table 5.9 shows the results using the simpler leave-one-out approach, and these results show higher average performance as compared to those in Table 5.8. SMltp combined with Harris3D gets as high as 95.3% recognition rate, which is competitive to the highest accuracy in the literature 95.7%.

Table 5.7: Performance of SMltp descriptor on the Weizmann robustness dataset under viewpoint changes.

Query sequence	Algorithms		**Proposed**		
	Gorelick et al. (2007)	Guha and Ward (2011)	Harris3D	Cuboid	Dense
Walking in 0°	walk	walk	walk	walk	walk
Walking in 9°	walk	walk	walk	walk	walk
Walking in 18°	walk	walk	walk	walk	walk
Walking in 27°	walk	walk	walk	walk	walk
Walking in 36°	walk	walk	walk	walk	walk
Walking in 45°	walk	walk	walk	walk	walk
Walking in 54°	walk	walk	walk	walk	walk
Walking in 63°	walk	**skip**	walk	walk	**jump**
Walking in 72°	walk	**skip**	walk	walk	**side**
Walking in 81°	walk	**side**	walk	walk	**side**

Table 5.8: Results on KTH dataset.

Method	Average Precision
Kovashka and Grauman (2010)	**94.53%**
Gilbert et al. (2011)	**94.5%**
Wang et al. (2011)	**94.2%**
Le et al. (2011)	93.9%
Yuan et al. (2009)	93.3%
I.Laptev et al. (2008)	91.8%
Klaser et al. (2008)	91.4%
Yeffet and Wolf (2009)	90.1%
Nowozin et al. (2007)	87.04%
Schuldt et al. (2010)	71.71%
Gorelick et al. (2005)	62.97%
Proposed	
Harris3D+SMltp	**94.0%**
Cuboids+SMltp	93.7%
Dense+SMltp	90.8%

Table 5.9: Results on KTH dataset (Leave-One-Out).

Method	Average Precision
Kovashka and Grauman (2010)	**95.7%**
Kim et al. (2007)	95.0%
Han et al. (2009)	94.33%
Liu and Shah (2008a)	94.1%
Uemura et al. (2008)	93.7%
Bregonzio et al. (2009)	93.2%
Yang et al. (2009b)	87.3%
Niebles et al. (2008)	81.5%
Dollar et al. (2005)	81.2%
Proposed	
Harris3D+SMltp	**95.3%**
Cuboids+SMltp	94.1%
Dense+SMltp	93.2%

The UCF sports dataset is a challenging one since the sequences are mostly acquired by moving cameras. Different from the Weizmann and KTH dataset using the synthetic actions, sequences in the UCF dataset are captured from sportscast on television channels. Thus, the performance of SMltp on this dataset can well reflect the effectiveness and robustness of algorithms on real-world action recognition. Table 5.10 shows the mean accuracies of the proposed methods, compared with various approaches in the literature. Combined with dense sampling, SMltp significantly outperforms the current state-of-the-art method Wu et al. (2011) by 5%. Note that our algorithm is computationally efficient, and does not need to enlarge the training set as Wang et al. (2009) and Zhu et al. (2004) do.

Table 5.11 shows the Mean AP for algorithms evaluated on Hollywood2 dataset. This dataset is the most challenging one in the literature that all the videos in it are from movies. Although the performance of SMltp descriptor is not as well as the state-of-the-art algorithms on this dataset, it outperforms algorithms that use other feature descriptors in Wang et al. (2009). This result further indicates the limitation of using local spatio-temporal feature detectors. Possible improvements on

Table 5.10: Results on UCF dataset

Method	Average Precision
Wu et al. (2011)	89.7%
Wang et al. (2011)	88.2%
Kovashka and Grauman (2010)	87.27%
Yao et al. (2010)	86.6%
Wang et al. (2009)	85.6%
Zhu et al. (2004)	84.3%
Yeffet and Wolf (2009)	79.2%
Rodriguez et al. (2008)	69.2%
Proposed	
Harris3D+SMltp	85.3%
Cuboids+SMltp	83.3%
Dense+SMltp	**95.3%**

performance can be expected if the dense trajectory presented in Wang et al. (2011) is used for feature detection. However, it generates many more features even than dense sampling, and thus requires large memory and computational time.

5.3 Summary

We presented an effective and computationally-efficient feature description algorithm, referred to as SMltp, to capture both the shape and motion information for the purpose of single view human action recognition. Experimental results demonstrated that SMltp is fast, robust and highly accurate. As an LBP-like feature descriptor, its simple calculations help save the computational cost greatly. As a local feature descriptor, SMltp can be easily combined with various local feature detectors to achieve the state-of-the-art accuracy. In addition, the proposed patch selection scheme makes the dense sampling practical for real application by dramatically reducing the number of features. Due to the robustness of SMltp, 3 spatial scales are enough for achieving better performance than the dense sampling used in Wang et al. (2009) on the testing datasets. The highest accuracy of the proposed SMltp on the UCF

Table 5.11: Results on Hollywood2 dataset.

Methods	Mean AP
Wang et al. (2011)	**58.3**%
Le et al. (2011)	53.3%
Gilbert et al. (2011)	50.9%
Wang et al. (2009)	47.7%
Taylor et al. (2010)	46.6%
Proposed	
Harris3D+SMltp	48.5%
Cuboids+SMltp	47.8%
Dense+SMltp	48.9%

action dataset, compared to other approaches in the literature, further suggested the feasibility of its real-world deployment.

Chapter 6

Application 2: Distributed Human Action Recognition

6.1 Introduction

With the emergence of distributed camera networks (DCNs) Chen et al. (2008), the deployment of multi-view action recognition in DCN becomes a logical next step. In the DCN environment, an object could be surrounded by multiple cameras and observed by these cameras from different views. These cameras have imaging, on-board processing, and wireless communication capabilities. Collaboratively they could solve computer vision problems through distributed sensing and processing. However, resource constraint is the major limitation of DCNs since each camera has only limited memory and power supply, and the communication between local cameras and base station is also expensive and limited by the bandwidth. These constraints have largely hindered the successful deployment of existing multi-view action recognition algorithms in DCNs.

To resolve the conflict between the constraint resource in DCNs and the need for real-time recognition, two issues need to be investigated. First, what information should be extracted by each local camera in order to satisfy the bandwidth requirement without jeopardizing recognition accuracy? Second, how to fuse the information collected from distributed cameras for recognition purpose? In the DCN environment, the transmission of the raw image data to the base station is not feasible due to the expensive transmission cost, and thus the distributed feature-based recognition becomes necessary.

In this chapter, we focus on the design of algorithms for *distributed* and *robust* multi-view action recognition, which requires low memory and bandwidth consumption. Here, "distributed" refers to the fact that each camera processes its data locally and just sends limited information to the base station, and "robust" refers to the fact that the recognition accuracy is not affected much if one or more of the cameras are malfunctioning and if different selections of cameras are used. Specifically, we propose a "distributed" and "robust" human action recognition framework based on *sparse coding* of the extracted features. The framework consists of three components, feature extraction, feature representation, and classification. We summarize the main contributions of the paper also from these three aspects.

From the feature extraction perspective, we use the proposed Motion Local Ternary Pattern (Mltp) as a new operator to describe the local intensity difference among frames caused by any motion. Mltp takes advantage of Local Binary Pattern (LBP) Ojala et al. (2002); Yeffet and Wolf (2009) in terms of computational efficiency, tolerance to illumination change and robustness in homogeneous regions. Mltp is distinctive for motion patterns without any additional information such as background subtracted silhouettes or 3D visual hull volumes.

From the feature representation perspective, the sparse coding technique is adopted to generate a histogram representation for every video to reduce the possible quantization error caused by the popular Bag-of-Words (BoW) model. Given a 50-frame video clip with image resolution of 120×160 pixels, instead of transmitting $960,000$ pixels, the amount of data transfer required by our system is only n ($n < 600$) bins where each bin is represented by 4 *bytes*. During the whole process, communication among cameras is unnecessary.

To fuse the information transmitted from local cameras, we use Naive Bayesian to integrate the prediction results from the Support Vector Machine (SVM) Chang and Lin (2001) to give the final decision. Compared to the existing algorithms Yan et al. (2008); Weinland et al. (2007); Liu and Shah (2008b) and applications Srivastava et al. (2009) on the similar DCN environment, our proposed framework achieves higher recognition accuracy, requires less memory and is with lower bandwidth consumption.

6.2 Proposed Methodology

Figure 6.1 shows the four steps involved in the proposed framework for distributed multi-view action recognition: 1) feature detection that localizes the positions of interest points from the feed videos; 2) feature description that generates the distinctive feature descriptors for any 3D patches around the detected interest points. In this chapter, we use the Mltp as feature descriptor; 3) feature representation which aims to generate the histogram representation for every action video; and 4) distributed multiple view action recognition that fuses information sent from local cameras to generate the final class label for a specific action.

6.2.1 Feature Detection

In this chapter, the Cuboid detector proposed by Dollar Dollar et al. (2005) is adopted for spatio-temporal interest points detection. We use the proposed Mltp as the feature descriptor. Figure 6.2 shows the result of Mltp on nearby frames from the IXMAS multi-view action dataset Weinland et al. (2007). Figure 6.2a are three subsequent frames representing the motion of “hands up”, while Figure 6.2c showing the opposite motion of “hands down”. The directions of motion are encoded in three gray scale values (0 or black, 125 or gray and 255 or white) as shown in Figure 6.2b and Figure 6.2d. Comparing the locations of pixels with the intensity value of 125 (motion caused by the difference between the current frame and the next frame) and

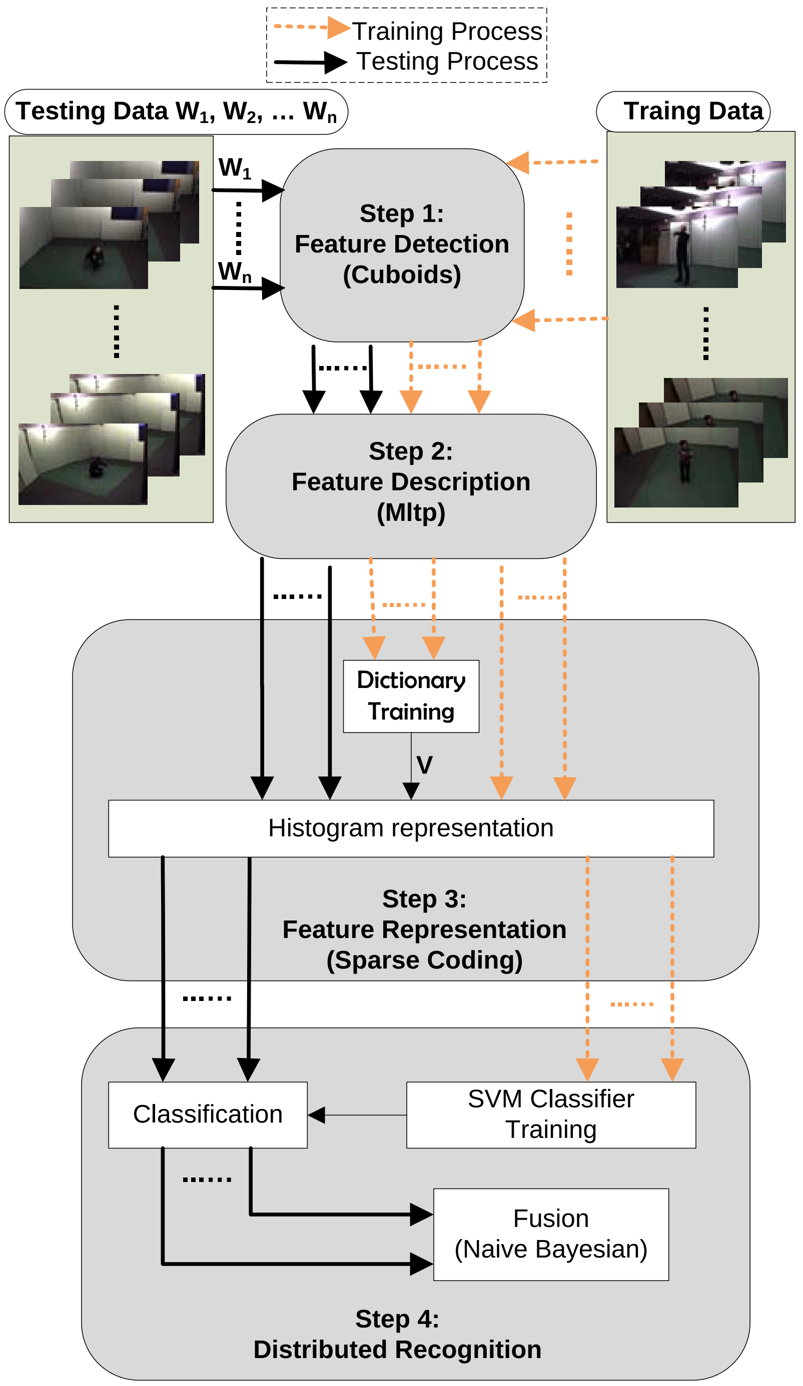


Figure 6.1: The proposed framework for distributed human action recognition.

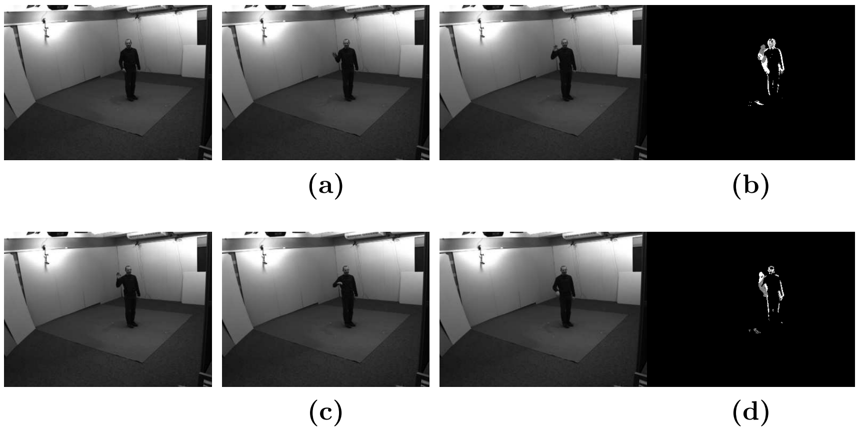
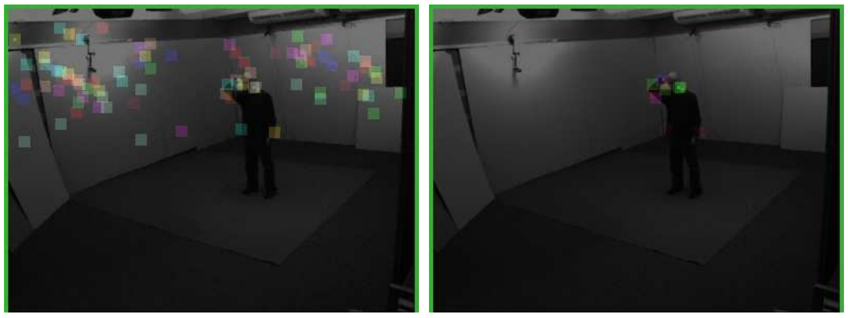


Figure 6.2: Mltp results on nearby frames from the IXMAS multi-view action dataset. (a) Three subsequent frames from the beginning of hand waving motion. (b) Motion image calculated by Mltp on (a). (c) Three subsequent frames from the end of hand waving motion. (d) Motion image calculated by Mltp on (c).

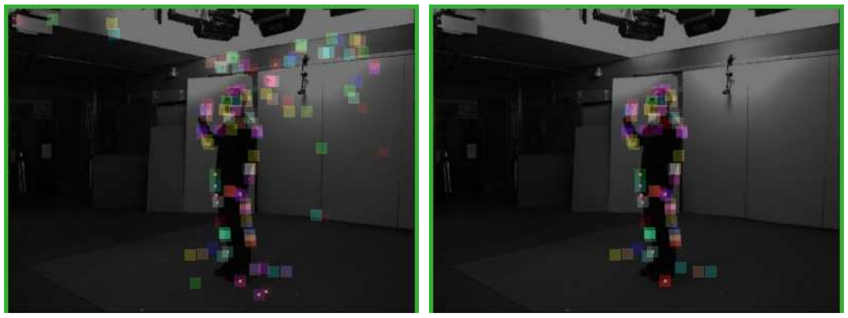
pixels with intensity value of 255 (motion caused by the difference between the current frame and the previous frame), the two opposite motions are distinctively described. Moreover, all the background pixels, as well as body parts with no contribution to the motion, are singled out and assigned with zero values.

6.2.2 3D Motion Patch Selection

After applying feature detection on the input videos, a specific number of 3D patches can be extracted. For pixels within any extracted 3D patch, 4-digit codes can be calculated on a 4-neighbor definition of Mltp. In total, there will be 81 possible codes that represent different motion patterns. The resulting 3D patch with assigned codes is referred to as the *3D motion patch*. Pixels with no motion have the code value as (0000). Any 3D motion patch can be considered as non-informative if the number of pixels with code (0000) is too large. Let m_0 be the number of pixels with no motion and N be the number of total pixels within the 3D motion patch. If the value of the ratio m_0/N is larger than a predefined threshold T_p, the 3D motion patch will be considered as non-informative and removed. Figure 6.3 shows the results of applying



(a) scratch head



(b) wave

Figure 6.3: Cuboid selection by Mltp values for two actions. Cuboids are marked as colorful squares. The cuboids contain large number of pixels with no motion are removed. Hence, cuboids located at background area are eliminated.

3D motion patch selection. The colorful squares show the extracted cuboids Dollar et al. (2005) in the spatial domain. For both figures, the left images reflect the locations of original patches, and the right images reflect the results after removing patches without obvious motion. It is clear to see that the noisy patches generated from background have been removed, which better highlights the foreground patches. Note that the threshold T_p is set to be 0.9 as a compromise between the number of selected features and the requirement for the informative 3D motion patches.

6.2.3 Feature Representation

In order to further reduce the transmission cost, a histogram representation of the collected feature descriptors (Mltp) from a video is calculated and transmitted to the base station. In this paper, the well known *Bag-of-Words (BoW)* model is used as a baseline. We further propose the *Sparse Coding* model for feature representation to reduce the approximation error and increase the robustness. For both models, the first step is to learn a set of representative vectors from a large number of feature vectors and the second step is to obtain the histogram representation of features according to the learned vectors. The set of learned representative vectors are referred to as the "codebook" in BoW and "dictionary" in sparse coding. The individual representative vectors are referred to as the "codewords" in BoW and "atoms" in sparse coding.

The BoW model uses k-means clustering to learn the codebook and every cluster center corresponds to a codeword. Then each feature vector can be assigned to a specific codeword that is closest to it in terms of the Euclidean distance. Considerable amount of quantization error would incur by the approximation process where each sample vector is assigned to the nearest codeword of BoW. This deficiency can be largely overcome by sparse coding that allows a linear and sparse combination of atoms to be used in the approximation process. The calculated sparse codes for one feature corresponds to the responses of that feature to all the atoms in the dictionary. Or put it another way, the sparse codes represent the coefficients in the linear combination of atoms.

Let $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N] \in \mathbb{R}^{f \times N}$ be a set of extracted features in a f-dimensional feature space, and $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{K \times N}$ be the sparse codes for these feature vectors. The approximation process can be represented by sparse coding Yang et al. (2009a):

$$\begin{aligned} \min_{D,X} = & \quad \sum_{m=1}^{N} \|\mathbf{y}_m - \mathbf{D}\mathbf{x}_m\|^2 + \lambda|\mathbf{x}_m| \\ \text{subject to} & \quad \|\mathbf{d}_k\| \leq 1, \qquad \forall k = 1, 2, \ldots, K \end{aligned} \tag{6.1}$$

where $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_K]^T \in \mathbb{R}^{f \times K}$ is the learned *overcomplete* dictionary with K atoms, and $\mathbf{x}_m$ is the calculated sparse codes for feature vector $\mathbf{y}_m$. The unit L_2-norm constraint on $\mathbf{v}_k$ is typically applied to avoid trivial solutions Yang et al.

(2009a). Note that the L_1 regularization on $\mathbf{x}_m$ enforces $\mathbf{x}_m$ to have a small number of nonzero elements. In other words, instead of assigning the nearest codeword as does in the BoW modeling, each feature vector is approximated by a linear and sparse combination of atoms in the learned dictionary.

To solve Eq. 6.1, two steps are involved by iteratively optimizing over $\mathbf{D}$ or $\mathbf{X}$ while fixing the other. By fixing $\mathbf{D}$, the optimization can be solved by:

$$\min_{\mathbf{x}_m} \|\mathbf{y}_m - \mathbf{D}\mathbf{x}_m\|^2 + \lambda|\mathbf{x}_m| \tag{6.2}$$

This is known as Lasso in the Statistical literature and can be efficiently solved by the *feature-sign search* algorithm Lee et al. (2006). By fixing $\mathbf{X}$, the objective function becomes a least square problem with quadratic constraints:

$$\begin{aligned} \min_{\mathbf{D}} \quad & \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 \\ \text{subject to} \quad & \|\mathbf{d}_k\| \leq 1, \quad \forall k = 1, 2, \ldots, K \end{aligned} \tag{6.3}$$

The Lagrange dual can be used to solve the optimization problem Lee et al. (2006).

In sparse coding, the dictionary $\mathbf{D}$ is learned in the training phase that collects a large number of features from training samples by iteratively optimizing Eqs. 6.2 and 6.3. In the coding phase, the sparse codes are retained by optimizing Eq. 6.2 given learned $\mathbf{D}$.

To yield the histogram representation after sparse coding, a pooling function $\mathbf{z} = \mathcal{F}(\mathbf{X})$ is applied in this paper. Note that different pooling functions construct different image statistics. As noted in Yang et al. (2009a), *max* pooling produces better performance than other pooling methods when using, for example, the square root of mean squared statistics and the mean of absolute values, as evaluation metrics. Therefore, we adopt the *max* pooling function:

$$z_j = \max\{|x_{1j}|, |x_{2j}|, \ldots, |x_{Mj}|\} \tag{6.4}$$

where z_j is the j-th element of $\mathbf{z} \in \mathbb{R}^K$, x_{ij} is the matrix element at the i-th row and the j-th column of $\mathbf{X}$, and N is the size of the extracted features (Mltp) set.

Using sparse coding as a tool for feature representation has two benefits. First, the objective function of sparse coding minimizing the reconstruction error with sparse coefficients reduces the approximation or quantization error. Second, the linear combination representation makes it robust to the different combination of multiple views and tolerant to the misalignment between training and testing samples.

6.2.4 Distributed Multiple View Action Recognition

In our system, one object is surrounded by several cameras capturing various views. Cameras from distinctive views will record the same action simultaneously and process the video data locally. The DCN systems like WiCa Kleihorst et al. (2006) and CITRIC Chen et al. (2008) are some examples of smart camera platforms where our algorithms can be implemented. To conduct the distributed multiple view action recognition, two phases are involved: training and testing. We use the sparse coding representation as an example to illustrate both phases and the BoW follows the similar process.

- **Training Phase**: As illustrated in Figure 6.1, the training phase is designed to learn the dictionary and train the classifier. To learn the dictionary, the extracted features (Mltp) should be collected from training samples. Note that only one dictionary is trained for all the actions. Each camera will store the learned dictionary locally. After the classifier is trained, the base station will store it for recognition purpose. All the training process is conducted off-line.

- **Testing Phase**: Each local camera extracts the features (Mltp) first, and then calculates the histogram representation by Eqs. 6.2 and 6.4 for every video sequence based on the learned dictionary. It should be pointed out that our system does not need data transmission between cameras and only the histogram representation of every video is transmitted to the base station. At the base station, the classification results obtained from these histograms based on the classifier are fused to obtain the final decision.

In this chapter, we use a Naive Bayesian technique to fuse the results from each view. We first apply the Support Vector Machine (SVM) Chang and Lin (2001) with the χ^2-kernel I.Laptev et al. (2008) for prediction based on each camera's input:

$$K(H_i, H_j) = \exp\left(-\frac{1}{2A}\sum_{n=1}^{S}\frac{(h_{in}-h_{jn})^2}{h_{in}+h_{jn}}\right) \tag{6.5}$$

where $H_i = \{h_{in}\}$ and $H_j = \{h_{jn}\}$ are the histograms calculated by BoW or sparse coding, and S is the number of codewords in codebook or atoms in dictionary. A is the mean value of distances between all training samples Zhang et al. (2007).

The second step performed at the base station is the integration of probability outputs of SVM Platt (2000) using the Naive Baysian technique for the purpose of multiple view action recognition. Same kernel function is used as Eq. 6.5. Each view W_j will have a probability for action label l_i as $P(l_i|W_j)$. Let the number of cameras be n and the number of action classes be c. Our objective is to calculate the probability of l_i when $W_1, \cdots, W_n$ are available, referred to as $P(l_i|W_1, W_2, \cdots, W_n), i = 1, \cdots, c$. To simplify the problem, we assume the selection of each camera is independent, so that $P(W_1, \cdots, W_n) = P(W_1) \times \cdots P(W_n)$. In addition, the probability of every action is assumed to be equal.

For any class label $i, i = 1, \cdots, c$ in an n-camera environment, its probability can be calculated as

$$\begin{aligned} P(l_i|W_1,\cdots,W_n) &= \frac{P(W_1,\cdots,W_n|l_i)P(l_i)}{P(W_1,\cdots,W_n)} \\ &= \frac{P(W_1|l_i)\cdots P(W_n|l_i)P(l_i)}{P(W_1)\cdots P(W_n)} \\ &= \frac{P(l_i|W_1)\cdots P(l_i|W_n)}{P(l_i)^{n-1}} \end{aligned} \tag{6.6}$$

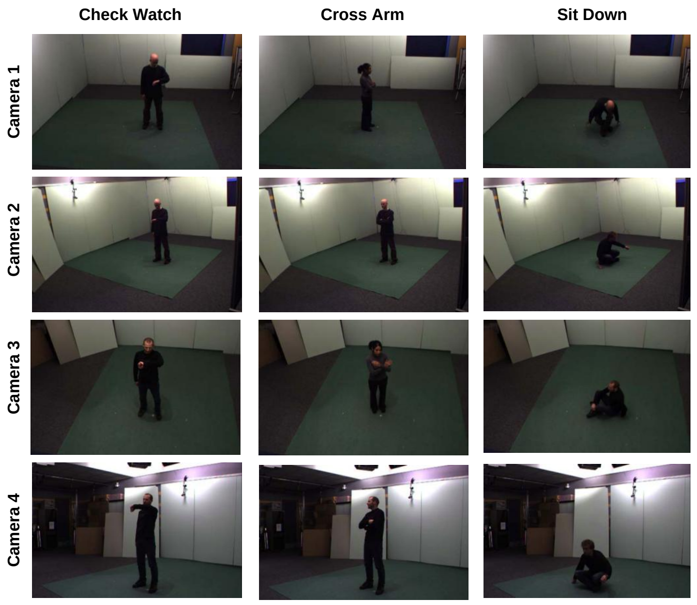
Since $P(l_1) = P(l_2) = \cdots P(l_c)$, the probability of action class i can be calculated by multiplying all the local probabilities $P(l_i|W_j), j = 1, \cdots, n$. For any test sequence, the class label can be assigned to the one with the highest value of Eq. 6.6.

6.3 Experiments and Results

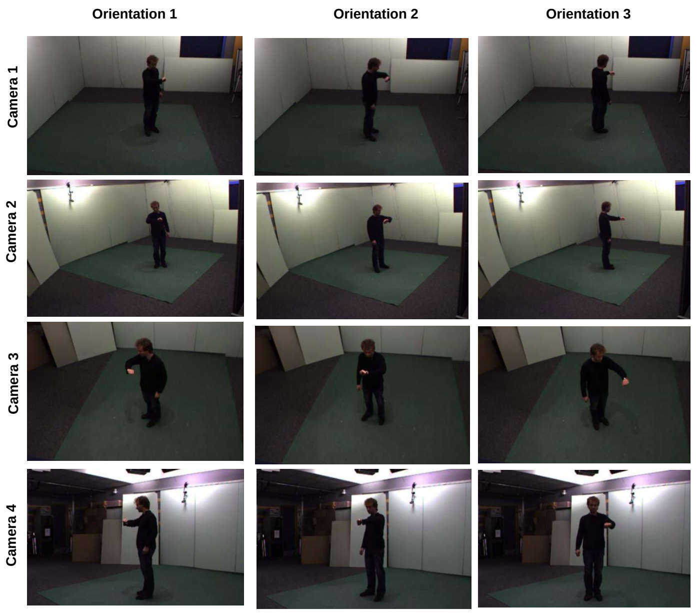
We evaluate our approaches on the IXMAS multi-view action dataset Weinland et al. (2007) which contains 13 daily-life motions performed each 3 times by 12 actors. The actors choose free positions and orientations. In order to compare with other algorithms, we choose the same experimental setting as Srivastava et al. (2009); Weinland et al. (2007) which use 11 action categories and 10 subjects. In addition, we select four views excluding the top view, same as Srivastava et al. (2009); Liu and Shah (2008b). Figure 6.4a shows sample images of actions "check watch", "cross arm", and "sit down", performed by different actors from three orientations. Figure 6.4b shows sample images of action "check watch" performed by the same actor from different orientations. Note that actors are free to choose their own style to perform the same action. For example, the actor in Figure 6.4b uses the *left hand* to perform the action "check watch" in one orientation, but changes to use the *right hand* in another orientation.

6.3.1 Parameter Setup

From each video 200 cuboids are extracted with spatial scale $\sigma = 2$ and temporal scale $\tau = 2.5$. The size of the cuboids (or 3D patches) is then $13 \times 13 \times 17$ pixels, which corresponds to 2873 pixels in total. For Mltp, the parameters for $\Delta t = 3$, $T_m = 0.1$, $t_m = 5$, and $D = 2$ are experimentally selected. In addition, the parameter T_p used for non-informative patches removal discussed in Sec. 5.1.1 is set to be 0.9. Note that the selection of the parameters are not sensitive and can be adjusted simply by choosing a few samples for testing purpose.



(a) Four views of three actions



(b) Four views and three orientations for action: check watch

Figure 6.4: Samples of the IXMAS dataset. (a) Four views of three selected action examples. (b) Three orientations of the action “check watch”. People may choose their own orientations and styles while performing the same action.

As described in Chapter 3, the dimension of feature descriptor **Mltp** depends on the number of small cells divided within every cuboid. In this paper, we divide the cuboids into $2 \times 2 \times 2$ cells to get a 128-dimensional feature descriptor (Mltp).

For feature representation, we follow the same setting as in the literature Yang et al. (2009a), and choose the size of dictionary (when using sparse coding) to be of 4 times the feature dimension, which is $4 \times 128 = 512$. For the size of codebook, we evaluate various numbers of codewords and set it to be 180 for BoW representation.

6.3.2 Multi-View Human Action Recognition

At the multi-view performance evaluation stage, we first evaluate the proposed framework in the case of all the four views and three orientations are used for training. During the testing phase, different number of views are involved. Both the Leave-One-Out strategy and the cross subjects strategy are used for evaluation. For the Leave-one-out strategy, videos from 9 actors are used for learning and the remaining one actor's videos are used for testing. For the cross subjects strategy, videos from 5 actors are used for learning and the remaining 5 actor's videos are used for testing. The classification result is based on the probability output of SVM as described in Sec. 6.2.4. It should be pointed out that our models do not need any preprocessing such as background subtraction or 3D visual hull volumes Yan et al. (2008); Weinland et al. (2007).

Table 6.1 lists the recognition accuracies of the proposed framework using different feature representation models (Mltp+BoW and Mltp+sparse coding) when the Leave-one-out and cross subject strategies are used, respectively. The performance of the state-of-the-art algorithms that use the Leave-one-out strategy are listed for comparison purpose. No cross-subject experimental results have been reported by these algorithms. For the proposed framework, the class label is assigned to the one with the highest value according to Eq. 6.6. The various combination of views are explored and the maximum and average recognition accuracies are reported in

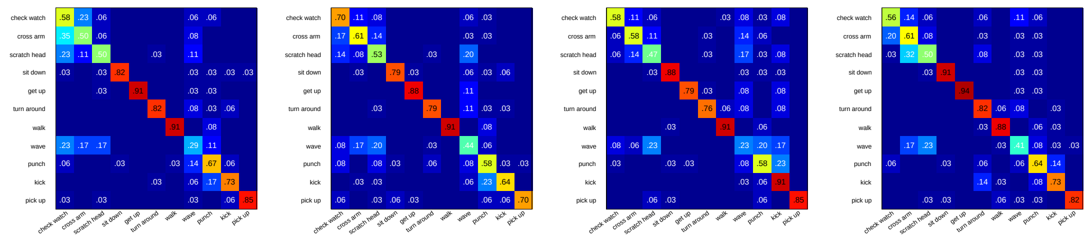
Table 6.1: Performance evaluation of multi-view action recognition. The maximum, averaged values and standard deviation of the proposed methods (Mltp+BoW and Mltp+sparse coding) are listed in the table. Symbol '*' indicates results are the highest accuracies can be found in the original paper.

Method		# Cameras used for testing			
		1	2	3	4
Liu et al. Liu et al. (2008)		**82.8%**	-	-	-
Liu & Shah Liu and Shah (2008b)		73.7%	-	-	82.8%
Yan et al. Yan et al. (2008)		64.0%	71%*	60%*	78%
Weinland et al. Weinland et al. (2007)		63.9%	81.3%*	70.2%*	81.3%
Srivastava et al. Srivastava et al. (2009)		69.1%	75.6%	79.1%	81.4%
Leave-One-Out (Mltp+)					
BoW	max	**77.6%**	**82.4%**	**83.9%**	**84.5%**
	avg	**75.7%**	80.3%	**82.6%**	**84.5%**
	std	1.27	1.32%	1.16%	0
Sparse Coding	max	**80.9%**	**85.2%**	**85.2%**	**86.4%**
	avg	**77.7%**	**82.9%**	**84.6%**	**86.4%**
	std	3.6%	2.25%	0.52%	0
Cross Subjects (Mltp+)					
BoW	max	73.4%	78.5%	77.7%	78.4%
	avg	68.9%	74.7%	77.2%	78.4%
	std	3.3%	2.7%	0.83%	0
Sparse Coding	max	**76.1%**	**82.2%**	**83.6%**	**84.0%**
	avg	**73.8%**	80.4%	**82.8%**	**84.0%**
	std	2.8%	1.6%	0.76%	0

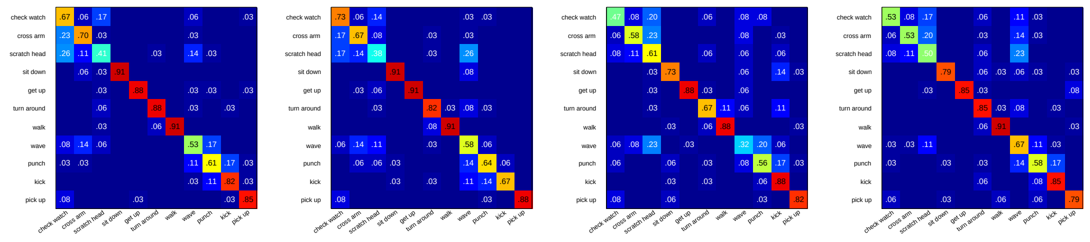
Table 6.1. Recall that the standard deviation of recognition accuracies using different combinations of views reflects the robustness of the evaluated algorithms - the smaller the derivation, the less affected the algorithm to the selection of different views.

BoW vs Sparse Coding

The performance of the sparse coding representation and the BoW representation are compared and listed in Table 6.1. It shows that the sparse coding representation consistently outperforms the BoW representation in all testing cases. Especially, if the number of actors used for training reduces from 9 to 5, the sparse coding



(a) Results based on Bag-of-Words



(b) Results based on Sparse Coding

Figure 6.5: The confusion matrix by using different representation methods when only one camera is available at test step. From left to right, the confusion matrices correspond to the results from camera 1 to camera 4.

representation outperforms the BoW representation by 5%. This is because the sparse coding representation introduces less quantization error during the feature representation step than BoW.

Figure 6.5 shows the confusion matrices of single view recognition by using the Leave-one-out strategy. From left to right, the confusion matrices correspond to camera 1 to camera 4. It is interesting to note that actions with large motions such as “sit down”, “get up”, “turn around” and “walk” have high accuracy for both BoW representation and sparse coding representation. In addition, sparse coding representation performs much better than BoW representation for camera 1 and camera 2. For camera 3 and camera 4, both representations are competitive to each other.

Algorithm Comparison

We further compare the results of the proposed framework with state-of-the-art algorithms using the Leave-one-out strategy. Accuracies obtained by our approaches that are higher than the literature are marked in bold fonts. Note that the work of Liu et al. (2008) uses knowledge transfer based on extracted features to achieve high accuracy, which has not been used in our system. Experimental results indicate that the proposed methods perform better than existing distributed multi-view action recognition algorithms, and even superior to the complicated algorithms that require preprocessing and large data transmission by 4% $\sim$ 14%. For example, the proposed methods achieve higher accuracies using 2 views than other algorithms using 4 views, as shown in Table 6.1. In addition, the proposed algorithms are less sensitive to the different combinations of views selected for testing, since the average accuracies are close to the highest accuracies and the standard deviations are small. Different from the work of Yan et al. (2008); Weinland et al. (2007), the performance of the proposed algorithms is steadily improved when more views are available at the testing stage. This further indicates that information from various views can be well represented and fused in our systems. The improvements of recognition accuracy are more obvious from 2 views to 3 views or 1 view to 2 views, than 3 views to 4 views. In other words, using the proposed algorithms, 3 views are enough to achieve competitive accuracy.

Leave-One-Out vs Cross Subjects

To evaluate the influence of the number of training actors to the results, we use both the Leave-One-Out strategy and the cross subjects strategy on multiple view recognition as shown in Table 6.1. For cross subjects, we randomly choose 5 actors for training and use the remaining 5 actors for testing. Note that we run each experiment 10 times to obtain the average accuracy for the cross subjects strategy. The performances of cross subjects are worse than the Leave-One-Out strategy since the information during training phase is limited when only half actors are used. Compared to the Leave-One-Out strategy, the accuracies of cross subjects by using the BoW representation decrease by 6%, as compared to the 2% by using the sparse coding representation. Therefore, sparse coding representation is more robust than

BoW. In addition, the performance of cross subjects by using sparse coding still outperforms the state-of-the-art algorithms that use more actors for training.

6.3.3 Robustness of the Proposed Framework

We further evaluate the robustness of the proposed framework by using more complicated experiments. As shown in Figure 6.4, the positions of four cameras are distinctive. Although the view angles of the four cameras are overlapping, there are still things can only be seen from one camera but not the rest. In addition, each actor performs actions three times by choosing different orientations. In this case, not only the positions of actors in each camera are different, the body movements of the same action can also be different. In this paper, we design two experiments for robustness evaluation. In the first experiment, we use two orientations for training, and test the samples on the untrained remaining orientation. In the second experiment, we use different numbers of cameras for training, and test on the remaining unseen cameras. Note that for both experiments, we use 9 actors for training and the remaining actor for testing. In other words, not only the views or orientations for testing are distinctive from that of training, the actors for training are not included for testing.

Cross Orientation

Table 6.2 shows the results by using different orientations for testing. In each case, we use two orientations for training and the rest one orientation for testing. Note that the viewpoints are different from different orientations as shown in Figure 6.4. Experimental results indicate that the proposed framework is tolerant to variance in orientations, since the accuracies in Table 6.2 are competitive to the results in Table 6.1. It is quite significant that even when the orientation used for testing is different from the orientations used for training, the performance of the proposed framework is not influenced much. In addition, the recognition rates increase with the number of cameras used during the testing phase. The performances of sparse

Table 6.2: Performance evaluation of multi-view action recognition under different orientations.

Test	Method	# Cameras used for testing			
		1	2	3	4
One	BoW	74.1%	76.2%	77.7%	78.2%
	Sparse	**77.0%**	**82.6%**	**85.2%**	**86.4%**
Two	BoW	73.6%	77.3%	79.1%	79.1%
	Sparse	76.8	81.2%	83.6%	84.5%
Three	BoW	75.7%	79.2%	79.5%	84.5%
	Sparse	75.5	79.2%	80.7%	83.6%

coding representation for orientation 1 and orientation 2 are much better than the BoW representation, and competitive for orientation 3.

Cross Views Recognition

In this experiment, we train the actions from selected views and test on the unseen views. For example, we use 3 views from 9 actors for training, and choose the fourth view from the remaining actor for testing. Table 6.3 lists the average accuracies when different numbers of views are available during the testing phase. If we choose 3 views for testing, only 1 view is available during the training phase. The accuracies are still acceptable for both feature representation models. The experimental results further indicate that the recognition rates increase with the number of unseen views available during the testing phase. In other words, even though the information learned is quite limited during the training phase, the performance of the proposed framework can still take advantage of the different information provided by multiple views during the test phase and improve the performance. It also reveals that our framework can be used to improve the accuracy if more unseen views are involved during the test phase.

Table 6.3: Performance evaluation of multi-view action recognition using different number of views for training and the remaining views for testing.

Method		# Unseen Cameras		
		1	2	3
Mltp+BoW	max	72.1%	74.5%	76.4%
	avg	67.4%	66.3%	73.5%
	std	5.0%	10.1%	2.4%
Mltp+Sparse Coding	max	**80.0%**	**81.2%**	**79.1%**
	avg	**70.8%**	**72.5%**	**75.0%**
	std	8.3%	10.6%	3.4%

6.3.4 Memory and Bandwidth Requirements and Computational Cost

For the 128-dimension Mltp, the trained codebook or dictionary need to be stored on every camera, which is 180 codewords for BoW and 512 atoms for sparse coding. Then, every camera only transmits the calculated histogram to the base station for classification. Therefore, the (Mltp+BoW) transmits 4 histograms (from 4 cameras) with 180 bins to the base station, and (Mltp+Sparse Coding) transmits 4 histograms with 512 bins to the base station. Using the same calculation as Srivastava et al. (2009), both the memory and bandwidth requirements for our system can be calculated. Table 6.4 compares the memory and bandwidth consumption of our models and the ones in Srivastava et al. (2009); Weinland et al. (2007). It shows that our models require even less resource on memory and transmission bandwidth. The difference of memory and bandwidth consumption of BoW and sparse coding is caused by the size difference between the learned dictionary and codebook.

The Mltp descriptor takes advantage of the LBP-like features in terms of computational efficiency. From Eq. 3.4, we see that the calculation of Mltp is focused on simple pixel comparison within local region. It can be considered as having the same complexity as gradient calculation. Furthermore, the histogram representation of Mltp does not need any angle calculation as most local descriptors require. We

Table 6.4: Memory and bandwidth requirements for our models.

Methods	Memory($Mbytes$)	Bandwidth($Kbytes/s$)
Srivastava et al. (2009)	0.315	2.7
Weinland et al. (2007)	1.72	47
Mltp+BoW	**0.092**	**0.96**
Mltp+sparse coding	0.26	2.7

test the computational cost of the proposed method on a $390 \times 291 \times 300$ video. The run-time estimates were obtained on a PC with Intel Xeon Processor X5550 (Nahalem) 2.66 GHz Quad Core and 12 GB memory. The coding language is C++. The proposed method has around 15 fps run time which shows the possibility of its real-time application.

6.4 Summary

In this chapter, we proposed a framework to perform distributed multiple view action recognition. We first use the presented feature descriptor Mltp to extract the motion information among frames. Combined with the Cuboid detector, Mltp helps to remove the non-informative patches generated from background. Next, two feature representation schemes, BoW and sparse coding were used to obtain the histogram representation of Mltp. The proposed approaches do not need complicated operation for feature extraction or description. Experimental results indicated that our algorithms outperform existing algorithms for distributed multi-view action recognition. Especially, the sparse coding representation was robust and tolerant to the misalignment between training and testing views. Compared with the existing algorithms, our proposed framework required less memory and bandwidth consumption. In addition, communication among local cameras was unnecessary for information fusion in the proposed system.

Chapter 7

Application 3: RGB-D Human Action Recognition

7.1 Introduction

Traditional human action recognition approaches focus on learning distinctive feature representations for actions from labelled videos and recognizing actions from unknown videos. However, it is a challenging task to label unknown RGB sequences due to the large intra-class variability and inter-class similarity of actions, cluttered background, possible camera movements and illumination changes.

Recently, the introduction of cost-effective depth cameras provides a new possibility to address difficult issues in traditional human action recognition. Compared to the monocular video sensors, depth cameras can provide 3D motion information so that the discrimination of actions can be enhanced and the influence of cluttered background and illumination variations can be mitigated. Especially, the work of Shotton et al. Shotton et al. (2011) provided an efficient human motion capturing technology to accurately estimate the 3D skeleton joint positions from a single depth image, which are more compact and discriminative than RGB or depth sequences. As shown in Figure 7.1, the action "drink" from the MSR DailyActivity3D dataset Wang

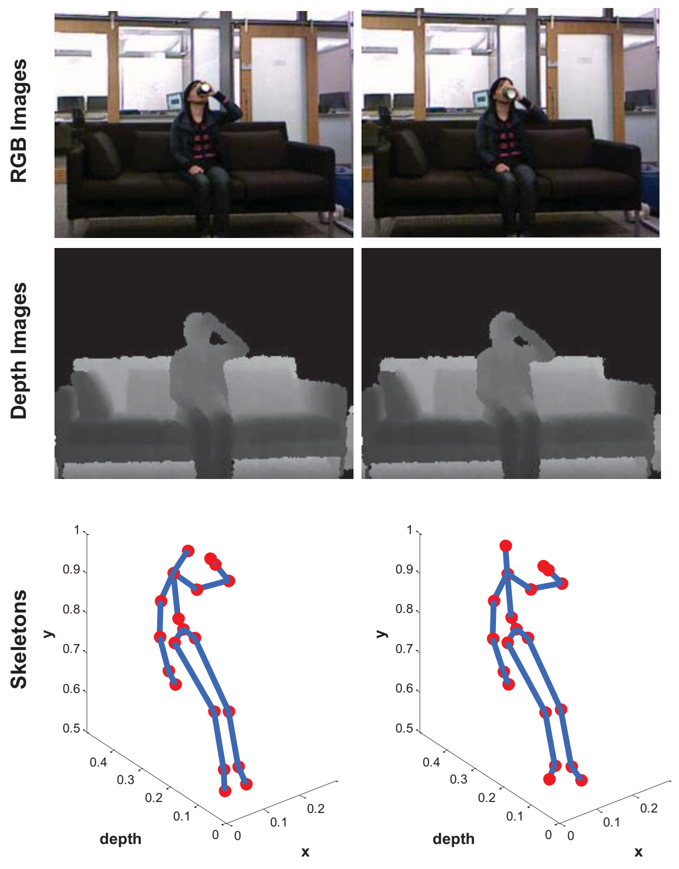


Figure 7.1: Sample images obtained by different cameras for the action "drink". The 3D joints are estimated by the method in Shotton et al. (2011).

et al. (2012c), can be well reflected from the extracted 3D joints by comparing the joints "head" and "hand" in the two frames. However, it is not that straightforward to tell the difference between the two frames from the depth maps or color images.

Although with strong representation power, the estimated 3D joints also bring challenges to perform depth-data based action recognition. For example, the estimated 3D joint positions are sometimes unstable due to the noisy depth maps. In addition, the estimated 3D joint positions are frame-based, which require representation methods to be tolerant to the variations in speed and time of actions.

To extract robust features from estimated 3D joint positions, relative 3D joint features have been explored and achieved satisfactory performance Wang et al.

(2012c); Farhadi and Tabrizi (2008); Yang and Tian (2012). To represent depth sequences with different lengths, previous research mainly focused on temporal alignment of sequences Farhadi and Tabrizi (2008); Lv and Nevatia (2006); Muller and Roder (2006) or frequencies evolution of extracted features Wang et al. (2012c) within a given period. However, the limited lengths of sequences, the noisy 3D joint positions, and the relatively small number of training samples may cause the overfitting problem and make the representation unstable.

In this chapter, a new framework is proposed for depth-based human action recognition. Instead of modeling temporal evolution of features, our work emphasizes on the distributions of representative features within a given time period. To keep the temporal information during the feature representation, a temporal pyramid matching (TPM) based on a pooling function $\mathbf{z} = \mathcal{F}(\mathbf{X})$ is used to yield the histogram representation for every depth sequence.

In the proposed framework, the feature representation is an important step and contribute most to the performance. In this chapter, both the classic sparse coding based representation and the proposed DL-GSGC in Chapter 4 are applied in the framework. Note that the depth features used in this chapter is based on the 3D relative joint positions, and details can be referred to Chapter 3. In addition, the performance on feature fusion from both RGB videos and depth maps are explored.

7.2 Sparse Coding Temporal Pyramid Matching

Given the depth features using the relative 3D joint positions (introduced in Chapter 3), the next step is feature representation. The main challenging issue is how to solve the "time alignment" problem since sequences are of various lengths and actions are of different paces. The common solutions using temporal modeling, such as the DTW Muller and Roder (2006) or HMM Lv and Nevatia (2006), are not optimal when only limited number of training samples is available which is always the case in the existing databases. Regardless of temporal order, the Bag-of-Words model can be applied to the extracted features. However, large quantization errors will incur due to the usage of K-means in the clustering process. In addition, all the temporal information is discarded in BoW. In this chapter, we propose to use a sparse coding scheme to learn representative features to reduce the quantization error. Next, a temporal pyramid structure based on max pooling is used for histogram representation of features. The whole representation process is referred to as Sparse coding Temporal Pyramid Matching (ScTPM).

7.2.1 Feature Representation

From the feature extraction step, every frame generates a 20×57 3D joint feature, where 20 is the number of joints and 57 is the dimension of the 3D joint feature vector. Since different joints have distinctive features even for the same action, we train 20 dictionaries for the 20 joints separately by Eq.4.1. To be specific, we use each joint position as reference joint, and calculate the relative positions of the rest 19 joints using Eq. 3.8. Then, each joint is treated independently to solve the sparse coding problem.

We first collect enough samples from the training dataset to learn the dictionary $\mathbf{D}_i$ for each joint i. Next, we obtain the sparse codes by solving Eq. 6.2 for every joint on each frame. After that, each frame can be represented by $K-$dimensional sparse codes. Then, a temporal pyramid matching based on a pooling function $\mathcal{F}(\mathbf{X})$ is used to yield the histogram representation $\mathbf{z}$ for every joint. Note that different pooling functions construct different image statistics. As noted in (Yang et al., 2009a), *max* pooling produces better performance than other pooling methods when using, for example, the square root of mean squared statistics and the mean of absolute values, as evaluation metrics. Therefore, we adopt the *max* pooling function in Eq. 6.4 to generate the histogram representation.

To keep the temporal order, we use a temporal pyramid matching approach to perform max pooling. First of all, the number of pyramid levels should be

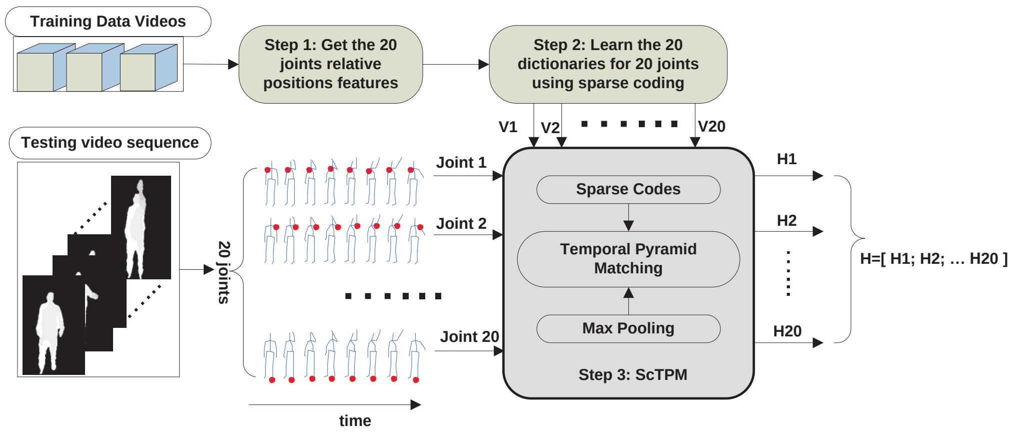


Figure 7.2: Illustration of the proposed feature extraction and representation by sparse coding and max pooling on 3D joint features.

defined. At level n, the video sequence is divided into 2^{n-1} segments. Next, on each segment at each level, the max pooling function is applied to generate the histogram representation $\mathbf{z}$ for that segment. Finally, all the histograms generated from all the segments are concatenated together to form a long histogram as the feature representation for this video sequence by one joint. There are two advantages of using this temporal pyramid: 1) the temporal information is well kept by segments; 2) it is not sensitive to the temporal shift or misalignment since lower level of the pyramid keeps less temporal information.

Figure 7.2 shows the process of the proposed scheme. The temporal pyramid approach gives a histogram representation of the whole input video sequence for every joint. Next, a long vector $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_{20}]$ is formed by concatenating all the 20 histograms $\mathbf{z}_i$ as the final histogram representation for one video.

7.3 DL-GSGC Temporal Pyramid Matching

Although sparse coding reduces the quantization error caused by K-means, it does not take the classification error into construction. As proposed in Chapter 4, we propose a discriminative dictionary learning algorithm DL-GSGC for better performance. In

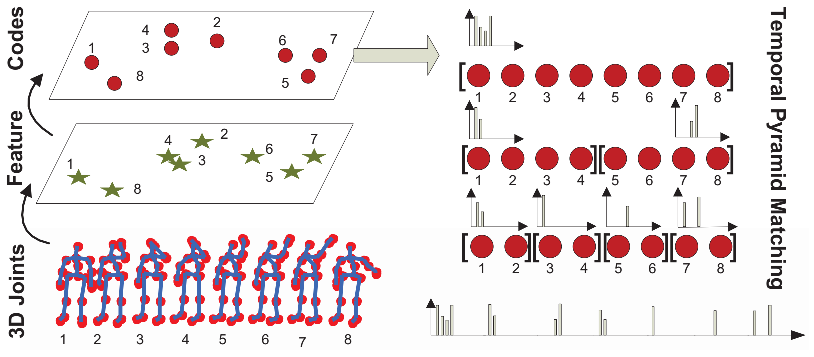


Figure 7.3: Temporal pyramid matching based on sparse coding.

this chapter, we will discuss how to apply the proposed algorithm in the problem of action recognition.

Similar to the ScTPM framework, we also uses the temporal pyramid matching to keep the temporal information. Also, the max pooling is selected as many literature work did Yang et al. (2009a); Wang et al. (2010). TPM divides the video sequence into several segments along the temporal direction. Histograms generated from segments by max pooling are concatenated to form the representation, as shown in Figure 7.3.

Different from the ScTPM structure that we train dictionary for each joint, the DL-GSGC based framework works well when we use the center hip as reference joint and calculate the relative positions. Therefore, for the DL-GSGC based framework, we train one dictionary, compared to the 20 dictionaries used in ScTPM.

During the training stage, the dictionary $\mathbf{D}$ can be calculated by alternatively updating the coefficients and dictionary atoms by Eqs. 4.9 and 4.16. Details of the optimization process can be referred to Chapter 4.

After constructing the discriminative dictionary $\mathbf{D}$ by using the proposed DL-GSGC algorithm, the coefficients for a given feature $\mathbf{y}$ can be calculated by solving the following optimization problem:

$$\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2 + \lambda_1 |\mathbf{x}|_1 + \lambda_2 \|\mathbf{x}\|_2^2 + \lambda_3 \sum_{i=1}^{J} \|\alpha_i - \mathbf{x}\|_2^2 w_i \tag{7.1}$$

Similar to the derivation in Chapter 4, the feature-sign search method Lee et al. (2007) can be used to obtain the coefficients.

7.4 Classification

For classification, the Support Vector Machine (SVM) Chang and Lin (2001) is the most popular classifier with strong discriminative power. The SVM is a binary classifier that aims to learn a decision function

$$f(\mathbf{z}) = \sum_{i=1}^{n} \alpha_i \mathcal{K}(\mathbf{z}, \mathbf{z}_i) + b \tag{7.2}$$

where $\{(\mathbf{z}_i, y_i)\}_{i=1}^{n}$ is the training set and $y_i \in \{-1, 1\}$ is the label.

In literature, the Bag-of-Words (BoW) representation must be applied together with a particular type of nonlinear Mercer kernels, e.g., *Chi-square kernel* to obtain good performance Yang et al. (2009a). Equation 7.3 shows the kernel function:

$$K(H_i, H_j) = \exp\left(-\frac{1}{2M} \sum_{n=1}^{S} \frac{(h_{in} - h_{jn})^2}{h_{in} + h_{jn}}\right) \tag{7.3}$$

where $H_i = \{h_{in}\}$ and $H_j = \{h_{jn}\}$ are the histograms calculated by BoW, and S is the number of codewords. M is the mean value of distances between all training samples Zhang et al. (2007). However, the complexity of nonlinear kernel is $O(n^2 \sim n^3)$ in training and $O(n)$ in testing, where n is the training size. For the linear kernel used in this chapter, the complexity is $O(n)$ in training and a constant in testing Yang et al. (2009a).

To speed up the process of training and testing, a linear kernel $\mathcal{K}(\cdot,\cdot)$ is used on the calculated histogram Yang et al. (2009a). Note that the accuracy can be even better if a nonlinear kernel is used at the price of speed. Given the training data $\mathbf{z}_i$ and the test sample $\mathbf{z}$, the kernal can be calculated as

$$\mathcal{K}(\mathbf{z},\mathbf{z}_i) = \mathbf{z}_i^T\mathbf{z} \tag{7.4}$$

Combining Eqs. 7.2 and 7.4, the binary SVM decision function can be further represented as

$$f(\mathbf{z}) = \left(\sum_{i=1}^{n}\alpha_i\mathbf{z}_i\right)^T\mathbf{z} + b = \mathbf{w}^T\mathbf{z} + b \tag{7.5}$$

For multi-class, the linear SVM is equivalent of learning L linear functions $\{\mathbf{w}_c^T\mathbf{z}|c \in \mathcal{Y}\}$, given the training data $\{(\mathbf{z}_i, y_i)\}_{i=1}^n$, $y_i \in \mathcal{Y} = 1,\ldots,L$. For a test sample $\mathbf{z}$, its class label is predicted by Yang et al. (2009a)

$$y = \max_{c\in\mathcal{Y}} \mathbf{w}_c^T\mathbf{z} \tag{7.6}$$

As used in Yang et al. (2009a), the one-against-all strategy is adopted to train the L linear SVMs by solving the optimization problem

$$\min_{\mathbf{w}_c} J(\mathbf{w}_c) = \|\mathbf{w}_c\|^2 + C\sum_{i=1}^{n}\ell(\mathbf{w}_c; y_i^c, \mathbf{z}_i) \tag{7.7}$$

where $\ell(\mathbf{w}_c; y_i^c, \mathbf{z}_i) = [\max(0, \mathbf{w}_c^T\mathbf{z}\cdot y_i^c - 1)]^2$ is the differentiable quadratic hinge loss and can be solved by gradient-based optimization methods.

7.5 RGB-D Human Action Recognition

In this chapter, we evaluate both the DL-GSGC based action recognition and the ScTPM based action recognition. Although the previous sections mainly focus on the depth based representation, the proposed framework can also be applied on the

RGB features. In this chapter, we conduct the proposed ScTPM on the CS-Mltp features that have been proposed in Chapter 3. First, enough features are collected from the training samples to learn the dictionary by Eq. 6.3. Next, sparse coefficients can be calculated for any given CS-Mltp features by Eq. 6.2. Similar to the 3D joint feature representation, the three-level temporal pyramid combined with max pooling technique is applied on every RGB video and a final histogram representation is formed by concatenating all the segments from the given RGB video.

To perform RGB-D human action recognition, we fuse the features of depth maps and color images at two levels, 1) a feature-level fusion where the histograms generated from two sources are simply concatenated together to form a longer histogram representation as the input to classifier, and 2) a classifier-level fusion where the classifiers for the two sources are trained separately and classifier combination is performed subsequently to generate final result.

To combine the classifier outputs of both the 3D joint features and the CS-Mltp features, we first interpret the classifier outputs as a probability measure. For each test sample $\mathbf{z}$, since the degree of confidence that it belongs to class i can be measured by $\mathbf{w}_i^T\mathbf{z}$, the probability representation of the classifier can be defined as:

$$p_i = \frac{\exp(\mathbf{w}_i^T\mathbf{z})}{\sum_{j=1}^{L}\exp(\mathbf{w}_j^T\mathbf{z})} \tag{7.8}$$

where p_i gives the probability of sample $\mathbf{z}$ belongs to class i.

Next, the *product* and *sum* aggregation operators Kittler et al. (1998) are applied as two classification confusion operators for evaluation. Different operators will generate different classification results. Let h_1 and h_2 be classifiers for the 3D joint features and the CS-Mltp features, respectively. The *product* operator can be expressed as:

$$y = \max_{c\in\mathcal{Y}} P(s_c|h_1)P(s_c|h_2) \tag{7.9}$$

if $P(s_i)$ is the prior probability and set to be equal among all classes, i.e., $P(s_1) = \ldots = P(s_L)$. The value of $P(s_c|h_i)$ can be calculated by Eq. 7.8. Similarly, the *sum* operator can be simplified into:

$$y = \max_{c \in \mathcal{Y}} (P(s_c|h_1) + P(s_c|h_2)) \tag{7.10}$$

7.6 Experiments

Two benchmark datasets, *MSR-Action3D dataset* Li et al. (2010) and *MSR Daily-Activity3D dataset* Wang et al. (2012c), are used for evaluation purpose. For both datasets, we compare the performance from two aspects, the effectiveness of the proposed framework (i.e., ScTPM and DL-GSGC+TPM) as compared to state-of-the-art approaches, the effectiveness of the proposed dictionary learning algorithm (i.e., DL-GSGC) as compared to state-of-the-art DL methods. In addition, since the second dataset also contains the RGB video sequence, we further compare the performance between using the RGB sequence and the depth map sequence, and the different fusion schemes that combine both RGB features and depth features. In all experiments, the proposed approaches constantly outperform the state-of-the-art.

7.6.1 Parameters Setting

For the 3D joint features, the threshold t_s and t_e need to be pre-defined for the key frame detection step as described in Chapter 3. We randomly choose 10 samples from the system, and find the values $t_s = 0.15 \sim 0.25$ and $t_e = 0.2 \sim 0.3$ are acceptable. We thus choose $t_s = 0.18$ and $t_e = 0.25$ for all the experiments. Note that the recognition performance is not very sensitive to the values of these two parameters and thus they can be coarsely selected.

For the Cuboid detector, the value of σ is set to be 2, and τ to be 3. Note that the patch size is then fixed to be $12 \times 12 \times 18$. We detect 200 cuboid from every video.

For the CS-Mltp, the radius of neighboring pixels is set to be 2. The frame step Δt is set to be 3. The threshold T_{cs} in Eq. 3.5 is set to be 0.1. To keep the spatial and temporal information, we divide the extracted cuboids into $2 \times 2 \times 2$ cells. Note that the recognition performance is not very sensitive to the values of these parameters and thus we choose to use fixed values to all the datasets in our experiments.

For the DL-GSGC dictionary learning, there are three parameters: λ_1, λ_2 and λ_3 that corresponding to group sparsity and geometry constraints, respectively. According to our observation, the performance is best when $\lambda_1 = 0.1 \sim 0.2$, $\lambda_2 = 0.01 \sim 0.02$ and $\lambda_3 = 0.1 \sim 0.2$. Initial sub-dictionaries are obtained by solving $\|\mathbf{Y}_i - \mathbf{D}_i\mathbf{X}_i\|_F^2 + \lambda_1 \sum_{j=1}^{N_i} |\mathbf{x}_j^i|_1 + \lambda_2\|\mathbf{X}_i\|_F^2$ using online dictionary learning Mairal et al. (2009) and the number of atoms is set to be 15 for each sub-dictionary. For geometry constraint, 1500 features are used to build the templates. Note that all these features are collected from a subset of training samples, and cover all the classes. Compared to the total number of training features, the number of templates is relatively small. For the TPM, the depth sequence is divided into 3 levels with each containing 1, 2 and 4 segments, respectively.

7.6.2 MSR Action3D Dataset

The MSR-Action3D dataset Li et al. (2010) contains 567 depth map sequences. There are 20 actions performed by 10 subjects. For each action, the same subject performs it three times. The size of the depth map is 640×480. Figure 7.4 shows the depth sequences of three actions: *draw x*, *draw circle*, and *forward kick*, performed by different subjects. For all experiments on this dataset, the 1500 templates used for geometry constraint are collected from two training subjects.

Compared with State-of-the-art Algorithms

We first evaluate the proposed algorithm (**ScTPM and DL-GSGC + TPM**) in terms of recognition rate and compare it with the state-of-the-art algorithms that

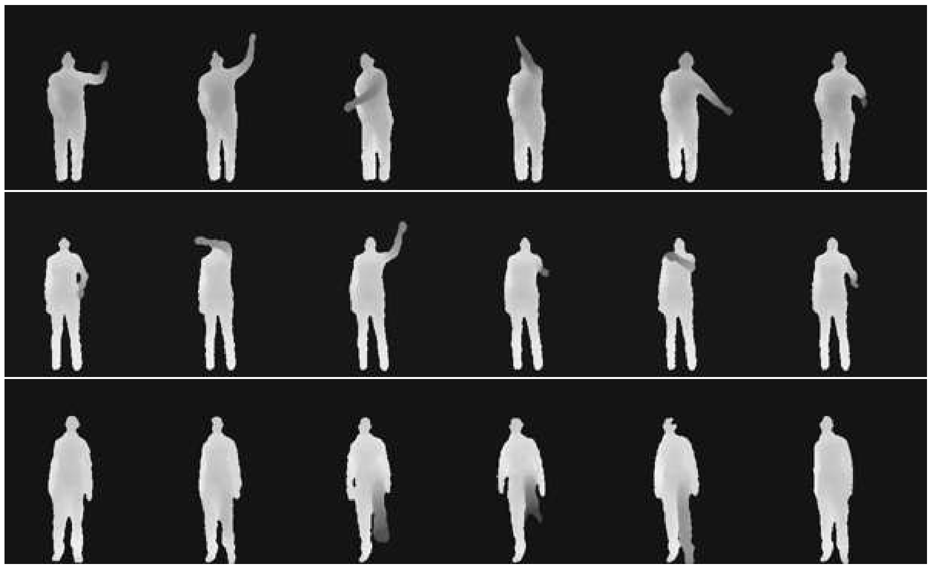


Figure 7.4: Sample frames from the MSR Action3D dataset. From top to bottom, frames are respectively from actions: Draw X, Draw Circle, and Forward Kick.

have been applied on the MSR Action3D dataset. For fair comparison, all results are obtained using the same experimental setting: 5 subjects are used for training and the rest 5 subjects are used for testing. In other words, it is a cross-subject test. Since subjects are free to choose their own styles to perform actions, there are large variations among training and testing features.

Table 7.1 shows the experimental results by various algorithms. Our proposed method achieves the highest recognition accuracy as 96.7%, and accuracies reduced to 95.2% and 94.2% if only one constraint is kept. Note that the work of Wang et al. (2012c) required a feature selection process on 3D joint features and a multiple kernal learning process based on the SVM classifier to achieve the accuracy of 88.2%, whereas our algorithm use simple 3D joint feature, combined with the proposed feature representation and a simple linear SVM classifier. Therefore, the proposed dictionary learning method and framework is effective for the task of depth-based human action recognition. Compared to the ScTPM framework that uses 20 dictionaries, the Dl-GSGC+TPM performs better and it just uses one dictionary.

Table 7.1: Evaluation of algorithms on the cross subject test for the MSRAction3D dataset.

Method	Accuracy
Recurrent Neural Network Martens and Sutskever (2011)	42.5%
Dynamic Temporal Warping Muller and Roder (2006)	54.0%
Hidden Markov Model Lv and Nevatia (2006)	63.0%
Bag of 3D Points Li et al. (2010)	74.7%
Histogram of 3D Joints Farhadi and Tabrizi (2008)	78.97%
Eigenjoints Yang and Tian (2012)	82.3%
STOP Feature Vieira et al. (2012)	84.8%
Random Occupy Pattern Wang et al. (2012a)	86.2%
Actionlet Ensemble Wang et al. (2012c)	88.2%
DL-GSGC+TPM	**96.7%**
DL-GSGC+TPM($\lambda_2 = 0$)	**95.2%**
DL-GSGC+TPM($\lambda_3 = 0$)	**94.2%**
ScTPM	**93.83%**

Figure 7.5 shows the confusion matrix of the DL-GSGC+TPM. Actions of high similarity get relative low accuracies. For example, action *Draw Tick* tends to be confused with *Draw X*.

Comparison with Sparse Coding Algorithms

To evaluate the performance of the proposed **DL-GSGC**, classic DL methods are used for comparison. These methods include K-SVD Aharon et al. (2006), sparse coding used for image classification based on spatial pyramid matching (ScSPM) Yang et al. (2009a), and the dictionary learning with structured incoherence (DLSI) Ramirez et al. (2010). In addition, for all the evaluated DL methods, the feature-sign search method is used for coefficients calculation, the TPM and max pooling are used to obtain the vector representation, and the linear SVM classifier is used for classification. We refer to the corresponding algorithms as K-SVD, ScTPM and DLSI for simplicity.

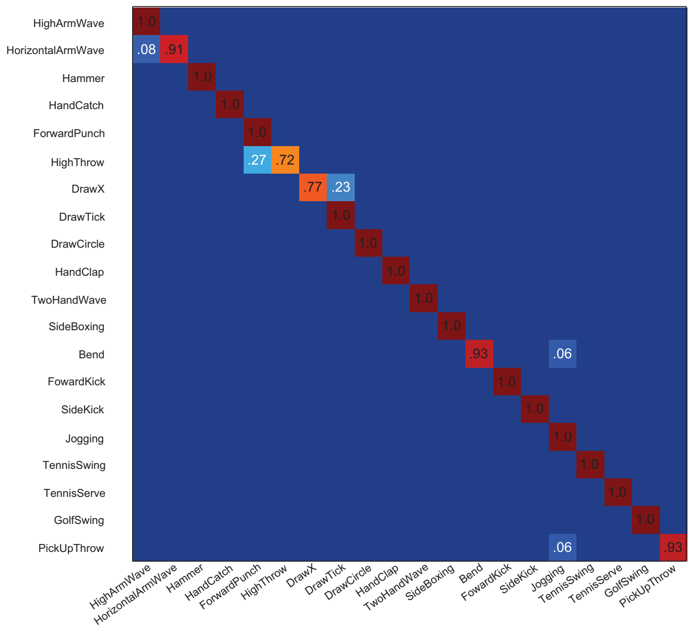


Figure 7.5: Confusion matrix for MSR Action3D dataset.

Comparisons are conducted on three subsets from the MSR Action3D dataset, as described in Li et al. (2010). For each subset, 8 actions are included. All the subsets(AS1, AS2 and AS3) are deliberately constructed such that similar movements are included within the group while A3 further contains complex actions with large and complicated body movements. On each subset, three tests are performed by choosing different training and testing samples. Since each subject will perform the same action 3 times, Test1 and Test2 choose 1/3 and 2/3 samples for training respectively. Test3 uses the cross subjects setting, which is the same as described in Sec. 7.6.2. Compared with Test1 and Test2, Test3 is more challenging since the variations are larger between training and testing samples.

Table 7.2 shows the results on the three subsets. Note that the overall accuracies based on all actions (20 actions) are also provided for each test. It shows that the performance of **DL-GSGC** is superior to other sparse coding algorithms in terms of accuracies on all tests. In addition, class-specific dictionary learning methods, such as **DL-GSGC** and DLSI, perform better than methods learning a whole dictionary

Table 7.2: Performance evaluation of sparse coding based algorithms on three subsets. Method 1 is proposed in Li et al. (2010), method 2 is proposed in Farhadi and Tabrizi (2008), and method 3 is proposed in Yang and Tian (2012).

Dataset		1	2	3	K-SVD	ScTPM	DLSI	DL-GSGC
Test 1	AS1	89.5	98.5	94.7	98.8	98.8	97.4	**100**
	AS2	89.0	96.7	95.4	95.6	95.6	98.1	**98.7**
	AS3	96.3	93.5	97.3	98.8	98.8	99.4	**100**
	Overall	91.6	96.2	95.8	97.8	97.3	97.6	**98.9**
Test 2	AS1	93.4	98.6	97.3	100	100	98.8	100
	AS2	92.9	97.9	98.7	98	98	97.2	**98.7**
	AS3	96.3	94.9	97.3	100	100	100	100
	Overall	94.2	97.2	97.8	98.9	98.9	97.9	98.9
Test 3	AS1	72.9	87.9	74.5	92.4	96.6	96.6	**97.2**
	AS2	71.9	85.5	76.1	91.9	92.9	93.7	**95.5**
	AS3	79.2	79.2	96.4	95.5	96.4	96.4	**99.1**
	Overall	74.7	79.0	82.3	92.0	92.7	93.2	**96.7**

simultaneously for all classes (i.e., K-SVD and ScTPM). Moreover, the proposed framework (i.e., sparse coding + TPM), is effective for action recognition, since accuracies when using different sparse coding methods outperform the literature work in both Tables 7.1 and 7.2. Especially, our method outperforms other algorithms in Table 7.1 based on 3D joint features by 15% $\sim$ 17% on test 3.

7.6.3 MSR DailyActivity3D Dataset

The MSR DailyActivity3D dataset contains 16 daily activities captured by a Kinect device. There are 10 subjects in this dataset, and each subject performs the same action twice, once in standing position, and once in sitting position. In total, there are 320 samples with both depth maps and RGB sequences available. Figure 7.6 shows the sample frames for the activities: *drink*, *write* and *stand up*, from top to bottom. As shown in Figure 7.6, some activities in this dataset contain small body movements, such as *drink* and *write*. In addition, the same activity performed in different positions have large variations in the estimated 3D joint positions. Therefore, this dataset is

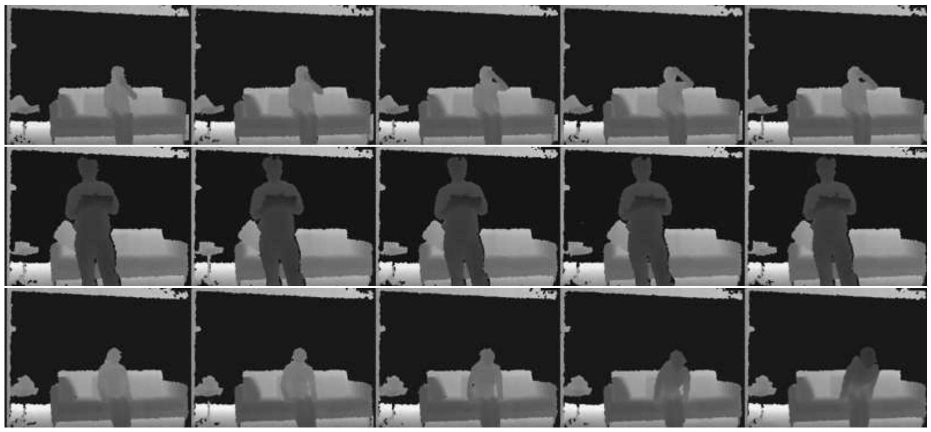


Figure 7.6: Sample frames for the MSR DailyActivity3D dataset. From top to bottom, frames are from actions: drink, write, and stand up.

more challenging than the MSR Action3D dataset. Experiments performed on this dataset is based on cross subjects test. In other words, 5 subjects are used for training, and the rest 5 subjects are used for testing. The number of templates is also 1500 which are collected from 2 training subjects. Table 7.3 shows the experimental results by using various algorithms.

Comparison with State-of-the-art Algorithms

We first compare the performance of **DL-GSGC** with literature work that have been conducted on this dataset. As shown in Table 7.3, the proposed method outperforms the state-of-the-art work Wang et al. (2012c) by 10% and the geometry constraint is more effective for performance improvement. In addition, other DL methods are incorporated in our framework for comparison, referred to as K-SVD, ScTPM and DLSI. Experimental results show that the performance of **DL-GSGC** is superior to other DL methods by 4% $\sim$ 5%. In addition, class-specific dictionary learning methods, i.e., DL-GSGC and DLSI, are better for classification task than K-SVD and ScTPM. Moreover, the proposed framework outperforms the state-of-the-art work Wang et al. (2012c) by 5% $\sim$ 10% when different DL methods are

Table 7.3: Performance evaluation of the proposed algorithm with eight algorithms. Algorithms marked with (*) are applied on RGB videos and all rest algorithms are applied on depth sequences.

Method	Accuracy
Cuboid+HoG*	53.13%
Harris3D+HOG/HOF*	56.25%
Dynamic Temporal Wrapping Muller and Roder (2006)	54%
3D Joints Fourier Wang et al. (2012c)	68%
Actionlet Ensemble Wang et al. (2012c)	85.75%
K-SVD	90.6%
ScTPM	90.6%
DLSI	91.3%
CS-Mltp (RGB)	65.63%
DL-GSGC	**95.0%**
DL-GSGC ($\lambda_2 = 0$)	**93.8%**
DL-GSGC ($\lambda_3 = 0$)	**92.5%**
ScTPM+CS-Mltp (**product**)	**90.63%**
ScTPM+CS-Mltp (**sum**)	**91.15%**
ScTPM+CS-Mltp (**concatenate**)	**92.5%**

used. Considering the large intra-class variations and noisy 3D joint positions in this dataset, the proposed framework is quite robust.

Comparison with RGB Features

Since both depth and RGB videos are available in this dataset, we also compare the performance of RGB features with that of depth features. For traditional human action recognition problem, spatio-temporal interest points based methods have been heavily explored. Two important steps are spatio-temporal interest point detection and local feature description. As for feature representation, Bag-of-Words representation based on K-means clustering is widely used. In this chapter, we follow the same steps to perform action recognition from RGB videos. To be specific, the classic Cuboid Dollar et al. (2005) and Harris3D I.Laptev et al. (2008) detectors are used for feature detection, and the HOG/HOF descriptors are used for description. The Bag-of-Words representation is used for feature representation.

Table 7.3 provides the recognition rates by using different feature detectors and descriptors on RGB video sequences. Compared to the other feature descriptors, the proposed CS-Mltp performs better on this dataset. However, compared with the performance of depth features, recognition rates on RGB sequences are lower. We argue the main reason to be that this dataset contains many actions with high similarity but small body movements, i.e., *Drink, Eat, Write, Readbook*. In this case, the 3D joint features containing depth information are more reliable than RGB features. In addition, the K-mean clustering method will cause larger quantization error than sparse coding algorithms. Therefore, depth information is important for the task of action recognition, and the sparse coding based representation is better for quantization.

Performance Evaluation of Various Fusion Schemes

The features extracted from different sources can be fused at two levels: feature level and classifier level. For the former, histograms generated by the two features are concatenated to form a longer histogram that used as input to the classifier. For the latter, the common combination rules such as *sum* and *product* are used for classifier fusion.

Table 7.3 shows the performance of different fusion methods compared with the single classifier. Performance is improved by combining the two classifiers even with the simple operator *sum*. In addition, concatenating both the CS-Mltp feature and 3D joint feature helps improve the recognition rate by 27% and 1.5% respectively compared to the usage of just the CS-Mltp feature or the 3D joint feature.

Figure 7.7 shows the accuracies when using the RGB and depth features separately, as well as the fused classifier for all 16 classes. For the CS-Mltp feature, it performs better on actions with large body movements such as *cheer up, walk, stand up* and *sit down*, but it fails to describe the small-scale body movements and performs worse on actions such as *call cell phone, use laptop* and *play guitar*. The 3D joint feature is less sensitive to the scale of body movements since its performance is stable on all 16

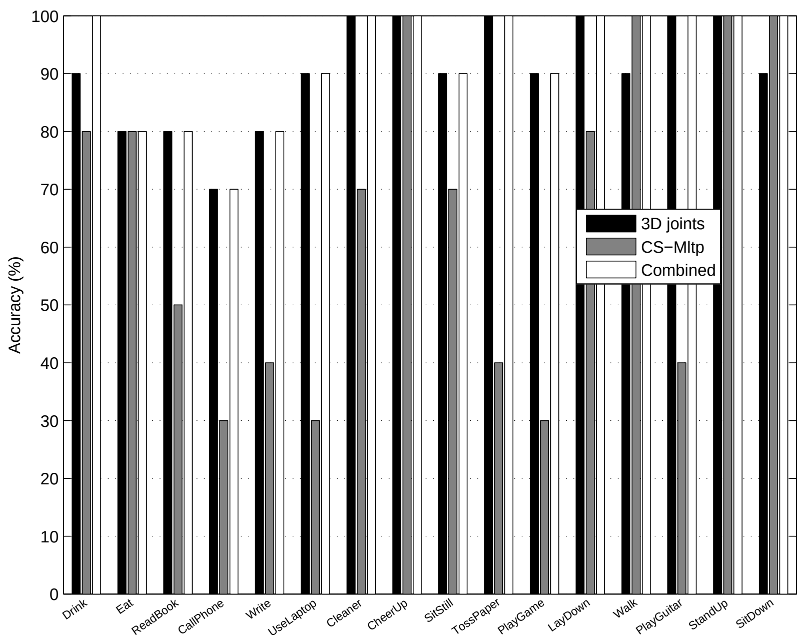


Figure 7.7: Comparison of the proposed RGB feature (CS-Mltp) and depth features (3D joints) used separately and in concatenation.

classes. By taking advantage of the two features, the accuracies of actions *drink* and *sit down* are improved after concatenating the two features together.

7.7 Summary

In this chapter, a new framework is used to perform human action recognition on RGB-D sensors. 3D joint features were extracted from the detected key frames on each depth sequence and CS-Mltp features were proposed to simultaneously extract both the spatial and temporal features from the RGB sequences. To achieve temporal alignment as well as preserve temporal information, a temporal pyramid approach combined with sparse coding and max pooling function was proposed for feature representation. Approximation error of extracted features was largely reduced by the linear and sparse combination of representative features. Compared to the BoW model, the proposed framework that based sparse coding algorithms perform better on depth based action recognition. In addition, the proposed DL-GSGC outperforms the other classic sparse coding algorithms (i.e, K-SVD and DLSI). Due to the strong representation capability, the proposed features and framework outperformed algorithms with complicated classifiers even with a simple linear SVM classifier.

Experimental results showed the significant improvement of the proposed algorithms and revealed the robustness of depth features and RGB features on different actions. On one hand, the proposed framework based on the 3D joint feature was tolerant to the large intra-class variations and small inter-class variations. On the other hand, the proposed framework based on the CS-Mltp features performed well on actions with large movements and was superior to other local representation approaches. In addition, simple fusion approaches, e.g., product, max and feature concatenation, used as aggregation operators, yielded better classification accuracies, especially on actions with small body movements or actions occluded by other objects.

Chapter 8

Conclusions and Future Work

This chapter summarizes the study findings of this dissertation and discusses possible future research directions. Section 8.1 addresses the main findings and evaluates to what extent the goals of this thesis have been achieved. Section 8.2 suggests possible future research directions.

8.1 Summary

The objective of this dissertation is to explore efficient algorithms for the task of human action recognition. Given different video input sources (i.e, RGB, depth or both), how to automatically generate the class labels for actions with the videos is a very challenging problem due to the large intra-class variations, camera motions, and cluttered background. In addition, the performance when both RGB and depth features are available needs to be evaluated and fusion schemes should be explored.

In order to extract more discriminative and efficient features, we first proposed the SMltp descriptor to describe detected 3D patches from RGB sequences. Different from the traditional LBP based descriptors with long vector representation, the SMltp contains 25 bins in total so that can be combined with existing spatial-temporal detectors. Compared to the popular HOG and HOF descriptor, the SMltp is more computationally efficient and generates better performance on benchmark datasets. Next, we propose a new descriptor, referred to as CS-Mltp, to capture motion information. Different from SMltp that compared on the intensity, the CS-Mltp compares the gradient difference among frames and assign a ternary pattern for measure. Compared to the SMltp, CS-Mltp is more sensitive to small motions and noise. We further proposed a discriminative dictionary learning method (DL-GSGC) for feature representation. Different from K-means that assign each query feature to the nearest cluster center, the proposed algorithm uses soft assignment by selecting atoms within the same class as the features to generate nonzero coefficients and keeping the spatial distribution simultaneously.

To evaluate the effectiveness of the proposed algorithms, three classic kinds of human action recognition are explored.

We first apply the proposed SMltp on the single view human action recognition. Compared to the popularly used feature descriptors, our proposed SMltp is much faster and achieves better performance when combined with the same feature detectors. Especially, combined with the dense sampling, the proposed algorithms significantly reduce the number of features, and achieve superior to or competitive with the state-of-the-art descriptors in realistic settings.

Next, the distributed human action recognition is used as another application of the proposed Mltp. In the distributed sensor network, our proposed algorithm is computational efficient, and robust to the positions or orientations of cameras. In addition, preprocessing and communication among local visual sensors are not necessary.

Finally, we apply the CS-Mltp and DL-GSGC on the task of RGB-D human action recognition. In addition, we propose a new framework for action recognition from depth maps. Using the 3D joint features and CS-Mltp features, both feature-level fusion and classifier-level fusion are explored. Experimental results indicate that recognition accuracy benefits from the fusion schemes. In addition, the proposed framework and dictionary learning algorithm outperforms existing state-of-the-art approaches on benchmark datasets.

8.2 Future Research

For the human action recognition based on RGB videos, existing work mainly focuses on BoW representation due to the large number of features extracted from each video sequence. Different from the depth features (i.e, 3D skeletons), many outliers existing in the extracted RGB features. In this case, how to remove the outliers plays an important role to the performance. In addition, although SMltp is tolerant to the camera motion to some extend, it still be influence when the camera motions is large and background is cluttered. In this case, possible feature selection algorithms can be helpful to improve the performance.

In this distributed camera network, although the proposed Mltp and sparse coding based framework performs better with less bandwidth consumption and computational cost, how to take advantage of the information among cameras can be further explored. Taking the advantage of transfer learning, it is possible to get good performance when use different training and testing views. In this case, how to apply the transfer learning algorithms to the distributed camera network can be essential, so that view invariant action recognition is possible since information from any smart camera is well explored and related to the rest.

As for the depth maps, features extracted from the 3D joint tracker are compact and discriminative. However, it is necessary to add more information when actions contain interaction between persons and objects. Therefore, besides the 3D joint positions, other features extracted directly from depth maps may need to be explored. In addition, when features from both RGB sequences and depth maps are both available, more complicated feature selection and feature fusion algorithms can be explored to fully explore the advantages of both sources.

Publication

1. **Jiajia Luo**, Wei Wang, and Hairong Qi, "Spatio-Temporal Feature Extraction and Representation for RGB-D Human Action Recognition", *Pattern Recognition Letter: Special Issue on Depth Image Analysis*, under revision.

2. **Jiajia Luo**, Wei Wang, and Hairong Qi, "Group Sparsity and Geometry Constraint Dictionary Learning for Human Action Recognition", *IEEE International Conference on Computer Vision (ICCV)*, Sydney, Australia, pp. 1-8, 2013.

3. **Jiajia Luo**, Wei Wang, and Hairong Qi, "Feature Extraction and Representation for Distributed Multi-view Human Action Recognition", *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 3, no. 2, pp. 145-154, 2013.

4. Wei Wang, **Jiajia Luo**, and Hairong Qi, "Action Recognition Across Cameras by Exploring Reconstructable Paths", *Seventh ACM/IEEE International Conference on Distributed Smart Cameras*, Palm Springs, California, USA, pp. 1-6, 2013.

5. **Jiajia Luo** and Hairong Qi, "Motion Local Ternary Pattern for Distributed Multi-view Human Action Recognition", *Sixth ACM/IEEE International Conference on Distributed Smart Cameras*, Hong Kong, China, pp. 1-6, 2012.

6. **Jiajia Luo**, Gongyao Wang, Hairong Qi, Yoshihiko Yokoyama, Peter K. Liaw, and Akihisa Inoue, "Interpreting temperature evolution of a bulk-metallic glass during cyclic loading through spatialtemporal modeling", *Intermetallics*, vol. 29, pp. 1-13, 2012.

7. Hairong Qi, Yilu Liu, Fran Li, **Jiajia Luo**, Li He, Kevin Tomsovic, Leon Tolbert, and Qing Cao, "Increasing the resolution of wide-area situational awareness of the power grid through event unmixing", *2011 44th Hawaii International Conference on System Sciences (HICSS)*, pp. 1-8, 2011.

8. Li He, **Jiajia Luo**, Hairong Qi, and Chiman Kwan, "A Comparative Study of Unsupervised Unmixing Algorithms to Detecting Anomalies in Hyperspectral Images", *2010 International Symposium on Spectral Sensing Research*, 2011.

9. **Jiajia Luo** and Hairong Qi, "Distributed Object Recognition via Feature Unmixing", *Fourth ACM/IEEE International Conference on Distributed Smart Cameras*, Atlanta, Georgia, pp. 73-80, 2010.

Bibliography

Aggarwal, J. K. and Ryoo, M. S. (2011). Human activity analysis: A review. *ACM Computing Surveys*, 43(3). 2

Aharon, M., Elad, M., and Bruckstein, A. M. (2006). K-svd: An algorithm for designing overcomplete dicitonaries for sparse representation. *IEEE Trans. Signal Processing*, 54:4311–4322. 100

Ahonen, T., Hadid, A., and Pietikainen, M. (2004). Face recognition with local binary pattern. *ECCV*. 16

Ali, S. and Shah, M. (2010). Human action recognition in videos using kinematic features and multiple instance learning. *IEEE Trans. PAMI*, 32(2):288–303. 62

Bay, H., Tuytelaars, T., and Gool, L. V. (2006). Surf: Speeded up robust features. *ECCV*. 15

Bondell, H. and Reich, B. (2008). Simultaneous regression shrinkage, variable selection and supervised clustering of predictors with oscar. *Biometrics*, 64(1):115–123. 41

Bregonzio, M., Gong, S., and Xiang, T. (2009). Recognition actions as clouds of space-time interest points. *CVPR*. 65

Chang, C.-C. and Lin, C. J. (2001). Libsvm: A library for suppot vector machines. 52, 70, 77, 94

Chen, P., Ahammad, P., Boyer, C., Huang, S., Lin, L., Lobaton, E., Meingast, M., Oh, S., Wang, S., Yan, P., Yang, A., Yeo, C., Chang, L., Tygar, D., and Sastry, S. (2008). Citric: A low-bandwidth wireless camera network platform. *ICDSC*, pages 1–8. 68, 76

Dalal, N., Triggs, B., and Schmid, C. (2006). Human detection using oriented histogram of flow and appearance. *ECCV*. 15

Dollar, P., Rabaud, V., Cottrell, G., and Belongie, S. (2005). Behavior recognition via sparse spatio-temporal features. *Proc. IEEE Int. Visual Surveillance and Performance Evaluation of Tracking and Survellance*, pages 65–82. 4, 12, 14, 21, 22, 49, 65, 70, 73, 104

Fang, Y., Wang, R., and Dai, B. (2012). Graph-oriented learning via automatic group sparsity for data analysis. *IEEE 12th International Conference on Data Mining.* 38, 39, 41

Farhadi, A. and Tabrizi, M. K. (2008). Learning to recognize activities from the wrong view point. *ECCV.* xi, 90, 100, 102

Gao, S., Tshang, I., Chia, L., and Zhao, P. (2010). Local features are not lonely - laplacian sparse coding for image classification. *CVPR.* 41

Georis, B., Maziere, M., Bremond, F., and Thonnat, M. (2004). A video interpretation platform applied to bank agency monitoring. *2nd Workshop on Intelligent Distributed Surveillance Systems (IDSS).* 2

Gilbert, A., Illingworth, J., and Bowden, R. (2011). Action recognition using mined hierarchical compound features. *IEEE Trans. PAMI*, 33(5):883–897. 63, 64, 67

Gorelick, L., Blank, M., Shechtman, E., Irani, M., and Basri, R. (2005). Efficient visual event detection using volumetric features. *ICCV.* 64

Gorelick, L., Blank, M., Shechtman, E., Irani, M., and Basri, R. (2007). Actions as space-time shapes. *IEEE Trans. PAMI*, 29(12):2247–2253. 51, 53, 55, 58, 61, 62, 63, 64

Gu, J., Ding, X., Wang, S., and Wu, Y. (2010). Action and gait recognition from recovered 3d human joints. *IEEE Transactions on systems, man and cybernetics, part B: cybernetics*, 40(4):836–949. 17

Guha, T. and Ward, R. K. (2011). Learning sparse representations for human action recognition. *IEEE trans. PAMI.* 61, 62, 63, 64

Gupta, R., Patil, H., and Mittal, A. (2010). Robust order-based mothods for feature description. *CVPR.* 16, 21, 23, 25, 49

Han, D., Bo, L., and Sminchisescu, C. (2009). Selection and context for action recognition. *ICCV.* 65

Han, L., Wu, X., Liang, W., Hou, G., and Jia, Y. (2010). Discriminative human action recognition in the learned hierarchical manifold space. *Image and Vision Computing*, pages 836–949. 17

Harris, C. and J., S. M. (1988). A combined corner and edge detector. *In Alvey Vision Conference.* 11

Heikkila, M., Pietikainen, M., and Schmid, C. (2009). Description of interest regions with center-symmetric local binary pattern. *Pattern recognition*, 42(3):425–436. 16, 29

Huang, H. J., Serre, T., Wolf, L., and Poggio, T. (2007). A biologically inspried system for action recognition. *ICCV.* 4, 49

I.Laptev, Marszalek, M., Schmid, C., and Rozenfeld, B. (2008). Learning realistic human actions from movies. *CVPR.* 4, 48, 49, 52, 64, 77, 104

Jiang, Z., Lin, Z., and Davis, L. S. (2011). Learning a discriminative dictionary for sparse coding via label consistent k-svd. *CVPR.* 38

Junejo, I. N., Dexter, E., Laptev, I., and Perez, P. (2009). View-independent action recognition from temporal self-similarities. *IEEE Trans. PAMI*, 99. 62

Kellokumpu, V., Zhao, G., and Pietikainen, M. (2008). Human activity recognition using a dynamic texture based method. *BMVC.* 16

Kim, T., Wong, S., and Cipollar, R. (2007). Tensor canonical correlation analysis for action classification. *CVPR*. 65

Kittler, J., Hatef, M., Duin, R., and Matas, J. (1998). On combining classifiers. *PAMI*, 20(3):226 239. 96

Klaser, A., Marszalek, M., and Schmid, C. (2008). A spatio-temporal descriptor based on 3d gradients. *BMVC*. 4, 16, 21, 49, 64

Kleihorst, R., Schueler, B., Danilin, A., and Heijligers, M. (2006). Smart camera mote with high performance vision system. *IWDSC*. 76

Kong, S. and Wang, D. (2012). A dictionary learning approach for classification: separating the particularity and the commonality. *ECCV*. 39, 40

Kovashka, A. and Grauman, K. (2010). Learning a hierarchy of discriminative space-time neighborhood features for human action recognition. *CVPR*. 63, 64, 65, 66

Laptev, I. and Lindeberg, T. (2004). Local descriptors for spatio-temporal recognition. *First International Workshop on spatial coherence for visual motion analysis*. 4, 49

Laptev, I. and Lindeberg, T. (2005). On space-time interest points. *International Journal of Computer Vision*, pages 107–123. 4, 11, 14, 21, 22, 49, 52

Le, Q. V., Zou, W. Y., Yeung, S. Y., and Ng, A. Y. (2011). Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. *CVPR*. 64, 67

Lee, H., Battle, A., Raina, R., and Ng., A. Y. (2006). Efficient sparse coding algorithms. *NIPS*. 75

Lee, H., Battle, A., Raina, R., and Ng, A. Y. (2007). Efficient sparse code algorithms. *NIPS*. 44, 45, 47, 94

Li, W., Zhang, Z., and Liu, Z. (2010). Action recognition based on a bag of 3d points. *In Human communicative behavior analysis workshop (in conjunction with CVPR).* xi, 17, 32, 97, 98, 100, 101, 102

Liu, J. and Shah, M. (2008a). Learning human actions via information maximization. *CVPR*. 65

Liu, J. and Shah, M. (2008b). Learning human actions via information maximization. *ICCV*. 70, 78, 81

Liu, J., Shah, M., Kuipers, B., and Savarese, S. (2008). Cross-view action recognition via view knowledge transfer. *CVPR*. 81, 83

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *IEEE Journal on Computer Vision*, 60(2). 15

Lv, F. and Nevatia, R. (2006). Recognition and segmentation of 3d human action using hmm and multi-class adaboost. *ECCV*, pages 359–372. 90, 100

Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2009). Online dictionary learning for sparse coding. *ICML*. 98

Marszałek, M., Laptev, I., and Schmid, C. (2009). Actions in context. *CVPR*. 53, 55, 58

Martens, J. and Sutskever, I. (2011). Learning recurrent neural networks with hessian-free optimization. *ICML*. 100

Muller, M. and Roder, T. (2006). Motion templates for automatic classification and retrieval of motion capture data. *In proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on compute animation*, pages 137–146. 17, 90, 100, 104

Niebles, J., Wang, H., and Fei-Fei, L. (2008). Unsupervised learning of human action categories using spatial-temporal words. *International Journal of Computer Vision*, 79:299–318. 62, 65

Nowozin, S., Bakir, G., and Tsuda, K. (2007). Discriminative subsequence mining for action classification. *ICCV*. 64

Oikonomopoulos, A., Patras, I., and Pantic, M. (2006). Spatio-temporal salient points for visual recognition of human actions. *IEEE Trans. System, Man, and Cybernetics, Part B*, 36(3):710–719. 4, 49

Ojala, T., Pietikainen, M., and Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trancations on pattern analysis and machine intelligence*, 24:971–987. 6, 16, 69

Platt, J. (2000). Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. *Cambridge: MIT Press*. 77

Poppe, R. (2010). A survey on vision-based human action recognition. *Journal of image and vision computing*, 28:976–990. 11, 14, 48

Ramirez, I., Sprechmann, P., and Sapiro, G. (2010). Classification and clustering via dictionary learning with structured incoherence and shared features. *CVPR*. 39, 40, 100

Riemenschneider, M. D. H. and Bischof, H. (2009). Bag of optical flow vojumes for image sequence recognition. *BMVC*. 62

Rodriguez, M., Ahmed, J., and Shah, M. (2008). Action mach a spatio-temporal maximum average correlation height filter for action recognition. *CVPR*. 2, 51, 53, 55, 58, 66

Schuldt, C., Laptev, I., and Caputo, B. (2004). Recognizing human actions: A local svm approach. *ICPR*. x, xiii, 30, 51, 53, 57, 58

Schuldt, C., Laptev, I., and Caputo, B. (2010). Recognizing human actions: A local svm approach. *ACCV*. 48, 63, 64

Scovanner, P., Ali, S., and Shah, M. (2007). A 3-dimensional sift descriptor and its application to action recognition. *ACM International conference on multimedia*. 4, 15, 21, 49, 62

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. (2011). Real-time human pose recognition in parts from single depth images. *CVPR*. xiv, 5, 17, 22, 32, 33, 88, 89

Srivastava, G., Iwaki, H., Park, J., and Kak, A. C. (2009). Distributed and lightweight multi-camera human activity classification. *ICDSC*. 70, 78, 81, 86, 87

Tan, X. and Triggs, B. (2007). Enhanced local texture feature sets for face recognition under difficult lighting conditions. *International workshop on Analysis and modeling of face and gestures*. 16

Taylor, G., Fergus, R., Lecun, Y., and Bregler, C. (2010). Convolutional learning of spatio-temporal features. *ECCV*. 67

Thurau, C. and Hlavac, V. (2008). Pose primitive based human action recognition in videos or still images. *CVPR*. 62

Turaga, P., Chellappa, R., Subrahmanian, V. S., and Udrea, O. (2008). Machine recognition of human activities: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 18:1473–1488. 2

Uemura, H., Ishikawa, S., and Mikolajczyk, K. (2008). Feature tracking and motion compensation for action recognition. *BMVC*. 65

Vieira, A. W., Nascimento, E. R., Oliveira, G., Liu, Z., and Campos, M. (2012). Stop: space-time occupancy patterns for 3d action recognition from depth map sequences. *17th Iberoamerican Congress on Pattern recognition*. 100

Wang, H., Klaser, A., Schmid, C., and Liu, C. (2011). Action recognition by dense trajectories. *CVPR*. 13, 15, 21, 63, 64, 66, 67

Wang, H., Ullah, M. M., Klaser, A., Laptev, I., and Schmid, C. (2009). Evaluation of local spatio-temporal features for action recognition. *BMVC*. x, 4, 13, 16, 49, 51, 52, 53, 56, 58, 60, 65, 66, 67

Wang, J., Liu, Z., Chorowski, J., Chen, Z., and Wu, Y. (2012a). Robust 3d action recognition with random occupancy patterns. *ECCV*. 17, 100

Wang, J., Liu, Z., Wu, Y., and Yuan, J. (2012b). Mining actionlet ensemble for action recognition with depth cameras. *CVPR*. 6, 17, 32, 34, 37

Wang, J., Liu, Z., Wu, Y., and Yuan, J. (2012c). Mining actionlet ensemble for action recognition with depth cameras. *CVPR*. 88, 89, 90, 97, 99, 100, 103, 104

Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., and Gong, Y. (2010). Locality-constrained linear coding for image classification. *CVPR*. 93

Wang, Y. and Mori, G. (2009). Human action recognition by semilatent topic models. *IEEE Trans. PAMI*, 31(10):1762–1774. 62

Weinland, D., Boyer, E., and Ronfard, R. (2007). Action recogntion from arbitrary views using 3d examplars. *ICCV*. 70, 78, 80, 81, 83, 86, 87

Willems, G., Tuytelaars, T., and Gool, L. V. (2008a). An efficient dense and scale-invariant spatio-temporal interest pint detector. *ECCV*. x, 15, 60

Willems, G., Tuytelaars, T., and Gool, L. V. (2008b). An efficient dense and scale-invariant spatio-temporal interest point detector. *ECCV*. 4, 12, 21, 49

Wong, S. F. and Cipolla, R. (2007). Extracting spatio-temporal interest points using global information. *ICCV*. 4, 49

Wu, S., Oreifej, O., and Shah, M. (2011). Action recognition in videos acquired by a moving camera using motion decomposition of lagrangian particle trajectories. *ICCV*. 65, 66

Xia, L., Chen, C. C., and Aggarwal, J. K. (2012). View invariant human action recognition using histograms of 3d joints. *workshop of CVPR*. 5, 17

Yan, P., Khan, S., and Shah, M. (2008). Learning 4d action feature models for arbitrary view action recognition. *CVPR*. 70, 80, 81, 83

Yang, J., Yu, K., Gong, Y., and Huang, T. (2009a). Linear spatial pyramid matching using sparse coding for image classification. *CVPR*. 19, 20, 37, 74, 75, 80, 91, 93, 94, 95, 100

Yang, M., Lv, F., Xu, W., Yu, K., and Gong, Y. (2009b). Human action detection by boosting efficient motion features. *Workshop video-oriented object and event classification at ICCV*. 65

Yang, M., Zhang, L., Feng, X., and Zhang, D. (2011). Fisher discrimination dictionary learning for sparse representation. *ICCV*. 38

Yang, X. and Tian, Y. (2012). Eigenjoints-based action recognition using naive bayes nearest neighbor. *CVPR 2012 HAU3D Workshop*. xi, 17, 34, 90, 100, 102

Yao, A., Gall, J., and Gool, L. V. (2010). A hough transform-based voting framework for action recognition. *CVPR*. 66

Yeffet, L. and Wolf, L. (2009). Local trinary patterns for human action recognition. *ICCV*. 16, 29, 61, 62, 64, 66, 69

Yuan, J., Liu, Z., and Wu, Y. (2009). Discriminative subvolume search for efficient action detection. *CVPR*. 64

Zhang, J., Marszalek, M., Lazebnik, S., and Schmid, C. (2007). Local features and kernels for classification of texture and object categories: A comprehensive study. *IJCV*, 73(2):213–238. 53, 77, 94

Zhang, Q. and Li, B. (2010). Discriminative k-svd for dicitonary learning in face recognition. *CVPR*. 38

Zhang, Z., Hu, Y., Chan, S., and Chia, L.-T. (2008). Motion context: a new representation for human action recognition. *ECCV*, 5305:817–829. 62

Zhu, Y., Zhao, X., Fu, Y., and Liu, Y. (2004). Sparse coding on local spatial-temporal volumes for human action recognition. *ICPR*. 65, 66

Zou, H. and Hastie, H. (2005). Regression and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67(2):301–320. 39, 41

Vita

Jiajia Luo was born in Zhejiang, China. After graduation in 2002 from Zhuji High School, she attended the University of Science and Technology Beijing, where she earned both a Bachelor of Science degree in 2006 and a Master of Science degree in 2009 from the Electrical and Computer Engineering Department. From fall 2009, she was enrolled in the department of Electrical Engineering and Computer Science at the University of Tennessee, Knoxville. At the same time, she joined in the advanced imaging and collaborative information processing (AICIP) group as a graduate research assistant. During her PhD study, she was involved in several projects besides this PhD dissertation work, including the hyperspectral image analysis and shear bands analysis from surface temperature captured by infrared camera. In the summer of 2011, she worked as an intern at the Aldis, Inc. in Knoxville, Tennessee, where she conducted research on in-zone vehicle detection. In the spring of 2014, she worked as a research intern at the Microsoft Corporation in Redmond, Washington, where she conducted research on furniture image retrieval. She will complete her Doctor of Philosophy degree in Computer Engineering in Spring 2014.