Local Part Model for Action Recognition in Realistic Videos

by

Feng Shi

Thesis submitted to

Faculty of Graduate and Postdoctoral Studies

in partial fulfillment of the requirements

for the Doctorate of Philosophy degree in Electrical Engineering

School of Electrical Engineering and Computer Science

Faculty of Engineering

University of Ottawa

Abstract

This thesis presents a framework for automatic recognition of human actions in uncontrolled, realistic video data such as movies, internet and surveillance videos. In this thesis, the human action recognition problem is solved from the perspective of local spatio-temporal feature and bag-of-features representation. The bag-of-features model only contains statistics of unordered low-level primitives, and any information concerning temporal ordering and spatial structure is lost. To address this issue, we proposed a novel multiscale local part model on the purpose of maintaining both structure information and ordering of local events for action recognition. The method includes both a coarse primitive level root feature covering event-content statistics and higher resolution overlapping part features incorporating local structure and temporal relationships. To extract the local spatio-temporal features, we investigated a random sampling strategy for efficient action recognition. We also introduced the idea of using very high sampling density for efficient and accurate classification.

We further explored the potential of the method with the joint optimization of two constraints: the classification accuracy and its efficiency. On the performance side, we proposed a new local descriptor, called GBH, based on spatial and temporal gradients. It significantly improved the performance of the pure spatial gradient-based HOG descriptor on action recognition while preserving high computational efficiency. We have also shown that the performance of the state-of-the-art MBH descriptor can be improved with a discontinuity-preserving optical flow algorithm. In addition, a new method based on histogram intersection kernel was introduced to combine multiple channels of different descriptors. This method has the advantages of improving recognition accuracy with multiple descriptors and speeding up the classification process. On the efficiency side, we applied PCA to reduce the feature dimension which resulted in fast bag-of-features

matching. We also evaluated the FLANN method on real-time action recognition.

We conducted extensive experiments on real-world videos from challenging public action datasets. We showed that our methods achieved the state-of-the-art with real-time computational potential, thus highlighting the effectiveness and efficiency of the proposed methods.

Acknowledgements

Firstly, I would like to thank my supervisor Dr. Emil M. Petriu and Dr. Robert Laganière for providing this opportunity to work with them. Their trust in my abilities and the academic latitudes they provided have proved to be extremely invaluable during my Ph.D. study. They have provided far more than just the required supervisory feedback and excellent guidance throughout the journey of the research and the production of this work. I would also like to thank my supervisor Dr. Nicolas D. Georganas for his thoughtful guidance and constant encouragement.

Secondly, I would like to thank all other members in my doctoral committee, Dr. Abdulmotaleb El Saddik and Dr. Dorina C. Petriu for their thoughtful comments on my thesis. Special thanks to the members of the Discover Lab for their suggestions and helps on my research.

Finally, I would also like to express gratitude to my wife, Dongmei, whose prompting inspired me down this academic journey. It is by the selfless supports and sacrifices that she made over the years that I was able to fulfil this accomplishment. Last but not least, I would like to thank my entire families, who always encourage me to trust myself and to follow my passion.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Recognizing human action from image sequences has been a main interest of many industry and research communities for years with the goal of automatic analysis of visual events. With over 100 hours of video uploaded to YouTube every minute[1], and millions of surveillance cameras all over the world, the amount of digital video has grown exponentially in recent years. With the ubiquity of cheap digital sensors and the technology advances of computing power, storage capacity and networking, the need for reliable, automatic recognition of the visual events in the video is critical for many important applications, such as intelligent video surveillance, content-based video annotation and retrieval, human-robot interaction, and smart home, etc.

Video surveillance plays a major role in the war against the crime. Law enforcement and government use surveillance videos to analyse human action and pose to identify illegal or suspicious behaviour. Surveillance cameras are used not only to help police find criminals after the crime happened, but also to prevent it from happening. For example, an automatic real-time video analysis system can detect a terrorist's action of dropping explosive bags in an airport and send alarm signal to security guards.

[1] http://www.youtube.com/yt/press/statistics.html

Action recognition is an important tool to understand the content of on-line videos. The traditional methods for video search have relied on text, such as those extracted from closed caption, speech analysis, or manual annotation. Annotating the unlabelled videos is labour-intensive, and as videos grow explosively, it is impractical to do manual annotation on all objects and events that occurred in a video.

Further application areas include smart home, human-computer interaction and games. A typical smart home application for action recognition is to perform the fall detection of elderly persons living alone at home. For human-computer interaction and games, Microsoft Kinect enables users to control and interact with Xbox without using a physical controller. This is achieved by full-body 3D motion analysis, facial recognition, voice recognition, and acoustic source localization.

These examples show the importance of automatic recognition of human actions.

1.1 Problem statement

There are different definitions for actions in the literature [10, 37, 70, 72]. In this study, we adopt the same action hierarchy as [70, 82]: action primitive, action and activity. *Action primitives* are atomic entities which consist of the motion of human body parts, such as feet, head and hands etc. An *action* can be decomposed into a series of subsequent action primitives. It often includes full-body motions, such as running, boxing and hand waving. However, there are certain types of actions involving only part of the body parts, such as chewing, which consists of periodic action primitives of lip and jaw movement. *Activities* often contain complex sequence of actions with much longer temporal durations. They are often performed by several humans.

The main theme of this thesis is to perform automatic recognition of human actions in uncontrolled, realistic video data such as movies, internet and surveillance videos.

We put an emphasis on the joint optimization of two constraints: the accuracy and the efficiency. We aim on automatic classification of actions without considering context explicitly. However, our approaches may include implicit context information which helps to improve the performance. For example, the dining table background in "chewing" and water background in "diving" can provide discriminative information.

1.2 Motivations and objectives

Local spatio-temporal features and bag-of-features(BoF) representations [21, 48, 50, 123] have recently become popular for action recognition due to their simplicity and good performance. The success of such approaches can be attributed to their relatively independent representation of events which has better tolerance to certain conditions in the video, such as illumination, occlusion, deformation and multiple motion.

While impressive progress has been made, there are still some problems that need to be addressed. Firstly, bag-of-features approach only contains statistics of unordered primitive "features" from the image sequences, and any information of temporal relation or global spatial structure is ignored. A more discriminative method should include global structure information and ordering of local events.

Secondly, most methods extract local spatio-temporal features by extending interest point detectors from 2D image domain. They were originally designed for feature matching, not for selecting the most discriminate patches for classification. Interest point detectors [123] or selected features [53] by unsupervised learning have been shown to be very useful for simple KTH dataset [95] with single, staged human actions and uncorrelated backgrounds. We argue that it is more suitable to include the background information for real-life challenging datasets [45, 62, 85] because some of the background features are highly correlated with the foreground actions(*e.g.* diving with water back-

ground and skiing with snow background), and thus provide discriminative information for the foreground categories.

Thirdly, most existing action recognition methods use computationally expensive feature extraction, which is a very limited factor considering the huge amount of data to be processed. Also, sparse interest point representations may miss important aspects of the scene and therefore do not generate enough relevant information for classification. In contrast, dense sampling methods can provide a very large number of feature patches and thus can potentially produce excellent performance. The better results are generally observed as the feature density increased [120, 123]. However, the increase in the number of processed points adds to the computation complexity even if simplified techniques are used, such as integral video and approximative box-filters.

It should also be noted that when dealing with large multi-class datasets [45, 85], the non-linear SVMs are often used for training and testing. The non-linear SVMs are more expensive when compared to their linear counterparts. Nevertheless, there are recent approaches [60, 115] using approximate additive kernels for fast training. To improve the classification accuracy, a recent trend is to combine multiple channels of different descriptors [121, 136], which leads to more computational complexity.

We aim to overcome these challenges in this study. Local spatio-temporal features and bag-of-features representations normally include three steps: how to extract local spatio-temporal patches, how to represent them and how to classify the video based on the statistics of the features. For all three steps we propose solutions to address these challenges. More specifically, we focus on efficiently extracting local features, improving the performance of feature descriptors, addressing the orderless issue of the bag-of-features representation, and improving the efficiency and accuracy of the classification process.

1.3 Main contribution of the thesis proposal

This thesis contributes to several research domains, including, but not limited to: computer vision, image processing, pattern recognition, machine learning and computational intelligence. This work makes several contributions showing the efficiency and effectiveness of the proposed methods for visual event classification:

- A novel multiscale local part model is proposed to maintain both structure information and ordering of local events for action recognition. The method includes both a coarse primitive level root model covering event-content statistics and higher resolution overlapping part models incorporating structure and temporal relationships. Experimental evaluation demonstrates the promise of the proposed method.

- We investigate a random sampling strategy for efficient action recognition. We introduce the idea of using very high sampling density for efficient and accurate classification. Compared with existing methods, a major strength of our method resides in its very high computational efficiency.

- We introduce a new local 3D descriptor based on histogram of oriented spatial-temporal gradients. It significantly outperforms popular HOG descriptor without losing high computational efficiency. We also improve the performance of the state-of-the-art MBH descriptor with a discontinuity-preserving optical flow method.

- A new method based on histogram intersection kernel is proposed to combine multiple channels of different descriptors. This method has the advantages of improving recognition accuracy with multiple descriptors and speeding up the classification process in comparison with popular multi-channel RBF-χ^2 SVM [121, 136].

- We evaluate the efficiency benefits on bag-of-words matching by applying the PCA

technique to reduce dimensionality and the fast approximate nearest neighbour search method.

1.4 Publications arising from this report

The following publications have arisen from the work presented in this thesis:

- F. Shi, E. Petriu and R. Laganière. Sampling Strategies for Real-time Action Recognition. *In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. (**Acceptance rate 25.2%**) [101]

- S. Wu, F. Shi, M. Jobin, E. Pugin, R. Laganière and E. Petriu. Event Detection Using Local Part Model and Random Sampling Strategy for Visual Surveillance. *In Proc. of TRECVID 2013*.

- C. Whiten, R. Laganière, E. Fazl-Ersi, F. Shi, G. Bilodeau, D. Gorodnichy, J. Bergeron and E. Choy. VIVA-uOttawa / CBSA at TRECVID 2012: Interactive Surveillance Event Detection. *In Proc. of TRECVID 2012*.

- M. R. Abid, F. Shi and E. Petriu. Dynamic Hand Gesture Recognition from Bag-of-Features and Local Part Model. *In Proc. IEEE International Workshop on Haptic Audio Visual Environments and Games (HAVE)*, 2012.

- F. Shi, E. Petriu and A. Cordeiro. Human Action Recognition from Local Part Model. *In Proc. IEEE International Workshop on Haptic Audio Visual Environments and Games (HAVE)*, 2011. [100]

Submitted paper:

- F. Shi, R. Laganière and E. Petriu. Local Part Model for Fast Action Recognition. Submitted to *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2014.

Notebook paper:

- F. Shi, R. Laganière, E. Petriu and H. Zhen. LPM for Fast Action Recognition with Large Number of Classes. *THUMOS: ICCV Workshop on Action Recognition with a Large Number of Classes*, 2013. [99]

1.5 Outline of the thesis

The thesis begins with a literature review on action recognition in Chapter 2. Chapter 3 introduces our approach of *local part model*. An action recognition framework is also illustrated. Chapter 4 investigates feature extraction by random sampling. The idea of using very high sampling density for efficient and accurate action classification is discussed. In Chapter 5, we present a new local spatial-temporal descriptor, called GBH. The performance of MBH descriptor is improved with a state-of-the-art optical flow method. A new method based on histogram intersection kernel is proposed to combine multiple channels of different descriptors. Chapter 6 gives the strategies to improve the efficiency of the system. Chapter 7 summarizes the major contributions with a brief conclusion as well as future works.

Chapter 2

Related Work

In this chapter, we review the state-of-the-art approaches for action recognition in realistic, uncontrolled videos. We first outline the related surveys. Then, we present the approaches on solving the unordered problem of bag-of-features representation. We will also discuss the efficient action recognition methods. Finally, we will briefly introduce the popular public human action datasets.

2.1 Human action recognition

This section reviews the state-of-the-art vision-based human action recognition methods. Generally speaking, human action recognition can be separated into three steps: feature extraction, feature representation and classification. From the perspective of feature representation, we divide existing methods into three categories:

- *Body modelling methods* represent a human action as 2D or 3D model of human body parts.
- *Global representation methods* represent an action using appearances and move-

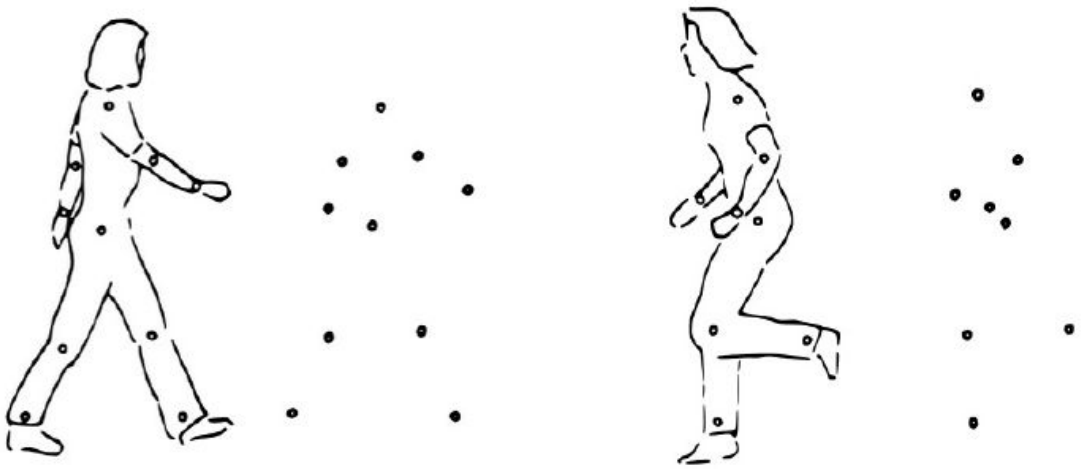


Figure 2.1: Illustration of walking and running from moving light displays (MLDs). It is shown that actions can easily be recognized by humans with only a few MLDs attached to the human body (reprinted from [39]).

ments of the whole human body without any description of body parts.

- *Local feature methods* represent human actions with a collection of independent local spatio-temporal regions.

2.1.1 Body modelling methods

Body modelling methods represent actions by employing human body part structure and positions with either 2D motion patterns [30, 83] or 3D body models [12, 61]. Approaches in this field are inspired by early Johansson's psychophysical work on visual interpretation of biological motion. In [39], he showed that humans can recognize actions solely from the motion of a few moving light displays (MLD) attached to the human body (Figure 2.1).

A significant amount of early research in action recognition belongs to this field due to its intuition and biologically plausible to action recognition. Such works include hierarchical 3D model based on cylindrical primitives [61, 88], blob model [12], stick figure model [30, 75], trajectories of joint positions [133], landmark points on the human

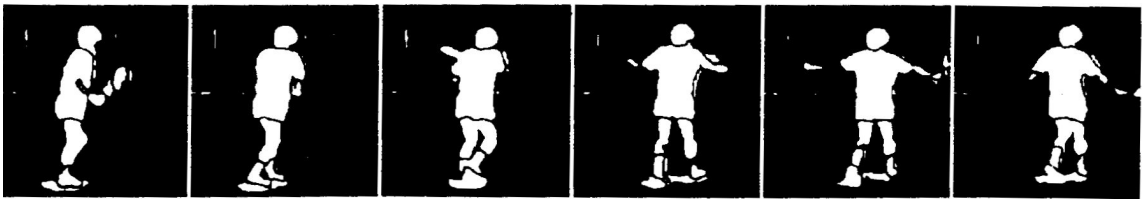


Figure 2.2: Silhouette shape masks of tennis strokes (reprinted from [130]).

body [15, 39] *etc.*

Body modelling methods show relatively good performance on simple, clean datasets. However, they heavily rely on the detection of human body parts and human poses. The detection of body parts in itself is not a trivial problem. It often involves very challenge techniques, such as image/video segmentation, human tracking, 3D reconstruction *etc.*, which are still open and active research areas. For uncontrolled, realistic video data, most body modelling methods can only achieve suboptimal performance due to occlusion, background clutter, scale change and multiple subjects etc. Some methods [2, 84, 112, 119] are able to achieve relatively better results with particular assumptions on certain type of motions. Such constrains limit their applications in general.

2.1.2 Global representation methods

Global representation methods represent an action using appearances and movements of the whole human body without any detection and labelling of individual body parts. Compared to approaches of body modelling, global representation methods only need to model global appearances and motions, which is more efficient and robust than to model every body part. In general, they can be classified into three main categories: *shape mask based methods*, *optical flow shape based methods* and *template based methods*.

Shape mask based methods

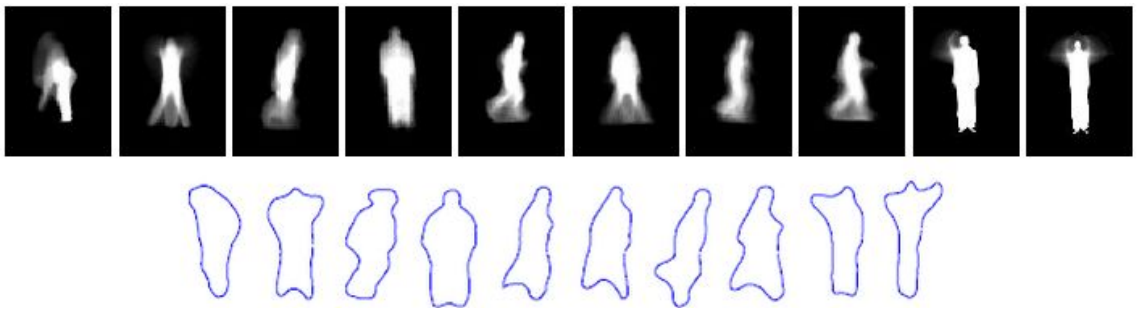


Figure 2.3: Examples of AMEs and MMSs of 10 different actions from a same subject (reprinted from [124]).

Typically, the shape mask based methods use silhouettes or contours of the whole human body from the image sequences. Yamato *et al.* [130] use silhouette images to recognize tennis actions (*c.f.* Figure 2.2). The authors quantize the silhouette image into a grid of cells, and represent the grid with the ratio of foreground to background pixels from each cell. They use hidden Markov models (HMM) to learn tennis actions from the grid representations.

Wang and Suter [124] convert a sequence of human silhouettes into two compact representations, *average motion energy* (AME) and *mean motion shape* (MMS) (as illustrated in Figure 2.3). The AME is computed with a sequence of binary silhouette images of the moving human based on periodical detection of motions. The MMS is obtained from the single-connectivity binary silhouette using a border. Supervised pattern classification method with various distance measurements is used for classification.

Weinland and Boyer [126] represent actions with a set of orderless static key-pose exemplars. An action sequence is matched against a set of silhouette exemplars. For each exemplar the minimum matching distance to any of the frames of the sequence is determined, and the resulting set of distances is concatenated into a vector representation. Bayes classifier with Gaussians is used to perform the final classification.

Silhouettes are efficient to compute and are insensitive to texture, colour and illu-

mination changes. However, the performance of silhouette-based representations heavily depends on a robust background segmentation, which is a very challenging problem in realistic settings due to the clutter, occlusion, camera motion and multiple actors *etc.*

Optical flow shape based methods

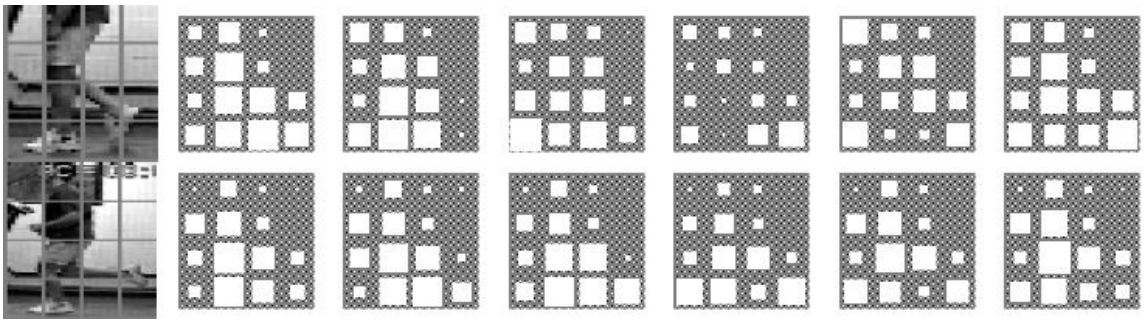
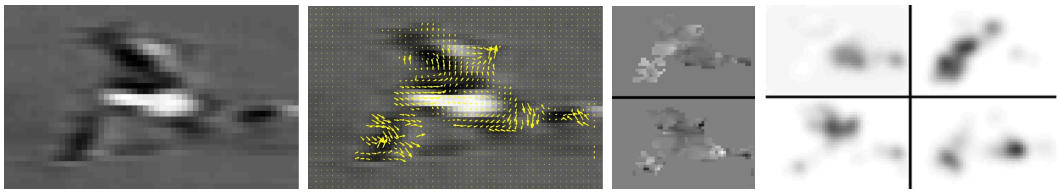


Figure 2.4: Optical flow magnitude accumulated in regular grid for a sample of walk (top) and a sample of run (bottom) (reprinted from [81]).

Another type of global representation methods uses dense optical flow for action recognition. In comparison with shape mask based methods, such approaches don't depend on background segmentation. Polana and Nelson [81] compute optical flow for region of interest (ROI) over the person, and accumulate flow magnitudes in a regular grid of non-overlapping bins as shown in Figure 2.4. Classification is performed by matching the descriptors in test sequences to reference motion templates.

Efros *et al.* [22] track soccer players in sports videos and compute optical flow over a very small human-centred image window. They split the flow field into four different channels related to positive and negative as well as horizontal and vertical components (as shown in Figure 2.5). A blurry process is followed to avoid noisy displacement. For classification, a test sequence is frame-wise aligned to a database of annotated actions, and four channels are matched separately. Similar method is used in [125].



(a) (b) (c) (d)

Figure 2.5: Motion descriptor using optical flow: (a) original image; (b) optical flow; (c) separating the x and y components of optical flow vectors; (d) final blurry motion channels from half-wave rectification of each component (reprinted from [22]).

Ali and Shah [4] derive a set of kinematic features from the optical flow. These features include divergence, vorticity, symmetric and anti-symmetric flow fields, second and third principal invariants of flow gradient and rate of strain tensor, and third principal invariant of rate of rotation tensor. These kinematic modes are computed by performing Principal Component Analysis (PCA) on the spatio-temporal volumes of the kinematic features. For classification, the multiple instance learning (MIL) is used to represent each action video with a bag of kinematic modes. Each video is then embedded into a kinematic mode-based feature space and the coordinates of the video in that space are used for classification with the nearest neighbour algorithm.

Template based methods

Unlike shape mark or optical flow methods, which are computed over a short time widows, template based methods typically learn the appearance over long sequence of video frames. The templates can be built with image models by stacking multiple silhouette images into a volumetric representation [9, 11]. Other methods compute templates with spatio-temporal volume filters, such as 3D FFT [87] and 3D Gaussian third derivative filters [91].

Bobick and Davis [11] are the first to introduce temporal templates for action recog-

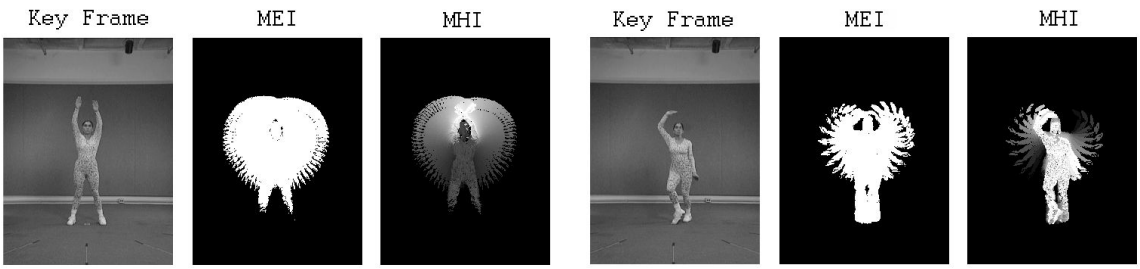


Figure 2.6: Motion history images (MHI) and motion energy images (MEI) (reprinted from [11]).

nition. They propose the idea of motion energy images (MEI) and motion history images (MHI). Figure 2.6 illustrates some examples of MEIs and MHIs. The MEI is a binary mask built by mapping successive silhouette frames into a single image. It represents regions of motion. The MHI, on the other hand, weights these regions as a function over time. Similar approaches include [3, 66], and motion history volumes (MHVs) [127] *etc.*

Figure 2.7: Space-time shapes (reprinted from [9]).

Blank *et al.* [9] and Gorelick *et al.* [28] build space-time shapes (as shown in Figure 2.7) by stacking a sequence of silhouette images, which are computed through background subtraction. The properties of the solution to the Poisson equation are explored to extract salient space-time features. The weighted moments over these features are calculated and described by a high-dimensional feature vector. For classification, these

global vectors are matched to space-time shapes in test sequences.

Ke *et al*. [40] separate the optical flow into its horizontal and vertical components, and use approximative box-filter operations and integral video structure to speed-up the feature extraction. They also apply the direct forward feature selection method to select a small subset volumetric features with a sliding-window approach and arrange them in a cascade for efficient action detection.

Jain *et al*. [34] propose an exemplar-based clustering approach to address issues of Kmeans, and use the exemplar-SVM to learn a discriminative distance metric for each cluster. They mine discriminative spatio-temporal patches as features and establish strong correspondence between spatio-temporal patches in training and test videos. In addition to action classification, their approach has the potential to perform object localization and fine-grained action detection by using label transfer techniques.

Other methods include dynamic time warping (DTW) [75, 80, 116], which is often used to deal with temporal data consists of actions with variable durations. Rodriguez *et al*. [87] use Maximum Average Correlation Height (MACH) filter to capture intra-class variability by synthesizing a single action MACH filter for a given action class. The Clifford Fourier transform is employed to analyse vector valued data obtained from the response of the filter. Action Bank [91], a high-level representation of video, represents a video as the collected output of many action detectors that each produce a correlation volume. 3D Gaussian third derivative filters are used to perform spatio-temporal orientation decomposition. Actions are treated as composition of energies along spatio-temporal orientations.

2.1.3 Local feature methods

Among all action recognition methods, local spatio-temporal features and bag-of-features (BoF) representations achieved remarkable performance. The success of such approaches can be attributed to their relatively independent representation of events which has better tolerance to certain conditions in the video, such as illumination, occlusion, deformation and multiple motion.

Feature detectors and dense feature points

Laptev and Lindeberg [48] are first to introduce space-time interest point by extending 2D Harris-Laplace detector. They propose the Harris3D detector. They compute a spatio-temporal second-moment matrix at each spatio-temporal point with different spatial and temporal scale, a separable Gaussian smoothing function and space-time gradients. The authors also apply an optional mechanism to select different spatio-temporal scales. Schüldt *et al*. [95] detect and use salient sparse spatio-temporal features with automatic scale selection based on Harris corner detector.

To produce denser space-time feature points, Dollár *et al*. [21] use a pair of 1D Gabor-filter to convolve with a spatial Gaussian to select local maximal cuboids. Interest points are the local maxima of the response convolution. Willems *et al*. [129] propose the Hessian3D detector and extend SURF descriptor to detect relatively denser and computationally efficient space-time points. The salient points are detected with the determinant of the 3D Hessian matrix. The position and scale of the interest points are automatically found without any additional iterative process.

A recent trend is the use of dense sampled points [123] and trajectories [35, 67, 120] to improve the performance. Most feature detectors are extended from computationally expensive image feature extraction methods, which is a very limiting factor considering

the huge amount of data to be processed. Also, sparse interest point representations may miss important aspects of the scene and therefore do not generate enough relevant information for classification. In contrast, dense sampling methods capture most information by sampling every pixel in each spatial scale.

Scovanner *et al.* [97] apply random sampling on a video at different locations, times, and scales to extract feature points. The authors extend 2D SIFT descriptor [58] into 3D SIFT to represent spatio-temporal patches. Wang *et al.* [123] evaluate and compare previously proposed space-time features in a common experimental setup. The authors evaluate Harris3D [48], Cuboid [21], Hessian3D and dense sampling, and use a standard bag-of-features SVM approach for action recognition. They demonstrate that dense sampling at regular space-time grids outperforms state-of-the-art interest point detectors. Liu *et al.* [57] select the most discriminative subset from densely sampled features using the AdaBoost Algorithm. [63, 117] are based on the idea that eye movement of the human viewers is the optimal predictor of visual saliency. They measure the eye movement of human observers watching videos, and use the data to produce an "empirical" saliency map. By using such saliency maps, they prune 20-50% of the dense features and achieve better results. The requirement of prior eye movement data renders such methods impractical for real applications.

Trajectory shapes encode local motion information by tracking spatial interest points over time. Uemura *et al.* [113] extract features with the KLT tracker [59] and SIFT descriptor as well as a method for estimating dominant planes in the scene. They use multiple interest point detectors to generate a large number of interest points for every frame. Messing *et al.* [67] obtain feature trajectories using KLT tracker. To represent feature trajectories with varying length, the authors apply uniform quantization in log-polar coordinates, with 8 bins for direction, and 5 for magnitude. Human actions are

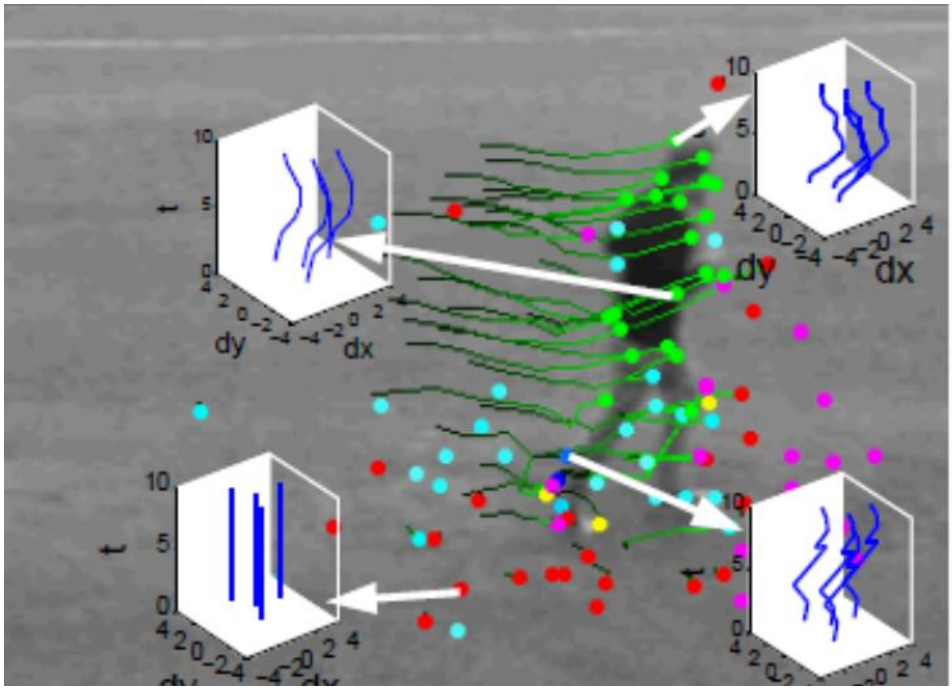


Figure 2.8: Trajectories are obtained by detecting and tracking spatial interest points, and are quantized to a vocabulary of fixed length trajectons (reprinted from [64]).

recognized using a generative mixture of Markov chain models. By matching SIFT descriptors between two consecutive frames, Sun *et al.* [108] compute trajectories with a hierarchical structure to model spatio-temporal contextual information. Actions are classified with intra- and inter-trajectory statistics. Sun *et al.* [107] track randomly sampled points within the region of the long-duration trajectories extracted through KLT tracker and SIFT descriptor matching.

Instead of using feature trajectories with varying length, Matikainen *et al.* [64, 65] employ fixed length feature trajectories for action classification (see Figure 2.8). Trajectories from the video are clustered by Kmeans. For each cluster centre an affine transformation matrix is computed. The final trajectory descriptor is represented with information containing both displacement vectors and elements of the affine transforma-

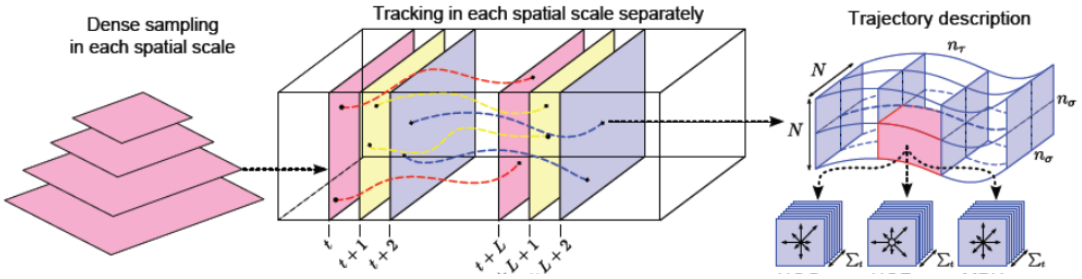


Figure 2.9: Illustration of dense trajectories description. The dense sampled feature points are tracked and aligned over frames into fixed length trajectories, which are represented with local appearance and motion descriptors (reprinted from [120]).

tion matrix for its assigned cluster centre. To generate dense trajectories [120], Wang *et al*. sample interest feature points at uniform intervals in space and time and track them based on displacement information using an efficient dense optical flow algorithm. Figure 2.9 illustrates the dense trajectories description. The HOG, HOF and MBH descriptors (as shown in Figure 2.10) are used to represent resulting dense trajectories for action recognition. Later, built on dense trajectories both Jiang *et al*. [38] and Jain *et al*. [34] explore the methods to improve the performance from better camera motion compensation.

Feature descriptors

For each spatio-temporal feature point (x, y, t, σ, τ), a feature descriptor is computed in a local neighbourhood centred at (x, y, t) to capture shape and motion information. The descriptors are critical for the performance of the video recognition.

Laptev and Lindeberg [49] compute and evaluate different type of descriptors: single and multi-scale higher-order derivatives (local jets), histograms of optical flow, and

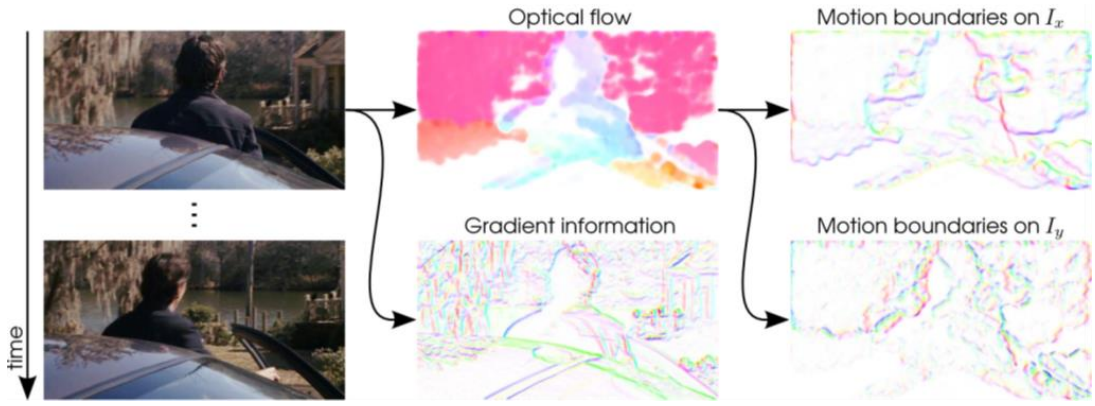


Figure 2.10: Illustration of the information captured by HOG, HOF, and MBH descriptors (reprinted from [120]).

histograms of spatio-temporal gradients. Histograms of optical flow and histograms of gradients are computed for each cell over a $M \times M \times M$ grid layout. The principal component analysis (PCA) is used to reduce the dimension of features, which are computed by concatenating optical flow or gradient components of each pixel. Their results show that descriptors based on histograms of optical flow and spatio-temporal gradients outperform others.

Dollár *et al.* [21] compare various local space-time descriptors based on normalized pixel values, brightness gradient, and windowed optical flow. They evaluate different strategies to build up the descriptor: simple concatenation of pixel values, a grid of local histograms, and a single global histogram. The authors also apply PCA to reduce the dimensionality of each descriptor representation. They report best results on concatenated gradients.

Scovanner *et al.* [97] extend 2D SIFT descriptor [58] to represent spatio-temporal patches, called 3D SIFT. After gradient magnitude and orientation are computed in 3D, each pixel has two values (θ, ϕ) which represent the direction of the gradient in three dimensions. A Gaussian weighted histogram for the 3D neighbourhood around a given

interest point is constructed. For orientation quantization, the gradients in spherical coordinates (θ, ϕ) are divided into equally sized bins, which are represented by a 8×4 histogram.

Willems *et al.* [129] propose an extension of 2D SURF descriptor [8] to 3D spatio-temporal space, called the extended SURF (ESURF) descriptor. The local spatio-temporal patch is divided into a grid of $M \times M \times M$ cells. Each cell is represented by a vector of weighted sums of uniformly sampled responses of Haar-wavelets along the three axes.

Laptev *et al.* [50] combine histograms of oriented spatial gradients (HOG) and histograms of optical flow (HOF) to include both local motion and appearance information. HOG descriptor is introduced by Dalal and Triggs in [19] for human detection. It is based on the popular image SIFT descriptor [58]. The gradients and dense optical flow are computed first. Each local region is subdivided into a grid of $N \times N \times M$ cells. For each cell, 4-bin HOG histograms and a 5-bin HOF histogram are computed, and then concatenated into the final HOG/HOF descriptors.

HOG3D descriptor is proposed by Kläser *et al.* [42]. It is built up on 3D oriented gradients. A 3D patch is divided into a grid of $M_c \times M_c \times N_c$ cells. Each cell is then divided into $M_b \times M_b \times N_b$ sub-blocks. A mean 3D gradient is computed for each sub-block. Each mean gradient is quantized using a polyhedron. The 3D histogram of oriented gradients for the 3D patch is formed by concatenating gradient histogram of all cells. Although it is very efficient to compute 3D gradients, the orientation quantization with polyhedron for each sub-blocks is relatively expensive to compute.

MBH descriptor is introduced by Dalal *et al.* in [20] and used by Wang *et al.* [120] on action recognition to achieved state-of-the-art performance. The derivatives are computed separately for the horizontal and vertical components of the optical flow, which

results in motion compensation. Due to its property of camera motion compensation, MBH descriptor is shown to outperform other state-of-the-art descriptors on human action classification.

Bag-of-features representation

Bag-of-features (BoF) representation is originally applied to document analysis [92]. It shows great popularity in object classification from images and action recognition from video data due to its simplicity and good performance. It models video as collections of local spatio-temporal patches. In a standard bag-of-features approach, a spatio-temporal patch is represented with one of the feature descriptors as a feature vector. A vocabulary of prototype features, called "visual words" or "codebook" is obtained by applying a clustering algorithm (*e.g.* Kmeans) on feature vectors computed from training data. A video is represented as a histogram of occurrences of local features by quantizing the feature vectors to their closest visual words. [21, 50, 73, 120] are among those methods which represent videos as bags of local spatio-temporal features and achieve remarkable performance for action recognition.

However, BoF only contains statistics of unordered "features" from the image sequences, and any information of temporal relations or spatial structures is ignored. It may have problem to discriminate between actions characterized by their structure and event-orderings, such as stand up and sit down. To preserve the "ordering of events", many works are introduced to add geometry and temporal information. Hamid *et al.* [31] propose an unsupervised method for detecting anomalous activities by using bags of event n-grams. In their method, human activities are represented as overlapping n-grams of actions. While overlapping n-grams can preserve the temporal order information of events, it causes the dimensionality of the space to grow exponentially as n increases. As

for more primitive motions, Thurau and Hlavác [111] introduce n-grams of primitive level motion features for action recognition. Laptev *et al.* [50] extend image representation of spatial pyramid [52] to the spatio-temporal domain. The authors divide a video into a grid of coarse spatio-temporal cells. The whole video is then represented by the ordered concatenation of the per-cell BoF models. Such ordered concatenation add global structural information. A greedy approach is used to learn the best combination of overlaid grids and feature types per action class. Gaidon *et al.* [26] focus on explicitly capturing the spatial and temporal structure of actions with structure model. Tang *et al.* [110] use a variable-length discriminative HMM model which infers latent sub-actions to explicitly model the presence of sub-events.

2.2 Efficient action recognition

With over 100 hours of video uploaded to YouTube every minute, and millions of surveillance cameras all over the world, the need for efficient recognition of the visual events in the video is crucial for real world applications. Most existing action recognition methods use computationally expensive techniques, which are very limiting factors considering the huge amount of data to be processed. Nevertheless, there are some methods which explore efficient action recognition.

Both Ke *et al.* [40] and Willems *et al.* [129] use approximative box-filter operations and integral video structure to speed-up the feature extraction. Patron-Perez and Reid [6] employ a sliding temporal window within the video and use first-order dependencies to effectively approximate joint distribution over feature observations given a particular action. Yeffet and Wolf [132] efficiently classify the action with Local Binary Patterns and an approximate linear SVM classifier. Yu *et al.* [134] extend the efficient 2D FAST corner detector to the 3D domain V-FAST detector, and apply semantic texton forests

for fast visual codeword generation. Whiten *et al.* [128] exploit very efficient binary bag-of-features matching with the Hamming distance rather than the Euclidean distance through an extension of the popular 2D binary FREAK descriptor.

2.3 Datasets

In this section, we present a brief review on human action benchmark datasets in the literature. Our discussion focuses on general human action recognition from 2D video data, rather than 3D data (*e.g.* Kinect sensors) or datasets for specific applications (*e.g.* gesture recognition and human posture analysis *etc.*).

Over the last decade, the advances in human action recognition have led to the emergence of many benchmarks for action recognition. In early years, as action recognition technologies are still at the beginning, the benchmarks are recorded "in the lab" based on staged environments under controlled settings, such as static, uncluttered backgrounds, single player without occlusion and fixed light conditions *etc*. They normally contain atomic actions with several categories. Two most extensively used such datasets are **KTH** [95] and **Weizmann** [9]. Another popular dataset is **IXMAS** [127] which includes synchronized, multiple viewpoints of each action. The performances on these datasets have saturated over the years, with 95.3% [121] on **KTH**, 100% [94] on **Weizmann** and 93.6% [121] on **IXMAS**.

Later, there are more interests in recognizing actions from realistic videos. The datasets focus on sports broadcasts, TVs and Movies. They include videos with large variation in human appearance, scale changes, dynamic backgrounds, illumination conditions, *etc*. The **UCF sports** [87] dataset has 200 videos with 9 categories from TV sport broadcasts. **Hollywood1** [50] and **Hollywood2** [62] collect videos from Hollywood Movies with 8 and 12 action classes, respectively. The **Hollywood2** is relatively

large, containing 3669 video clips. Although it only has 12 classes with high quality clips, it is a very challenging dataset due to large intra-class variation, multiple persons, camera motion, unconstrained and cluttered background etc.

More recently, benchmarks use videos from various "in-the-wild" sources, such as motion pictures, Youtube and Google videos. They often contain both high and low quality clips, captured either by professionals or amateurs. Also, they have a large number of action categories with significantly more videos than earlier datasets. Such benchmarks include **UCF50** [85], **HMDB51** [45] and **UCF101** [105]. We will discuss them in details next.

Other realistic human action benchmarks include TRECVID multimedia event detection (MED) and surveillance event detection (SER). The TREC Video Retrieval Evaluation (TRECVID) [78] is a TREC-style video analysis and retrieval evaluation, which has provided benchmarks to test the system performance for over ten years. Among five tasks, MED provides a development and evaluation collection of Internet multimedia (i.e., video clips containing both audio and video streams) clips. The data consist of publicly available, user-generated videos from various Internet video sites, with a set of 98,000 search clips. The 2013 evaluation consists of 20 "pre-specified" and 20 new "ad-hoc" event kits. SER focuses on detecting human behaviors efficiently in vast amounts surveillance video. The development data consist of about 150-hours airport surveillance video data, with 100-hours development data and 50-hours evaluation data. Both data sets were collected in the same busy airport environment.

Our objective is to perform automated recognition of human actions in uncontrolled, realistic video data. To test the effectiveness and efficiency of our approaches, we will use most recent realistic large scale datasets, *e.g.* **UCF50**, **HMDB51** and **UCF101**. As a comparison, we will also use an earlier **KTH** dataset.

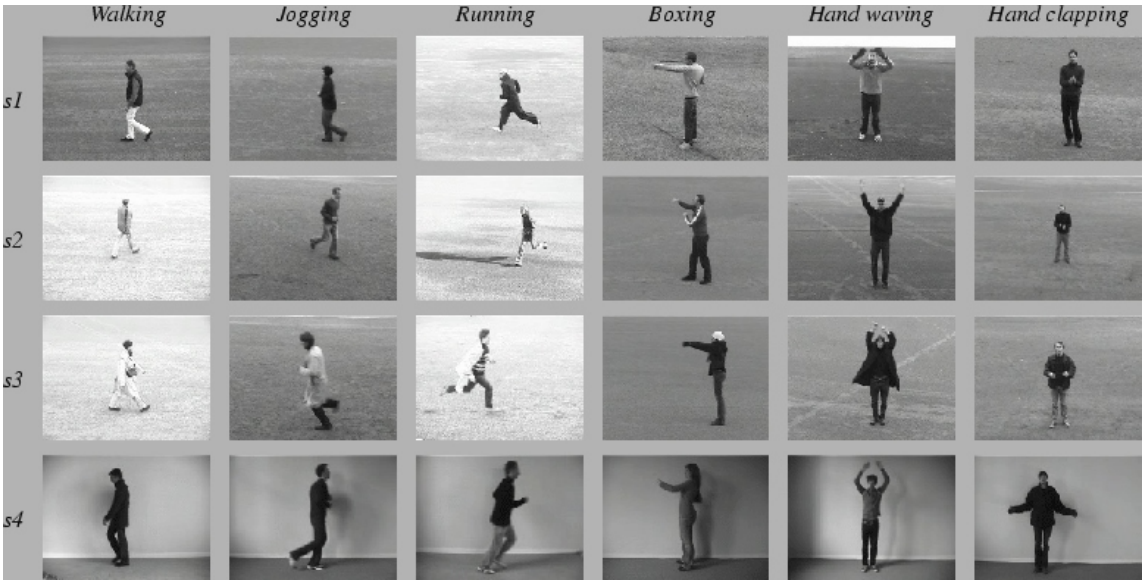


Figure 2.11: KTH action dataset: examples of sequences corresponding to different types of actions and scenarios (reprinted from [95]).

2.3.1 KTH action dataset

The **KTH** action dataset [95] is one of the most used datasets in evaluation of action recognition. It contains six classes of human actions: walking, jogging, running, boxing, hand waving and hand clapping. There are totally 2391 sequences. The sequences have spatial resolution of 160 x 120 pixels and a length of four seconds in average with 25 FPS frame rate. The sequences are recorded with 25 subjects in four different scenarios: outdoors s_1, outdoors with scale variation s_2, outdoors with different clothes s_3 and indoors s_4. The background is homogeneous and static in most cases. The typical clips are shown in Figure 2.11. The performances on this dataset have saturated over the years, with state-of-the-art result of 95.3% [121].

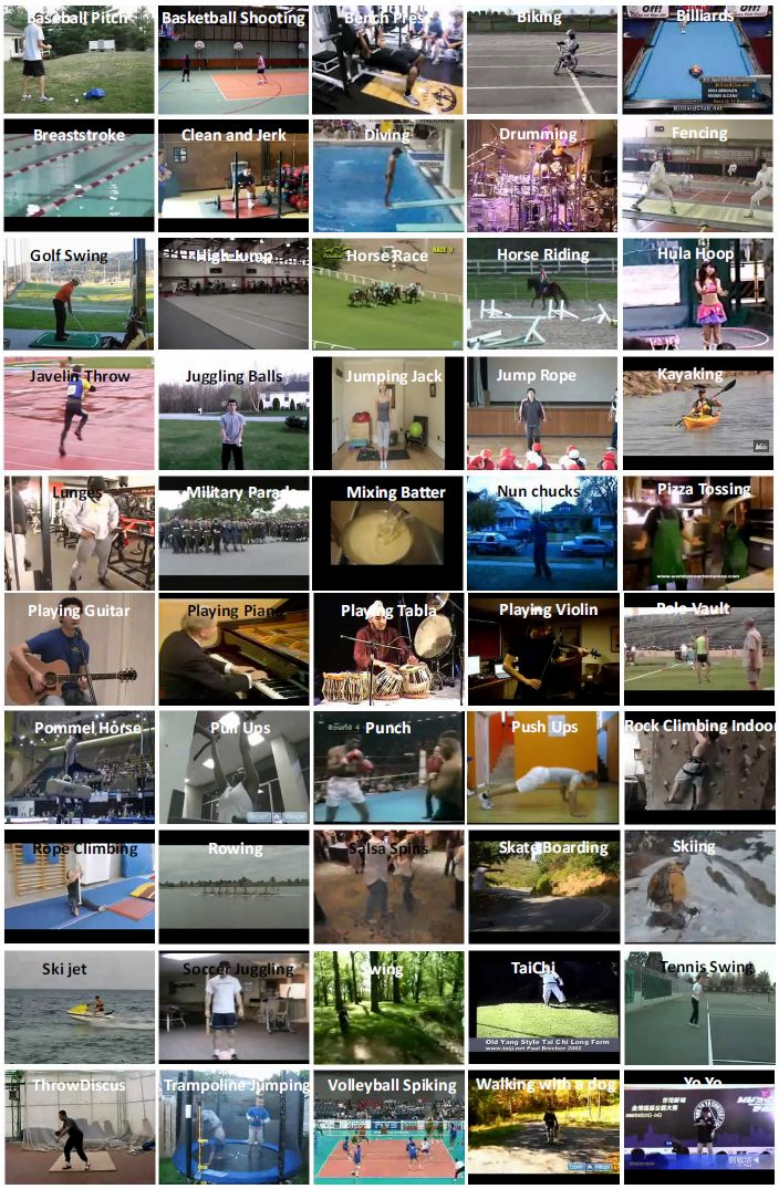


Figure 2.12: UCF50 dataset: examples of sequences corresponding to different types of actions and scenarios (reprinted from [85]).

2.3.2 UCF50 action dataset

The **UCF50** dataset [85] contains 50 classes and 6680 realistic videos taken from YouTube. All clips have fixed frame rate and resolution of 25 FPS and 320×240 respectively. The videos are grouped into 25 groups, where each group consists of a minimum of 4 action clips. The video clips in the same group may have similar background or be played by the same subjects. The dataset is very large and relatively challenging due to camera motion, cluttered background, large scale change and illumination variation, etc. Samples of video frames from **UCF50** are shown in Figure 2.12. Currently the best performance is reported in [122] with a mean accuracy of 91.2% based on Leave-One-group-Out (LOGO) training and testing splits.

2.3.3 UCF101 action dataset

The **UCF101** dataset [105] is by far the largest human action dataset with 101 classes and 13320 realistic video clips taken from YouTube. As an extension of **UCF50** dataset, it includes all clips of 25 groups from **UCF50** and adds additional 51 categories. The action categories can be divided into five types: 1)Human-Object Interaction; 2) Body-Motion Only; 3) Human-Human Interaction; 4) Playing Musical Instruments; 5) Sports. All clips have fixed frame rate and resolution of 25 FPS and 320×240 respectively. The clips of one action class are divided into 25 groups which contain 4-7 clips each. The clips in one group may have similar background or be played by the same subjects. The sample clips are shown in Figure 2.13. It is a challenging dataset with largest number of classes to date. The state-of-the-art result on this dataset is reported in [122] with 85.9% mean accuracy based on 3-splits evaluation scheme.

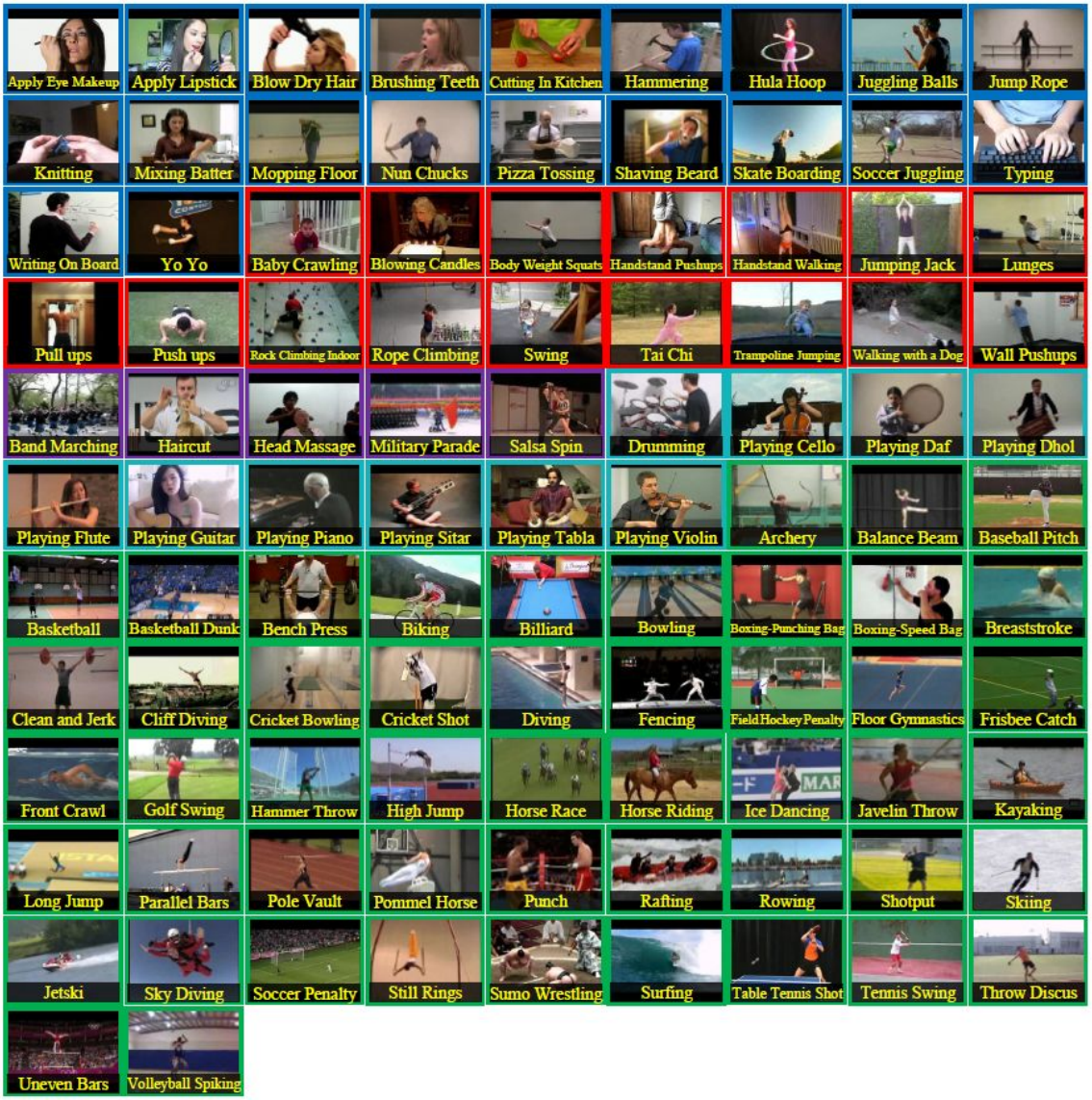


Figure 2.13: UCF101 dataset: examples of sequences corresponding to different types of actions and scenarios (reprinted from [105]).

2.3.4 HMDB51 action dataset

The **HMDB51** dataset [45] contains 51 action categories, with at least 101 clips for each category. The dataset includes a total of 6,766 video clips extracted from Movies, the Prelinger archive, Internet, Youtube and Google videos. The videos are encoded with a resolution of 240 pixels in height with preserved aspect ratio. The clips have various video quality, ranged from "High" with detailed visual elements such as the fingers and eyes to "Low" with large body parts not identifiable. The minimum quality standard is a minimum of 60 pixels in height for the main actor. The action categories can be grouped into five types: 1) General facial actions: smile, laugh, chew, talk; 2) Facial actions with object manipulation: smoke, eat, drink; 3) General body movements: cartwheel, clap hands, climb, climb stairs, dive, fall on the floor, backhand flip, handstand, jump, pull up, push up, run, sit down, sit up, somersault, stand up, turn, walk, wave; 4) Body movements with object interaction: brush hair, catch, draw sword, dribble, golf, hit something, kick ball, pick, pour, push something, ride bike, ride horse, shoot ball, shoot bow, shoot gun, swing baseball bat, sword exercise, throw; 5) Body movements for human interaction: fencing, hug, kick someone, kiss, punch, shake hands, sword fight. Three distinct training and testing splits have been selected from the dataset, with 70 training and 30 testing clips for each category. Figure 2.14 shows the examples corresponding to different types of actions. It is perhaps the most realistic and challenging dataset. The state-of-the-art result on this dataset reaches only 57.2% in [122].

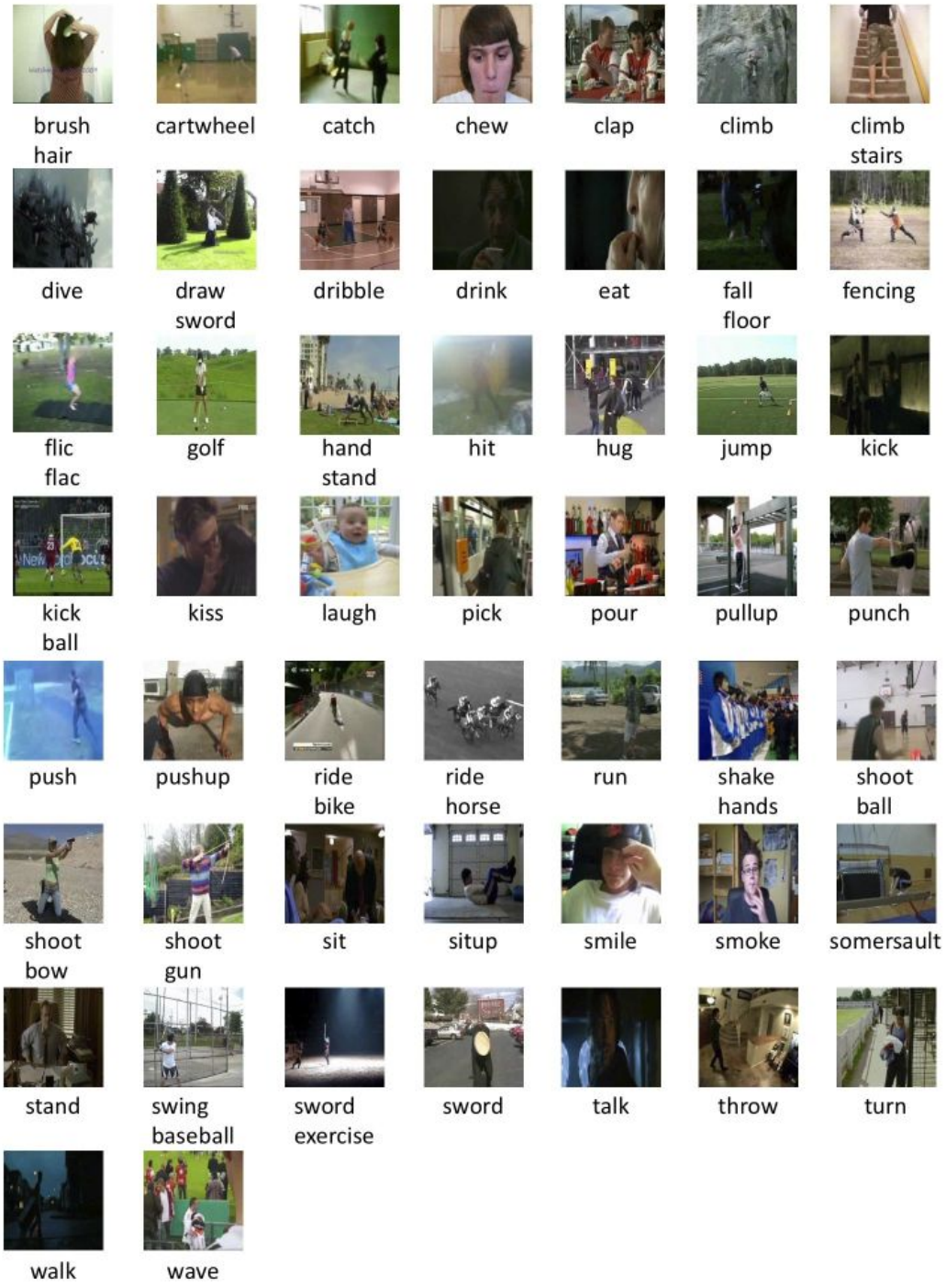


Figure 2.14: HMDB51 dataset: examples of sequences corresponding to different types of actions (reprinted from [45]).

Chapter 3

Local Part Model

3.1 Introduction

Recent studies [21, 48, 50, 123] have shown that local spatio-temporal features can achieve remarkable performance when represented by popular bag-of-features method. The success of such approaches can be attributed to their relatively independent representation of events which has better tolerance to certain conditions in the video, such as illumination, occlusion, deformation and multiple motion. Also, the BoF method has the ability to represent videos with statistical information of local features, with no requirement for the detection of humans, body parts or joint locations which is very challenging in uncontrolled realistic videos.

BoF approach was originally applied to document analysis. It gained great popularity in object classification from images and action recognition from video data due to its discriminative power. However, BoF only contains statistics of unordered "features" from the image sequences, and any information of temporal relation or spatial structure is ignored. In [31], Hamid *et al*. argued: "Generally activities are not fully defined by their event-content alone; however, there are preferred or typical event-orderings. Therefore a

model for capturing the order of events is needed." On the purpose of addressing out-of-ordering problem of bag-of-features, we propose a novel multiscale local part model to maintain both global structure information and ordering of local events.

This chapter is structured as follows: The next section will introduce a novel Local Part Model. The proposed framework will be presented in section 3. Sections 4 will describe some experimental results and analysis. The chapter is completed with a brief conclusion.

3.2 3D multiscale local part model

In this section, we propose a novel 3D multiscale part model to address the unordered problem of bag-of-features representation.

3.2.1 Deformable part model

Our method is inspired by the work of multiscale *deformable part model* (**DPM**) [24]. The multiscale **DPM** uses deformable part models to detect and localize objects of a generic category from the images. **DPM** represents objects by a "root" and a collection of "parts", arranged in a deformable configuration. Each part captures local appearance properties of an object, while the deformable configuration is characterized by star-like connections among the parts. Figure3.1 shows an example of the multiscale **DPM**. The **DPM** is defined by a coarse template covering an entire object, several higher resolution part templates and a spatial model for the location of each part. The spatial model defines a set of allowed placements for a part relative to a detection window, and a deformation cost for each placement. To achieve the high performance of object recognition, the discriminative training method is used to generalize SVMs for handling latent variables

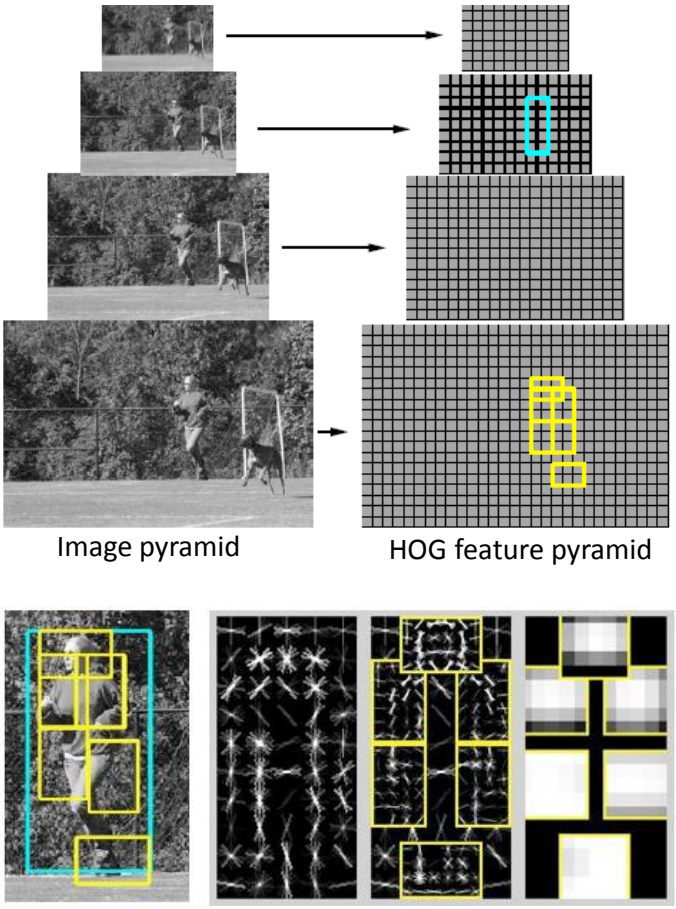


Figure 3.1: Example detection obtained with the person model of *multiscale deformable part model* from Felzenszwalb *et al.* (reprinted from [24]). The model is defined by a coarse template, several higher resolution part templates and a spatial model for the location of each part.

such as part position.

We aim to maintain both global structure information and ordering of local events for action recognition. Our method should incorporate both spatio structural information as [27, 52, 50] and ordering of the events as [31, 111], but avoid the increased dimensionality of the n-Grams method [31]. Inspired by **DPM**, we propose a 3D multiscale part model for video event analysis. However, instead of adopting deformable "parts", we use overlapped "parts" with fixed size and location on the purpose of maintaining both

structure information and ordering of local events for action recognition. In addition, unlike **DPM** which requires bounding box labels for the positive examples, our method is weakly supervised. It relies neither on part annotations, nor whole object bounding boxes or pre-trained object detectors.

3.2.2 Local part model

The local part model includes both a coarse primitive level root feature covering event-content statistics and higher resolution overlapping parts incorporating local structure and temporal relations. The underlying building blocks for our models are local spatio-temporal (ST) volume patches, which can be extracted by dense sampling or local spatio-temporal feature detectors, *e.g. Harris3D* [48], *Cuboid* [21], *Hessian3D* [129]. The local ST features can be represented by different types of 3D descriptors, such as *HOG/HOF* [50], *HOG3D* [42], *MBH* [120], *ESURF* [129] *etc.*

As shown in Figure 3.2, for a video V_p with size of $W \times H \times T$, we create a new video V_r with size $W/2 \times H/2 \times T$ by down-sampling. We then extract local 3D patches at regular positions and scales in space and time from V_r as coarse features for "root" model. For every "root" model, a group of finer "part" models are extracted from the video V_p with respect to the location where the coarse patch serves as a reference position.

Our model consists of a coarse root patch and a group of higher resolution part patches. The histograms of root patch and all part patches are concatenated to create a histogram representation of a local ST feature. Both the coarse root patch and the higher resolution part patches are described by any of the 3D descriptors (*HOG/HOF* [50], *HOG3D* [42], *MBH* [120], *ESURF* [129] *etc.*), and act with the same classification power as the the used descriptor. However, our local part model incorporates the structure and temporal ordering information by including local overlapping "events". Thus, it provides

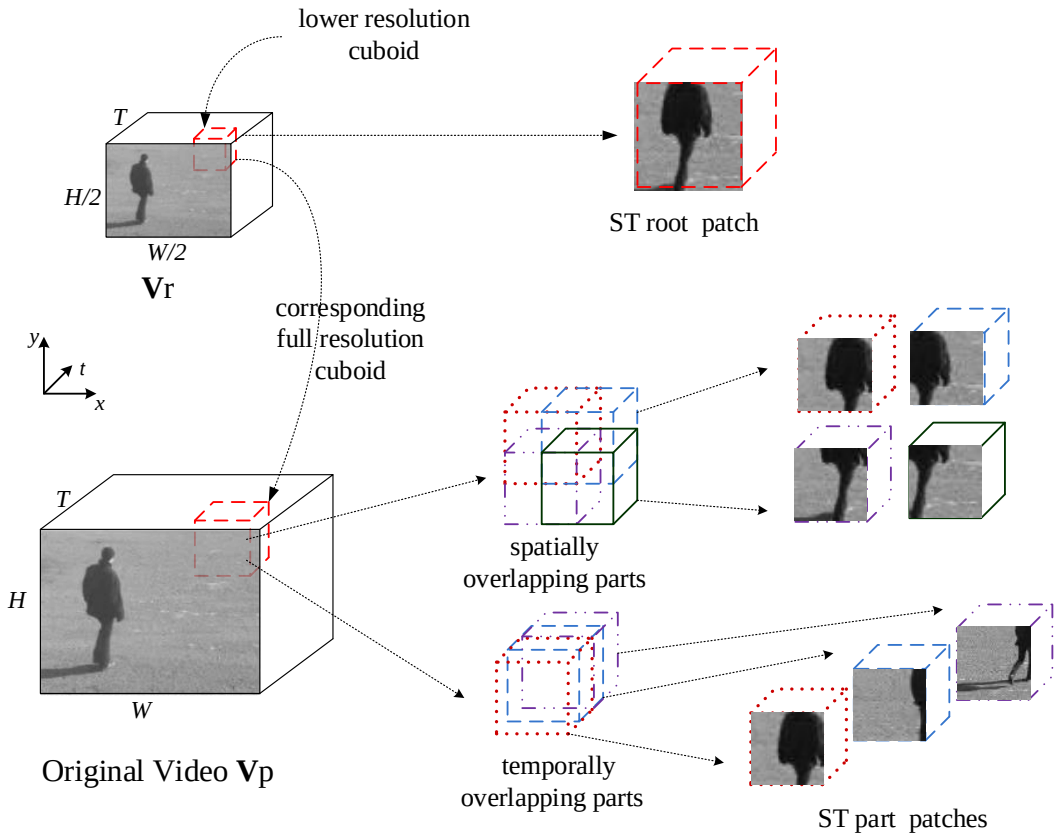


Figure 3.2: Example of Local Part Model defined with a root and an overlapping grid of parts. The model includes both a coarse primitive level root patch covering event-content statistics and higher resolution overlapping part patches incorporating structure and temporal relationships.

more discriminative power for action recognition

3.3 Implementation details

In order to evaluate the proposed local part model with comparable results, we closely followed the dense sampling experiments of Wang *et al.* [123]. In summary, we applied the proposed local part model to extract local spatio-temporal features by dense sampling at

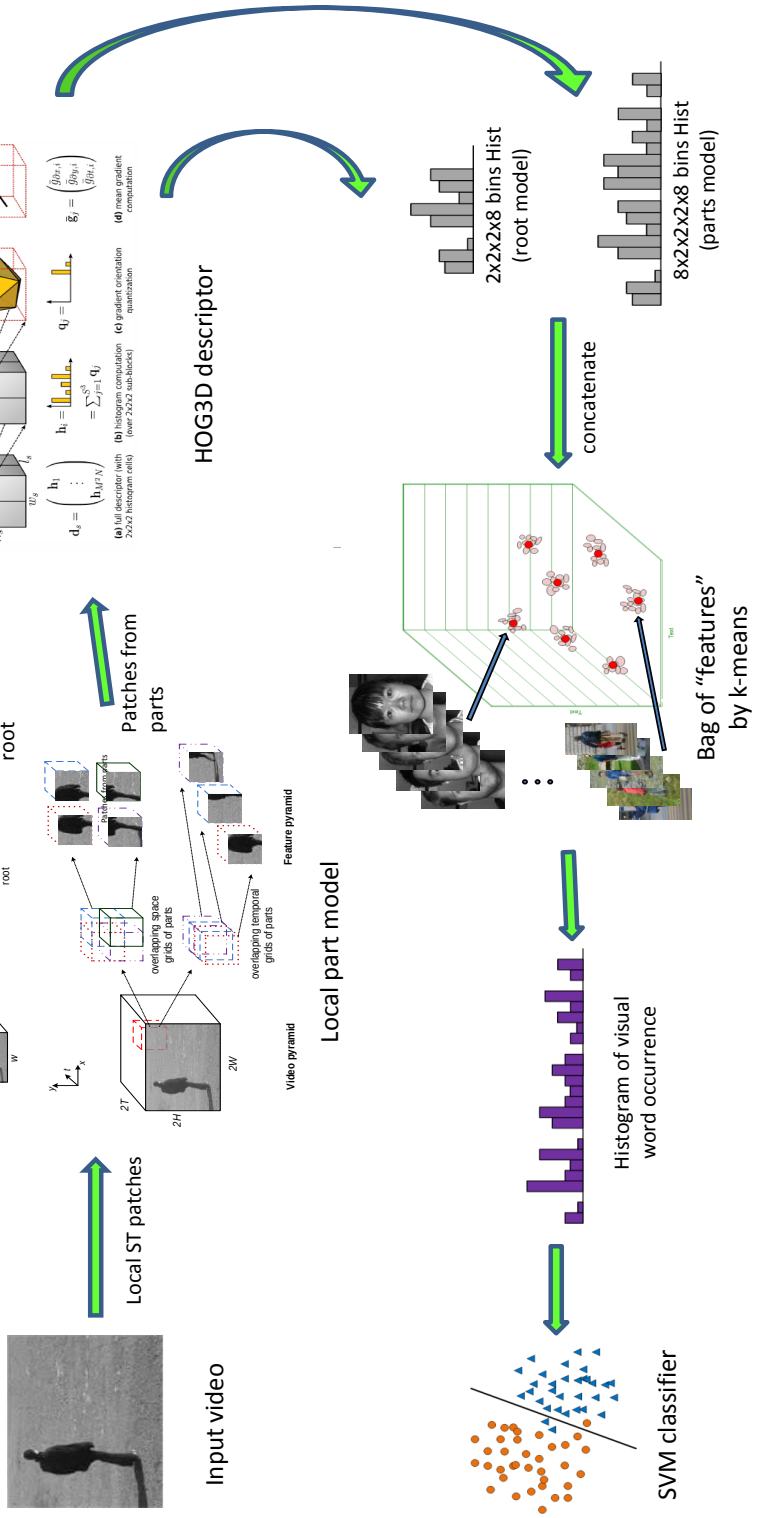


Figure 3.3: A framework of the proposed method. Note: the HOG3D descriptor is reprinted from [42]

multiple spatial and temporal scales, and used them to represent a video with a standard bag-of-features approach. The sampled 3D ST patches were represented by *HOG3D* [42] descriptor, and quantized into their nearest codewords with Euclidean distance as histograms of visual word occurrences. To generate codewords, we randomly selected 120,000 training features from training data, and used k-means to cluster them into visual vocabulary. The resulting histograms of visual word occurrences were fed into a non-linear SVM with RBF(radial basis function) kernel for classification. Figure3.3 shows the framework of the proposed method.

3.3.1 spatio-temporal features by dense sampling

We used the dense sampling to extract local spatio-temporal patches from the video at different scales and locations. In order to compare the performance fair, we used similar sampling strategies as those in [42, 123]. A sampled ST patch is decided by 5 parameters $[x, y, t, \delta, \tau]$, where δ and τ are the spatial and temporal scale, and (x, y, t) is its space-time location in the video. For a 3D patch $s = (x_s, y_s, t_s, \delta_s, \tau_s)$, a feature can be computed for a local ST region with the size of width w_s , height h_s and length l_s given by

$$w_s = h_s = \delta_0 \times \delta_s \quad and \quad l_s = \tau_0 \times \tau_s \tag{3.1}$$

where δ_0 and τ_0 are the initial spatial and temporal scales, respectively, and δ_s and τ_s are the spatial and temporal step factors for consecutive scales.

For the "root" model, we used the similar dense sampling method as the approach [42, 123]. As stated in Section 3.2.2, we extracted coarse features from the video V_r . In our experiments, we defined the initial values of δ_0 and τ_0 , and set scale steps as $\delta_s = \tau_s = (\sqrt{2})^i$, given $i = 0, 1, 2, ..., K$ as the i^{th} step of the scales.

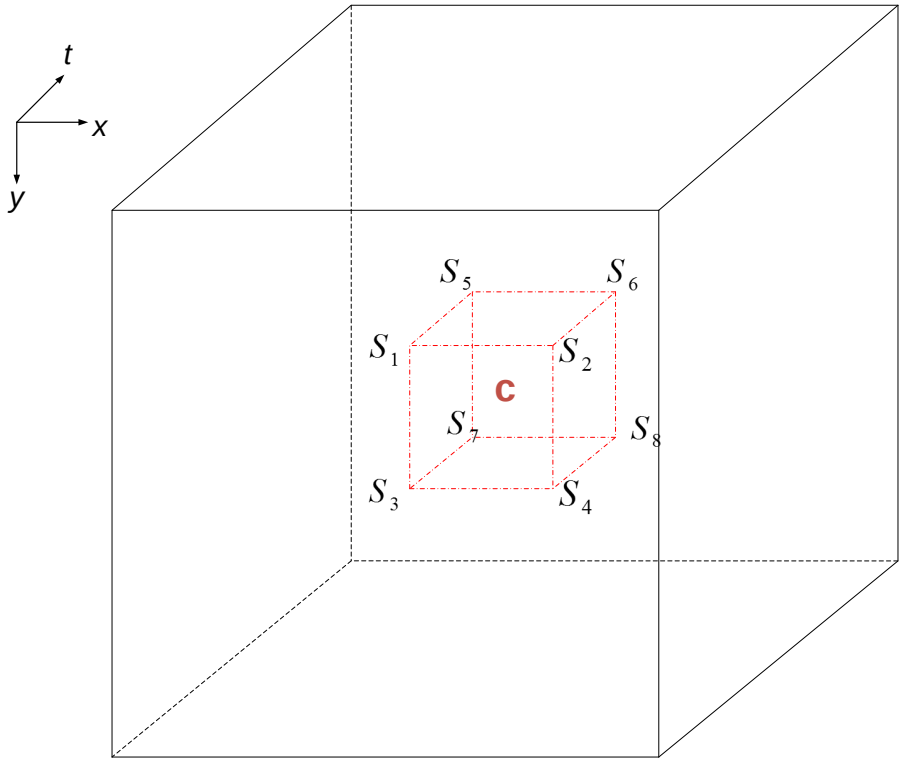


Figure 3.4: Integral video feature computation. Volume C can be computed with eight array references: $S_8 - S_7 - S_6 + S_5 - S_4 + S_3 + S_2 - S_1$.

As show in Figure 3.2, given a "root" 3D patch with the size of $w_s \times h_s \times l_s$ at the location (x_s, y_s, t_s) of video V_r , we first extracted a ST patch of size $2w_s \times 2h_s \times l_s$ at the location $(2x_s, 2y_s, t_s)$ of the high resolution video V_p . To construct a group of fine "part" models, this patch was then divided into a set of overlapping $M_s \times M_s \times N_s$ sub-patches. In our experiments, the neighbouring sub-patches had 50% overlapping area, which showed good performance.

3.3.2 Integral video

In our experiments, we sampled the volume patches with different spatial and temporal scales at various location and time for both "root" and "part" videos. One strategy

to improve computational efficiency is to use spatio-temporal "pyramids" as those in [50, 51, 120]. However, for each spatio-temporal scale, the video needs to be rescaled and stored. If we build a spatio-temporal pyramid with $\sigma_{xy} = \sigma_t = \frac{\sqrt{2}}{2}$ over a total of 8 spatial scales and 2 temporal scales, the total memory needed would be increased by a factor of:

$$z = (1 + \frac{\sqrt{2}}{2}) \times \frac{1 - (\frac{\sqrt{2}}{2})^8}{1 - \frac{\sqrt{2}}{2}} \approx 5.46. \tag{3.2}$$

Since our local part model includes a "part" video and a "root" video at half the resolution, the total memory needed for extra data would be $z' \approx 5.46 \times (1 + \frac{1}{4}) \approx 6.83$ times the processed video.

Therefore, we employed the same strategies as those in [40, 42] by using integral video, a memory-efficient alternative. Integral video is a spatio-temporal extension of the integral image proposed by Viola and Jones [118] for efficient computation of Haar features. An integral video at spatial location (x, y) and time t is defined as the sum of all pixel values at locations less than or equal to (x, y, t). The integral video iv can be described as:

$$iv(x, y, t) = \sum_{t'<=t} \sum_{y'<=y} \sum_{x'<=x} i(x', y', t'), \tag{3.3}$$

where $i(x', y', t')$ is the pixel value at the location (x', y', t') of the input image sequence. The integral video can be computed efficiently in one pass over the image

sequence using the following recurrence:

$$s_1(x,y,t) = s_1(x,y-1,t) + i(x,y,t) \tag{3.4a}$$

$$s_2(x,y,t) = s_2(x-1,y,t) + s_1(x,y,t) \tag{3.4b}$$

$$iv(x,y,t) = iv(x,y,t-1) + s_2(x,y,t), \tag{3.4c}$$

where $s_1(x,0,t) = s_2(0,y,t) = iv(x,y,0) \equiv 0$. Given the input video with size of (W,H,T), the computed integral video size is $(W+1,H+1,L+1)$. Therefore the total memory requirement for integral video is only marginally more than original video. Also, with integral video, the sum of values within any rectangular volume can be calculated with 7 additions/subtractions, independent of the volume's size and location. As shown in 3.4, for the 3D rectangular volume $\mathbf{c} = (x_0, y_0, t_0, w, h, l)$ with width (w), height (h), and length (l) at location of (x_0, y_0, t_0), we can compute its sum as:

$$\begin{aligned} S_0 = {} & iv(x_0+w, y_0+h, t_0+l) - iv(x_0, y_0+h, t_0+l) \\ & - iv(x_0+w, y_0, t_0+l) + iv(x_0, y_0, t_0+l) - iv(x_0+w, y_0+h, t_0) \\ & + iv(x_0+w, y_0, t_0) + iv(x_0, y_0+h, t_0) - iv(x_0, y_0, t_0). \end{aligned} \tag{3.5}$$

3.3.3 HOG3D descriptor

We used *HOG3D* [42] descriptor to represent the local ST features. *HOG3D* is built up on 3D oriented gradients, and can be seen as an extension of the 2D *SIFT* [58] descriptor to video data. As shown in Figure 3.5 , it can be computed as follows:

- Spatio-temporal gradients are computed for each pixel over the video. The mean

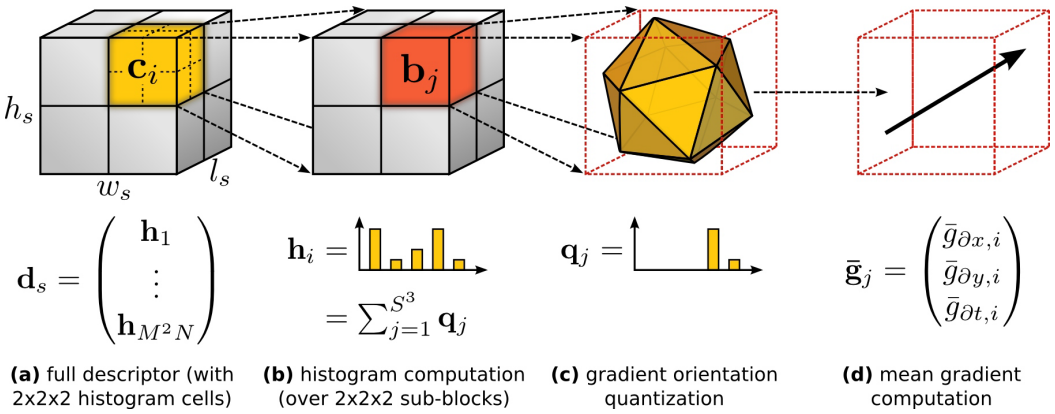


Figure 3.5: *HOG3D* descriptor computation (reprinted from [42]); (a) the support region around a point of interest is divided into a grid of gradient orientation histograms; (b) each histogram is computed over a grid of mean gradients; (c) each gradient orientation is quantized using regular polyhedrons; (d) each mean gradient is computed using integral videos.

gradients for each 3D patches are computed efficiently with integral video method.

- A 3D patch is divided into a grid of $M_c \times M_c \times N_c$ cells. Each cell is then divided into $M_b \times M_b \times N_b$ sub-blocks. A mean 3D gradient is computed for each sub-block.
- Each mean gradient is quantized using a regular polyhedron.
- For every cell, a 3D histogram of oriented gradients is obtained by summing the quantized mean gradients of all its blocks.
- The 3D histogram of oriented gradients for the 3D patch is formed by concatenating gradient histograms of all cells.

Orientation quantization with polyhedron

Built on HOG descriptor, HOG3D uses regular polyhedron to quantize spatio-temporal 3D gradient into a n-bin histogram. A regular polyhedron with congruent faces is re-

ferred to as platonic solid. There are only five of them: the tetrahedron (4-sided), cube (6- sided), octahedron (8-sided), dodecahedron (12-sided), and icosahedron (20-sided). Kläser *et al*. [42] used the dodecahedron or the icosahedron for 3D gradient quantization.

3.3.4 Bag-of-features representation

We used a standard bag-of-features method to represent a video sequence. Given a video sequence, a set of local spatio-temporal features are extracted and quantized into visual words. The classification is performed by measuring the frequency of visual word occurrences in the video. This method requires a visual vocabulary (or codebook). To construct word vocabulary, the local ST features are densely sampled from the training videos and represented with the HOG3D descriptors. The k-means [7] method is applied to cluster the features into k centres. The word vocabulary of k visual words is thus created with the centres. Each feature from a video sequence can be assigned to the closest (Euclidean distance) word from the vocabulary, and videos can be represented as the histogram of visual word occurrences.

3.3.5 Classification: support vector machines

For classification, we used a non-linear support vector machine (SVM) with a RBF (radial basis function) kernel:

$$K(x_i, x_j) = \exp(-\gamma \, ||x_i - x_j||^2), \quad \gamma > 0. \tag{3.6}$$

For SVM, we used the LIBSVM [18] library. In our experiments, the vector $\mathbf{x}_i$ was computed as the histogram of visual word occurrences. Data scaling was performed on all training and testing samples. The best parameters for the SVM and kernel were found

through 10-fold cross-validation procedure on training data. For multi-class classification, we used the one-versus-one approach implemented by max-wins voting.

3.4 Experiments

We have performed a number of experiments to test the performance of the proposed method. In order to evaluate our method with comparable results, we closely followed the dense sampling experiments of Wang *et al*. [123]. In summary, we used the proposed local part model to extract local spatio-temporal features by dense sampling ST patches at multiple spatial and temporal scales. The ST patches were described with the HOG3D descriptor and quantized into visual words using visual vocabulary built with k-means from training data. The bag-of-features SVM approach was applied for the classification.

We performed our experiments on **KTH** [95] action dataset 2.3.1. In our experiment, we followed the experimental settings as those in [42, 120, 123] by dividing the sequences into testing set (2,3,5,6,7,8,9,10 and 22) and training set (the remaining subjects). We trained a non-linear SVM on training set and reported the average accuracy over six classes on testing set.

3.4.1 Parameter settings

We evaluated the overall recognition performance of different experiments based on the various sampling and HOG3D parameters. Such parameters are:

Dense sampling parameters

We first down-sampled original video by 2 into 80×60, with the temporal scale unchanged. The "root" model was extracted from this video by dense sampling. The

Feature size	Number of codewords		
	2k words	3k words	4k words
360	87.83%	89.19%	90.26%
540	88.97%	90.88%	91.19%
864	89.33%	90.79%	91.77 %

Table 3.1: Average accuracy on KTH dataset with 2000, 3000, 4000 codewords. The experiments were performed to evaluate the proposed LPM, with 1 root and 8 parts.

minimal (initial) spatial size of sampling was set as 12, and the further scales were sampled with a scale factor of $\sigma_s = \sqrt{2}$ with up to 8 scales. For the temporal length, the minimal size of 6 and 10 frames was evaluated, each combined with 2 and 3 sampling scales with a scale factor of $\tau_s = \sqrt{2}$, respectively. The overlapping rate for spatio-temporal dense patches was 50%. As for "part" models, their location and sampling parameters were decided by the "root" model, given the sampling was performed in the original high resolution video.

HOG3D parameters

Different combinations of following parameters were examined: number of histogram cells, number of blocks and polyhedron types(icosahedron or dodecahedron). Other HOG3D parameters were set based on the optimization of those in [42, 123]: the orientation type as full orientation, number of supporting mean gradients $s = 3$, and cut-off value $c = 0.25$.

BoF visual vocabulary

The codebook size was evaluated with the value of 2000, 3000 and 4000 visual words.

3.4.2 Experimental results

In our experiments, we chose parameter settings to make it computationally tractable, mainly by controlling number of cells to limit the vector size of features. Table 3.1 shows our experimental results with different feature dimensions. In general, the best performance is observed with 4000 codewords with feature dimension 864. In all experiments, 4000 codewords give better results than 3000 and 2000 codewords . For feature dimension, it is shown that the performance improves as feature dimension increases from 360 to 860.

We obtained the average accuracy of 91.77% with following optimal parameters: codebook size $V = 4000$; minimal patch size $\sigma_0 = 12, \tau_0 = 6$; total sampling scales $8 \times 8 \times 3$; number of histogram cells $2 \times 2 \times 2$; polyhedron type $dodecahedron(12)$; and number of parts per root $2 \times 2 \times 2$. The dimension for a root model is $2 \times 2 \times 2 \times 12 = 96$. The vector size of a LPM feature is $96 \times (1(root) + 8(parts)) = 864$.

We also observed a slight decreased performance of 91.19% in a feature dimension of 540 with parameters: codebook size $V = 4000$; minimal patch size $\sigma_0 = 12, \tau_0 = 10$; total sampling scales $8 \times 8 \times 2$; number of histogram cells $1 \times 1 \times 3$; polyhedron type $icosahedron(20)$; and number of parts per root model $2 \times 2 \times 2$. The feature dimension is $1 \times 1 \times 3 \times 20 \times (1(root) + 8(parts)) = 540$. A performance of 90.26% was achieved when the dimension is reduced to 360 by changing above parameters as: minimal patch size $\sigma_0 = 12, \tau_0 = 6$; total sampling scales $8 \times 8 \times 3$; and number of histogram cells $1 \times 1 \times 2$.

We compared our method with other dense sampling results on the KTH dataset. Table 3.2 shows the comparison between the average class accuracy of our results and those reported in an evaluation framework [123]. Compared with the other approaches adopted dense sampling for feature extraction, our method achieves 3.77% improvement.

As for the computational complexity, since the dense sampling for root model is

HOG/HOF [50]	HOG [50]	HOF [50]	HOG3D[42]	Ours
86.1 %	79.0%	88.0%	85.3 %	91.77 %

Table 3.2: Comparison of average accuracy on KTH with other results of dense sampling methods in the literature.

performed on the video with half the spatial resolution, the total number of patches is far less than those in [123] with full spatial resolution. Although we need to extract the part patches in the original video, the total number of features is only decided by the root model, and the computation of HOG3D descriptor on part patches is handled efficiently with integral video.

3.5 Conclusions

In this chapter, we presented a novel local part model for human action recognition from video. We used HOG3D descriptor and bag-of-features method to represent video. To overcome the unordered events of bag-of-features approach, we proposed a multiscale local part model to preserve temporal orderings. The method includes both a coarse primitive level root model covering event-content statistics and higher resolution overlapping part models incorporating structure and temporal relations.

Our system builds upon several recent ideas including dense sampling, local spatio-temporal (ST) features, HOG3D descriptor, bag-of-features representation and non-linear SVMs. The preliminary results showed that our approach outperformed other dense sampling methods on KTH action dataset.

Chapter 4

Feature Extraction by Random Sampling

4.1 Introduction

In last chapter, we proposed a Local Part Model to address the orderless issue of bag-of-features representation. The preliminary results showed that it outperformed the state-of-art dense sampling methods on KTH action dataset. In this chapter, we will focus on improving the performance of the Local Part Model.

A recent trend in vision recognition is to use dense sampling over sparse interest point detectors for better performance. Dense sampling has been shown to produce good results for image classification [1, 54]. For action recognition, Wang *et al*. demonstrated in [123] that dense sampling at regular space-time grids outperformed state-of-the-art interest point detectors. Similar results have also been observed in [35, 120]. Compared with interest point detectors, dense sampling captures most information by sampling every pixel in each spatial temporal scale. However, such approaches are often computationally intractable for large video datasets.

Uniform random sampling [76], on the other hand, can provide performance comparable to dense sampling. A recent study [117] showed that action recognition performance could be maintained with as little as 30% of the densely detected features. Mathe and Sminchisescu also showed similar results in [63]. Given the effectiveness of the uniform sampling strategy, one can think of using biased random samplers in order to find more discriminative patches. Yang *et al.* [131] were able to identify more features on the object of interest by using a prior distribution over patches of different locations and scales. Liu *et al.* [57] selected the most discriminative subset from densely sampled features using the AdaBoost Algorithm. [63, 117] were based on the idea that eye movement of the human viewers is the optimal predictor of visual saliency. They measured the eye movement of human observers watching videos, and used the data to produce an "empirical" saliency map. By using such saliency maps, they pruned 20-50% of the dense features and achieved better results. The requirement of prior eye movement data renders such methods impractical for real applications. In addition, because of computational constraints, these methods didn't explore high sampling density schemes to improve their performance.

Inspired by the success of random sampling approach in image classification [76], we apply random sampling for action recognition in this chapter. We will use very high sampling density for efficient and accurate action classification.

4.2 Sampling strategies

Dense sampling grid

We perform uniform random sampling on a very dense sampling grid. We follow the same multi-scale dense sampling method as in [100] but with denser patches. A feature

Spatial resolution	Cuboid[21]	Dense[123]	Ours
80 x 60			767
160 x 120			4,295
360 x 288	44	643	27,950

Table 4.1: Average number of generated features per frame for different methods. Our numbers are based on a video length of 160 frames. The numbers of Cuboid and Dense are based on the report in [123].

point is determined by 5 parameters $[x, y, t, \sigma, \tau]$. A 3D video patch centred at (x, y, t) is sampled with a patch size determined by the multi-scale factor (σ, τ). The consecutive scales are obtained by multiplying σ and τ by a factor of $\sqrt{2}$. In our experiments with HOG3D, we set the minimal spatial size to 16 x 16 pixels and minimal temporal size to 10 frames. With a total of 8 spatial scales and 2 temporal scales, we sampled the video 16 times.

A key factor governing sampling density is the overlapping rate of sampling patches. We explore very high sampling density with 80% overlap for both spatial and temporal sampling. Table 4.1 shows the comparison of the average number of generated features per frame for different methods. The features produced with cuboid and dense sampling in [123] were sampled from videos with resolution of 360 x 288 pixels. At same resolution, we generate 43 times more features than the dense sampling method in [123].

Random sampling strategies

For an image of size $n \times n$, the number of possible sampled patches is n^4 [46]. Nowak *et al*. have shown in [76] that the performance was always improved as the number of randomly sampled patches was increased with as many as 10000 points per image. For video recognition, such an approach would be computationally prohibitive. Therefore, we have to use some strategies to reduce number of sampled points per frame and at the

same time maintain an adequate sampling density.

One solution is to apply sampling at lower spatial resolution. As discussed above, the Local Part Model is well suitable for maintaining sampling density. The dense sampling grid is determined by the *root* filter, which is applied at half the resolution of the processed video. Additional approach is to perform bias random sampling which selects a small subset of the potential feature samples.

On the purpose of efficiency, we use random sampling instead of directly applying dense sampling. The total number of feature patches obtained from dense sampling is determined by sampling parameters and size of the video. With sampling parameters fixed, number of features generated from dense sampling varies largely depending on the size of the video. For example, on HMDB51, some clips can have over 100K features, whereas others (with 19 frames per video) have only 4K features. Therefore, we randomly sample features for a good compromise between efficiency and a sufficient number of features.

In order to achieve fast processing and make the comparison across different dataset fair, we use the same spatial video resolution as [93] in our experiments. For the videos with different spatial resolution, we down-sample them to the same height of 120 pixels with the preserved aspect ratio. Table 4.2 shows the average number of dense points (the third column) per video for different datasets. It also includes the average percentage of random samples *vs*. the total number of densely sampled points. For example, the average video size of HMDB51 is 182 x 120 pixels and 95 frames, and we randomly sample 10000 patches from the dense grid of 87,249 points. The dense grid is decided by *root* model, which is performed on half the video size (91 x 60 pixels and 95 frames).

We randomly select 4000, 6000, 8000 and 10000 features for each video, and report the classification results for each of them. They are chosen uniformly from the dense

Dataset	Avg. video size	Dense samples	10,000 samples
KTH	80 x 60 x 94	72,324	13.83%
UCF50	80 x 60 x 199	129,150	7.74%
HMDB51	91 x 60 x 95	87,249	11.46 %

Table 4.2: The sampling percentage of 10,000 random samples *vs*. total points (the third column) of dense sampling for different datasets.

grid, so samples at finer scales predominate. We set the maximal video length to 160 frames. If the video is longer than 160 frames, we simply divide it into several segments, and select features at same rate for each segment. The 10k random samples represent 7.74-13.83% of the total dense points (depending on dataset). Our sampling density is much higher than [120, 123]. Also, compared with [117] which randomly discarded up to 60-70% of dense points, our method uses a much higher pruning rate.

4.3 Experiments

To demonstrate the performance of our sampling strategy, we evaluated our method on three public action benchmarks, the KTH [95], the UCF50 [85] and the HMDB51 [45] datasets. We randomly sampled 3D patches from the dense grid, and used them to represent a video with a standard bag-of-features approach. To generate codewords, we randomly selected 120,000 training features, and used k-means to cluster them into 4000 and 6000 visual words.

The sampled 3D patches were represented with HOG3D descriptor [42] as feature vectors. Feature vectors were matched to their nearest visual word with Euclidean distance. The resulting histograms of visual word occurrences were fed into a non-linear SVM with RBF kernel for classification. In order to demonstrate the effectiveness of our method, instead of performing parameter optimization with a greedy search, we used the

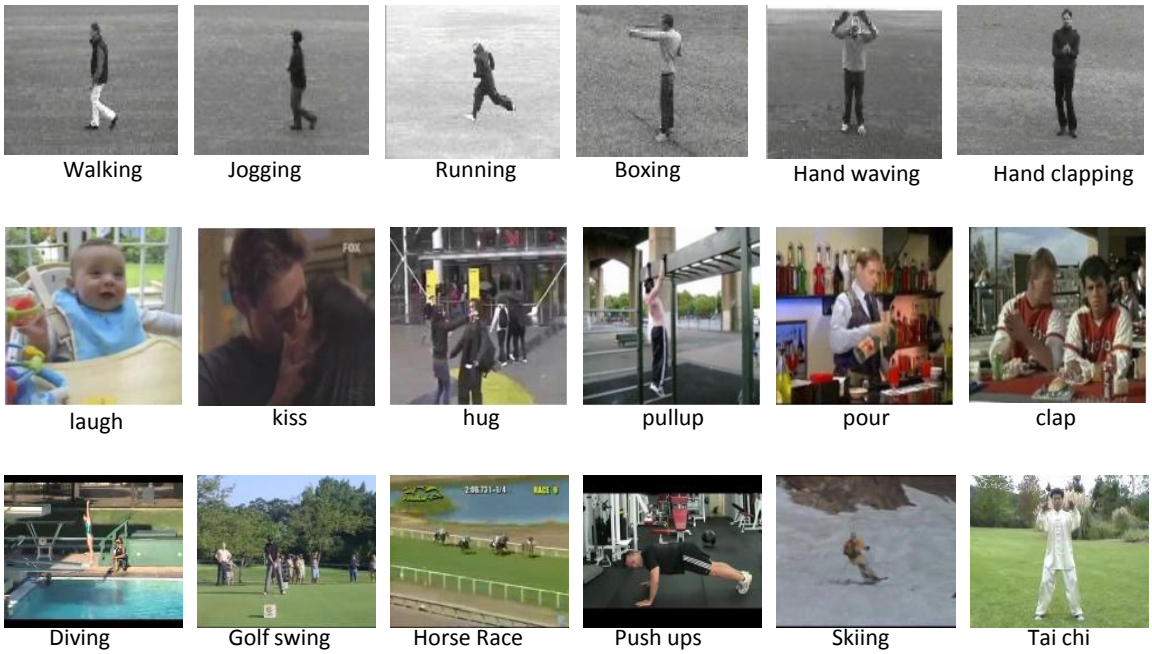


Figure 4.1: Sample of frames from KTH (first row), HMDB51 (second rom) and UCF50 (last row). The frames from KTH share same background for all categories. The cluttered background are shown on both HMDB51 and UCF50 datasets.

fixed SVM parameters as $C = 62.5, \gamma = 0.00225$ for all our experiments. For multiclass SVM, we used OpenCV one-versus-one approach implemented by max-wins voting.

To compensate for the random sampling, we repeated every experiment 3 times, and reported average accuracy and standard deviation over 3 runs.

4.3.1 Datasets

The **KTH** dataset [95] is an older dataset. It contains six action classes: walking, jogging, running, boxing, hand waving and hand clapping. Each action is performed by 25 subjects in four different scenarios: outdoors, outdoors with scale variation, outdoors with different clothes and indoors. The background is static and homogeneous. The better performances have been obtained on the methods which focus on the foreground

human motion. Such systems include feature point approaches [21, 95, 129] and template based approaches [91]. Because we randomly selected the features, all parts of the scene have equal probability. In our experiments, we followed the experimental settings as those in [120, 123] by dividing the videos into testing set (2,3,5,6,7,8,9,10 and 22) and training set (the remaining subjects). We trained a non-linear SVM on training set and reported the average accuracy over six classes on testing set.

The **UCF50** dataset [85] contains 50 classes and 6680 realistic videos taken from YouTube. The videos are grouped into 25 groups, where each group consists of a minimum of 4 action clips. The video clips in the same group may have similar background or be played by the same subjects. The dataset is very large and relatively challenging due to camera motion, cluttered background, large scale variation, etc. We followed the experimental setup of Action Bank [91] with 10-fold video-wise cross-validation and 5-fold group-wise cross-validation.

The **HMDB51** dataset [45] has 51 action categories, with at least 101 clips for each category. It is perhaps the most realistic and challenging dataset. The dataset includes a total of 6,766 video clips extracted from movies, the Prelinger archive, Internet, Youtube and Google videos. Three distinct training and testing splits have been selected from the dataset, with 70 training and 30 testing clips for each category. We used the non-stabilized videos with the same three train-test splits as the authors [45], and reported the mean accuracy over the three splits.

4.3.2 Parameters

There are a few parameters for our methods, which determine the feature dimensions.

Local part model. The *root* patches of *local part model* are sampled from the dense sampling grid of the processed video with half resolution. For each *root* patch, we sample

8 (2 by 2 by 2) overlapping *part* patches from the full resolution video. Both *root* and *part* patches are represented with HOG3D descriptor. The histograms of 1 *root* patch and 8 *part* patches are concatenated as one local ST feature vector. Therefore, each feature is 9 times the feature dimension of one HOG3D descriptor.

HOG3D. We focus on computational efficiency in setting HOG3D parameters. For each feature, we have 9 HOG3D descriptors (1 *root* and 8 *parts*) to compute. With up to 10000 features per video, even though we use integral video for fast 3D cuboid computation, it is still computationally challenging. Therefore, instead of optimizing the performance with the authors' original parameters [42], we choose the HOG3D parameters for fast computation.

We test our method with three different HOG3D feature dimensions: 60, 96 and 144. For the dimension 60, the parameters are: number of histogram cells $M = 1$, $N = 3$; number of sub-blocks $2 \times 2 \times 2$; and polyhedron type icosahedron(20) with full orientation. For the dimension 96, the parameters are: number of histogram cells $M = 2$, $N = 2$; number of sub-blocks $2 \times 2 \times 3$ for KTH, $1 \times 1 \times 3$ for UCF50 and HMDB51; and polyhedron type dodecahedron(12) with full orientation. For the dimension 144, the parameters are: number of histogram cells $M = 2$, $N = 3$; number of sub-blocks $2 \times 2 \times 2$ for KTH, $1 \times 1 \times 2$ for UCF50 and HMDB51; and polyhedron type dodecahedron(12) with full orientation. For all cases, the cut-off value is $c = 0.25$. With one HOG3D descriptor at dimension of 60, 96 or 144, our local part model feature (with 1 *root* and 8 *parts*) has a dimension of 540, 864 or 1296, respectively.

4.3.3 Results

Table 4.3 illustrates our experimental results for all three datasets. All the tests were run with the parameters listed in previous subsection. For each video, we randomly sampled

Feature size	Samples	KTH		UCF50		HMDB51	
		4k words	6k words	4k words	6k words	4k words	6k words
540	4000	87.6% ± 0.33	87.9% ± 0.42	60.5% ± 0.37	61.9% ± 0.13	28.1% ± 0.44	29.2% ± 0.29
	6000	87.9% ± 0.85	88.8% ± 0.77	61.1% ± 0.05	62.0% ± 0.08	29.0% ± 0.12	29.7% ± 0.13
	8000	88.8% ± 0.61	88.1% ± 0.60	61.4% ± 0.18	62.7% ± 0.08	28.8% ± 0.05	29.6% ± 0.09
	10000	88.6% ± 0.42	88.8% ± 0.31	61.6% ± 0.14	62.8% ± 0.15	29.3% ± 0.53	30.6% ± 0.38
864	4000	91.9% ± 0.37	92.8% ± 0.24	63.5% ± 0.11	65.1% ± 0.01	29.9% ± 0.28	30.5% ± 0.34
	6000	92.4% ± 0.12	92.5% ± 0.12	64.2% ± 0.36	65.9% ± 0.22	30.2% ± 0.58	30.7% ± 0.14
	8000	92.7% ± 0.29	92.8% ± 0.13	64.8% ± 0.18	66.2% ± 0.38	30.6% ± 0.03	31.3% ± 0.58
	10000	92.7% ± 0.29	**93.0%** ± 0.29	65.1% ± 0.24	**66.6%** ± 0.15	30.7% ± 0.18	31.2% ± 0.06
1296	4000	91.7% ± 1.40	92.7% ± 0.31	63.9% ± 0.07	64.4% ± 0.36	29.8% ± 0.08	30.7% ± 0.57
	6000	91.9% ± 0.31	92.4% ± 0.27	64.7% ± 0.08	65.2% ± 0.50	30.0% ± 0.31	31.2% ± 0.13
	8000	92.0% ± 0.71	92.7% ± 0.41	65.2% ± 0.38	65.7% ± 0.31	30.4% ± 0.37	31.0% ± 0.24
	10000	92.4% ± 0.24	92.7% ± 0.18	65.7% ± 0.37	66.1% ± 0.25	30.6% ± 0.53	**31.7%** ± 0.29

Table 4.3: Average accuracy on all three datasets with 4000, 6000, 8000 and 10000 randomly sampled features per video. The table gives the mean and standard deviations over 3 runs with 4000 codewords and 6000 codewords, respectively. 5-fold group wise cross-validation is used for UCF50. Note: if video has more than 160 frames, more features are sampled at the same sampling rate as the first 160 frames.

4000, 6000, 8000 and 10000 LPM features (1 root + 8 parts), and performed classification using a standard bag-of-features approach with 4000 and 6000 codewords, respectively. The feature dimensions of 540, 864 and 1296 were tested. To compensate the sampling randomness, all the tests were run 3 times. The mean accuracy and standard deviation were reported.

In general, the best performance is observed with 6000 codewords combined with 10000 features per video. The performance is almost always improved as the number of patches sampled from the video is increased. This is consistent with the results of random sampling for image classification [76]. Also, 6000 codewords give better results than 4000 codewords. For feature dimension, there is no clear performance advantage when using 1296 over 864. However, both of them show better results than the dimension of 540. In practice, it is preferable to use the dimension of 864 for better computational efficiency without sacrificing the performance.

On both KTH and UCF50 datasets, the best results are achieved using 6000 codewords and 10000 sampled patches at a feature dimension of 864. For HMDB51, the best

Method	Accuracy
3D Harris [95]	71.7
HOF+Dense [123]	88.0
HOG3D [42]	91.4
Local Part Model [100]	91.8
Multi-scale 3D Harris[50]	91.8
Dense Trajectories [120]	94.2
Mined hierarchical features [27]	94.5
ActionBank [91]	**98.2**
Ours	93.0 ±0.3

Table 4.4: Comparison of average accuracy on KTH dataset with state-of-the-art methods.

result is obtained using 6000 codewords and 10000 samples at a feature dimension of 1296.

One very important observation from Table 4.3 is that all values of the standard deviation are very low. Such low standard deviation demonstrates effectiveness and consistency of our methods in spite of the random sampling. As discussed on Section 4.2, 10000 random samples represent 7.74-13.83% of the dense points. The consistency of the results at such high rate of randomness can be explained by very high sampling density used by our approach.

4.3.4 Comparison to state-of-the-art

KTH is a simple dataset with six classes sharing the same homogeneous, uncorrelated backgrounds. The best results are observed on methods focusing on foreground human motion. State-of-the-art result 98.2% was achieved with Action Bank [91], which applied convolution with templates cropped spatially to cover the extent of the human motion within them. The Dense Trajectories [120] used tracked dense optical flow which also focused on the moving human objects. In our method, we applied uniform sampling

Method	HMDB51	UCF50-G	UCF50-V
HMDB51 [45]	23.2	47.9	-
ActionBank [91]	26.9	57.9	76.4
Pooling [16]	27.84	-	-
MIP [43]	29.17	-	-
Subvolume [93]	31.53	-	-
Ours	**31.7**±0.3	**66.6**±0.2	**91.5** ±0.2

Table 4.5: Comparison of average accuracy on UCF50 and HMDB51 with state-of-the-art methods in the literature(-V specifies video-wise 10-fold cross-validation, -G specifies group-wise 5-fold cross-validation).

strategy to the entire video. Therefore the similar background of all classes from KTH dataset may reduce the classification power. This is consistent with conclusion in [136] that, for "easier" datasets, using foreground and background features together does not improve the performance for image classification. Nevertheless, we obtained 93%, which is better than the uniform dense sampling methods [50, 100, 123].

For the **UCF50** dataset, we significantly outperformed the state-of-the-art method [91] by 8.7% and 15% with group-wise 5-fold cross-validation and video-wise 10-fold cross-validation, respectively. We achieved 31.7% on the **HMDB51** dataset, which also surpassed the state-of-the-art [93]. The detail performance comparison of our results with others is listed in Table 4.5. Note that the results listed in the table were reported before year 2013. For the latest performance comparison, please refer to our improved method in next chapter (as shown in Table 5.11).

Our method demonstrates very good performance on large scale challenging datasets with more realistic scenarios. One possible explanation for the good performances on real life videos is because our random sampling is conducted on a very high dense sampling grid. Compared with interest point detectors, our method has more patches sampled from the test videos, and with uniform random sampling our method also includes correlated background information. Such background information may improve discriminative

Feature size	Samples	Speed (frames per second)						
		Integral video	HOG3D & Sampling		BoF matching		Total fps	
					4k words	6k words	4k words	6k words
540	6000	75.15	457.40	cpu	124.09	81.92	41.82	36.49
				gpu	440.49	308.78	58.26	57.05
	10000	76.18	286.53	cpu	76.35	50.31	33.43	27.50
				gpu	271.67	185.24	49.22	46.62
864	6000	75.98	471.38	cpu	81.01	51.77	36.87	28.34
				gpu	271.80	185.49	49.17	51.65
	10000	77.64	293.49	cpu	49.33	32.79	27.44	21.30
				gpu	178.25	110.86	48.35	39.18
1296	6000	75.98	447.70	cpu	53.16	35.33	29.23	22.83
				gpu	186.52	126.91	49.68	45.84
	10000	74.27	280.50	cpu	31.62	21.15	20.42	15.60
				gpu	109.99	76.91	39.37	35.50

Table 4.6: Average computation speed at different stages in frames per second for HMDB51 dataset. Note: the classification stage is not included.

power for recognition on real-life videos.

Our parameters were optimized for real-time process at half spatial resolution. We expect the performance to improve further with parameter tuning in the case of full resolution videos.

4.3.5 Computational efficiency

Compared with existing methods, a major strength of our method resides in its high computational efficiency. By using random sampling, we bypass the feature detection stage, which often consists of a large portion of total computational cost. For example, Yu *et al*. [134] extended FAST 2D corners [90] into a spatio-temporal domain, and proposed a high efficient V-FAST detector for real-time action recognition. However, their feature detection stage with V-FAST detector still consists of about 1/3 total processing time.

We also use integral video to improve the efficiency. With integral video, the descriptor of a 3D patch can be computed very efficiently through 7 additions/subtractions, independent of the volume's size and location. It has the complexity of $O(N)$ where N is the number of local features. The naive computation (without integral video) has the

complexity of $O(N \times W \times H \times T)$ where W is the width of the patch, H is the height of the patch and T is the total frames of the patch. If using integral image as in [121, 122], it has the complexity of $O(N \times T)$. In addition, as discussed in Section 3.3.2, the use of integral image results in a factor of $z \approx 5.46$ more memory usage.

Table 4.6 summarizes average computational speed at different stages for HMDB51 dataset. Similar speed is observed on KTH and UCF50 datasets, on which we tested with same spatial resolution. The computation time was estimated on an Intel i7-3770K PC with an AMD HD7770 gpu @1050 Hz. Our prototype was implemented in C++. The run-time estimates for "Integral video" and "HOG3D & random sampling" were obtained on CPU parallelized with OpenMP. The speed for integral video varies little, and it only depends on the size of videos. Because there is no feature detection for random sampling, the introduction of integral video has greatly improved the speed for random sampling and HOG3D, with 280 to 471 frames per second depending on number of samples. We simply used brute-force matching to assign sampled features to their nearest visual word. Brute-force matching was the most time-consuming stage, but very suitable for GPU computing. We tested it with and without GPU. The results show that our system runs at over 30 frames per second with low feature dimensions by using only CPU and at high feature dimensions using GPU.

4.4 Discussion: dense sampling *vs.* random sampling

The video size in the datasets varies a lot for different clips, especially for total number of frames. For example, the shortest clip on HMDB51 dataset has a size of $560 \times 240 \times 19$, while the longest clip has a size of $360 \times 240 \times 1063$. Table 4.7 shows the

Video size	Sampling 1	Sampling 2	Sampling 3
280 x 120 x 19	458	1,998	9,768
280 x 120 x 160	11,679	81,252	196,298

Table 4.7: Number of generated features per video from dense sampling with three different sampling parameters, represented as "Sampling 1", "Sampling 2" and "Sampling 3".

number of generated features per video from dense sampling with three different sampling parameters. If we use dense sampling method with the parameters of "Sampling 3" and generate 9,768 features for the clip with 19 frames, the total number of features generated from a video of 160 frames would be 196,298. Such large number of features for a 160-frames video is both computationally expensive and unnecessary. For the clips with very long duration, most actions consist of repeated motions and similar structures. Schindler and Val Gool [94] showed that snippets of 5-7 frames were enough to achieve a comparable performance on staged, controlled datasets. For realistic videos, it may need more frames to classify them due to the scale changes, different viewpoints, multiple players and various backgrounds. However, it is a reasonable practice to sample features more densely from a clip with less frames. In practical applications using dense sampling, it is impossible to change the parameters based on the video duration.

Random sampling, on the other hand, can control the total number of features without any parameter changes. As shown in Table 4.7, we can use the parameters of "Sampling 2" and densely sample 1,998 features from the clip with 19 frames. At the same time, randomly choose 10,000 features on the 160-frame video. Thus, random sampling can generate denser features for clips with less frames given the fixed parameters. It also improves efficiency by processing less features for videos with long duration.

4.5 Conclusions

This chapter has introduced a random sampling method for fast action recognition. In combined with local part model, it achieved good recognition performance on two realistic large scale datasets, UCF50 and HMDB51. Compared with existing methods, a major strength of our method is real-time performance. The high computational efficiency is obtained through using integral video and fast random sampling. Our results show that good action recognition performance can be achieved by randomly selecting as few as 10% of total patches from a high dense sampling grid.

Chapter 5

Local Feature Descriptors: A Multi-channel Approach

5.1 Introduction

Recent studies [50, 120, 123, 121] have shown that local features and their descriptors have significant impact on the performance of the video recognition. Dollár *et al*. [21] evaluated different descriptors based on dense features extracted from normalized pixel values, image gradients, and windowed optical flow. They reported best results on concatenated gradients. Laptev and Lindeberg [49] compared different type of descriptors: local jets, histograms of optical flow, and histograms of spatio-temporal gradients. Their results showed that descriptors based on histograms of optical flow and spatio-temporal gradients outperformed others. Later, Laptev *et al*. [50] combined histograms of oriented spatial gradients (HOG) and histograms of optical flow (HOF) to include both local motion and appearance information. In their method, each local region was subdivided into a grid of $N \times N \times M$ cells. For each cell, a 4-bin HOG histogram and a 5-bin HOF histogram were computed, and then concatenated into the final HOG/HOF descriptor.

To extend 2D image descriptors to 3D spatio-temporal space, Scovanner *et al.* [97] proposed the 3D SIFT descriptor based on popular 2D SIFT descriptor [58]. For orientation quantization, the gradients in spherical coordinates (θ, ϕ) were divided into equally sized bins, which were represented by a 8×4 histogram. Willems *et al.* [129] extended the efficient 2D SURF descriptor [8] to 3D spatio-temporal space, called the extended SURF (ESURF) descriptor. Kläser *et al.* [42] introduced the HOG3D descriptor. In their method, the mean spatio-temporal oriented gradients were quantized using a polyhedron into a n-bin histogram. Yeffet and Wolf [132] presented the efficient Local Binary Patterns descriptor by comparing pixel values of a local region to the previous and to the next frame. Wang *et al.* [120] used MBH descriptor [20] in action recognition, which outperformed other descriptors by encoding motion boundary and suppressing camera motion.

More recently, trajectory-based methods became popular due to the good performance. Messing *et al.* [67] used KLT tracker [59] to extract feature trajectories, and quantized them in log-polar coordinates with 8 bins for direction, and 5 for magnitude. Matikainen *et al.* [64, 65] employed fixed length feature trajectories, and represented the trajectory descriptor with information containing both displacement vectors and elements of the affine transformation matrix for its assigned cluster centre. Sun *et al.* [107] tracked randomly sampled points within the region of the long-duration trajectories extracted through KLT tracker and SIFT descriptor matching. Wang *et al.* [120] proposed dense trajectories by sampling feature points on a dense grid in each frame and tracking them using an efficient dense optical flow algorithm. When combined with camera motion compensated MBH descriptor, dense trajectories achieved state-of-the-art performance on various datasets. Later, built on dense trajectories [120], both Jiang *et al.* [38] and Jain *et al.* [34] explored the methods to improve the performance from

better camera motion compensation.

In addition to mining the most discriminative features and their descriptors, a recent trend is to improve the recognition performance from a combination of multiple complementary features/descriptors. Reddy and Shah [85] applied late fusion using probabilistic fusion of motion descriptors (*e.g.* 3DSIFT and MBH) and scene context descriptor (*e.g.* GIST). In [120, 121], dense trajectories, HOG, HOF and MBH descriptors were fused with a RBF-χ^2 kernel SVM in a multichannel approach. Jiang *et al.* [38] combined amended dense trajectories, HOG, HOF and MBH descriptors. The combination was done by simply averaging the kernels computed from different representations. Jain [35] used the same multichannel χ^2 SVM as those in [120, 121] to combine trajectories, HOG, HOF, MBH and DCS descriptors. Oneaţă *et al.* [77] employed a late fusion strategy to linearly combine classifier scores computed from dense SIFT features and MBH features. Lan *et al.* [47] proposed a double fusion method to take advantage of both early fusion and late fusion strategies in fusing multiple features, such as SIFT, CSIFT, MoSIFT, STIP, ASR, OCR and GIST.

In this chapter, we focus on mining various discriminative features and combining such multiple complementary features to perform an accurate recognition of videos. We also introduce a new efficient spatio-temporal descriptor to represent local features in video, and improve the performance of the state-of-the-art MBH descriptor using a discontinuity-preserving optical flow algorithm.

5.2 Feature descriptors

In last chapter, we applied the random sampling method on our proposed Local Part Model to extract local 3D patches at regular positions and scales in space and time. That is, for each given sample point (x, y, t, σ, τ), a feature descriptor is computed for a 3D

video patch centred at (x, y, t). The descriptors are very important for the performance of the video recognition.

5.2.1 Local feature descriptors

The low-level local spatio-temporal features and bag-of-features(BoF) [99, 101, 120] representation or alternative Fisher vector encoding [122, 77] can achieve good performance for action recognition on realistic datasets. A key factor for high performance is the local descriptors, which should include both local structure and motion information. In the literature, a number of local descriptors are proposed to encode local motion pattern and structure information.

HOG descriptor was introduced by Dalal and Triggs in[19] for human detection. It is based on the popular SIFT descriptor [58]. In our implementation, image gradients are computed by applying [-1, 0, 1] filter along x- and y-axis with no smoothing. The orientation $\theta(x, y)$ and magnitude $r(x, y)$ are computed from the intensity gradients for every pixel in the image, and the orientated histograms are voted with weighting based on gradient magnitudes. For colour images, we simply use the colour channel with the maximal value from each pixel for gradient computation.

HOF descriptor was first used by Laptev *et al.* [50] to combine with HOG to incorporate both local motion and appearance. Instead of using image gradients $(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$, the horizontal and vertical components of optical flow (I^x, I^y) are employed to compute orientation and magnitude, which are used as weighted votes into local orientated histograms in the same way as for the standard HOG.

MBH descriptor was introduced by Dalal *et al.* in[20] and used by Wang *et al.* [120] on action recognition to achieve state-of-the-art performance. The horizontal and vertical components of the optical flow (I^x, I^y) are treated as two independent "images", and their

local gradients $((\frac{\partial I^x}{\partial x}, \frac{\partial I^x}{\partial y}), (\frac{\partial I^y}{\partial x}, \frac{\partial I^y}{\partial y}))$ are calculated separately. The result derivatives are used as weighted votes into local oriented histograms for each "image" in the same way as for the standard HOG.

HOG3D descriptor was proposed by Kläser [42]. It is built up on spatio-temporal oriented gradients. The ST gradients $(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}, \frac{\partial I}{\partial t})$ are computed for each pixel over the video, and saved in three integral videos. Three gradient components of a ST patch can be computed efficiently from the integral videos. A 3D patch is divided into a grid of $M_c \times M_c \times N_c$ cells, with $M_b \times M_b \times N_b$ sub-blocks per cell. A mean 3D gradient is computed for each sub-block, and quantized using a polyhedron into a n-bins histogram. The 3D histogram of oriented gradients for the 3D patch is formed by concatenating gradient histograms of all cells.

3D SIFT descriptor was extended from popular 2D SIFT descriptor [58] by Scovanner *et al.* [97] to represent spatio-temporal patches. After gradient magnitude and orientation are computed in 3D, each pixel has two values (θ, ϕ) which represent the direction of the gradients in three dimensions. A Gaussian weighted histogram for the 3D neighbourhood around a given interest point is constructed. For orientation quantization, the gradients in spherical coordinates (θ, ϕ) are divided into equally sized bins, which are represented by an 8×4 histogram. Such an representation leads to singularity problems as bins get progressively smaller at the poles.

The HOG descriptor are very efficient to compute. However, it only encodes appearance of local features and lacks of local motion information for video classification. Therefore, it is often combined with HOF descriptor to improve the performance. For HOG3D descriptor, although it is very efficient to compute ST gradients, the orientation quantization with polyhedron for each sub-block is relatively expensive considering a lot of patches sampled. Also, its descriptor has much higher dimensionality (default 960)

than HOG/HOF (default 96), which puts more constraints on efficiency. In addition, using regular polyhedron with congruent faces to quantize the ST gradients may not be an optimal option because the gradients in spatial domain and those in temporal domain should have different characters. Similar as HOG3D, 3D SIFT also has increased dimensionality (default/optimal 2048), and represents spatial gradients and temporal ones with same quantities.

Both HOF and MBH need to compute dense optical flow, which is computationally expensive. In addition, the MBH includes two descriptors, MBHx and MBHy, which add the dimensionality and complexity for codeword quantization. The optical flow computes the absolute motion which includes camera motion, and thus leads to suboptimal performance for HOF. With MBH, constant camera motion is suppressed by the gradients of the optical flow, and only information about changes in the flow field is kept. The MBH achieved state-of-the-art performance.

5.2.2 Improved MBH descriptor

MBH descriptor with local part model

In local part model, for each clip, we compute two integral videos, one for the *root* model at half resolution, and another one for the *part* models at full resolution. Note that we compute the dense optical flow for HOF and MBH at full resolution for part models, and then reuse them for root by down-sampling. In comparison to down-sampling the video frames and recomputing the dense optical flow at half resolution, there are two benefits: 1) no extra computation cost of the dense optical flow for root model; 2) improving the sub-pixel accuracy of optical flow. Dalal *et al.* [20] have observed that deep sub-pixel accuracy is very important for the performance of MBH descriptor. This is because the optical flow vectors can be very short, so any rounding to sub-pixel accuracy can

have significant impact on performance. We will provide experimental evaluation on this subject.

Depending on the spatial resolution, the dense optical flow computed from half resolution may lose more deep sub-pixel accuracy than computed from full resolution and resized by down-sampling. We use Farnebäck's approach [23] to compute dense optical flow, which adopts iterative and multi-scale displacement estimation. It starts at a coarse spatial scale to get a rough but reasonable displacement estimate and propagates it through finer scales to obtain increasingly more accurate estimates. However, if the video resolution is too low, it becomes difficult to compute reliable motion estimates at the coarser levels. In our case, the resolution for root model is 80×60, which is too low for accurate multi-scale displacement estimation.

Optical flow estimation on MBH descriptor

Dalal *et al*. [20] have shown that the accuracy of dense optical flow estimation has significant impact on the performance of MBH descriptors. Motion analysis algorithms have been developed for decades, but the state-of-the-art optical flow methods do not produce the expected results for many real-world video sequences [56]. In computing optical flow for MBH descriptor, we follow Wang *et al*. [121] to adopt Farnebäeck's algorithm [23] as a good trade-off between speed and accuracy. In this section, we will explore the potential benefits of using different optical flow algorithms.

Wang *et al*. [121] compared efficient Farnebäeck optical flow algorithm with a state-of-the-art method, large displacement optical flow (LDOF) from Brox and Malik [14]. They found that the overall performance of the two optical flow algorithms is similar. To estimate very fast moving small human body parts, LDOF combines descriptor matching and the continuation method to calculate arbitrarily large displacements. A continuous

optimization is desirable for optical flow as sub-pixel precision is required, but descriptor matching with discrete optimization can only achieve pixel level accuracy. Thus, adopting descriptor matching in optical flow estimation may lead to difficulty in distinguishing small/slow motions, which has equal importance to human action as fast moving small body parts. Also, the use of descriptor based on spatial histograms may cause the inaccuracies at motion discontinuities. Discontinuities in the flow field often appear in the area with high image gradients, which is critical to motion boundary encoding for MBH descriptor. When using LDOF on MBH, Wang *et al*. [121] observed a 3% improvement over YouTube dataset and 0.8% drop in performance over Hollywood2 dataset. This could attest our aforementioned hypothesis because YouTube dataset contains sports video with fast moving body parts, whereas Hollywood2 dataset has many slow motions, such as Kiss and AnswerPhone.

To this end, we would like to explore more accurate optical flow algorithm for "discontinuity-preserving". We evaluate duality-based TV_L1 (Dual_TV_L1) method from Zach *et al*. [135]. It is based on classical homogeneous regularization method of Horn and Schunck [32] approach. The original work of Horn and Schunck uses quadratic L^2-regularity, which does not allow for discontinuities in the optical flow field. Neither does Farnebäeck's algorithm. The Dual_TV_L1 employs L^1-regularity to better preserve discontinuities.

5.2.3 GBH descriptor

In this section, we propose a new local spatio-temporal descriptor. Our objective is to avoid the expensive dense optical flow computation, but adopt high efficient gradient-based method. We also intend to encode both local static appearance and motion information. However, we want to avoid using three gradient components as 3D SIFT

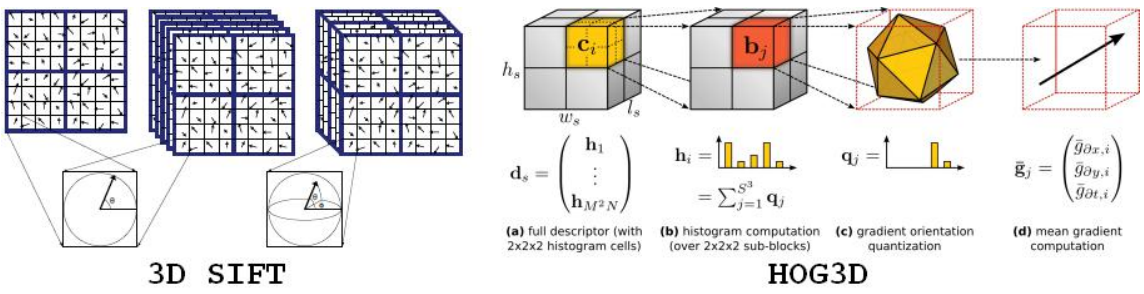


Figure 5.1: Illustration of 3D SIFT and HOG3D descriptors (reprinted from [97] and [42]). Compared to HOG/HOF with a dimension of 96, the 3D SIFT and HOG3D have a default dimension of 2048 and 960, respectively, due to the third temporal gradient component.

and HOG3D descriptors do, which leads to high dimensionality and relatively expensive quantization cost (as shown in Figure 5.1). Instead, we will use compact HOG-like descriptor with two gradient components.

For each frame in a video, we first compute image gradients using simple 1-D [-1,0,1] Sobel masks on both x and y directions. Then, we apply a [-1, 1] temporal filter over two consecutive gradient images. Thus, for each pixel, we have:

$$I_{t,x} = \frac{\partial}{\partial t}(\frac{\partial I}{\partial x}), \qquad I_{t,y} = \frac{\partial}{\partial t}(\frac{\partial I}{\partial y}) \tag{5.1}$$

Now the gradient magnitude and orientation for each pixel are defined as follows:

$$r(x,y) = \sqrt{I_{t,x}^2 + I_{t,y}^2}, \qquad \theta(x,y) = \arctan(\frac{I_{t,y}}{I_{t,x}}) \tag{5.2}$$

Our new descriptor uses a similar histogram of orientation based method voting with θ and r as in SIFT and HOG descriptors. However, instead of using image gradients, we use time-derivatives of image gradients, which show moving edge boundaries. We call this descriptor gradient boundary histogram (GBH).

Figure 5.2 illustrates the comparison of image gradients and gradient boundaries.

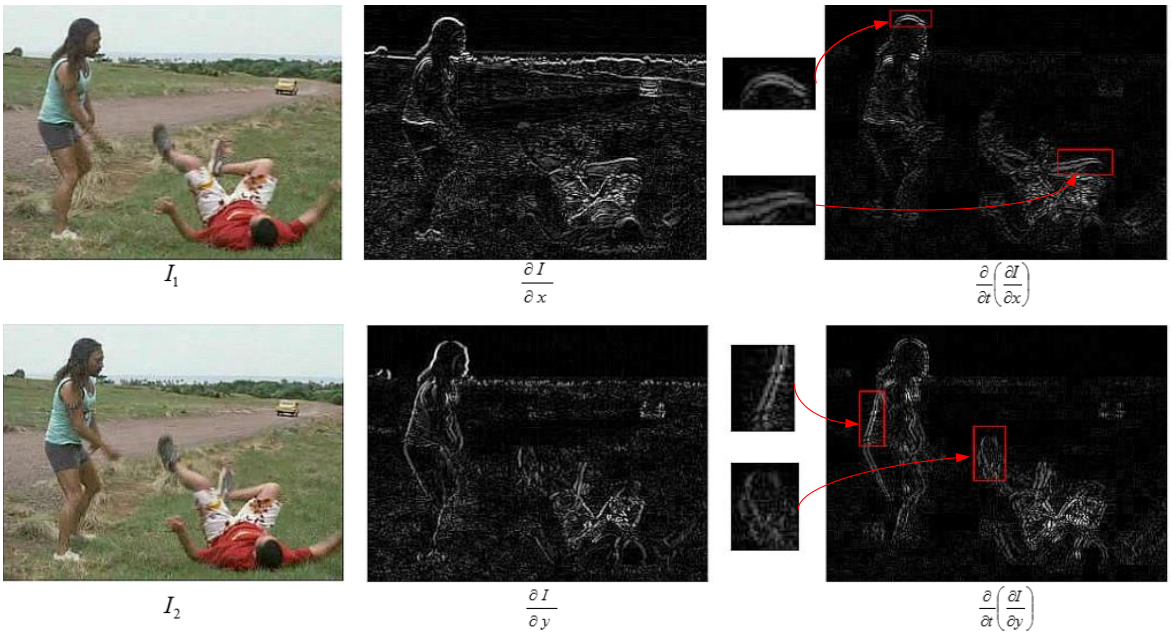


Figure 5.2: Illustration of gradients and gradient boundaries for a "fall floor" action. Compared to image gradients, gradient boundaries have less background noise. More important, gradient boundaries encode motion information. The areas inside red bounding boxes show the double edges with various distances decided by the speed of the moving body parts.

We have two important observations. First, the subtraction of two consecutive image gradients results in the removal of the backgrounds of the video sequences. Two gradient images in the centre show a lot of background noise, while the gradient boundary images on the right show clear human shapes with far less background noise. More important, gradient boundaries encode moving human shapes. As demonstrated with red bounding boxes in the Figure, the double edges with various distances are proportional to the moving speed of the human body parts. For example, the distance between double leg edges is larger than the double head edges, which represents that the leg moves faster than the head of another person in the upper right image.

It seems like that the simple gradient subtraction works well only when the camera and background are largely static. However, our in-depth experimental evaluations

showed that it achieved good performance on realistic datasets. One possible explanation may reside in that the changes of the human gradient boundaries (as shown in Figure 5.2) reflect the speed of the moving body parts.

5.3 Improved local part model

We have proposed a local part model for action recognition. Our model consists of a coarse root patch and a group of finer part patches. Both the coarse root patch and the higher resolution part patches can be represented by any of the local descriptors (*HOG/HOF* [50], *HOG3D* [42], *MBH* [120], *ESURF* [129] *etc.*). In addition, our local part model also incorporates the local structure relations and temporal ordering information by including local overlapping "events". It thus provides discriminative power for action recognition

In our previous chapters (Chapter 2, Chapter 3), each of these patches was represented by a histogram of a local descriptor, and histograms from all patches were concatenated into one vector that was 9 (1 root + 8 parts) times the original feature dimension of the used descriptor. Such an approach, however, results in large codeword quantization errors, and it also occludes the individual discriminative power of independent root model and part models. Therefore, we propose a new multi-channel approach, in which we substantially improve the method by treating the root and 8 parts as two separate channels. For each channel, a standard bag-of-features approach is applied. The resulting histograms of visual word occurrences from root and parts are concatenated into one histogram for SVM classification. We will discuss this in details in our experimental section.

5.4 Experimental setup

We evaluated our method on three public large scale action benchmarks, the UCF50 [85], the HMDB51 [45] and the UCF101 [105] datasets. We focused our evaluation on using a standard bag-of-features approach. However, we also tested our system with Fisher Vector (FV) [36] encoding. Unless stated otherwise, all the results were reported with bag-of-features approach.

Notation: we define the sampling grid based on root video, which is half the spatial resolution of the processed video. The root patches are randomly chosen from this half size video, and we will refer to it as "root video". The part patches are sampled from the processed video at full spatial resolution, which is referred to as "part video" or "processed video", interchangeably. We also use "original video" to represent the original spatial resolution of the clips from the datasets.

5.4.1 Bag-of-features approach

For each clip, we randomly sample 3D patches from the dense grid, and use them to represent a video with a standard bag-of-features approach. To generate codewords, we randomly select 120,000 training features, and use k-means to cluster them into 2000 and 4000 visual words.

The sampled 3D patches are represented by descriptors, and the descriptors are matched to their nearest visual words with Euclidean distance. The resulting histograms of visual word occurrences are fed into a non-linear SVM implemented by LIBSVM [18] with histogram intersection kernel [109]. For multi-class SVM, we use one-versus-all approach, which is observed in [35] to have better results than one-against-one multi-class SVM. However, we will also evaluate the performance of our system with one-against-one multi-class SVM and provide a comparison of both approaches.

Different visual features can be extracted from a single video and represented with various descriptors to provide complementary information. These features need to be fused to improve the classification performance. In general, based on when the information is combined, there are two types of fusion strategies: early fusion and late fusion [103].

Early fusion: the features are combined before performing a classification training. Early fusion can better learn the correlation among features. However, it may have the over-fitting problem due to limited amount of training data. Also, it often suffers from high feature dimensionality.

Late fusion: classifiers are trained for different descriptors, then the classification results are combined. Various features from different feature spaces can be fused more easily, but their relationships are often not learnt. Also, some late fusion strategies have the pre-requirement of conditional independence for different descriptors [85].

In our approach, we use early fusion method by concatenating different feature descriptors and then training a classifier. To combine multiple channels of different descriptors, most early fusion methods use RBF-χ^2 kernel [121, 136]:

$$K(x_i, x_j) = \exp(-\sum_c \frac{1}{A^c} D(x_i^c, x_j^c)), \tag{5.3}$$

where $D(x_i^c, x_j^c))$ is the χ^2 distances between the samples for the c-th channel, and A^c is the mean value of the χ^2 distance between the training samples for the c-th channel. Using mean χ^2 distance A^c has an advantage of parameter free for each channel. While this approach produced "comparable results" [136], we argue that the mean value of the χ^2 distance is not representative to the discriminative power of each individual channel. For instance, in our experiments, the MBH outperforms HOG on HMDB51 dataset by a large margin (56.4% *vs*. 28.3%). Therefore, it is intuitive to add more weight to

descriptors with higher classification power. We thus propose a histogram intersection kernel for multi-channel classification:

$$K_{IH}(x_i, x_j) = \sum_{c=1}^{C} \frac{w^c}{max(w^c)} \sum_{n_c=1}^{N_c} min(x_i^c, x_j^c) = \sum_{n=1}^{C \times N_c} \frac{w^c}{max(w^c)} min(x_i^c, x_j^c), \tag{5.4}$$

where w^c is classification accuracy for the c-th channel, which can be learnt from the training data. $max(w^c)$ is the maximal value from w^c of all channels, and N_c is the feature dimension for the c-th channel. The histogram intersect kernel is a positive semi-definite kernel [29]. Thus, the proposed kernel is Mercer kernel because each learnt w^c is positive and Mercer kernels are closed under both addition and scaling by a positive constant[98].

One advantage of this approach is its computational efficiency. Our histogram intersect kernel is similar as the approach in Spatial Pyramid Matching [52]. It is simply a weighted sum of histogram intersections. Given $w_i min(a, b) = min(w_i a, w_i b)$ for positive numbers, we can concatenate the weighted histograms of all channels, and use a single efficient intersection kernel SVM [60].

5.4.2 Fisher vector

To encode features, in addition to using bag-of-features, we will also evaluate Fisher vector (FV) [36]. In recent studies [106, 77, 121], Fisher vector has showed the improved performance over standard bag-of-features method on action recognition. FV extends the BoF by encoding high-order statistics (first and, optionally, second order) between the descriptors and a Gaussian Mixture Model (GMM). Therefore, in addition to including codewords' occurrences, it also encodes additional information about the distribution of the descriptors.

Due to the fact that decorrelated data can be fitted more accurately by a GMM with diagonal covariance matrices, it is favourable to apply PCA dimensionality reduction on Fisher vector. In addition, for a D dimension descriptor, the FV signature with K words has an increased dimension of $2KD$ (VLAD has a dimension of KD). Therefore we apply PCA on the computed LPM features. The dimensions of root descriptor and part descriptor are reduced into their 1/2 and 1/8, respectively. The detail implementation can be found in Chapter 6.2.

We use the parameters as in Section 5.4.4 to compute LPM features for GBH and MBH descriptors. The 4000 codewords are used for bag-of-features encoding. For Fisher vector, we first apply PCA to reduce root vector from 64 to 32 and part vector from 512 to 64. We set the number of quantization cells to $K = 256$ and $K = 128$ for FV, which speeds up the codewords matching process in comparison with BoF ($K = 4000$).

We use VLFeat library [114] for Fisher vector encoding. We adopt improved fisher vector [79] by applying the signed square-rooting and followed with L_2 normalization, which significantly improves the performance when combined with linear classifiers. For classification with FV, we use linear SVM with fixed parameter $C = 32.5$. To combine multiple descriptors, we simply concatenate the vectors from different channels.

5.4.3 Datasets

We perform the evaluate on three large scale action datasets, the UCF50 [85], the HMDB51 [45] and the UCF101 [105]. We provide a short review over these datasets. The details for all these datasets can be found in Section 2.3.

The **UCF50** dataset [85] contains 50 classes and 6680 realistic videos taken from YouTube. The videos are grouped into 25 groups, where each group consists of a minimum of 4 action clips. The dataset is very large and relatively challenging. We report

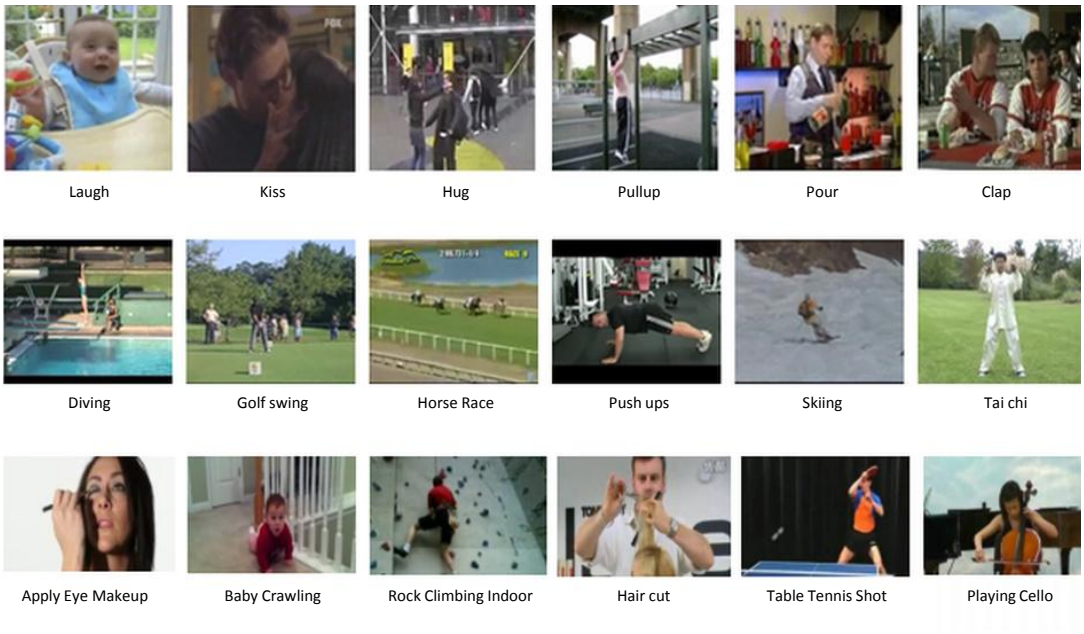


Figure 5.3: Sample of frames from HMDB51 (first row), UCF50 (second row) and UCF101 (last row).

our results with Leave-One-Group-Out (25-fold group-wise) Cross-Validation.

The **UCF101** dataset [105] is by far the largest human action dataset with 101 classes and 13320 realistic video clips taken from YouTube. As an extension of **UCF50** dataset, it includes all clips of 25 groups from **UCF50** and adds 51 categories. The dataset is relatively challenging due to camera motion, cluttered background, large scale variations, etc. We follow the original experimental setup of the authors by reporting mean accuracy over three distinct training and testing splits. For split 1, split 2 and split 3, clips from groups 1-7, groups 8-14 and groups 15-21 are selected respectively as test samples, and the rest for training.

The **HMDB51** dataset [45] is perhaps the most realistic and challenging action dataset. It has 51 action categories, with at least 101 clips for each category. The dataset includes a total of 6,766 video clips extracted from Movies, the Prelinger archive,

Internet, Youtube and Google videos. We use the original non-stabilized videos with the same three train-test splits as the authors [45], and report the mean accuracy over the three splits in all experiments.

For efficiency, unless stated otherwise, we down-sample the UCF50, UCF101 and HMDB51 videos to half the spatial resolution for all our experiments except for HOG descriptor, which is fast to compute. Since UCF101 is an extension of UCF50 and includes all clips of 25 groups from UCF50 with additional 51 categories, we will focus our experiments on UCF101 and HMDB51. However, we will compare our performance on UCF50 with state-of-the-art. Fig. 5.3 shows samples of the frames from the datasets.

5.4.4 Parameters

There are few parameters for our method, which determine the feature dimensions. Our parameters are optimized for fast process at half the spatial resolution of the tested datasets. In addition, we use the simplified HOG3D, HOG, HOF and MBH descriptors, mainly by reducing the dimensionality through controlling the total number of patch cells.

Local part model. The root patches are randomly sampled from the dense sampling grid of the processed video at half the resolution. For each root patch, we sampled 8 $(2 \times 2 \times 2)$ overlapping part patches from the full resolution video. Both root and part patches are represented as histograms with a local descriptor. However, the histograms of 1 root patch and 8 part patches are treated as two separate channels. The root channel has the same dimension as that of the used descriptor, but the histograms of 8 part patches are concatenated to create a part channel with 8 times the dimension of the root channel.

HOG3D. We evaluate our method with a descriptor dimension of 64 instead of the

default 960 in [42]. The parameters are: number of histogram cells $M = 2$, $N = 2$ (default $M = 4$, $N = 3$); number of sub-blocks $1 \times 1 \times 3$; and polyhedron type dodecahedron(12) with full orientation. The minimal patch size is $16 \times 16 \times 10$. With one HOG3D descriptor at dimension of 96 ($2 \times 2 \times 2 \times 12$), our local part model feature has a dimension of 96 and 768 for root channel and part channel, respectively.

MBH, HOF and **GBH**. The minimal patch size is $20 \times 20 \times 14$. Each patch is subdivided into a grid of $2 \times 2 \times 2$ cells, with no sub-block division. With 8-bins quantization, one descriptor of GBH, HOF , MBHx or MBHy has a dimension of 64. Therefore, for any one of these descriptors, the local part model feature has a dimension of 64 for root channel and 512 for part channel.

HOG. HOG is fast to compute. Unlike all other descriptors, instead of down-sampling the video clips into half size, we use full size videos. The minimal patch size is $24 \times 24 \times 14$. Each patch is subdivided into a grid of $2 \times 2 \times 2$ cells, with no sub-block division. With 8-bins quantization, one descriptor of HOG has a dimension of 64. The local part model feature has a dimension of 64 for root channel and 512 for part channel.

For all gradient-based descriptors (HOG3D, HOG and GBH), unless stated otherwise, we simply define r and θ at each pixel using grayscale frames.

5.4.5 Normalization

Normalization is very important to the performance. For all descriptors used, we apply the $L2$-norm over the feature vectors. For part channel, we apply $L2$-norm over each patch vector (with dimension of 64), and then renormalize the whole concatenated vector.

For bag-of-features normalization, we simply apply $L1$-norm to convert the feature into unit-length vector to eliminate the difference between short and long documents. We do not use the *tf-idf* scheme. Our experiments show no significant improvement

Dataset	GBH	HOG	HOG3D	HOF	MBH	HOG+HOF	HOG+GBH	HOF+GBH
HMDB51	38.8%	28.4%	36.2%	35.5%	51.5%	43.6%	43.0%	46.6%
UCF101	68.5%	54.1%	61.4%	61.8%	77.1%	71.8%	73.0%	73.7%

Table 5.1: Performance comparison of the proposed GBH descriptor and other local descriptors.

Spatial resolution	Speed (frames per second)				
	GBH	HOG	HOG3D	HOF	MBH
160×120	192.2	206.7	113.2	68.3	47.3
320×240	92.5	94.1	90.2	17.5	14.3

Table 5.2: Efficiency comparison of the proposed GBH descriptor and other local descriptors. Feature sampling and extraction stages are included in the run time as frames per second.

when using *tf-idf* scheme.

5.5 Results

This section evaluates our proposed methods on three datasets, HMDB51, UCF101 and UCF50. To compensate for the random sampling, we repeated every experiment 3 times, and reported average accuracy and standard deviation over 3 runs. For all descriptors, we simply concatenated bag-of-features vector from root channel and part channel without weight. Same strategy was applied to combine MBHx and MBHy channels. However, to combine multiple descriptors, we used the weighted multichannel approach (*c.f.* Eq. 5.4). Unless stated otherwise, the optical flow was computed with Farnebäeck's algorithm [23] for a good trade-off between speed and accuracy.

5.5.1 Evaluation of GBH descriptor

We performed a number of experiments with both BoF and FV to evaluate the proposed GBH descriptor.

GBH descriptor with bag-of-features

Table 5.1 shows the performance comparison of the GBH descriptor and other local descriptors. The evaluation was performed in a common experimental setup. We used the default parameters as in Section 5.4.4, and randomly chose 10K features (10K root patches + 80K part patches) from each clip with up to 160 frames.

The GBH descriptor gives surprisingly good results by itself, with 38.7% on HMDB51 and 68.5% on UCF101. It outperforms HOG, HOG3D and HOF descriptors on both datasets. However, the MBH descriptor outperforms all other descriptors by a large margin. We also evaluated the combination of GBH with HOG or HOF descriptor (the combination was applied without weight). The results consistently show improvement when combining two descriptors.

We further analyzed the computational complexity of the feature extraction stage for all descriptors. The experiments were performed on UCF101 dataset. The computation time was estimated on an Intel i7-3770K PC with prototype implemented in C++. In order to avoid built-in multi-core processing of OpenCV library, we set only one core active @ 3.5Ghz in Bios, and disabled both Hyper-threading and Turbo-boost. For all experiments, 10K features were randomly sampled from each clip. The detail results are shown in Table 5.2. As expected, the GBH is fast to compute, with similar complexity as that of HOG. Even with simplified version, the extraction of HOG3D is relatively expensive given the fact that it involves quantization cost with polyhedron. The flow based descriptors, HOF and MBH, are more expensive due to the dense optical flow estimation. As resolution increases, there is more increased cost on HOF and MBH descriptors.

Smoothing	HMDB51			UCF101		
	HOG	HOG3D	GBH	HOG	HOG3D	GBH
Yes	29.4%±0.6	37.8%±0.6	44.4%±0.3	60.6%±0.2	64.5%±0.9	74.6%±0.3
No	30.0%±0.3	38.2%±0.3	40.2%±0.7	61.2%±0.5	64.7%±0.4	73.0%±0.6

Table 5.3: The performance impact of Gaussian smooth on different descriptors. The experiments are performed on the video with original resolution.

Resolution	HMDB51			UCF101		
	HOG	HOG3D	GBH	HOG	HOG3D	GBH
364 x 240	30.0%±0.3	38.2%±0.3	44.4%±0.3	61.2%±0.5	64.7%±0.4	74.6%±0.3
182 x 120	27.5%±0.5	36.5%±0.2	44.7%±0.3	55.4%±0.4	63.6%±0.2	74.2%±0.4
91 x 60	23.7%±0.4	33.0%±0.7	45.3%±2.4	50.5%±0.4	56.6%±0.7	73.6%±0.9

Table 5.4: The performance comparison of three gradient-based descriptors in different spatial resolutions.

GBH descriptor with Fisher vector

We first evaluated the impact of the Gaussian smoothing. The detail results are shown in Table 5.3. If smoothing is "Yes", a Gaussian filter was applied on all frames before computing the gradients. All the experiments were performed on original video at full resolution. Note that for color images (in this and next experiments), we simply chose the color channel with the largest value.

For HOG and HOG3D descriptors, we observed slight performance drops on all cases when applying Gaussian filter before computing gradients. The similar performance drop was reported on HOG [19] on human detection with smoothing. The performance of GBH, on the other hand, increased significantly by pre-smoothing, with 4.2% on HMDB51 and 1.6% on UCF101. Such a performance increase may be explained by the fact that the second order derivatives are more sensitive to noise. Nevertheless, this could be a good property given that the sub-sampling with reduced spatial resolution works similar as Gaussian smoothing in a sense of losing high frequencies. Thus, the

Sampling #	Resolution	Speed (frames per second)				Accuracy
		Integral video	Sampling	FV encoding	Total fps	
4K	182 x 120	52.7	267.6	89.3	29.0	43.3%
10K	182 x 120	52.9	108.4	37.5	18.3	44.7%

Table 5.5: Average computation speed on a Toshiba Netbook with an AMD-E350 cpu and 2GB memory. The experiments are performed on HMDB51 dataset. $K = 128$ codewords per channel is used for FV encoding. 4K and 10K features are sampled in two different experiments. The dimensionality reduction process is included in FV encoding.

performance could be preserved even if the spatial resolution is reduced.

Table 5.4 shows the performance comparison of three gradient-based descriptors in different spatial resolutions. For HOG and HOG3D descriptors, the performance is consistently and significantly decreased for both HMDB51 and UCF101 when the spatial resolution is reduced. Such results are consistent with observations in [123] on Hollywood2 dataset. As resolution is reduced, the background noise gradients could confuse and blur the human gradients.

For GBH descriptor, one very important observation is that the accuracy is preserved on HMDB51 and with little (1%) loss on UCF101 when the spatial resolution is reduced by a factor $\lambda = 4$. This leads to huge benefits in efficiency considering that the subsampling in resolution by λ results in a reduction by a factor of λ^2 on both number of processed pixels and memory usage. Kläser *et al*. analyzed in HOG3D [42] that using integral video can result in a factor of $z \approx 21$ saving in memory usage than spatio-temporal "pyramids". Our method could have far less memory usage when processing video at low resolution.

When processing video at a very low resolution, we observed a relatively high standard deviation on performance for both HMDB51 (2.4) and UCF101 (0.9). This is probably due to the fact that the random sampling was performed on the very low resolution. At such low resolution, a sampled ST patch could have large differences even with a one pixel

Dataset	Descriptor	Root	Parts	Root + Parts
HMDB51	MBH	48.8%	49.9%	51.5%
	HOF	33.3%	33.5%	35.5%
	HOG	25.0%	26.4%	28.8%
	GBH	34.5%	36.2 %	38.8%
UCF101	MBH	74.6%	75.7%	77.1%
	HOF	57.8%	59.0%	61.8%
	HOG	50.1%	52.4%	54.1%
	GBH	63.2%	65.6%	68.5%

Table 5.6: Evaluation of our system on the HMDB51 and UCF101 datasets. *Root* uses only a root model with 10K patches at half spatial size of the processed video. *Parts* is a part-based system with 80K patches at full spatial size, but no root model. *Root+Parts* includes both root and part models, which are combined as two channels.

displacement in its location. At this point, it is unclear why the GBH descriptor performs better in very low spatial resolution (91 x 60) on HMDB51 than on UCF101. Our hypothesis is sampling with high density on clips with fewer frames may include denser information, which could provide bias benefits for short clips on HMDB51. Moreover, the HMDB51 has a quality standard of a minimum of 60 pixels in height for the main actor, which may improve the robustness when the spatial resolution is greatly reduced.

Since FV uses fewer quantization cells than BoF ($K = 128$ *vs.* $K = 4000$), it is much faster to perform feature encoding. To further demonstrate the efficiency of GBH descriptor, we tested its runtime on a Toshiba Netbook with an AMD-E350 CPU and 2GB memory. We processed the video at half the original video resolution. Table 5.5 lists average computation speed on an AMD-E350 CPU, which shows a high processing frame rate. This proves the high efficiency of GBH descriptor, and demonstrates its potential for real-time applications as well as mobile recognition.

5.5.2 Evaluating the effectiveness of LPM

We also investigated the effectiveness of the local part model by evaluating its different components on MHDB51 and UCF101 datasets with four descriptors. We computed the part models at the half spatial resolution of the original videos, and the root model at 1/4 resolution. The 10K patches were randomly sampled for root model and 80K patches for part models. Table 5.6 summarizes results of different models. For all descriptors, it shows that the use of both root and parts can improve the recognition accuracy, which demonstrates the effectiveness of our local part model. In addition, the part models show better performance than root model in all experiments. This is expected as there are 8 times as many part patches as root ones, and the root patches are sampled at half the resolution of the video on which we compute the part patches.

With only root model and MBH descriptor, we obtain 48.8% on HMDB51 and 74.6% on UCF101, which outperform many sophisticated methods in the literature (*c.f.* Table 5.11). Such high performance is achieved on 1/4 of the spatial resolution of the video. When computing the root model, we reused the optical flow computed from part model by down-sampling it into half. As analysed before, such an approach may reduce the loss of the deep sub-pixel accuracy, and therefore preserve the performance. Also, the very high sampling density may provide more information for classification, especially for clips with few frames (some clips from HMDB51 only have 19-25 frames).

5.5.3 Resolution influence of root model

For root model, recall that we reused the dense optical flow computed with full resolution from part models. Such an approach has the benefit in reducing the loss of the deep sub-pixel accuracy than computing the optical flow from low resolution frames directly.

To investigate the performance influence of such an approach, we performed two

Dataset	MBH		HOF	
	Before	After	Before	After
HMDB51	48.8%	46.8%	33.3%	32.3%
UCF101	74.6%	70.3%	57.8%	55.0%

Table 5.7: Comparison of performance on the optical flow computation for *root* model "Before" and "After" sub-sampling. "Before" stands for computing optical flow at full resolution and then down-sampling it for *root* model, while "After" computes the optical flow on the down-sampled frames.

Optical flow methods	HMDB51	UCF50	UCF101
Farnebäeck [23]	51.5%±0.3	87.7%±0.2	77.1%±0.4
Dual_TV_L1 [135]	56.4%±0.2	90.1%±0.3	81.5%±0.1

Table 5.8: Comparison of average accuracy with MBH descriptor built on two different optical flow algorithms, Farnebäck and Dual_TV_L1.

groups of experiments. In the first one, we computed optical flow from the full resolution video. Then, we down-sampled the optical flow images into half size, and used them to compute MBH and HOF descriptors. We refer to this approach as "Before". In the second "After" experiments, we computed the optical flow on the down-sampled frames. We set all other parameters same, and computed the performance with only *root* model.

In comparison to "After", when computing optical flow from the full spatial resolution, we achieved 2% and 4.3% improvements in mean accuracy on HMDB51 and UCF101 datasets for MBH descriptor, respectively. For HOF descriptor, the performance also improved on both cases. The detail comparison is shown in Table 5.7. Such results are consistent with observation on optical flow influence in [20], which has reported that even 1/10 of a pixel of flow values' rounding causes the performance to drop significantly.

5.5.4 Influence of different optical flow algorithms

Motion analysis algorithms have been developed for decades, but the state-of-the-art optical flow methods do not produce the expected results for many real-world video sequences [56]. In last section, we have shown that the accuracy of optical flow has large impact on the recognition performance. As discussed on Section 5.2.2, there may be potential benefits of using state-of-the-art optical flow algorithms. In this section, we will evaluate performance impact when using an efficient Farnebäeck optical flow algorithm and using a discontinuity-preserving optical flow method, Dual_TV_L1 from Zach *et al*. [135].

Table 5.8 shows the performance comparison on MBH descriptor built on two different optical flow methods, Farnebäeck and Dual_TV_L1. For both methods, we use the OpenCV implementation with default parameters for Dual_TV_L1 and optimized parameter for Farnebäeck. On all three datasets, the results of the MBH descriptor from Dual_TV_L1 achieve significantly better performance. One possible explanation for such improvement is that the Dual_TV_L1 estimates more accurate flow field with both over human motions and at motion discontinuities.

For HOF descriptor, one very interesting observation is that the performance drops by around 1-2% when using Dual_TV_L1 method. Brkić *et al*. [13] also observed slight performance penalty with Dual_TV_L1 than with Farnebäeck method when using HOF-like descriptor. One possible reason is that Farnebäeck estimates more smooth flow field, whereas Dual_TV_L1 and LDOF produce more sub-pixel accuracy with noisy global motions. Another possible explanation is that Dual_TV_L1 and LDOF may also estimate more accurate camera motion, which offsets the benefit of the more accurate human motions. Compared with MBH, the HOF has no mechanism to reduce the camera motion. The results in dense trajectories [121] show higher (around 4%) performance

penalty for HOF when using LDOF. One possible explanation is that the trajectory tracking in dense optical flow field may also be affected by noisy global motions. The MBH descriptor in their case is also impacted by trajectory tracking from noisy global motions. Unlike dense trajectories, we don't track points over dense optical flow field, and the optical flow estimation can only affect the MBH and HOF descriptors alone. Therefore, the MBH descriptor in our system can benefit the improvement over dense optical flow estimation more directly.

Note that the Dual_TV_L1 is computationally expensive. However, Zach *et al*. reported a real-time GPU implementation in [135].

5.5.5 Evaluation of the improved LPM

As discussed in Chapter 5.3, our previous approach in concatenating the histograms of root and parts into a single vector may result in suboptimal performance due to the large codeword quantization errors. Therefore, we proposed an improved LPM by treating the root and 8 parts as two separate channels. To validate this hypothesis, we performed experiments with 4 descriptors on three datasets. For all experiments, we fixed all parameters except to treat root and parts as single channel or separate channels. In addition, for single channel approach, we used 4K codewords for HOG, HOF, GBH descriptors, and 8K codewords for MBH descriptor. For the improved LPM with separated channels, we used 4K codewords for HOG, HOF, GBH, MBHx and MBHy descriptors.

Figure 5.4 illustrates the performance comparison of our previous single channel LPM and the improved LPM with separated channels. In the figure, "1C" stands for single channel approach, while "2C" represents the separate channel approach. On all three datasets, we observe consistent recognition performance improvement over all descriptors

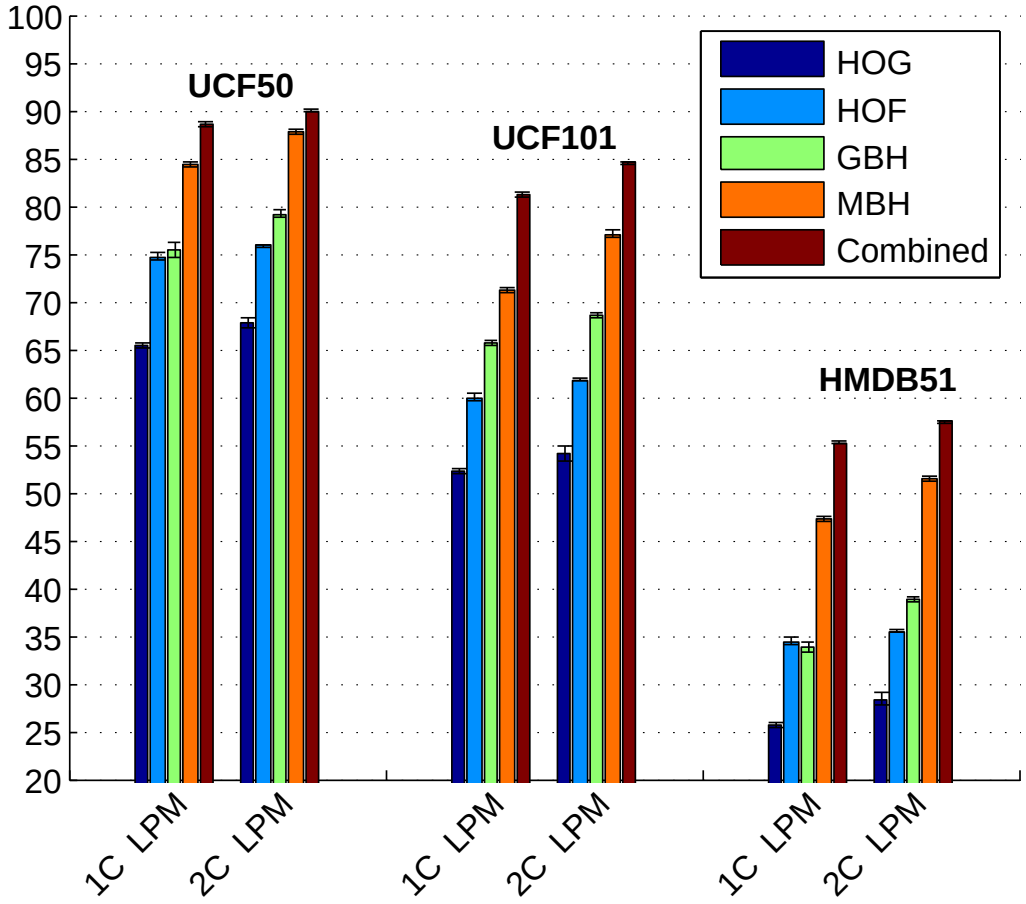


Figure 5.4: Performance comparison of single channel LPM and separated channel LPM on different datasets. The average accuracy in percentage (with the standard deviation denoted by error bar) is plotted against different descriptors.

with separate channel LPM. The most significant gains are obtained with MBH descriptor, namely, 3.3% on UCF50, 5.9% on UCF101 and 4.2% on HMDB51. The similar improvement is also observed on GBH descriptor, with an increase of 3.8% on UCF50, 2.9% on UCF101 and 5.8% on HMDB51.

Dataset	Method	Descriptor				
		HOG	HOF	HOG3D	MBH	Combined
HMDB51	OVO	24.8%	33.5%	34.7%	43.0%	47.8%
	OVA	25.7%	34.4%	36.3%	47.2%	54.8%
UCF101	OVO	51.1%	58.7%	59.1%	66.9%	73.8%
	OVA	52.2%	60.0%	62.9%	71.2%	77.1%
UCF50	OVO	58.6%	69.7%	72.4%	80.1%	83.3%
	OVA	62.6%	74.7%	76.1%	84.4%	88.5%

Table 5.9: Comparison of different multi-class SVM approaches, one-versus-one(OVO) *vs*. one-versus-all(OVA).

5.5.6 Multi-class SVM: one-versus-one *vs.* one-versus-all

Support Vector Machines (SVM) is originally designed for binary classification. It can be extended to multi-class classification by decomposing the multi-class problem into a number of two-class problems, and applying a standard SVM for each of them. There are two popular decomposition methods: one-versus-one(OVO) and one-versus-all(OVA).

The **one-versus-one**. For n classes, OVO constructs $n(n-1)/2$ SVM classifiers. The SVM models are learnt with data from any two of the all classes. After all models are trained, the prediction is implemented by max-wins voting, with class label assigned to the class with the largest vote from the models.

The **one-versus-all**. For n classes, OVA constructs n SVM models. The ith SVM is trained with all of the examples from the ith class as positive labels, and remaining examples from other classes as negative labels.

The one-versus-one strategy is often substantially faster than one-versus-all method. In the literature, there are disagreements on which multi-class SVM performs better. [5, 18, 33] reported better recognition performance with OVO, while others have shown different results [69, 86]. For action recognition on large scale datasets, Jain *et al*. [35] reported significantly better performance with OVA.

Dataset	Descriptor	Bag-of-features	Fisher vector	
			$K = 128$	$K = 256$
HMDB51	MBH	56.4%	59.4%	60.7%
	GBH	38.8%	43.9%	44.5%
UCF101	MBH	81.5%	84.7%	85.4%
	GBH	68.5%	74.8%	75.7%

Table 5.10: Comparison of performance on feature encoding with bag-of-features and Fisher vector.

A number of experiments have been performed to evaluate the OVA and OVO. We used same parameters for OVA and OVO on all experiments. The single channel SVM approach was used, with 4K codewords for HOG, HOF, HOG3D descriptors, and 8K codewords for MBH descriptor.

The detail performance comparison of one-versus-one strategy and one-versus-all strategy is shown in Table 5.9. For all three datasets, our experiments show significantly better recognition accuracies when using OVA on all descriptors. Such results are consistent with the recent report [35] on action recognition with large scale datasets.

5.5.7 BoF *vs.* FV

Table 5.10 presents the performance comparison of feature encoding with BoF and FV. The Fisher vector consistently shows better performance than bag-of-features approach on both MBH and GBH descriptors with either of the tested datasets. For FV with $K = 256$, we observed around 4% performance improvement for MBH descriptor and $6-7\%$ for GBH descriptor. Such results are consistent with recent reports [106, 77, 121] on action recognition. In all their respective results, they have shown better results by applying feature encoding (*e.g.* Wang and Schmid [122] achieved improvement of 4% on UCF50 and 5.1% on HMDB51 with FV over BoF). For FV, using more quantization cells also results in better performance in all tests.

Discussion: the main challenge in using FV is the resulting high dimensionality, which is expensive in classification stage even by using linear SVM. For a D dimension descriptor ($D = 32 + 64$ for GBH after applying PCA), the FV signature with K codewords ($K = 256$ and $K = 128$ in our experiments) has an increased dimension of $2KD$. In comparison with BoF (often with $K = 4000$ codewords), the FV representation is not much as a "Compact Feature Set" as the claim made in [77]. The original FV approach [36], however, reduced the high dimensional FV into a compact low dimensional vector and observed improved performance on image search. We did use PCA to reduce the FV from $d = 24576$ (GBH) or $d = 49152$ (MBH) to $d = 3570$, and observed identical classification results on HMDB51, with a classification speed-up of approximately an order of magnitude. Note that there are only 3570 training samples from HMDB51, and the largest dimension after applying PCA (with retained Variance = 1) is 3570. For a comprehensive study on the effect of PCA on action recognition, more training data are desirable.

5.5.8 Comparison to state-of-the-art

Table 5.11 shows the comparison of our method with the state-of-the-art. Most state-of-the-art methods use multiple descriptors and apply some feature encoding algorithms to improve the performance. For example, Jain *et al*. [35] combined five compensated descriptors and applied VLAD representation. Wang and Schmid [122] used four descriptors and Fisher Vector encoding. They also improved the performance with human detection and extensive camera motion compensation.

We used the parameters listed in Section 5.4.4. For all descriptors, we used 4000 codewords (BoF) and $K = 128$ (FV) for both root and part channels. For MBH descriptor, we used Dual_TV_L1 [135] method to compute optical flow. For efficiency, we used the

Method			HMDB51	UCF101	UCF50
HMDB51 [45]			23.2%	–	47.9%
ActionBank [91]			26.9%	–	57.9%
MIP [43]			29.17%	–	72.68%
Subvolume [93]			31.53%	–	–
GIST3D [104]			29.2%*	–	73.7%*
UCF50 [85]			27.02%*	–	76.90%*
UCF101 [105]			–	43.9%	–
MRP [38]			40.7%*	–	–
DCS [35]			52.1%*	–	–
Actons [137]			54.0%*	–	–
FV coding[77]			54.8%*	–	90.0%*
Trajectories [122]			57.2%*	85.9%*	91.2%*
Ours	BoF	HOG	28.4%±0.6	54.1%±0.8	67.8%±0.5
		HOF	35.5%±0.2	61.8%±0.2	75.9%±0.1
		GBH	38.8%±0.3	68.5%±0.2	79.2%±0.4
		MBH	56.4%±0.2	81.5%±0.1	90.1%±0.3
		Combined	60.6%±0.1*	84.5%±0.1*	91.5%±0.2*
	FV	GBH	43.9%±0.3	74.8%±0.4	83.4±0.3
		MBH	59.4%±0.1	84.7%±0.1	90.7±0.2
		GBH+MBH	**62.2**%±0.1*	**86.6**%±0.2*	**92.1**±0.1*

Table 5.11: Comparison of average accuracy on HMDB51, UCF101 and UCF50 with state-of-the-art methods in the literature. Those marked with * are results with combined descriptors. Leave One Group Out Cross-validation is used for UCF50.

original datasets' video resolution to compute the HOG descriptor, and down-sampled them into half the resolution to compute other descriptors. For bag-of-feature approach, we combined all 4 descriptors (as shown in Table 4.5) with the weighted histogram intersection kernel (*c.f.* Eq. 5.4), which showed about 0.5% improvement in performance than with no weights. For Fisher vector, we simply concatenated MBH and GBH feature vectors and use a linear SVM for classification with the fixed parameter of $C = 32.5$. The one-versus-all strategy was used for multi-class SVM.

On **HMDB51**, our method achieves 59.4% on single MBH descriptor (FV) and 62.2% on two combined descriptors, which outperforms the state-of-the-art result (57.2% [122]) by 5%. With MBH computed from efficient Farnebäeck optical flow method, we obtain 57.4% (BoF) on 4 descriptors, which also exceeds state-of-the-art results. Note that our results are obtained from two descriptors with Fisher vector. The $K = 128$ is used in the experiments. If using $K = 256$, we achieve 63.2%.

On **UCF50**, we achieve 91.5% with four descriptors when using bag-of-features approach. By using Fisher vector, we obtain 92.1% with only two descriptors. Both of these results surpass the state-of-the-art (91.2% [122]). On **UCF101**, an extension of the UCF50, we report 84.7% with MBH descriptor and 86.6% with two descriptors (FV encoding). It is slightly outperformed the state-of-the-art result [122] (85.9%), which is obtained with four descriptors and Fisher Vector encoding as well as extensive camera motion estimation. Our method is based on pure random sampling with no extra cost on feature detection and motion compensation. We expect better performance with advanced motion compensation.

The confusion matrices for three tested datasets are illustrated in Figure 5.5, Figure 5.6 and Figure 5.7.

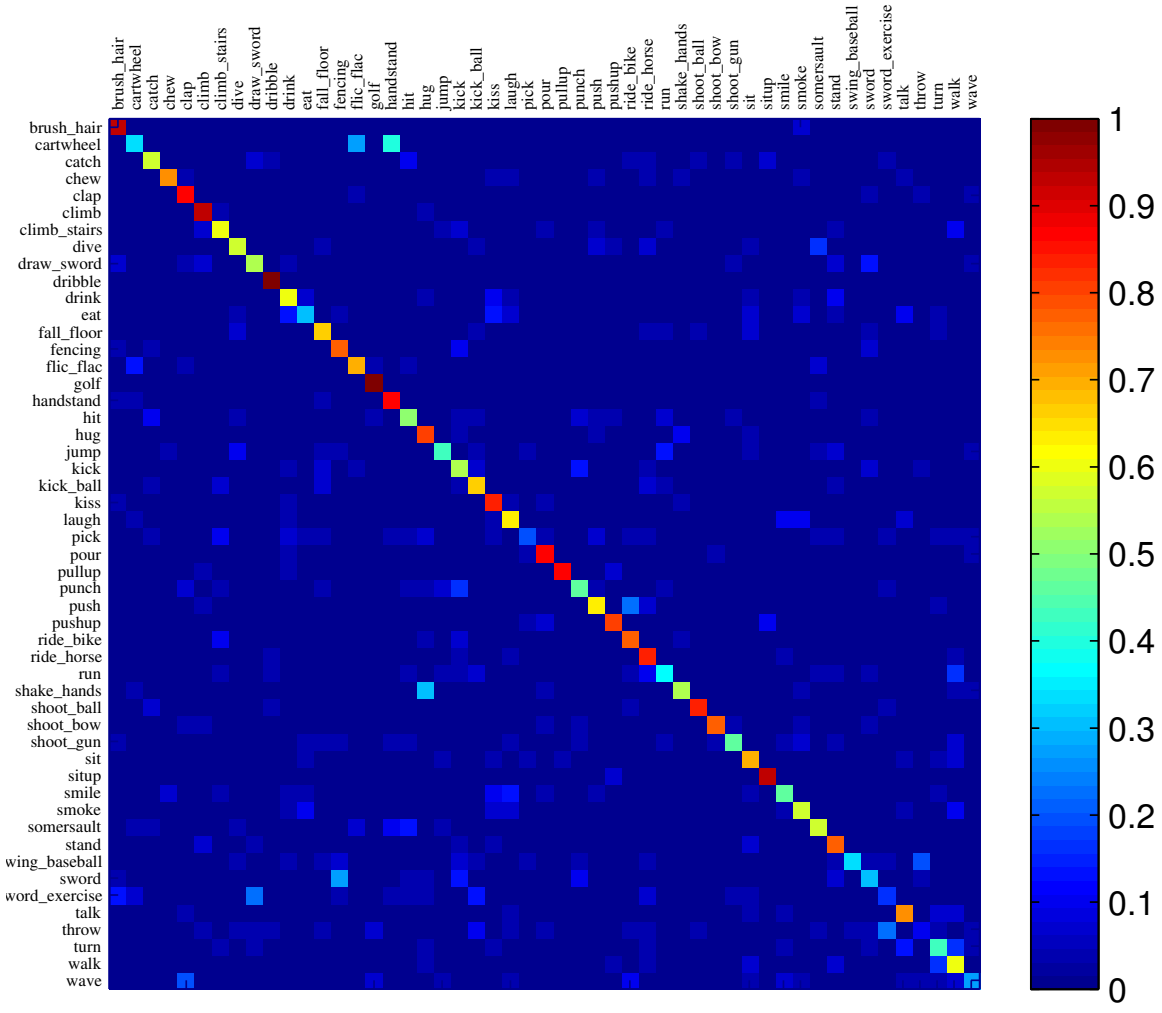


Figure 5.5: Confusion matrix for HMDB51 dataset obtained with 4 descriptors. The confusion matrix is based on the results reported in Table 4.5.

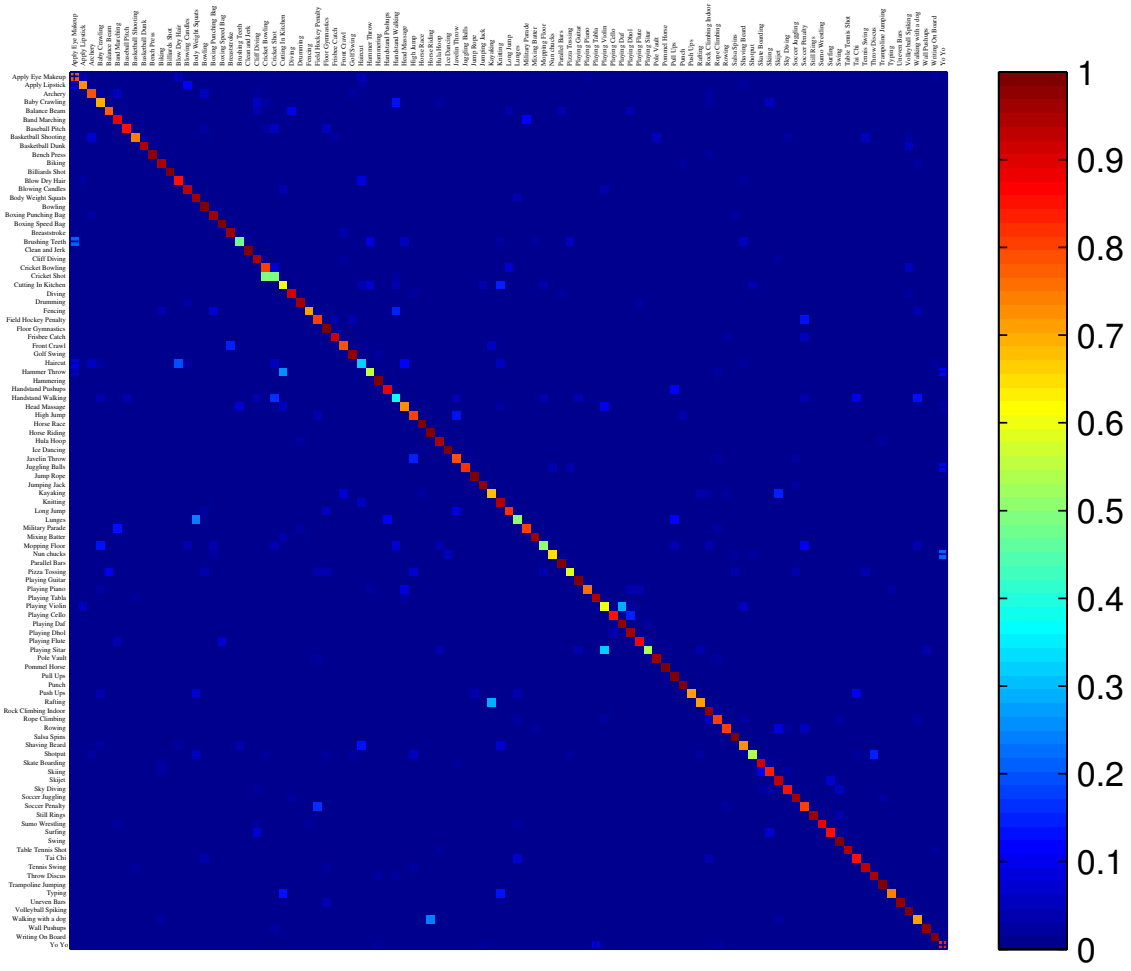


Figure 5.6: Confusion matrix for UCF101 dataset obtained with 4 descriptors. The confusion matrix is based on the results reported in Table 4.5.

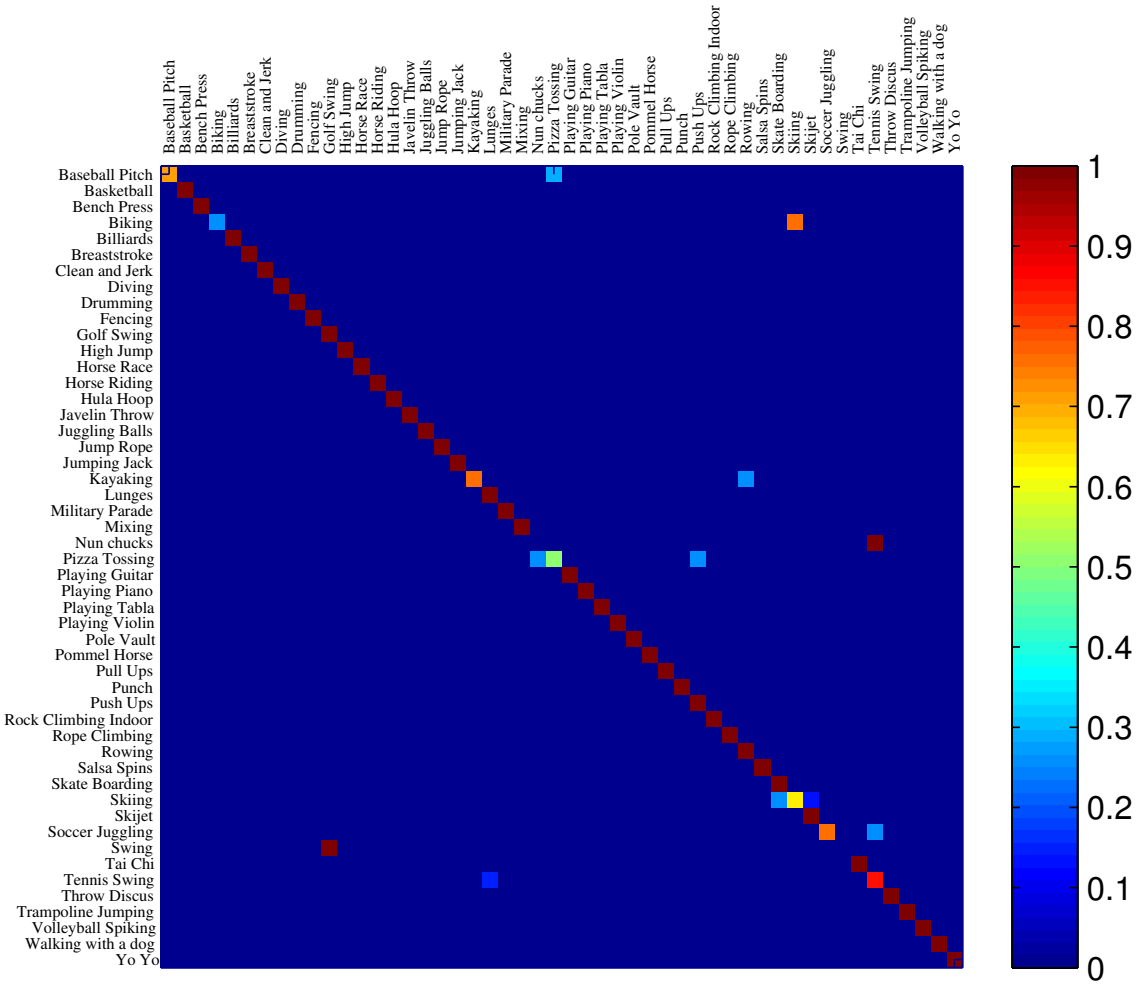


Figure 5.7: Confusion matrix for UCF50 dataset obtained with 4 descriptors. The confusion matrix is based on the results reported in Table 4.5.

5.6 Discussion

In comparison to other methods, the LPM works better on more challenging HMDB51 than UCF101 and UCF50. One possible factor is the use of high sampling density with random sampling. The HMDB51 has an average of 94.8 frames per clip with many clips at the range of 19-30 frames, whereas the UCF101 has 186.5 frames per clip on average. Thus, the other methods may have difficulty in generating sufficient number of features for short clips on HMDB51. As discussed in Chapter 4.4, this could be an advantage of using random sampling instead of dense sampling. Nevertheless, processing videos at low resolution may provide bias benefits for short clips on HMDB51 as sampling with high density on clips with fewer frames could generate more information.

Our method demonstrates very good performance on large scale challenging datasets with more realistic scenarios. One possible explanation for such good performance may reside in our random sampling conducted on an extremely dense sampling grid. For 90K (10K root + 80K parts) patches per video on HMDB51, we have around 900 patches per frame, which are more than those in [121]. However, our sampling density is much higher because the sampling is performed on one quarter size of that in [121]. Compared with interest point detectors, we have more patches sampled from the test videos, and with uniform random sampling our method also includes correlated background information. Such background information may improve discriminative power for recognition on real-life videos.

5.7 Conclusions

A new local 3D descriptor based on spatial-temporal gradient was proposed on the purpose of capturing both local static appearance and motion information. It significantly outperformed popular HOG descriptor with similar high computational efficiency. We also explored the potential benefits on using state-of-the-art optical flow algorithms to improve the performance with MBH descriptor.

We further improved the performance of Local Part Model through the use of multiple channels. For multi-class SVM, we experimentally showed that one-verse-all strategy outperformed one-verse-one strategy on human action recognition with large scale realistic datasets. A new method based on histogram intersection kernel was proposed to combine multiple channels of different descriptors. Our system outperformed state-of-the-art results on three large challenging realistic datasets, namely, HMDB51, UCF50 and UCF101.

Chapter 6

Fast Action Recognition

6.1 Introduction

We have proposed a Local Part Model, and our method achieved state-of-the-art performance on realistic large scale datasets when combined with random sampling method applied on a very dense sampling grid. Under the local part model, a feature consists of a coarse global *root* patch and several finer overlapped *part* patches. To improve the efficiency of LPM computation, two integral videos are computed, one for the *root* model at half resolution, and another one for the part models at full resolution. The descriptor of a 3D patch can then be computed very efficiently through 7 additions/subtractions multiplied by the total number of root and parts. Apart from descriptor quantization, most cost associated with feature extraction is spent on accessing memory through the integral videos.

Because it uses random sampling, the method does not require feature detection, which greatly improves processing speed. One LPM feature includes a *root* patch and a group of *part* patches. The histograms from all patches are concatenated into one vector that is 9 (1 *root* + 8 *part*s) times the original feature dimension. In last chapter, we

substantially improved this approach by treating the root and 8 parts as two separate channels. Nevertheless, the concatenated vector from part channel is 8 times the original feature dimension. Such high feature dimensionality results in a high computational cost for bag-of-features matching, which is the most computationally expensive component in our method.

In this chapter, we aim to improve the efficiency of our human action recognition system, mainly by reducing the computational cost in bag-of-features matching. To this end, we want to investigate two strategies:

- Reducing the high feature dimensionality with Principal Component Analysis (PCA). The description vectors are compared using the Euclidean distance. Considering we need to match 10K features with 4K/8K codewords for each clip, the reduction in each feature's dimensionality could result in substantial benefits in computation efficiency.

- Applying fast approximate nearest neighbour search (FLANN). Compared with linear, brute-force search, FLANN can provide large speed-ups at the cost of the method not always returning the exact match.

6.2 PCA-based LPM

Principal Component Analysis (PCA) is a standard tool for dimensionality reduction and has been used to solve various computer vision problems, such as feature descriptor [41], object classification [55], face recognition [17], image representation [36], and human detection [19, 96]. Our PCA dimensionality reduction is similar to PCA-SIFT [41]. In their work, Ke and Sukthankar applied PCA on keypoint image patches, and found that it improved the feature matching speed and outperformed SIFT for keypoint based

matching. Note, however, that a different observation on performance was reported in [68].

For Local Part Model, we randomly sample 90K 3D patches (10K "root" patches and 80K "part" patches) from every clip with up to 160 frames. The root and parts are either concatenated into one vector or treated as two separate channels, and each patch is represented by a local descriptor. However, the concatenated feature vector from a group of 8 part patches has 8 times the dimension of the used descriptor. Using more feature points generally leads to better recognition performance [120, 123]. LPM controls the complexity of a very high sampling density by representing a group of sampled part patches with a single high dimension vector. Because the root patch and its 8 part patches are sampled from overlapping volumes, they tend to produce a certain amount of redundant information. LPM is therefore a good candidate for feature reduction.

Let $d = D_0$ be the feature dimension of the descriptor we use. For the approach with separate channels, one LPM feature has two channels, with 1 root patch at dimension of $d_r = d = D_0$ and 8 part patches at dimension of $d_p = 8 \times d = 8 \times D_0$. We empirically found that, without noticeable performance penalty, the root dimension can be reduced by half and the part dimension can be reduced to 1/8 at the same time. After applying PCA, the root channel dimension is $d'_r = \frac{1}{2}D_0$, and part channel dimension is $d'_p = D_0$, which is equal to the dimension of an original descriptor. In comparison with other methods [41, 36, 121], which reduce the feature dimension by half, our approach uses a much higher reduction rate (1/8) for the part model.

There are some advantages of using PCA. On the efficiency side, reducing the dimensionality of the feature has obvious benefit. In a bag-of-features approach, the description vectors are compared using the L_2 distance. Considering we need to match 10K features with 4K words for each clip, the reduction in dimensionality to 1/8 results in significant

benefits in computational efficiency. On the performance side, there may be positive impact and practical meaning when applying robust feature encoding strategies, such as Fisher vector(FV) encoding [36] or VLAD encoding [36], a simplified version of the Fisher vector. In recent evaluations, both FV [106, 77, 121] and VLAD [106, 35] showed improved performance over standard bag-of-features methods. Fisher vector extends the BoF by encoding high-order statistics (first and, optionally, second order) between the descriptors and a Gaussian Mixture Model (GMM). Due to the fact that decorrelated data can be fitted more accurately by a GMM with diagonal covariance matrices, it is favourable to apply PCA dimensionality reduction on Fisher vector. In addition, for a D dimension descriptor, the FV signature with K words has an increased dimension of $2DK$ (VLAD has a dimension of KD). Therefore, reducing the feature dimension makes it more appealing to apply Fisher vector or VLAD on local part model.

6.3 Bag-of-features matching

For bag-of-features approach, a standard step is vector quantization which matches extracted features to their nearest visual words, often by Euclidean distance. It can be defined as follow: given a set of codewords $W = \{w_1, w_2, ..., w_n\}$ in a feature space X, for a new query feature $q \in X$, find the codeword in W that has minimal Euclidean distance to q. Thus, the most computationally expensive component of our approach consists of searching for the closest codeword given a high-dimension query feature.

Brute-force matching produces least quantization error with linear complexity. Given the high feature dimension and large databases, the brute-force search is too costly. In our case, the concatenated feature vector contributes to increased dimensionality of 8 times the original descriptors. The BF matching with such high dimension features is the most time consuming part in our system, and the real-time performance is only

achieved through the recourse to multi-core CPU and GPU processing (as shown in Chapter 4).

PCA feature reduction can reduce computational cost of brute-force matching by reducing the feature dimensionality. As analysed in Section 6.2, the dimensions of root descriptor and part descriptor are reduced into 1/2 and 1/8, respectively. Such strategies could result in 2 and 8 times speed-ups in feature matching process for the respective root and part models, plus extra costs in PCA projection.

To apply PCA projections, we compute the covariance matrix based on all 4000/2000 visual words computed from k-means. Then, the eigenvectors associated with the most energetic eigenvalues from the covariance matrix are stored on memory and used as the projection matrix M for dimensionality reduction. To use PCA for bag-of-features approach, we first apply PCA with the projection matrix M on 4000/2000 visual words (performed on each channel separately) to reduce root vector from 64 to 32 and part vector from 512 to 64. The vectors of both root and part channels from sampled ST patches are also reduced to 32 and 64, respectively, and then matched (brute-force) to the codewords with reduced dimensionality.

Fast approximate nearest neighbour search is a very important technique to handle large databases in computer vision applications. Such approximate algorithms can be orders of magnitude faster than linear, brute-force matching search. However, as an efficient approximate search method, it often compromises performance due to the approximation error.

The classical kd-tree algorithm [25] is the most widely used algorithm for nearest neighbour search. It divides the data in half at each level of the tree on the dimension for which the data shows the largest variance. It is efficient in low dimensions. Silpa-Anan and Hartley [102] improved the kd-tree with randomized kd-trees by choosing

Dataset	# of words	Descriptor				
		HOG	HOF	GBH	MBH	Combined
HMDB51	2K words	26.4%	35.3%	37.0%	50.0%	56.0%
	4K words	28.4%	35.5%	38.8%	51.5%	57.4%
UCF101	2K words	53.9%	59.6%	67.4%	76.3%	78.9%
	4K words	54.1%	61.8%	68.5%	77.1%	81.2%
UCF50	2K words	67.9%	74.1%	77.8%	85.9%	88.8%
	4K words	67.8%	75.9%	79.2%	87.7%	90.0%

Table 6.1: Performance comparison of 2K and 4K visual words with different descriptors.

the divided dimension randomly from the first d dimensions with the greatest variance. Nister and Stewenius [74] used hierarchical k-means tree by splitting the points at each level into K distinct areas with k-means clustering.

6.4 Experimental results

To demonstrate the performance and efficiency improvement of various BoF matching methods, we evaluated our method on the same three large-scale action benchmarks as last chapter. To ensure that our results are comparable, we used the same experimental settings as those in Section 5.4 of last chapter, and restricted our changes to the bag-of-features matching step.

For approximate nearest neighbour search, we used OpenCV implementation of Muja and Lowe's FLANN [71]. For all experiments, the runtime was estimated on an Intel i7-3770K PC with prototype implemented in C++. In order to avoid built-in multi-core processing of OpenCV library, we set only one core active @ 3.5Ghz in Bios, and disabled both Hyper-threading and Turbo-boost. Thus, we only used a single core of the CPU.

6.4.1 Influence of the codewords

We first evaluated the impacts on performance and efficiency when using different number of codewords. Table 6.1 illustrates performance comparison of 2K and 4K visual words with different descriptors. For all three datasets, we observed performance gains on all descriptors when using 4K codewords instead of 2K codewords. Such results are expected because more codewords yield less error from bag-of-words vector quantization.

Table 6.2 shows average computation speed for different BoF matching methods in frames per second. The MBH descriptor is used with 2K and 4K words per channel, and 10K features are sampled in all experiments. For PCA32-64, the computation cost for PCA feature reduction is included. The results show that brute-force matching with 4K words takes more than twice as much time as that with 2K words. Applying feature reduction with PCA (noted as PCA32-64 in the table) on brute force matching results in over 2× speed-ups for 2K words and 3× speed-ups for 4K words. The FLANN can speed up matching process with up to 18×, with most efficiency gains on 4K words. One important observation for FLANN is that using more words has little additional cost. Thus, it is favourable to have more codewords when using FLANN due to the performance advantages of more codewords. Note that there is performance penalty for FLANN method (*c.f.* Section 6.4.2).

Using more codewords normally improves the recognition accuracy at a cost of increasing computational complexity for bag-of-words matching. Therefore, depending on the applications, it is advisable to choose appropriate number of codewords for a good compromise between speed and accuracy.

Dataset	# of words	BoF matching speed (fps)		
		Brute-force	PCA32-64	FLANN
HMDB51	2K words	15.3	34.5	130.1
	4K words	7.1	21.6	123.0
UCF101	2K words	22.1	59.3	192.8
	4K words	10.7	34.2	185.5

Table 6.2: Average computation speed with single core for different BoF matching methods in frames per second. The MBH descriptor is used with 2K and 4K words per channel, and 10K features are sampled in all experiments. For PCA32-64, the computation for PCA feature reduction is included.

6.4.2 Evaluation of BoF matching methods

Fig. 6.1 and Fig. 6.2 illustrate the performance comparison of different BoF matching methods on HMDB51 and UCF101 dataset with 2K and 4K visual words. First, in all cases, there is a clear performance benefit in using 4000 visual words than 2000 visual words. Also, in comparison to 4K words, there is more performance penalty for 2K words in using FLANN or PCA instead of brute-force. One possible reason is vector quantization with fewer words is more prone to quantization errors from approximation, which may result in the similar ST patch being assigned to different visual words in different clips. Such penalty is more evidential on HMDB51 dataset, which is more challenge and therefore more sensitive to quantization errors.

For 4000 visual words, applying PCA to reduce dimensionality shows no performance penalty except for $0.5 - 1.5\%$ for HOF. As discussed in Section 6.2, this is a clear advantage considering we reduce the high dimension part channel into its $1/8$. We will evaluate the computational efficiency of different BoF matching methods in Section 6.4.3.

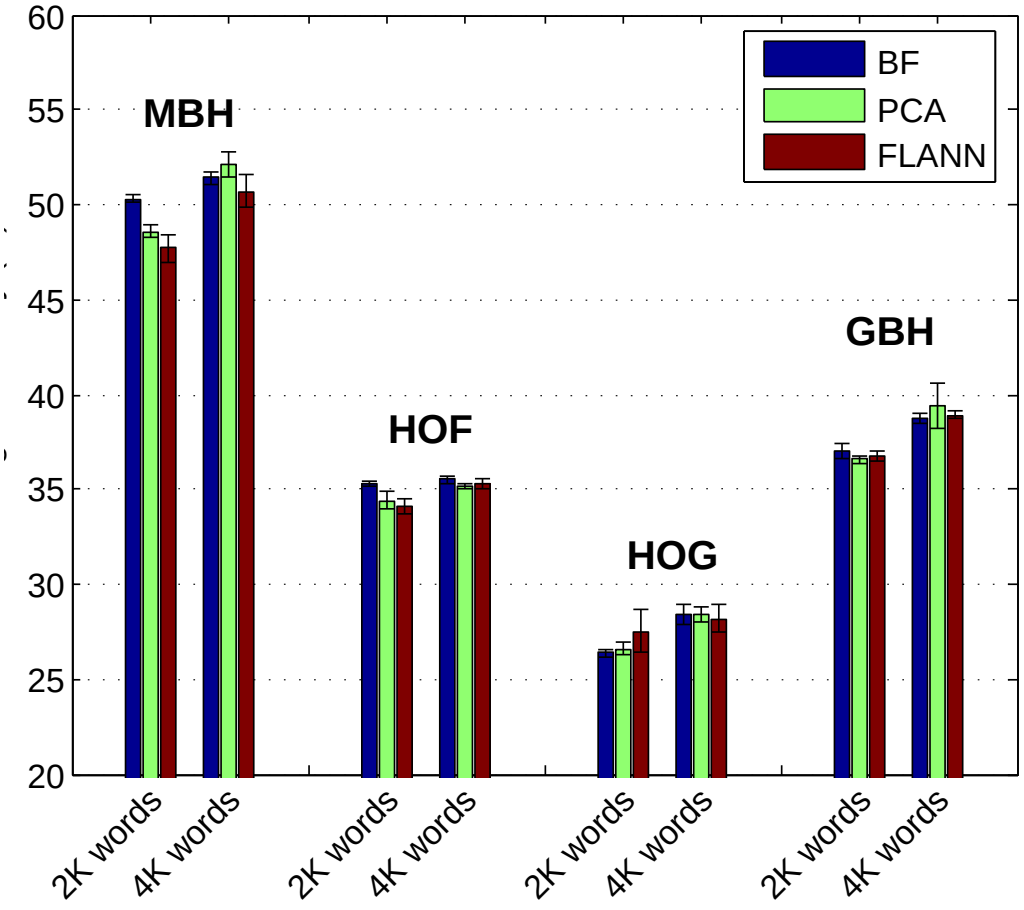


Figure 6.1: Performance comparison of different BoF matching methods on HMDB51 dataset with 2K and 4K visual words. The average accuracy in percentage (with the standard deviation denoted by error bars) is plotted against different descriptors.

6.4.3 Computational efficiency

We also evaluated the computational complexity on all four descriptors. Brute-force, FLANN and PCA methods were evaluated and compared. Except for codewords matching, all other stages were same. For all experiments, 4K words were used. We used the default parameters as in Section 5.4.4, and randomly chose 10K features (10K root patches + 80K part patches) from each clip with up to 160 frames. The optical flow for MBH and HOF was computed with Farnebäeck algorithm. The runtime for HOG was

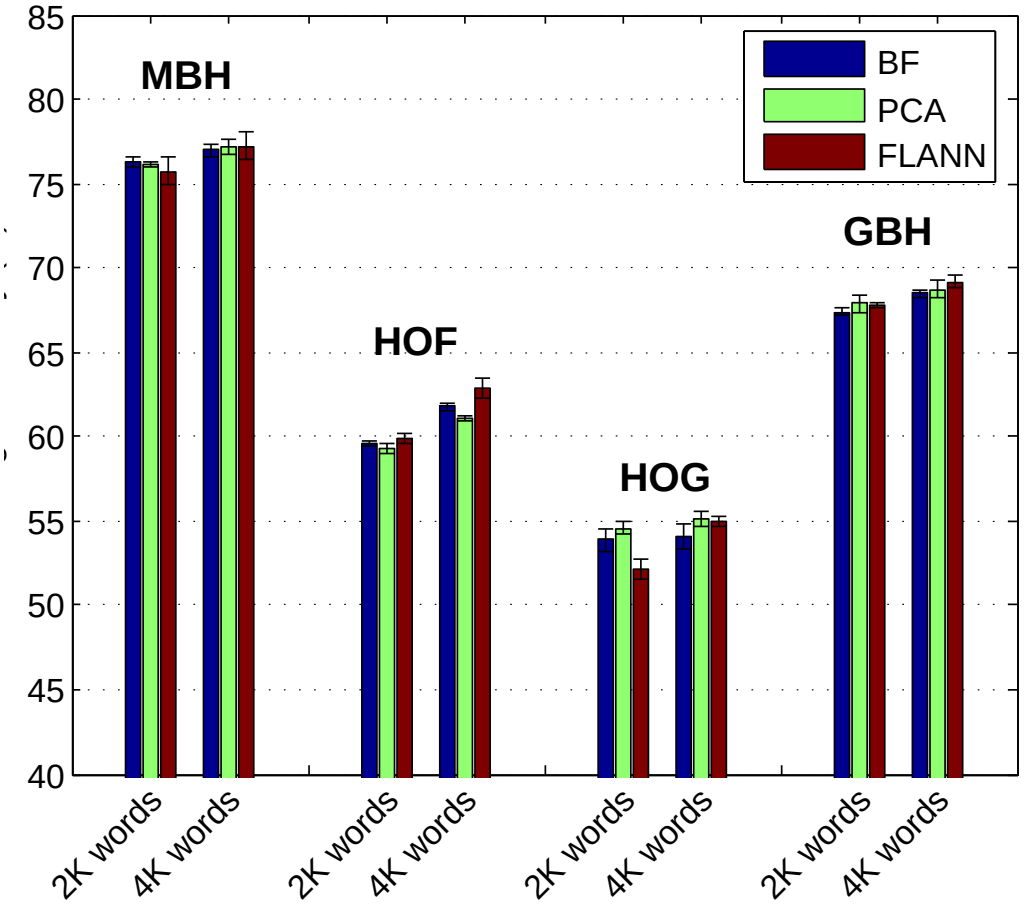


Figure 6.2: Performance comparison of different BoF matching methods on UCF101 dataset with 2K and 4K visual words. The average accuracy in percentage (with the standard deviation denoted by error bars) is plotted against different descriptors.

estimated with the original datasets' video resolution, and for other descriptors with half resolution.

Table 6.3 summarizes the efficiency comparison at different stages for HMDB51 and UCF101 datasets when using different descriptors. Applying feature reduction with PCA on Brute Force matching results in over 3× speed-ups in "BoF matching" and 2× speed-ups in total system process. The FLANN can speed up matching process by 18×. The detail results are listed in Table 4.6. There are small speed differences between HMDB51

and UCF101. One explanation is that HMDB51 videos have variable resolution. Also, it has fewer frames per video, and we sampled more features per frames on HMDB51 at 10K features per video.

In general, considering random sampling variation, there is no significant performance difference among three matching methods. The PCA matching performs similar as Brute-force. This is a very good result due to the fact that we reduced the dimension of part channel to its 1/8. The largest penalty for FLANN is 0.8% on HMDB51 for MBH descriptor. Our previous work [101] has shown larger performance drop for FLANN. The possible explanation is that using separate root and part channel makes it more robust to approximation errors.

Compared with existing methods, a major strength of our method resides in its very high computational efficiency. We achieve state-of-the-art classification results by analyzing the videos at half the resolution (except for HOG descriptor). Also, by randomly sampling ST patches, we are able to use integral video to accelerate the processing. In case of [122], the use of curve trajectories limits it to use integral image only. It also employs very expensive camera motion compensation method and state-of-the-art human detection algorithm to improve the performance.

Dataset	Descriptor	BoF matching method	Speed (frames per second)				Mean accuracy
			Integral video	Sampling	BoF matching	Total fps	
HMDB51	MBH	FLANN	53.1	131.5	123.0	30.51	$50.7\% \pm 0.83$
		PCA32-64			21.6	13.8	$52.1\% \pm 0.65$
		Brute Force			7.1	6.0	$51.5\% \pm 0.33$
	HOF	FLANN	71.5	265.6	245.0	47.1	$35.3\% \pm 0.26$
		PCA32-64			44.3	25.3	$35.2\% \pm 0.16$
		Brute Force			14.7	11.6	$35.5\% \pm 0.23$
	HOG	FLANN	101.1	305.4	251.1	59.2	$28.2\% \pm 0.75$
		PCA32-64			43.8	28.0	$28.4\% \pm 0.39$
		Brute Force			14.2	11.9	$28.4\% \pm 0.56$
	GBH	FLANN	217.1	325.3	267.2	89.9	$38.9\% \pm 0.18$
		PCA32-64			44.5	33.3	$39.4\% \pm 1.22$
		Brute Force			14.9	13.4	$38.8\% \pm 0.26$
UCF101	MBH	FLANN	60.8	198.8	185.5	37.0	$77.2\% \pm 0.83$
		PCA32-64			34.2	19.9	$77.2\% \pm 0.43$
		Brute Force			10.7	8.7	$77.1\% \pm 0.38$
	HOF	FLANN	75.5	398.0	392.2	58.0	$62.9\% \pm 0.59$
		PCA32-64			69.3	34.2	$61.1\% \pm 0.19$
		Brute Force			21.0	15.7	$61.8\% \pm 0.21$
	HOG	FLANN	111.3	463.6	384.4	74.7	$55.0\% \pm 0.32$
		PCA32-64			68.9	39.8	$55.1\% \pm 0.43$
		Brute Force			21.1	17.0	$54.1\% \pm 0.80$
	GBH	FLANN	248.8	483.7	410.4	124.3	$69.2\% \pm 0.38$
		PCA32-64			67.9	48.5	$68.7\% \pm 0.51$
		Brute Force			21.0	18.5	$68.5\% \pm 0.22$

Table 6.3: Average computation speed with single core at different stages in frames per second. The 4K words per channel are used, and 10K features are sampled in the experiments. The optical flow computation is included in "Integral video". We use full spatial size video for HOG descriptor, and half spatial size for all other descriptors. Note that the classification stage is not included.

6.5 Summary

In this chapter, we focused on reducing the computational cost of the most computationally expensive component of our system, namely bag-of-words matching. More specifically, we applied fast approximate nearest neighbour search method and PCA feature reduction technique to improve the efficiency of our approaches on action recognition.

The brute-force matching gives least quantization error with increased computational cost given the high feature dimension and large databases. To improve the computational efficiency of brute-force matching, the high feature dimension can be reduced by applying PCA dimensionality reduction. Fast approximate nearest neighbour search (FLANN) method is a very important technique to handle large databases in computer vision applications. However, as an efficient approximate search method, it often compromises performance due to the approximation error.

The evaluation shows that the feature dimensions can be reduced by 7/8 through PCA while preserving high accuracy and speeding up matching process with up to 3×. Applying fast approximate nearest neighbour search can result in as much as 18× speed-ups at the cost of less than 1% performance loss for large codewords.

Chapter 7

Conclusions and Future Work

This thesis describes a fast action recognition system based on multiscale local part model. The high performance is obtained by including both coarse global local root model and high resolution part model with overlapping patches. Our system remains high performance even after a significant dimensionality reduction of the feature vector to 96 (root 32 + part 64) dimension, *i.e.*, of the same size as a single original HOF/HOG/MBHx/MBHy vector. A random sampling strategy based on local part model is also proposed for efficient action recognition. In addition, we introduce the idea of using very high sampling density for efficient and accurate classification. Our system is both efficient and accurate, yielding state-of-the-art results on realistic large scale datasets, namely, 62.2% on HMDB51, 92.1% on UCF50 and 86.6% on UCF101.

7.1 Conclusions

The most notable conclusion is that, without losing efficiency, random sampling with very high density can generate a large number of patches, and therefore achieves good performance. With local part model, the high feature density can be obtained by per-

forming sampling with root model at low resolution. Also, random sampling can generate enough patches for clips with less frames, and at the same time improve efficiency by processing less features for videos with long duration.

Another very important fact for the performance resides in the local motion descriptors, which ideally should accurately encode both local structure and motion information. As one of such descriptors, MBH has been shown to outperform other descriptors by encoding motion boundary and suppressing camera motion. Yet, a more accurate optical flow estimation can significantly improve the MBH performance. In addition, better performance can be achieved by combining multiple descriptors with complementary information.

Finally, Low resolution does not necessarily lead to low performance. We are not dealing with gait recognition and face recognition, which focus on differentiating different persons. For action classification the high performance resides in better generalization (intra-class variation) and preventing overfitting. Similarly, a recent approach [44] (ILSVRC-2012 competition winner) showed the best image classification results by processing the images with lower resolution (256x256 by cropping and down-sampling).

The action recognition system normally includes three steps: how to extract spatio-temporal features, how to represent them, and how to classify the video based on the feature representations. For all the three steps, we have proposed solutions to address the recognition challenges with follow key contributions:

Feature extraction. We have presented a novel multiscale local part model on the purpose of maintaining both structure information and ordering of local events for action recognition. The method includes both a coarse primitive level root model covering event-content statistics and higher resolution overlapping part models incorporating structure and temporal relations. Such an approach is robust to "out-of-ordering" problem of bag-

of-features method, and thus shows good performance. Unlike Structured Models [26] and Spatio-temporal Pyramids [50] which include global information, lPM adds weak local low-level relationships to address unordered BoF method. Compared with existing methods, a major strength of our method is fast processing. The high computational efficiency is obtained by using integral video and fast random sampling. Since no feature detection is required for random sampling, most cost associated with feature extraction is spent on accessing memory through the integral videos. Experiments show very high efficiency even by using very high sampling density, which leads to accurate classification.

Feature representation. We have proposed a new local 3D descriptor based on histograms of oriented spatial-temporal gradients to encode both local static appearance and motion information. It significantly outperformed popular HOG descriptor with similar high computational efficiency. By using a discontinuity-preserving optical flow method, we improved the performance of the state-of-the-art MBH descriptor significantly. We further applied fast approximate nearest neighbour search method and PCA feature reduction technique to improve the efficiency of our approaches on action recognition. Our experiments have shown that the feature dimension can be reduced by 7/8 through PCA while preserving high accuracy and speeding up matching processing with low dimension features. We also showed that applying fast approximate nearest neighbour search could speed up the bag-of-words matching processing substantially at the cost of less than 1% performance loss for large codewords.

Classification. A new method based on histogram intersection kernel was proposed to combine multiple channels of different descriptors. In addition to the performance benefit, this approach has the advantage of high efficiency by using the fast SVM method [60]. We further improved the classification performance of Local Part Model through the use

of multiple channels. For multi-class SVM, we experimentally showed that one-verse-all strategy outperforms over-verse-one strategy on human action recognition with large scale realistic datasets. In comparison with current state-of-the-art methods, our system achieved the best performance on three large challenging realistic datasets, namely, HMDB51, UCF50, and UCF101.

We have shown that processing videos at low resolution can achieve good performance with high efficiency. However, it may be more advisable to use full resolution for certain applications. One example is recognizing group activities in the video with multiple subjects, which normally include collective behaviours of individuals in the group. In this case, there may be subtle inter-person interaction and very small human subjects. Another example is fine-grained activity recognition [89], which tries to differentiate more subtle activities with high inter-class similarity and high intra-class variation.

7.2 Future work

Our method is capable of recognizing realistic human action on large scale video. It could also be applied in the context of action localization, abnormal detection and video retrieval, *etc*. In the future, we would like to include the following improvements:

Extensions of local part model. We have shown the efficiency and effectiveness of local part model for action recognition. In the future, we would like to evaluate the benefits in using different combinations of parts, or parts with learnt weights. Additional approach includes a compact representation of LPM model in computing root and parts with same spatial resolution. We also want to explore the deeper part hierarchies (*i.e.* parts with parts). Considering that the parts are built on the half resolution, we expect the performance improves further when adding another layer of parts on full resolution

videos. Our preliminary experiments on GBH descriptors with FV show that adding another level of parts improves the performance from 44.7% to 46.0% on HMDB51.

Bias sampling with salient feature selection. Our random sampling strategy achieved state-of-the-art performance with high efficiency. This could be improved by using biased random samplers in order to find more discriminant patches. One aspect is to reduce the computational complexity by selecting the minimal subset of the most discriminative patches from our high density sampling grid, while at the same time preserving performance. A second aspect is to improve the performance with salient features. Some recent approaches have shown promises in using visual saliency on action recognition. Such methods include selection by AdaBoost [57], viewer's eye movement [63, 117] and human detection [122] *etc*.

Improvement on descriptors. Our results show the importance of the features and their descriptors on performance. The MBH descriptor shows excellent results for action recognition. The possible improvements include camera motion compensation and the evaluation of using more state-of-the-art optical flow methods. The gradient boundary histogram (GBH) could be improved by removing the background information. One such improvement is to only encode moving human gradients through using human detection and human tracking. A robust background and foreground subtraction algorithm and camera motion removal technique also could benefit the GBH.

Additional future works could include performing action recognition with explicit context information, and evaluating different feature encoding methods (*e.g*. Fisher Vector and VLAD) for better performance. We also want to extend our local part model in other applications, such as action localization and gesture recognition *etc*.

Appendix A

Glossary of Terms

weakly labelled video The video is only labelled as a positive or negative action class without using any training ground truth information, such as humans, body-parts or joint locations bounding box.

GBH A local descriptor based on pure spatio-temporal gradients. The gradients are computed by applying simple 1-D [-1,0,1] Sobel masks on both x and y directions, followed by a [-1, 1] temporal filter over two consecutive gradient images.

bag-of-features (BoF) Also called bag-of-words, which is originally applied to document analysis. It can be applied to image/video classification, by treating visual features as words. In computer vision, a bag of visual features is a vector of occurrence counts of a vocabulary of local image/video features.

Fisher vector (FV) A FV is a statistics encoding the distribution of a set of local image/video descriptors. FV extends the BoF by encoding high-order statistics (first and, optionally, second order) between the descriptors and a Gaussian Mixture Model (GMM). Therefore, in addition to including codewords' occurrences, it also encodes additional information about the distribution of the descriptors.

one-verse-one (OVO) Also called one-against-one (OAO), a method to perform multi-class classification by decomposing the multi-class problem into a number of two-class problems, and applying binary classification for each of them. For n classes, OVO constructs $n(n-1)/2$ SVM classifiers. The SVM models are learnt with data from any two of the all classes. After all models are trained, the prediction is implemented by max-wins voting, with class label assigned to the class with the largest vote from the models.

one-verse-all (OVA) Also called one-against-all (OAA), a method to perform multi-class classification by decomposing the multi-class problem into a number of two-class problems, and applying binary classification for each of them. For n classes, OVA constructs n SVM models. The ith SVM is trained with all of the examples from the ith class as positive labels, and remaining examples from other classes as negative labels.

support vector machines (SVMs) According to Wikipedia, "SVMs are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier".

Bibliography

[1] Ankur Agarwal and Bill Triggs. Hyperfeatures - multilevel local coding for visual recognition. In *ECCV*, 2006.

[2] Ankur Agarwal and Bill Triggs. Recovering 3d human pose from monocular images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(1):44–58, 2006.

[3] Briassouli Alexia, Tsiminaki Vagia, and Kompatsiaris Ioannis. Human motion analysis via statistical motion processing and sequential change detection. *EURASIP Journal on Image and Video Processing*, 2009, 2009.

[4] Saad Ali and Mubarak Shah. Human action recognition in videos using kinematic features and multiple instance learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(2):288–303, 2010.

[5] Erin L Allwein, Robert E Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *The Journal of Machine Learning Research*, 1:113–141, 2001.

[6] Ian Reid Alonso Patron-perez. A probabilistic framework for recognizing similar actions using spatio-temporal features. In *BMVC*, 2007.

[7] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

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

[9] Moshe Blank, Lena Gorelick, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. In *ICCV*, pages 1395–1402, 2005.

[10] Aaron F Bobick. Movement, activity and action: the role of knowledge in the perception of motion. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 352(1358):1257–1265, 1997.

[11] Aaron F. Bobick and James W. Davis. The recognition of human movement using temporal templates. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(3):257–267, 2001.

[12] Matthew Brand, Nuria Oliver, and Alex Pentland. Coupled hidden markov models for complex action recognition. In *CVPR*, pages 994–999. IEEE, 1997.

[13] Karla Brkić, Srdan Rašić, Axel Pinz, Siniša Šegvić, and Zoran Kalafatić. Combining spatio-temporal appearance descriptors and optical flow for human action recognition in video data. In *Croatian Computer Vision Workshop, Year 1*, 2013.

[14] Thomas Brox and Jitendra Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3):500–513, 2011.

[15] Lee W Campbell and Aaron F Bobick. Recognition of human body motion using phase space constraints. In *ICCV*, pages 624–630. IEEE, 1995.

[16] Liangliang Cao, Yadong Mu, Natsev Apostol, Shih-Fu Chang, Gang Hua, and John R. Smith. Scene aligned pooling for complex video recognition. In *ECCV*, pages 688–701, 2012.

[17] Zhimin Cao, Qi Yin, Xiaoou Tang, and Jian Sun. Face recognition with learning-based descriptor. In *CVPR*, pages 2707–2714. IEEE, 2010.

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

[19] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, volume 1, pages 886–893. IEEE, 2005.

[20] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*, pages 428–441. Springer, 2006.

[21] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *Proc. 2nd Joint IEEE Int Visual Surveillance and Performance Evaluation of Tracking and Surveillance Workshop*, pages 65–72, 2005.

[22] Alexei A Efros, Alexander C Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 726–733. IEEE, 2003.

[23] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Proceedings of the 13th Scandinavian Conference on Image Analysis*, 2003.

[24] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, pages 1–8, 2008.

[25] Jerome H. Friedman, Jon Louis Bentley, and Raphael A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, 1977.

[26] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Actom sequence models for efficient action detection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3201–3208. IEEE, 2011.

[27] Andrew Gilbert, John Illingworth, and Richard Bowden. Action recognition using mined hierarchical compound features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):883–897, 2011.

[28] Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(12):2247–2253, 2007.

[29] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, pages 1458–1465. IEEE, 2005.

[30] Yan Guo, Gang Xu, and Saburo Tsuji. Understanding human motion patterns. In *Pattern Recognition, Proceedings of the 12th IAPR International Conference on*, volume 2, pages 325–329. IEEE, 1994.

[31] Raffay Hamid, Amos Johnson, Samir Batta, Aaron Bobick, Charles Isbell, and Graham Coleman. Detection and explanation of anomalous activities: Representing activities as bags of event n-grams. In *Computer Vision and Pattern Recognition,*

2005. IEEE Computer Society Conference on, volume 1, pages 1031–1038. IEEE, 2005.

[32] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1):185–203, 1981.

[33] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425, 2002.

[34] Arpit Jain, Abhinav Gupta, Mikel Rodriguez, and Larry S Davis. Representing videos using mid-level discriminative patches. In *CVPR*, 2013.

[35] Mihir Jain, Hervé Jégou, and Patrick Bouthemy. Better exploiting motion for better action recognition. In *CVPR*, 2013.

[36] Hervé Jégou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Pérez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(9):1704–1716, 2012.

[37] Odest Chadwicke Jenkins and Maja J Mataric. Automated modularization of human motion into actions and behaviors. Technical Report CRES-02-002, Center for Robotics and Embedded Systems, University of S. California, 2002.

[38] Yu-Gang Jiang, Qi Dai, Xiangyang Xue, Wei Liu, and Chong-Wah Ngo. Trajectory-based modeling of human actions with motion reference points. In *ECCV*, pages 425–438, 2012.

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

[40] Yan Ke, R. Sukthankar, and M. Hebert. Efficient visual event detection using volumetric features. In *ICCV*, volume 1, pages 166–173, 2005.

[41] Yan Ke and Rahul Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *CVPR*, volume 2, pages II–506. IEEE, 2004.

[42] Alexander Kläser, Marcin Marszałek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, pages 995–1004, 2008.

[43] Orit Kliper-Gross, Yaron Gurovich, Tal Hassner, and Lior Wolf. Motion interchange patterns for action recognition in unconstrained videos. In *ECCV*, pages 256–269, 2012.

[44] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, volume 1, page 4, 2012.

[45] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: A large video database for human motion recognition. In *ICCV*, pages 2556–2563, 2011.

[46] Christoph H Lampert, Matthew B Blaschko, and Thomas Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *CVPR*. IEEE, 2008.

[47] Zhen-zhong Lan, Lei Bao, Shoou-I Yu, Wei Liu, and Alexander G Hauptmann. Multimedia classification and event detection using double fusion. *Multimedia Tools and Applications*, pages 1–15, 2013.

[48] I. Laptev and T. Lindeberg. Space-time interest points. In *Computer Vision, 2003. ICCV 2003. IEEE 9th International Conference on*, pages 432–439, 2003.

[49] Ivan Laptev and Tony Lindeberg. Local descriptors for spatio-temporal recognition. In *ECCV Workshop, Spatial Coherence for Visual Motion Analysis*, pages 91–103. Springer, 2004.

[50] Ivan Laptev, Marcin Marszałek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[51] Ivan Laptev and Patrick Pérez. Retrieving actions in movies. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, 2007.

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

[53] Quoc V. Le, Will Y. Zou, Serena Y. Yeung, and Andrew Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, pages 3361–3368, 2011.

[54] Thomas Leung and Jitendra Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43:2944, 2001.

[55] Oskar Linde and Tony Lindeberg. Composed complex-cue histograms: An investigation of the information content in receptive field based image descriptors for object recognition. *Computer Vision and Image Understanding*, 116(4):538–560, 2012.

[56] Ce Liu, William T Freeman, Edward H Adelson, and Yair Weiss. Human-assisted motion annotation. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008.

[57] L. Liu, L. Shao, and P. Rockett. Human action recognition based on boosted feature selection and naive bayes nearest-neighbor classification. *Signal Processing*, pages 1521–1530, 2012.

[58] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, pages 91–111, 2003.

[59] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

[60] Subhransu Maji, Alexander C Berg, and Jitendra Malik. Classification using intersection kernel support vector machines is efficient. In *CVPR*. IEEE, 2008.

[61] David Marr and Herbert Keith Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 200(1140):269–294, 1978.

[62] Marcin Marszałek, Ivan Laptev, and Cordelia Schmid. Actions in context. In *CVPR*, pages 2929 – 2936, 2009.

[63] Stefan Mathe and Cristian Sminchisescu. Dynamic eye movement datasets and learnt saliency models for visual action recognition. In *ECCV*, pages 842–856, 2012.

[64] Pyry Matikainen, Martial Hebert, and Rahul Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In *Computer Vi-*

sion Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on, pages 514–521. IEEE, 2009.

[65] Pyry Matikainen, Martial Hebert, and Rahul Sukthankar. Representing pairwise spatial and temporal relations for action recognition. In *ECCV 2010*, pages 508–521. Springer, 2010.

[66] Hongying Meng, Nick Pears, and Chris Bailey. A human action recognition system for embedded computer vision application. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007.

[67] Ross Messing, Chris Pal, and Henry Kautz. Activity recognition using the velocity histories of tracked keypoints. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 104–111. IEEE, 2009.

[68] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615–1630, 2005.

[69] Jonathan Milgram, Mohamed Cheriet, and Robert Sabourin. "one against one" or "one against all" : Which one is better for handwriting recognition with svms? In *Tenth International Workshop on Frontiers in Handwriting Recognition*, 2006.

[70] Thomas B Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2):90–126, 2006.

[71] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application (VISSAPP'09)*, pages 331–340, 2009.

[72] H-H Nagel. From image sequences towards conceptual descriptions. *Image and vision computing*, 6(2):59–74, 1988.

[73] Juan Carlos Niebles, Hongcheng Wang, and Li Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *International Journal of Computer Vision*, 79(3):299–318, 2008.

[74] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2161–2168. IEEE, 2006.

[75] Sourabh A Niyogi and Edward H Adelson. Analyzing and recognizing walking figures in xyt. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 469–474. IEEE, 1994.

[76] Eric Nowak, Frédéric Jurie, and Bill Triggs. Sampling strategies for bag-of-features image classification. In *ECCV*, pages 490–503, 2006.

[77] Dan Oneaţă, Jakob Verbeek, and Cordelia Schmid. Action and event recognition with fisher vectors on a compact feature set. In *IEEE Intenational Conference on Computer Vision (ICCV)*, 2013.

[78] Paul Over, Jon Fiscus, Greg Sanders, Martial Michel, George Awad, Alan F Smeaton, Wessel Kraaij, and Georges Quénot. Trecvid 2013-an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *TRECVID 2013*, 2013.

[79] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010.

[80] Massimiliano Pierobon, Marco Marcon, Augusto Sarti, and Stefano Tubaro. Clustering of human actions using invariant body shape descriptor and dynamic time warping. In *Advanced Video and Signal Based Surveillance, 2005. AVSS 2005. IEEE Conference on*, pages 22–27. IEEE, 2005.

[81] Ramprasad Polana and Randal Nelson. Low level recognition of human motion (or how to get your man without finding his body parts). In *Motion of Non-Rigid and Articulated Objects, 1994., Proceedings of the 1994 IEEE Workshop on*, pages 77–82. IEEE, 1994.

[82] Ronald Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976–990, 2010.

[83] Deva Ramanan and David A Forsyth. Automatic annotation of everyday movements. In *Advances in neural information processing systems*. MIT Press, Cambridge, MA, 2003.

[84] Michalis Raptis and Leonid Sigal. Poselet key-framing: A model for human activity recognition. In *CVPR*, 2013.

[85] Kishore K. Reddy and Mubarak Shah. Recognizing 50 human action categories of web videos. *Machine Vision and Applications Journal*, pages 1–11, September, 2012.

[86] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004.

[87] Mikel D. Rodriguez, Javed Ahmed, and Mubarak Shah. Action mach: a spatio-temporal maximum average correlation height filter for action recognition. In *Com-*

puter Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, 2008.

[88] Karl Rohr. Towards model-based recognition of human movements in image sequences. *CVGIP: Image understanding*, 59(1):94–115, 1994.

[89] Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele. A database for fine grained activity detection of cooking activities. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1194–1201. IEEE, 2012.

[90] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *ECCV*, pages 430–443, 2006.

[91] S. Sadanand and J. J. Corso. Action bank: A high-level representation of activity in video. In *CVPR*, pages 1234–1241, 2012.

[92] Gerard Salton. *Automatic Information Organization and Retrieval*. McGraw Hill Text, 1968.

[93] Michael Sapienza, Fabio Cuzzolin, and Philip H.S. Torr and. Learning discriminative space-time actions from weakly labelled videos. In *ECCV*, 2012.

[94] Konrad Schindler and Luc Van Gool. Action snippets: How many frames does human action recognition require? In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008.

[95] Christian Schüldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: A local svm approach. In *ICPR (3)*, pages 32–36, 2004.

[96] William Robson Schwartz, Aniruddha Kembhavi, and Larry S Harwood, Davidand Davis. Human detection using partial least squares analysis. In *ICCV*, pages 24–31. IEEE, 2009.

[97] Paul Scovanner, Saad Ali, and Mubarak Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th international conference on Multimedia*, pages 357–360. ACM, 2007.

[98] John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[99] Feng Shi, Robert Laganiere, Emil Petriu, and Haiyu Zhen. Lpm for fast action recognition with large number of classes. In *THUMOS: ICCV Workshop on Action Recognition with a Large Number of Classes*, 2013.

[100] Feng Shi, E. M. Petriu, and A. Cordeiro. Human action recognition from local part model. In *Proc. IEEE Int Haptic Audio Visual Environments and Games (HAVE) Workshop*, pages 35–38, 2011.

[101] Feng Shi, Emil Petriu, and Robert Laganiere. Sampling strategies for real-time action recognition. In *Proc. IEEE Conf. Comput. Vision Pattern Recognition. IEEE*, 2013.

[102] Chanop Silpa-Anan and Richard Hartley. Optimised kd-trees for fast image descriptor matching. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008.

[103] Cees GM Snoek, Marcel Worring, and Arnold WM Smeulders. Early versus late fusion in semantic video analysis. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 399–402. ACM, 2005.

[104] Berkan Solmaz, Shayan Modiri Assari, and Mubarak Shah. Classifying web videos using a global video descriptor. *Machine Vision and Applications*, pages 1–13, 2012.

[105] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. Technical Report CRCV-TR-12-01, CRCV, University of Central Florida, 2012.

[106] Chen Sun and Ram Nevatia. Large-scale web video event classification by use of fisher vectors. In *Applications of Computer Vision (WACV), 2013 IEEE Workshop on*, pages 15–22. IEEE, 2013.

[107] Ju Sun, Yadong Mu, Shuicheng Yan, and Loong-Fah Cheong. Activity recognition using dense long-duration trajectories. In *Multimedia and Expo (ICME), 2010 IEEE International Conference on*, pages 322–327. IEEE, 2010.

[108] Ju Sun, Xiao Wu, Shuicheng Yan, Loong-Fah Cheong, Tat-Seng Chua, and Jintao Li. Hierarchical spatio-temporal context modeling for action recognition. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2004–2011. IEEE, 2009.

[109] Michael J Swain and Dana H Ballard. Color indexing. *International journal of computer vision*, 7(1):11–32, 1991.

[110] Kevin Tang, Li Fei-Fei, and Daphne Koller. Learning latent temporal structure for complex event detection. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1250–1257. IEEE, 2012.

[111] Christian Thurau and Václav Hlavác. Pose primitive based human action recognition in videos or still images. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008.

[112] Yicong Tian, Rahul Sukthankar, and Mubarak Shah. Spatiotemporal deformable part models for action detection. In *CVPR*, 2013.

[113] Hirofumi Uemura, Seiji Ishikawa, and Krystian Mikolajczyk. Feature tracking and motion compensation for action recognition. In *BMVC*, pages 30.1–30.10,, 2008.

[114] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[115] Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(3):480–492, 2012.

[116] Ashok Veeraraghavan, Rama Chellappa, and Amit K Roy-Chowdhury. The function space of an activity. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 959–968. IEEE, 2006.

[117] Eleonora Vig, Michael Dorr, and David Cox. Space-variant descriptor sampling for action recognition based on saliency and eye movements. In *ECCV*, pages 84–97, 2012.

[118] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[119] Chunyu Wang, Yizhou Wang, and Alan L Yuille. An approach to pose-based action recognition. In *CVPR*, 2013.

[120] Heng Wang, A. Kläser, C. Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *CVPR*, pages 3169–3176, 2011.

[121] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision*, pages 1–20, 2013.

[122] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *International Conference on Computer Vision*, 2013.

[123] Heng Wang, Muhammad Muneeb Ullah, Alexander Kläser, Ivan Laptev, and Cordelia Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, pages 127–137, 2009.

[124] Liang Wang and David Suter. Informative shape representations for human action recognition. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 2, pages 1266–1269. IEEE, 2006.

[125] Yang Wang, Payam Sabzmeydani, and Greg Mori. Semi-latent dirichlet allocation: A hierarchical model for human action recognition. In *Human Motion–Understanding, Modeling, Capture and Animation*, pages 240–254. Springer, 2007.

[126] Daniel Weinland and Edmond Boyer. Action recognition using exemplar-based embedding. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008.

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

[128] Chris Whiten, Robert Laganiere, and Guillaume-Alexandre Bilodeau. Efficient action recognition with mofreak. In *Computer and Robot Vision (CRV), 2013 International Conference on*, pages 319–325, 2013.

[129] Geert Willems, Tinne Tuytelaars, and Luc J. Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *ECCV*, pages 650–663, 2008.

[130] Junji Yamato, Jun Ohya, and Kenichiro Ishii. Recognizing human action in time-sequential images using hidden markov model. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*, pages 379–385. IEEE, 1992.

[131] Lei Yang, Nanning Zheng, Jie Yang, Mei Chen, and Hong Chen. A biased sampling strategy for object categorization. In *ICCV*, pages 1141–1148, 2009.

[132] Lahav Yeffet and Lior Wolf. Local trinary patterns for human action recognition. In *ICCV*, pages 492–497, 2009.

[133] A Yilma and Mubarak Shah. Recognizing human actions in videos acquired by uncalibrated moving cameras. In *ICCV*, volume 1, pages 150–157, 2005.

[134] Tsz-Ho Yu, Tae-Kyun Kim, and Roberto Cipolla. Real-time action recognition by spatiotemporal semantic and structural forests. In *BMVC*, pages 1–12, 2010.

[135] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l1 optical flow. In *In Ann. Symp. German Association Pattern Recognition*, pages 214–223, 2007.

[136] J. Zhang, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: a comprehensive study. *International Journal of Computer Vision*, 73:2007, 2007.

[137] Jun Zhu, Baoyuan Wang, Xiaokang Yang, Wenjun Zhang, and Zhuowen Tu. Action recognition with actons. ICCV, 2013.