# Learning Semantic Features for Visual Recognition

by

JINGEN LIU
M.S., University of Central Florida, 2008
M.E., Huazhong University of Science and Technology, 2003
B.S., Huazhong University of Science and Technology, 2000

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

Fall Term
2009

Major Professor: Mubarak Shah

# ABSTRACT

Visual recognition (e.g., object, scene and action recognition) is an active area of research in computer vision due to its increasing number of real-world applications such as video (image) indexing and search, intelligent surveillance, human-machine interaction, robot navigation, etc. Effective modeling of the objects, scenes and actions is critical for visual recognition. Recently, bag of visual words (BoVW) representation, in which the image patches or video cuboids are quantized into visual words (i.e., mid-level features) based on their appearance similarity using clustering, has been widely and successfully explored. The advantages of this representation are: no explicit detection of objects or object parts and their tracking are required; the representation is somewhat tolerant to within-class deformations, and it is efficient for matching.

However, the performance of the BoVW is sensitive to the size of the visual vocabulary. Therefore, computationally expensive cross-validation is needed to find the appropriate quantization granularity. This limitation is partially due to the fact that the visual words are not semantically meaningful. This limits the effectiveness and compactness of the representation. To overcome these shortcomings, in this thesis we present principled approach to learn a semantic vocabulary (i.e. high-level features) from a large amount of visual words (mid-level features). In this context, the thesis makes two major contributions.

First, we have developed an algorithm to discover a compact yet discriminative semantic vocabulary. This vocabulary is obtained by grouping the visual-words based on their distribution in videos (images) into visual-word clusters. The mutual information (MI) between the clusters and the videos (images) depicts the discriminative power of the semantic vocabulary, while the MI between visual-words and visual-word clusters measures the compactness of the vocabulary. We apply the information bottleneck (IB) algorithm to find the optimal number of visual-word clusters by finding the good tradeoff between compactness and discriminative power. We tested our proposed approach on the state-of-the-art KTH dataset, and obtained average accuracy of 94.2%. However, this approach performs one-side clustering, because only visual words are clustered regardless of which video they appear in. In order to leverage the co-occurrence of visual words and images, we have developed the co-clustering algorithm to simultaneously group the visual words and images. We tested our approach on the publicly available fifteen scene dataset and have obtained about 4% increase in the average accuracy compared to the one side clustering approaches.

Second, instead of grouping the mid-level features, we first embed the features into a low-dimensional semantic space by manifold learning, and then perform the clustering. We apply Diffusion Maps (DM) to capture the local geometric structure of the mid-level feature space. The DM embedding is able to preserve the explicitly defined diffusion distance, which reflects the semantic similarity between any two features. Furthermore, the DM provides multi-scale analysis capability by adjusting the time steps in the Markov transition matrix. The experiments on KTH dataset show that DM can perform much better (about 3% to 6% improvement in average accuracy) than other manifold learning approaches and IB method. Above methods use only single type of features. In order to combine multiple heterogeneous features for visual recognition, we further propose the Fielder Embedding to capture the complicated semantic relationships between all entities (i.e., videos, images, heterogeneous features). The discovered relationships are then employed to further increase the recognition rate. We tested our approach on Weizmann dataset, and achieved about 17% 21% improvements in the average accuracy.

Dedicated to my family.

## ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Mubarak Shah, for his invaluable guidance, encouragement and generous support. I really appreciate his great patience to help me make progress in all respects like academic research, communication capability, and so on. His hard-working and persistent attitudes motivated me to do my best over the past years.

I would like to thank my mentor, Dr. Jiebo Luo, for all his valuable help to me when I was doing summer intern in Kodak research labs, and in the following collaboration.

I would like to thank my committee members, Dr. Niels Lobo, Dr. Joseph J. LaViola Jr. and Dr. Xin Li, for their precious services in my committee and valuable comments on my research work.

I also want to thank my colleagues including Yun Zhai, Saad Ali, Yang Yang, Kishore Reddy, Enrique G. Ortiz, Arslan Basharat, Imran Saleemi, Mikel Rodriguez, and Naveed Imran Syed.

Specially, I would like to thank my parents, my wife and my brothers and sisters for their continued support over the past years.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER 1: INTRODUCTION

Visual recognition is one of the fundamental functionalities of the human vision system in daily life. According to our experience, a human being has no difficulty in quickly recognizing new objects, even as a child. Research scientists from various fields such as biology, brain cognitive science, computer science, and so on have attempted to discover the secret of the human vision system. In recent decades, with a vast amount of image and video data generated every day around the world, visual recognition is gaining increasing attention in the computer vision community. The goal of this dissertation is to develop new techniques for visual recognition, which includes scene and object recognition from an image and action recognition from a video. Although image and video have different dimensionality, we aim to discover the underlying connection between them and design a common representation and recognition model for object/scene recognition and action recognition.

Object recognition is tasked to judge whether a specific object is contained in the image or not. Figure 1.1 shows some images selected from the famous CalTech object data set, in which each image contains some specific object with various scales, lighting conditions, view points, and so on. In the context of this work, the scene is defined as the physical setting of the environment where the image is taken. Some examples of image scenes include outdoor, indoor, beach, mountain, forest, office, and urban landscape. Figure 1.2 shows some scene images from the publicly available fifteen-scene data set [107]. In general, the image scene cannot be described by a single object. Rather, it is often a collection of objects. For instance, an office scene may be composed of the interior building walls, desks, chairs, computer monitors, and so on.

Object and scene recognition has a wide range of applications, such as image understanding, robot navigation, and content-based image indexing and retrieval (CBIR) [10] [17]

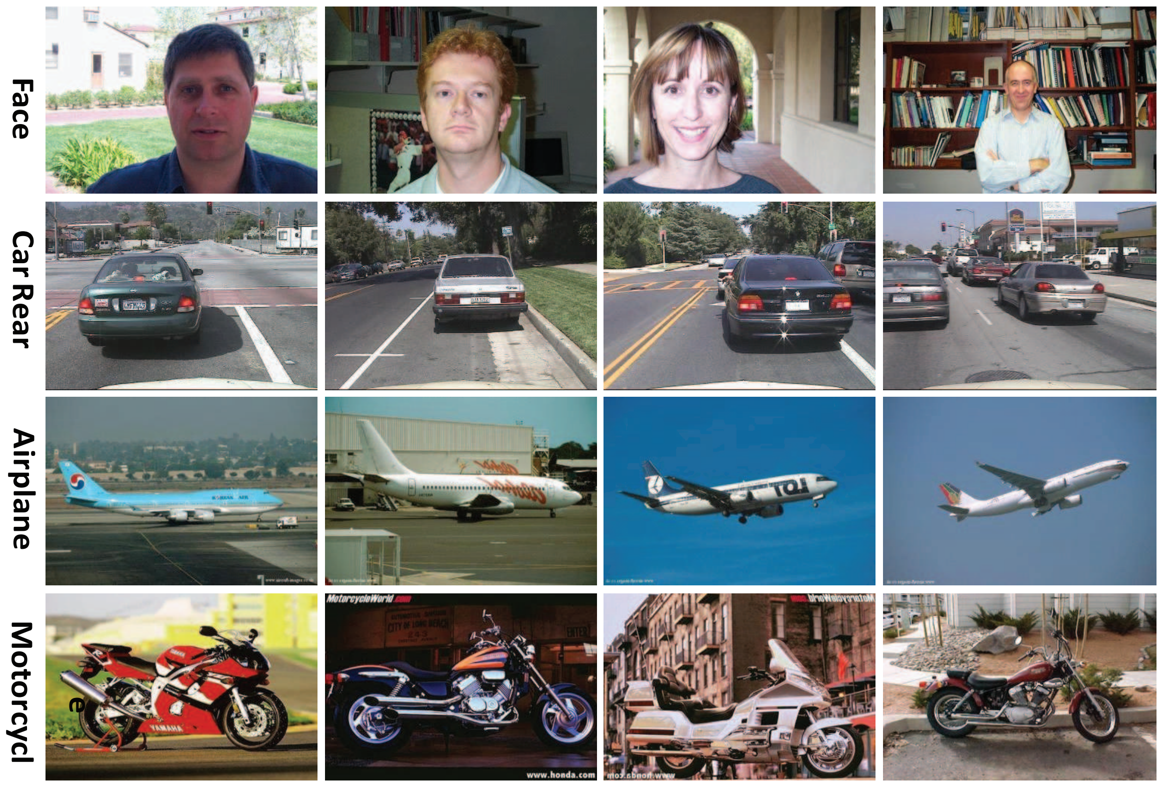


Figure 1.1: Example object images selected from Caltech-6 dataset showing the variation in scales, viewpoints, and illumination changes. Each row corresponds to one category.

[20] [29] [88] [125]. Most commercial image search engines retrieve images using the file-names, attached tags, or surrounding web texts rather than the visual content of the images. Therefore, the search quality is very poor. Scene and object recognition can be employed to automatically label the images based on their visual content, such that we can index the images based on their semantic labels for fast and high quality search. Another important application is in robotics. In certain situations, robots can be used to substitute for humans for dangerous tasks. Recognizing the location (i.e., scene recognition) is critical for robot navigation, and further localizing the target objects is also significantly important.

In the context of this dissertation, the goal of action recognition is to discriminate varied human actions (e.g. running, walking, horseback riding) from videos. Figures 1.3 and 1.4 demonstrate some examples from the frequently used action data sets: KTH [21], Weizmann

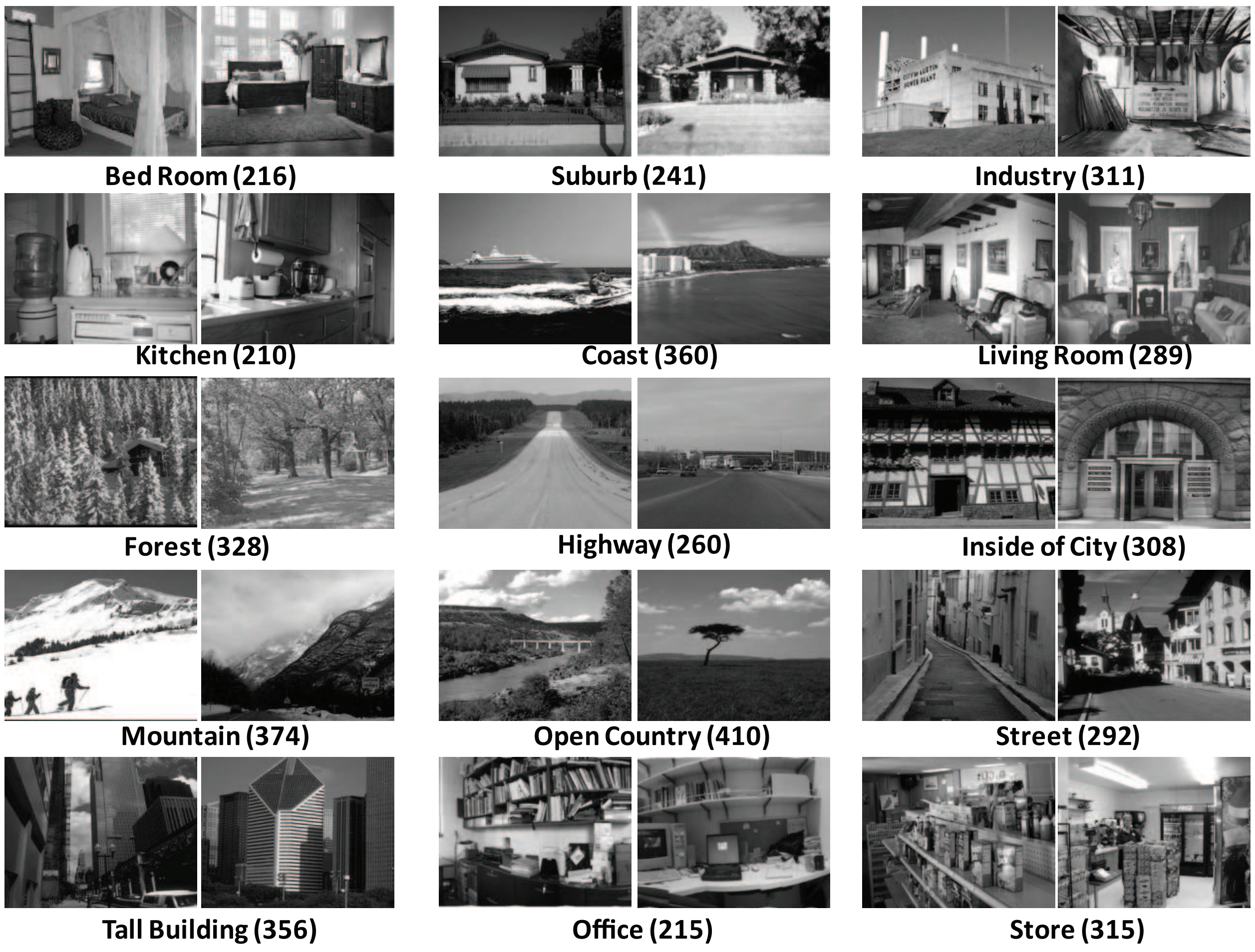


Figure 1.2: Example scene images selected from the fifteen-scene data set. The number of images contained in each category is shown under the images.

[34], and IXMAS [31]. All the three data sets were taken in human-controlled environments. In general, the backgrounds of the actions are simple and the cameras are almost static. Unlike object and scene recognition, recognizing actions from those simple videos is very difficult, because the acquisition of dynamic features from actions is not trivial. Recently, research on action recognition has moved to the more complicated and realistic action data sets [54] [47]. Figure 1.6 demonstrates some action examples from the UCF YouTube data set, which consists of unconstrained videos. Generally, the unconstrained videos contain significant camera motion, cluttered backgrounds, and changes in object appearances, scales, illumination conditions, and viewpoints.

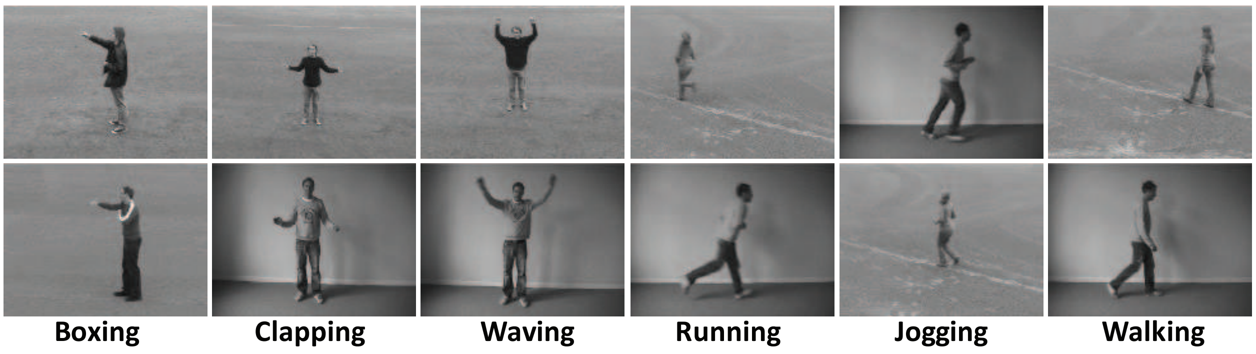


Figure 1.3: Example actions selected from the KTH dataset. Each column shows two action examples from one category. It has 6 categories with about 600 action videos in total.

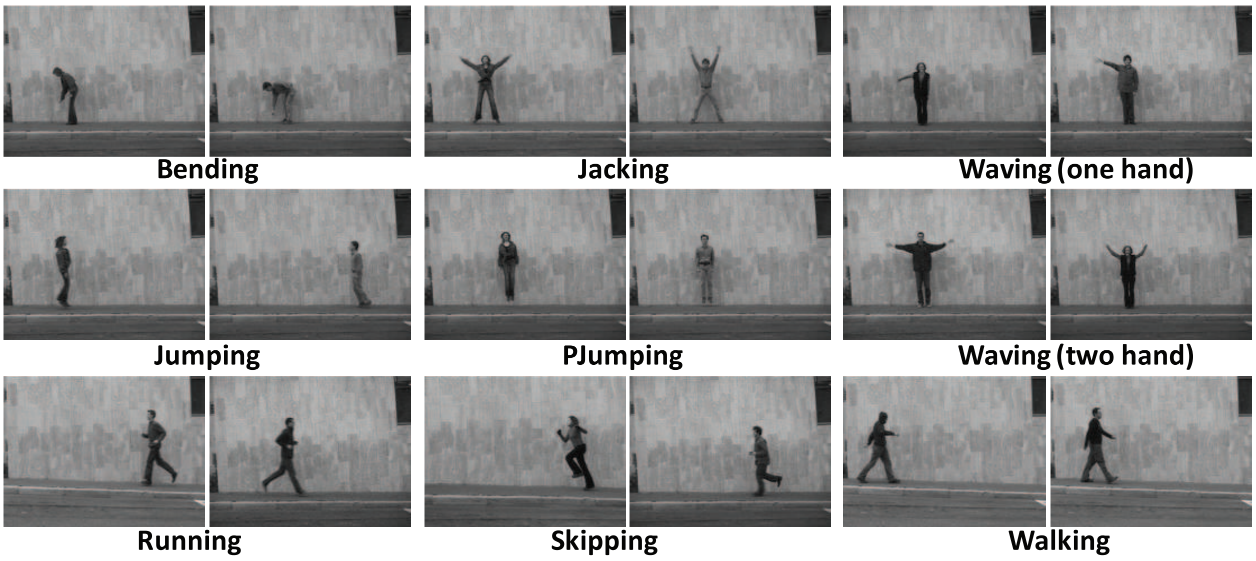


Figure 1.4: Example actions from the Weizmann action data set. It contains 9 actions with about 81 action videos in total.

Automatic action recognition can benefit video content understanding, video content based indexing and retrieval, human-computer interaction, intelligent video surveillance, and robotics. For instance, detecting abnormal actions in surveillance or retrieving similar actions from large amounts of archived videos is critical for video content analysis in modern intelligent systems. What is more, in robotics, a practical smart robot must be able to understand the actions from the partners (humans or machines) and make the correct response.

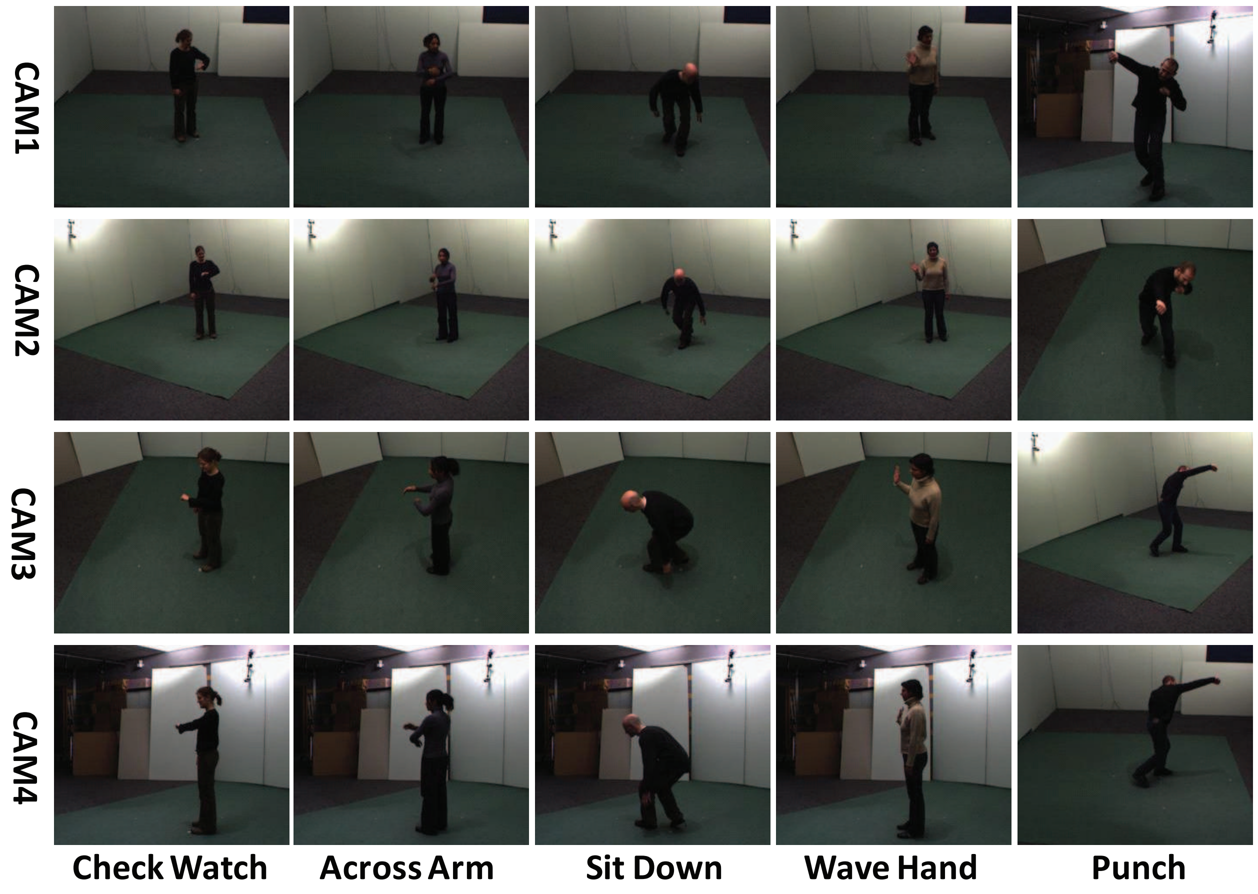


Figure 1.5: Four views of five selected action examples from IXMAS dataset. It has 13 action categories with 5 camera views and about 2,000 video sequences in total.

Although we recognize objects and scenes from still images and actions from videos, we have developed a generic visual representation schema for objects, scenes and actions. This results from the observation that both images and action videos can be modelled by *bag of features*. Therefore, the recognition techniques proposed in this work can be applied to both object or scene recognition and action recognition.

## 1.1 Motivations

Meaningful visual representation is of fundamental importance for visual recognition. Just like the physical world is composed of rich structures (e.g. electrons, atoms, molecules, and

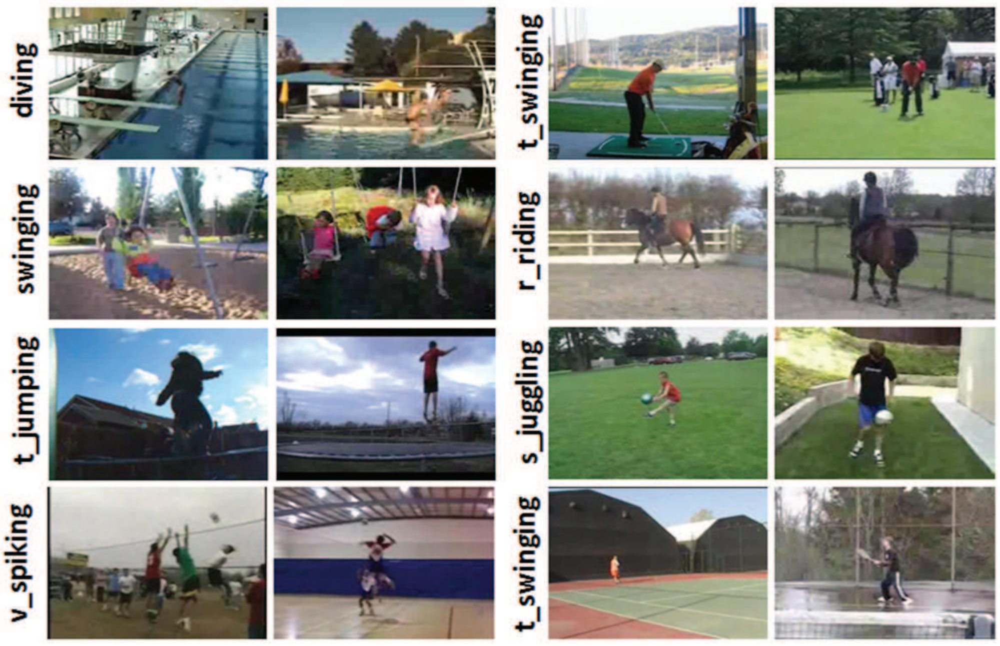


Figure 1.6: Examples actions from UCF YouTube action data set. It contains 11 action categories (here, eight categories are listed). Each category contains more than 100 video clips.

polymers) and documents consist of a large number of units (such as characters, words, phrases, and sentences), images and videos can also be treated as a collection of elements (pixels, voxels, edges, patches, etc.) [112]. The early studies on image processing discovered that neighboring pixels are highly correlated, which can be measured by the conditional entropy of a pixel given its neighbor's information [27]. Due to the information redundancy, we can transform an image into a new space with less correlated or ideally independent bases (e.g. wavelet transform, Principle Component Analysis), where an image is represented by its coefficients associated with this lower dimensional space [15] [87] [113]. Therefore, this discovery can be used for image coding and redundancy reduction.

Just as a chemical analyst differentiates a material by checking the proportion of its ingredients, the machine may be able to recognize a scene or object image by analyzing the statistics of the image components or elements. One widely adopted approach is to apply different frequency filters on each pixel, and then compute the first or second moment of the response values of the entire image or local patch. For instance, the well-known GIST feature descriptor [7] [11] is computed on the output of filters tuned to different orientations and scales (i.e., Gabor filters with 6 orientations and 4 scales [11]). In order to further capture the coarse spatial information, the response image can be down-sampled to have spatial resolution of M×M pixels [11].

A histogram of the filter responses is another widely used image representation, such as texton (i.e., quantized filter responses) histogram [66] [87] [115]. Figure 1.7 shows 38 filter banks with varied orientations and scales to capture some basic structures such as edge, spot, bar, etc [87]. These filters are convolved with the image and generate 38 response values at each pixel. In the next step, all the pixels that are represented by 38-dimension vectors are grouped into $N$ clusters as a texton dictionary, with which the image can be represented by the histogram of textons. Texton-based approach was also used for object recognition recently in [66]. Another statistic on pixels is a color histogram that depicts the color distribution of an image. Color histogram is one of the earliest and most popular image representations for image classification, indexing and retrieval [17] [29] [88] [51]. Going beyond pixel statistics, some recent work perform recognition using the statistics of image components consisting of a group of correlated pixels (e.g. contours [8] [119], object shapes, local patches [4] [18] [79] [100]), because the components are able to capture the local geometric structure of the pixels.

Recently, bag of local patches (i.e., 2-D image patches or 3-D video cuboids) based approaches are receiving increasing attention in object, scene and action recognition due to their computational simplicity and surprising good performance. This was first used in object recognition to cope with partial occlusion problems, then it was widely used in scene and action recognition. Inspired by the success of bag of words ($BOW$) approaches in text categorization [23] [114], computer vision researchers have discovered the connection between local patches in images (or videos) and words in documents. In the BOW text representation, a document is represented as a histogram of words. In order to employ the BOW to represent

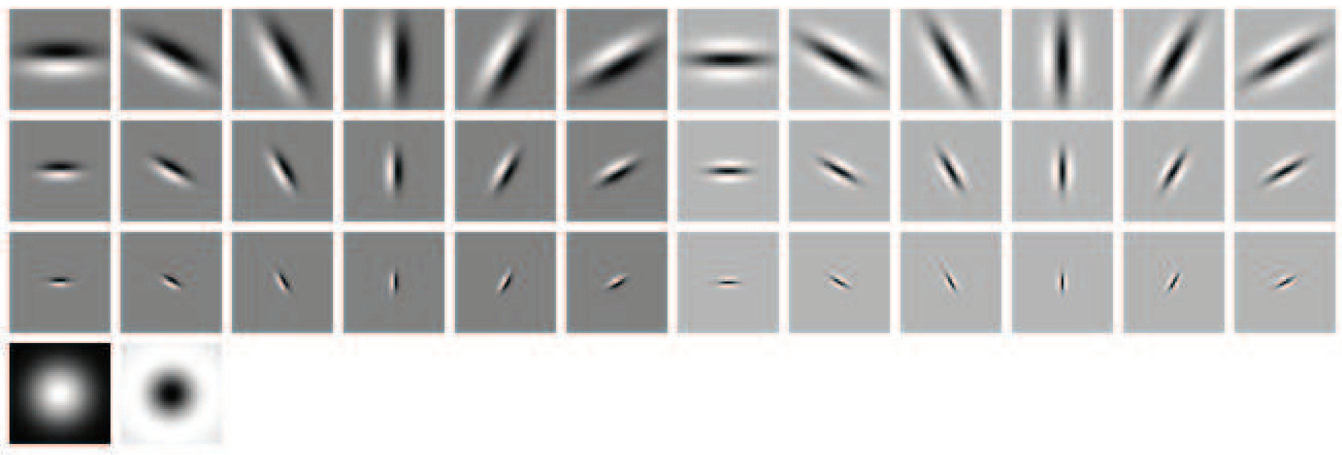


Figure 1.7: The Maximum Response filters ( this figure is taken from [87] ). They include two anisotropic filters with 3 scales at 6 orientations capturing the edges (the dark images of the first three rows) and bars (the light images of the first three rows), and 2 rotationally symmetric filters (a Gaussian and a Laplacian Gaussian).

an image or video, first we need to quantize the local patches into visual words. The $k$-means algorithm is commonly used to construct an initial visual vocabulary due to its simplicity. The visual representation in terms of the visual-words histograms is generally called the bag of visual words (BOVW) approach. The advantages of $BOVW$ are various. First of all, it can handle partial occlusion, because histogram intersection is used to match images. With the introduction of varied feature detectors and descriptors [71], BOVW is somewhat invariant to appearance transformations, such as changes of viewpoint, scale, orientation, and translation. In addition, BOVW makes the model somewhat tolerant to within-class deformations by adopting the appropriate granularity in feature quantization. Finally, it is not only efficient in computation time but also in feature storage. The time efficiency results from the fact that the features quantized into the same cluster (visual word) are deemed matched. So the image matching is pre-computed to some extent by feature quantization. On the other side, the system stores the feature labels instead of high-dimensional feature descriptors.

However, the BOVW model has some drawbacks. First of all, the recognition performance of BOVW is sensitive to the vocabulary size. Usually, one common concern of BOVW is what optimal vocabulary size is appropriate. Although coarse quantization may handle within-class deformation as aforementioned, it also makes the feature less discriminative. In practice, larger vocabulary size (i.e., fine quantization) can achieve better performance. Yet, this also brings up new problems, such as the visual representation (i.e., vector) is high dimensional and sparse, which may cause the model to be sensitive to noise and inefficient for weaker classifiers (e.g. $KNN$ classifier). Therefore, learning a compact vocabulary with an optimal number of *visual words* is necessary.

Another major problem in the BOVW model is that the visual words are not semantically meaningful. For example, given the local patches extracted from an image, a human being may have the capability to deduce the high level concepts implied in them based on their prior knowledge about the local visual structures. A *visual word* to the machine, however, is only a collection of patches with similar structure. In other words, the machine is unable to predict the possible contexts of a *visual word* without further information. The underlying reason is that the quantization criterion (i.e., the clustering measure) is only based on the appearance similarity (i.e., Euclidean distance). Hence, $k$-means clustering is unable to capture the semantic relationships between the patches. This semantic relationship is the so-called semantic gap between low level vision and high level concepts. Discovering this relationship is useful for image and video understanding.

In addition, because BOVW approaches disregard all information about the spatial layout of the features, it is incapable of depicting the entire shape of a scene or the global structure of an object. This severely limits the descriptive capability of the model. For example, in Figure 1.2, the three categories of scenes: *street*, *tall building*, and *inside of city*, may share certain local patches (e.g., construction patches from buildings), but they have different structural layouts. For example, *street* may have “road” patches in the mid-bottom of the scene, and *tall building* may have “sky” patches on the top of the scene.

In order to overcome these shortcomings, we developed novel algorithms to acquire the semantic meaning of the *visual words*, as well as their structural information. Given an initial visual vocabulary $V = \{x_1, x_2, ..., x_n\}$, where $x_i \in \mathcal{R}^d$ (e.g. d=128 for SIFT descriptor), which is generated from a collection of training patches $x \in \mathcal{R}^d$ by $k$-means clustering, our goal is to discover a semantic vocabulary $\hat{V} = \{\hat{x}_1, \hat{x}_2, ..., \hat{x}_m\}$, such that $m << n$ and $\forall x_i \in V$, $\exists \hat{x}_k \in \hat{V}$ includes $x_i$. In other words, $\hat{x}_k$ is a cluster of visual words $x_i$. However, the visual words grouped into one $\hat{x}_k$ are supposed to possess a similar semantic meaning. They do not necessarily share the same appearance. Namely, patches with varied appearances may be assigned into one *visual word* cluster $\hat{x}_j$, if they are semantically similar. For instance, the patch from "chin" and "eye" may be highly associated with some *visual word* cluster $\hat{x}_j$ since this *visual word* cluster depicts the "face" concept. Hence, in general, the distortion of cluster $\hat{x}_j$ in terms of Euclidean distance may be larger than that of $x_i$.

In fact, the discovery of semantic vocabulary $\hat{V}$ is to find a soft or hard mapping between $\hat{V}$ and $V$. The mapping aims to identify and discriminate between different contexts of *visual words* without the appeal to high level analysis. We have to face two significant issues: the *polysemy* problem in which one *visual word* may have multiple meanings, and the *synonym* problem, which means that several *visual words* may characterize the same object or scene context [95]. Figure 1.8 demonstrates both problems. The top *visual word* has the *polysemy* problem, as the majority of patches are from "forehead" while some are from "wheels". Therefore, this *visual word* is highly correlated to the "face" context, but possibly also occurs in the context of "motorcycle". The bottom *visual word* also demonstrates the same case as the top one. Although these two *visual words* have different appearances, they occur in the same "face" context. This is the *synonym* problem.

Thus, semantically similar *visual words* must be associated to a concept context. The degree of co-occurrence between pairs of *visual words* can be one of the measures telling their semantic similarity. Given a training data set, two visual words are semantically similar if they have very close distribution on the data set. Therefore, the thesis of this work is to:

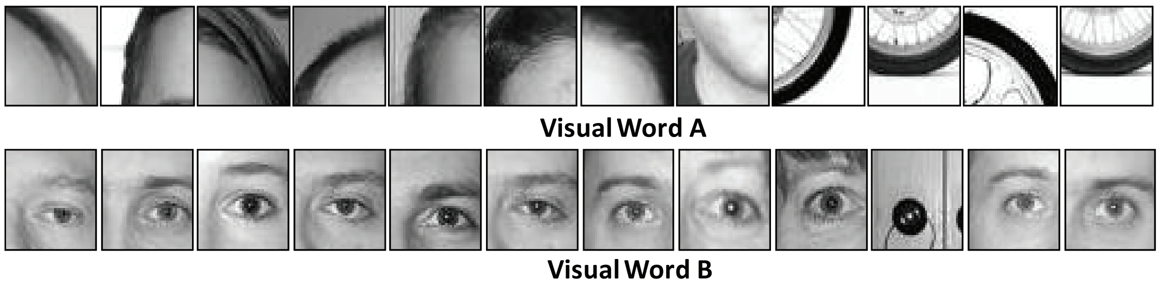


Figure 1.8: Two visual words demonstrate the *polysemy* and *synonym* problems in visual vocabulary learning.

*Exploit the distributions of the* visual words *(mid-level features) on the training data (i.e. images or videos) and then discover the semantic* visual word *clusters (high-level features) based on their distributions on the data, hence acquiring a compact yet discriminative visual vocabulary for recognition.*

For this purpose, in this dissertation we explored this problem from the following two aspects: feature *clustering* and *projection*.

In the *clustering* framework we aim to group the *visual words* into *visual word* clusters (VWCs). This can also be considered as finding a mapping function between the *visual words* and VWCs. This mapping can be either "soft" ($n$ to $n$ mapping) or "hard" ($n$ to one mapping). For simplicity, we only consider the "hard" mapping, which actually is a partition of the *visual words* into several mutually exclusive clusters. In general, the partition quality is measured by the overall distortion of the clusters. A good cluster is supposed to have small distortion, which means it includes highly "similar" *visual words*. However, the dissimilarity metric is defined on the "semantic" space (e.g., the basis of the space consists of the semantic concepts like *face*, *wheel*, *windows* etc.), instead of "appearance" space (e.g., color or edge feature space for image patches). In our work, we adopt Mutual Information (MI) between *visual words* and videos or images, estimated from the co-occurrence matrix, as our clustering criteria.

Rather than directly clustering the features (*visual words*), we can embed each feature into a low-dimensional semantic space that can preserve the semantic "structure" of the data, which means semantically similar features are placed closely in the new space. This is the idea of feature *projection.* The essence of the manifold embedding is that the features can be descried with far fewer parameters than the actual number of parameters. Namely, the features are located on some manifold of a high-dimensional space. There are numerous methods proposed for manifold embedding, including linear projections (e.g. Principle Component Analysis) and non-linear methods such as the local method Diffusion Map and the global method ISOMAP [82]. Both *Fiedler Embedding* [14] and *Diffusion Map* [102] employed in our work can be categorized into non-linear local embedding methods.

## 1.2 Proposed Work and Contributions

In this section, we briefly review the motivation and basic idea of the approaches developed in our work. Before the detailed discussion, we would like to summarize our major contributions in this dissertation into the following five points:

- Automatically discovering the optimal number of semantic visual features (*visual words* clusters) for action recognition by maximizing the mutual information between actions and the *visual words* [53].
- Simultaneously clustering images and *visual words* to extract semantic features for static scene classification using information-theoretic co-clustering approach [52].
- Developing the Spatial Correlogram Match Kernel (SCMK) that is able to capture the spatial correlation between features to make up for the absence of spatial distribution of features in BOVW [52] [53].
- Automatically exploiting the semantic relationships between heterogenous visual entities (i.e., images, videos, and different types of features) by embedding all the entities

into the common low-dimensional semantic space using *Fiedler Embedding*. The discovered relationships result in performance improvement on visual recognition [55].

- Systematically developing a framework to learn a compact yet discriminative semantic vocabulary for visual learning using Diffusion Map, which is able to capture the intrinsic geometric structure of feature space at multiple scales. [56]

### 1.2.1 Action recognition via maximization of mutual information

The first approach that we developed is to automatically learn a compact yet discriminative appearance-based human action model. It starts by treating a video sequence as a bag of features (i.e., cuboids). The cuboid is detected at the location where the local maximal response value is obtained by applying an one-dimensional Gabor filter in time direction on the video. The collection of cuboids is further quantized into the so-called *video words* by a clustering algorithm (e.g., $k$-means). Then a video is represented by a histogram of the *video words*. This is the procedure of the well-known BOVW approach.

Beyond BOVW, our proposed approach is able to automatically discover the optimal number of *video word* clusters by the Maximization of Mutual Information (MMI). The goal is to obtain a compact video representation, but we also want to keep the discriminative capability of the representation. In other words, to find the optimal number of *video word* clusters is to obtain a good tradeoff between the compactness and the discriminative capability. If we treat the *visual words* and videos as two random variables $\mathbf{X}$ and $\mathbf{Y}$, the Mutual Information (MI) between them is the measurement of how much information of one variable is contained in another one. To maintain the discriminative capability of the visual vocabulary is to preserve the MI between the *video words* and action videos. In practice, grouping the *video words* into clusters will surely decrease the mutual information. Nevertheless, we can maximize the final mutual information between *video words* and videos. We adopt the

Information Bottleneck [92] [91] approach to implement it, which greedily merges pairwise *video words* or *video words* clusters at each step.

Aside from finding optimal numbers of *video words* clusters, MMI clustering also endows semantic meaning with each cluster. Unlike the traditional $k$-means algorithm which topically clusters cuboids based on their appearance similarity, the MMI clustering can group features that are highly correlated to each other. This group of features may have a diverse appearance, but they may be related to the same concept. For instance, one particular cluster may contain the cuboids related to the "raising the hands" motion in different actions. These cuboids may be varied in appearance, but they have high co-occurrence. This means that separating them into different clusters does not help strengthen the discriminative capability of the clusters, so we can merge them into the same cluster. This is the underlying concept of MMI clustering.

As aforementioned, the Bag-of-Features based approaches are unable to preserve the layout of the features in the spatial and/or temporal space. In order to overcome this drawback, we proposed the spatial correlogram to capture the spatial distribution of the features. The correlogram is a probability distribution which depicts the likelihood to obtain semantically similar features at some distance away. This model is somewhat invariant to translation, rotation, and scale changes.

### 1.2.2 Scene recognition using MMI co-clustering

The second approach we proposed is to utilize the MMI co-clustering to model the static scenes for recognition. Just like representing a video by bag of *video words*, we extract image patches from a scene image and describe them with the SIFT descriptor. Furthermore, the scene is modelled as a bag of *visual words*, which are quantized SIFT descriptors. As aforementioned, the *visual words* may be further clustered on the basis of images or videos in which they co-occur, just like MMI clustering. The underlying assumption is that *visual* *words* that typically appear in the same instance (image or video) are usually associated with the same concepts. Then the clustering results are used to improve the visual recognition performance.

Let us look at the "document-words" matrix whose columns are documents (i.e., images and videos in the context of this dissertation) and whose rows are *visual words*. Most existing clustering is one-side clustering, either documents (columns) clustering that is based on the *visual words* (rows) distribution or *visual words* clustering that is determined by their co-occurrence in documents. This shows a duality between documents and *visual words*. Therefore, instead of one-side clustering, our proposed approach conducts co-clustering by simultaneously grouping both *visual words* and documents to boost the quality of both clusterings. The essence of our co-clustering method is to find a specified *visual words* and documents partition which can maximize the mutual information between documents (clusters) and *visual words* (clusters).

We treat the "document-words" matrix as a joint probability distribution between two random variables. The co-clustering method is to find two mapping functions from documents (columns) to document clusters and *visual words* (rows) to *visual word* clusters. Given the number of clusters for both row and column, the optimal mapping functions are supposed to give the largest mutual information between the new clusters. The procedure is similar to the $k$-means algorithm. It starts with random partitions, and in the following stages it alternately updates the cluster centroids by measuring the probability distribution of each row or column to the so-called "centroid".

In addition, we propose to use the Spatial Correlogram Match Kernel and the Spatial Pyramid Match Kernel [107] to capture the layout of the features. The experiments were conducted extensively on two very challenging datasets (the 15 scene categories and the LSCOM dataset [77]). The results demonstrated the advantages of co-clustering.

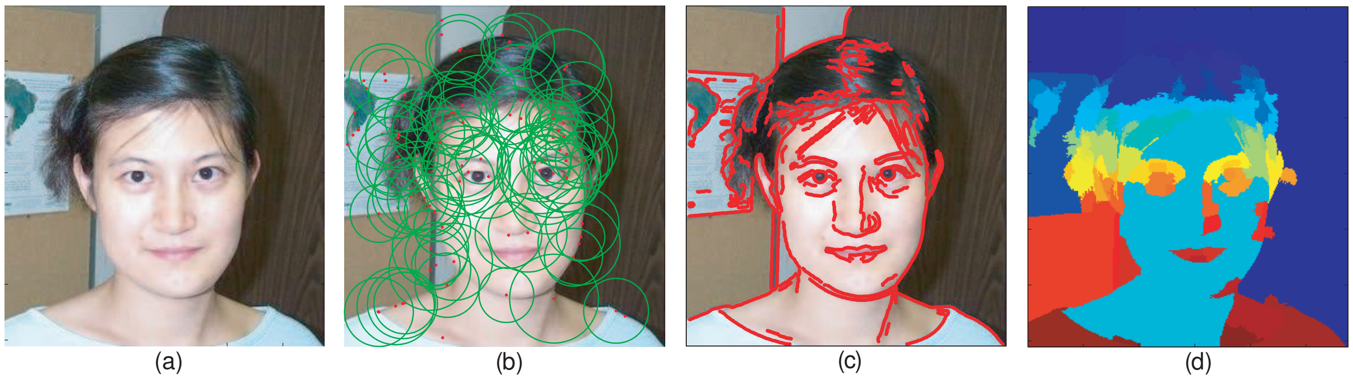


Figure 1.9: Representation of an image in terms of multiple features. (a) The original image. (b) Interest points (SIFT) representing local features. (c) Contours representing shape features. (d) Segments representing region features.

### 1.2.3 Visual recognition using multiple features

Many problems in the field of computer vision require analysis of entities that convey inherently different information, but are tied together due to explicit and implicit relationships between them. Such relationships, if properly identified, can play a crucial role in solving the problem at hand. To further illustrate this point, we give examples of two such problems from the domain of object recognition. In an object recognition task, the given visual information can be considered as an mixture of multiple classes of features such as interest points, contours and region segments. As a result, the appearance of any target object in the image stems from the interaction of these classes of features in the form of homogenous relationships (i.e., among features of the same type as in region segment to region segment, contour to contour, etc.) and heterogenous relationships (i.e., among features of different types as in interest point to contour, contour to region segment, etc.). Figure 1.9 provides a visual description of this point, where the relationship between the interest points on the eye and the nose can be categorized as a homogenous relationship, while the relationship between the interest points on the eye and the contour around the eye-brow can be categorized as a heterogenous relationship. It is quite evident that a framework capable of discovering such relationships will be immensely useful for solving the object recognition task.

We propose an algorithm that embeds different entities into a common Euclidian space and thus enables us to use simple Euclidian distances for discovering these relationships. The algorithm starts by treating different classes of entities as nodes in a graph where explicit relationships between entities are encoded by edges between nodes. The problem is then recast as a graph embedding problem where the entire graph is embedded into *one* $k$-dimensional space, such that the nodes which have a stronger relationship between them are closer to each other. This is achieved through *Fiedler embedding*, which is an algebraic method that explicitly optimizes the closeness criteria. Once all classes of entities are embedded into a common $k$-dimensional space, relationships between entities are discovered by using simple Euclidian distances. With the new space containing heterogenous entities, we are able to visualize the semantic connection amongst them, which will be propitious to visual understanding. What is more, all the discovered explicit and implicit relationships in turn can help improve the visual recognition performance. Therefore, our proposed approach seamlessly integrates visual recognition and relationship mining into one common framework. We applied it to object recognition and action recognition. The results demonstrate that fusion of multiple features helps in achieving improved performance.

### 1.2.4 Learning semantic visual vocabularies using diffusion distance

As aforementioned, *Fiedler Embedding* is capable of acquiring the semantic relationships amongst varied entities by embedding the entity graph into one *common* $k$-dimensional space via minimizing the geometric distance amongst the entities. However, it does not provide an explicit metric to measure the relative distance between any pair of features. Therefore, we propose the "diffusion distance" to learn a semantic visual vocabulary. The diffusion distance reflects the connectivity of the feature points (geometric structure between the points), to measure the semantic distance between two feature points when constructing a compact semantic vocabulary. The diffusion distance is derived from diffusion maps (DM) [102], which embeds the manifold points into a lower-dimensional space while preserving the intrinsic local geometric data structure.

The diffusion process begins by organizing the data points into a weighted graph (where the weight between two feature points is the feature similarity), which is a good way to represent the complex relationships between the feature points. Once we normalize the weight matrix and also make it symmetric and positive, we can further interpret the pairwise similarities as edge flows in a Markov random walk on the graph. In this case, the similarity is analogous to the transition probability on the edge. Then, utilizing the spectral analysis on the Markov matrix of the graph, we can find the dominant $k$ eigenvectors as the coordinates of the embedding space and map the feature points to a low-dimensional space while preserving their local geometric structures. In addition, by adjusting the time of the Markov chain, DM can be used to employ multi-scale analysis on the data. This multi-scale analysis is similar to the Pyramid Match Kernel (PMK) [70], which performs matching under different resolutions of the feature space. If we consider the embedding process as clustering, DM embeds semantically similar features into the same cluster (i.e. some concept). The size of the cluster or the range of the concept is defined by the diffusion time. A larger diffusion time corresponds to a bigger cluster, which means a larger range of concept. For instance, “sport” is on a larger scale than “baseball” and “football”, and “baseball” is on a larger scale than “team”. With the multi-scale data analysis, we can match the data under different scales.

## 1.3 Organization of the Thesis

The rest of this dissertation is organized as follows. Chapter 2 contains the literature review on visual recognition and bag-of-visual-words approaches. In Chapter 3, we present MMI clustering to find the optimal number of *video words* clusters for action recognition. Chapter 4 presents our framework for scene recognition using MMI co-clustering. Chapters 5 and 6 describe two *projection* based approaches: *Fiedler Embedding* and diffusion maps for visual recognition separately. Finally, Chapter 7 describes our future research plan.

# CHAPTER 2: LITERATURE REVIEW

Visual recognition has been studied for many decades, starting with object recognition. Early object recognition research focused on recognizing an instance of an object using 3-D geometric information, such as range data and 3-D CAD models. Since the 1980's, researchers have concentrated on 2-D image information for recognition. In the beginning, for the sake of simplicity, they focused on images with uniform background, but in the 1990's, they started to use natural images with varied backgrounds. Meanwhile, the search of natural images and scene recognition were also gaining more attention. On the other hand, inspired by Johansson's moving-light displays (MLD) [38] experiments, which demonstrate the importance of global motion for action perception, most early action recognition approaches are also model-based (i.e., the configuration of human body via markers, rectangular patches, sticks, etc.). This is reminiscent of the earlier object recognition. In fact, we can notice that both object or scene recognition and action recognition have many common characteristics in modelling and recognition methods. In this chapter, we focus on the review of recent work on object and scene recognition in Section 2.1 and action recognition in Section 2.2. Since our methods for visual modelling and recognition are based on *bag of features*, we address the related work in Section 2.3. Also, learning semantic visual features is addressed in detail.

## 2.1 Object and Scene Recognition

Visual representation is of fundamental importance for both scene and object recognition. The recognition approaches are varied according to the specific representation. Below we review some well-known object and scene modelling methods. A scene can be treated as one single object or a group of objects. In general, the modelling and recognition methods for scenes and objects are highly correlated, hence, in the following, we put them together for discussion.

### 2.1.1 Geometry-Based Models

Due to the limitation of the computation capability of machines, many early works assumed simple scenes presented with stable illumination and uniform backgrounds. With reliable edge contours extracted from objects, people can match them to 3-D models constructed by Computer-Aided Design (CAD) techniques. For instance, in the 1970's and 80's, recognizing objects from range data was popular [61] [98] [117]. Unlike the intensity images, range data can provide the depth information of the scene. This make it easy to obtain reliable object contours and regions. Then the research advanced to working on intensity values of images. However, extracting reliable object boundaries from images with complex backgrounds after many years research is still infeasible. Therefore, earlier work generally focused on scenes with uniform background.

Recognition is conducted by projecting the 3-D object models onto the image plane and searching for the best match between the 2-D projections and the unknown object image. We can coarsely group the recognition techniques into two categories: (1) Alignment techniques and (2) Hash indexing techniques. Alignment techniques: have two major steps. First is the hypothesis phase where a correspondence is found between a pose of the 3-D object model and image points or line segments. In verification, the model is projected onto the image, and all of the evidence is used to make a judgement. The representative works are [24] and [25]. In order to increase the search speed, the interpretation tree which integrates geometric constraints between primitives is introduced to explore the space of all the possible correspondences [120]. Another approach to reduce the number of potential correspondence hypotheses is the RANdom SAmple Consensus (RANSAC) algorithm proposed by Fischler and Bolles [81]. The basic idea is to compute the aligning transformation from a minimal set of randomly sampled correspondences. The degree of its consensus with other correspondences is used to measure how good one transformation is . The correct one should obtain a large support from other correct correspondences, while an incorrect one may only have a small number of supported correspondences. Hash indexing techniques: Hash indexing was well studied in the field of document indexing. It adopts a hash function to directly localize an item in the space (i.e., in terms of hash table). The very similar idea also was used by Lamdan and Wolfson [40] [124] to index the geometric features that are extracted from the template images. Given an unknown image, a set of features are extracted and matched against the hash table.

Most of the geometry-based models are useful for recognizing the specific object instances since they attempt to match the image objects to 3-D object models. It makes full use of the geometric structure constraints of the object primitives, so it is able to perform 3-D object recognition. However, its success relies on two preconditions: shape is discriminative for distinguishing the specific object, and object shapes (boundaries) are available in real images. Both of them may not be easy to be satisfied in the real applications. Hence, appearance-based models were further proposed to make up it.

### 2.1.2 Appearance-based Models

Appearance is one of the critical information sources for human vision systems. Geometry-based models can describe the shape of the objects, but for some object categories, knowing the object profile is not enough to recognize them. Their interior appearance, however, may provide more distinctive information to identify them (e.g., face recognition). In addition, most real image retrieval or scene classification systems have to cope with real natural images that generally consist of complex scenes with multiple categories of objects and complicated backgrounds. In these scenarios, the statics of some low-level features (i.e., color feature, edge feature, etc.) is easier to model and more robust than geometric shapes.

We can categorize the appearance-based models into two classes: pixel-level and patch-level methods. The former focuses on the statistics of the raw features or certain filter outputs at each single pixel. A histogram of these pixel features is the most frequently used mechanism. We can treat it as a global model, but it can also handle certain occlusion, since the matching is conducted between the bins of normalized histograms, namely *histogram intersection*, which is partial matching. With the introduction of patch-level methods, partial matching has becoming more efficient and effective. The patch-level methods are more computationally efficient than the pixel-level methods, due to the fact that only a small number of interest patches are considered for recognition. In addition, thanks to many good feature detector and descriptors [71] [72], the interest patches are not only repeatable and robust, but also invariant to scale, translation, rotation, and affine transforms. Therefore, they are more effective for object and scene recognition.

**Pixel-Level methods** Appearance features include the very basic features such as pixel color, intensity, gradient. Diverse statistics can be used to model appearance, such as color moments [57] and color correlograms capturing the spatial correlation of colors [51]. The most straightforward statistical mechanism is histogram. For example, various color histograms computed from different color spaces (e.g., RGB color space or HSV color space) were proposed for image classification in the hope of capturing different human perceptual information. In addition, histogram of oriented gradients (HOG) is another simple yet widely used features for object detection and image classification [89] [97], which is similar to edge orientation histograms [57], scale-invariant feature transform descriptors [79], and shape contexts [103].

The texture feature is another type of appearance features used as an image representation. In general, texture refers to metrics calculated to quantify the perceived texture of an image. Various representations of texture have been proposed based on grey-level co-occurrence matrices, wavelet features, Gabor features. Among them various Gabor filters were used to produce different texture features [16] [66] [87] [115]. The main idea is to convolve a set of filters (i.e., Gabor filters with different combinations of orientations and scales) with an image such that each pixel has a set of filter response values. There are two popular schemas to model them. One is a histogram-based approach [66] [87] [115]. It first quantizes all the pixels into *textons*. As each pixel is represented by a vector in which each dimension contains one filter response, we cluster them into a fixed number of centers, and treat each center as a *texton*. Then, an image is represented by the histogram of the *textons*. In general, the histogram based models are global. In order to capture coarse spatial layout of the texture feature, an image is split into multiple regions (cells), and treat each cell as a super-pixel [16]. If each cell is approximated as a homogenous region, it can be described by the mean and variance of all the filter response values of the pixels in the region. The image is simply represented by a vector consisting of these mean and variance values.

The Haar-like feature is another widely used appearance feature in object recognition [22] [96]. It was originally proposed by Papageorgiou et al. [22] for object detection, where they used two rectangles; they were extended by Viola [96] to three rectangles. Its name came from its intuitive similarity with Haar wavelets. The filter values indicate characteristics of some particular area of the image, which may reflect the local texture of the image.

**Patch-Level Methods** All the diverse image representation approaches discussed above are based on different pixel properties. The pixel-level texture feature can reflect somewhat local structure information of the image (i.e., the spatial frequency of the image). However, the spatial frequency does not fully reflect the local geometric structure. This is because each fixed filter can only acquire one specific frequency and scale information. Recently, scene and object recognition obtained surprising results thanks to the image representation by local patches (regions). The basic idea is to use a patch detector to acquire interesting patches or sample patches randomly from an image, and then depict them with certain feature descriptors (i.e. SIFT [79]). The recognition is performed by directly matching patches, or matching the histogram of *visual words*.

To our best knowledge, Schemid *et al.* [20] were first to propose local gray-value invariants for image retrieval (object images). Interest points at which the signal changes in both directions are detected by the Harris interest operator [19]. As the neighborhoods of a point can be described by a set of its derivatives that are the so-called *local jet* [67], we can compute different invariants from these *local jets*. Finally, these invariants are stacked in a vector $X$ which is considered to be the feature descriptor. These descriptors are invariant to rigid displacements and rotations. In order to cope with scale changes, these descriptors are computed at multiple scales. However, it did not deal with the affine transformation. All the descriptors were indexed by hash tables. When retrieving images ( or recognizing objects ), they used voting schema to find similar images or image labels (i.e. object classification). The authors tested this approach on both the COIL data set and aerial imagery.

Along this research stream, a diversity of feature detectors were advanced in the computer vision literature. We can coarsely categorize them into one of the following groups: corners [42] [62] [108], blobs [58] [71], edges [49] [32] [116] and ridges [116]. The corner feature is the most frequently used feature in object recognition. Generally, a corner is defined as the intersection of two edges. What is more, it can be a point which has two dominant edge directions. Aside from the famous Harris corner detector, there are several other common corner detectors, such as the Moravec corner [42], Shi and Tomasi corner [62], SUSAN corner [108], etc. In order to deal with scale changes, all the detectors can be applied on multiple image scales such as the multi-scale Harris detector. Blob detector aims to retrieve the homogenous regions that are either darker or brighter than the neighboring areas in an image. Most detectors use either differential methods [71] [79] or methods based on local intensity extrema [58]. Maximally stable extremum regions (MSER) [58] is the representative technique. With all possible thresholds applied on the image, a sequence of nested contiguous regions is obtained. Then MSER is used to find the regions with an approximately stationary area under different thresholds. With the introduction of some strategies, both corner and blob detectors are invariant to translations, rotations and scales. However, the images may be subject to perspective distortions. Hence, it is necessary to obtain interest points that are more robust to perspective transformation. This can be achieved by applying affine shape adaption to the detected regions. Mikolajczyk *et al.* [71] has extensively explored this topic, and proposed the Harris-affine and Hessian-affine feature detectors.

Edges (line segments, fragments, or contours) have recently been used for object recognition [8] [63] [100] [119]. This idea is similar to the earlier stage of object recognition with 3-D CAD models. The concern is that for certain types of object (e.g. bottles), the profile or the interior contours information may be more discriminative. Unlike early object recognition, now only 2-D images are used for training and testing. There are mainly two ways to use contours. Opelt *et al.* [8] and Shotton *et al.* [63] directly applied contour matching algorithm for recognition, but also devised a shape model of the contours to make the contours more discriminative. Instead of performing basic contour matching, Ferrari *et al.* [118] [119] discovered contour networks (i.e. several spatially connected contours) as the basic units for matching. Another way to use contours is to sample points from the contours and extract a local patch around the sampled points. This mechanism was first used in [100]. The advantages of this method are that it feature detection is efficient and it captures both appearance and contour information.

**Local descriptors** Once a collection of patches are detected, next is to find an efficient way to depict the patches. Using the flattened patch intensities is the most straightforward. For example, supposing the patch size is 10 by 10, then the descriptor is a 100-dimensional numerical vector. The similarity between two patch descriptors can be measured by cross-correlation or the sum of squared differences. Instead of intensity values, we also can use the gradient values to make the descriptor invariant to illumination changes. The above mentioned *differential invariants* based descriptor is also formed by different statistical measurements of the gradient. In order to make it rotation invariant, *steer filters* [121] with Gaussian derivatives can be convolved with the local patch, which basically rotates the Gaussian derivatives at different orientations. Shape orientated descriptors such as *shape context* [103], and *spin image* [106] are also used in certain applications.

The SIFT descriptor [79] is one of the most well-known and widely used patch representations. As figure 2.1 illustrates, an image patch is split into a $4 \times 4$ spatial grid (in the figure for demonstration it is $2 \times 2$). In each cell of the grid, a histogram of gradient orientations

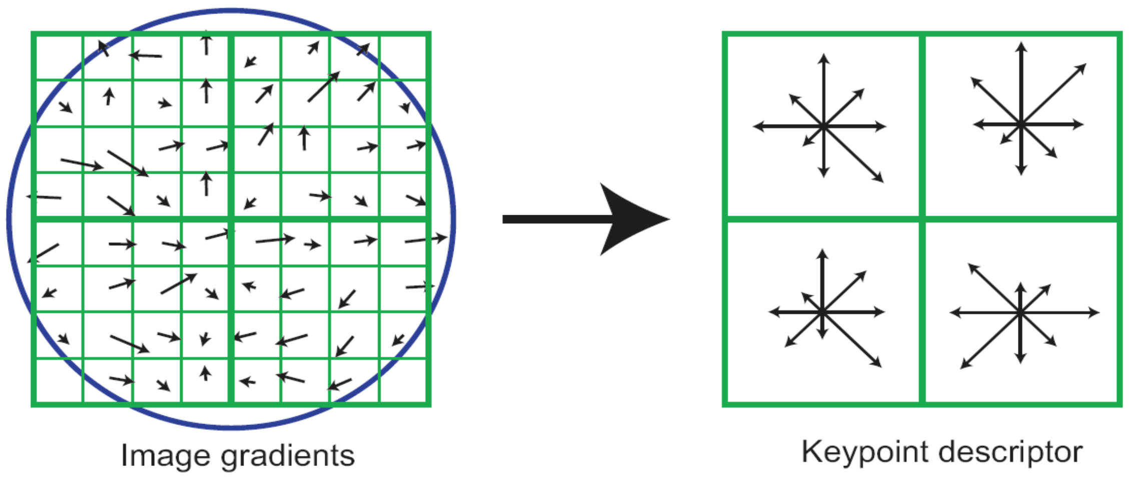


Figure 2.1: The demonstration of SIFT descriptor (this figure is taken from [79]). The left panel shows the gradients of an image patch that is divided into 2×2 subregions. The overlaid circle is the Gaussian window weighting the gradients. These gradients are accumulated into orientation histograms, as shown on the right panel. The length of the arrow represents the sum of the gradient magnitudes in the corresponding direction bin.

is computed by quantizing the orientation into 8 bins. This results in a descriptor with 128 dimensions. The orientation of the maximum peak is selected as the reference direction such that the descriptor is orientation invariant. Gradient location-orientation histogram (GLOH) [71] is the extension of the SIFT descriptor. Instead of using a grid to split the spatial space, it uses log-polar schema to arrange the space. Its design goal is to improve the robustness and distinctiveness. PCA-SIFT [75] is a training-base SIFT, which attempts to reduce the dimensionality of the vector.

We can represent an image as a *bag of features* (BoF). In the training phase, we create the database of patch features with labels from the training data. In the testing phase, the detected patches are matched to the stored patches in the training database. Then the content of the image is judged by a simple voting mechanism. This is suitable for instance recognition. For category level recognition, a histogram of the quantized patches is a very effective image representation. In fact, this is the so-called *bag of visual words*. The local patches can be quantized by any clustering algorithm ( e.g. $k$-means ). *Bag of visual words*

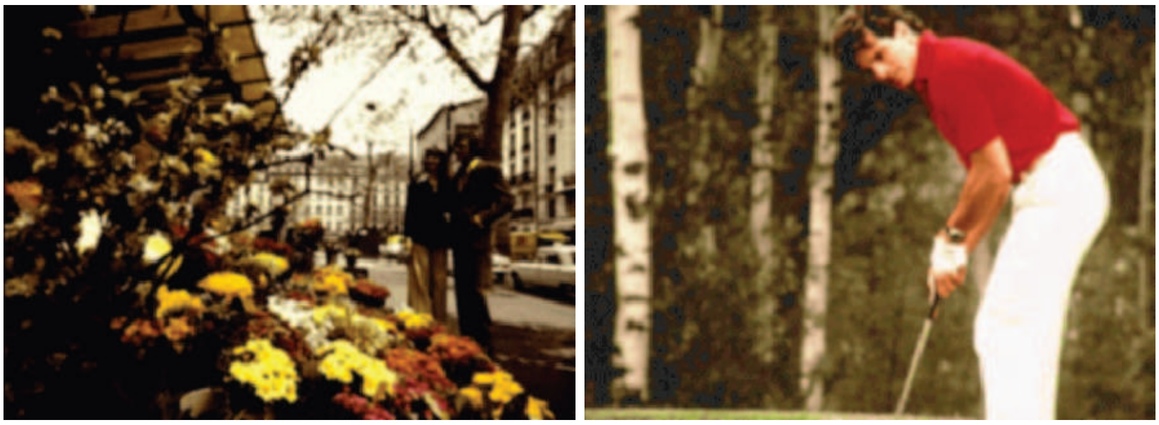


Figure 2.2: Two images having similar color histograms (the images are original in [51]).

has been extensively applied in object and scene recognition and it has shown impressive performance.

**Spatial Layout of Features** The underlying idea of most appearance-based methods (either pixel-level or patch-level appearance) is the *bag of features* (BoF) image representation. The image matching is conducted by a patch to patch, or a histogram bin to bin (i.e., matching a batch of patches). Therefore, the spatial relationships between features or spatial distribution of features are missed in BoF image representation. However, the spatial layout of features may be critical to distinguish some types of images. A very simple example is illustrated in Figure 2.2, where two totally different images have similar color histograms [51]. In fact, the colors in the left image are very dispersive, while the right images contain very concentrated colors. Thus, Huang et al. proposed *color correlogram* to make up the shortcomings of the general color histogram. *Color correlogram* essentially is the distribution of a given color as a function of the distance between two pixels. It depicts how the spatial correlation of pairs of colors changes with distance.

For object recognition, the shape information is as important as the appearance. Fergus *et al.* proposed the famous probabilistic constellations of the patches [100]. As *color correlogram* measures the pair-wise relationship between two colors at a certain distance, the *constellation* *model* aims to describe the probabilistic distribution of patch appearance and pair-wise patch correlation, which are modelled by a Gaussian. Essentially, the *constellation model* represents the relative positions between pair-wise distinctive patches. The parameters of the model are learned from the training data. Given an unknown image, the interest patches are first extracted. Next the patches are compared to the learned category models to judge whether it is generated by this category model.

One simple way to integrate coarse layout information of the features is to partition the image into relatively small grid cells [57] and then compute a local histogram for each cell. The image is described by concatenating the local histograms into a long vector. We call this the grid-based approach. Theoretically, it is a location quantization method, which integrates coarse location information into the *visual words*. Basically, it is similar to [99] in which absolute patch locations were integrated into the *probabilistic Latent Semantic Analysis* (pLSA) models [4] [95] [114]. This simple approach works very well in solving realistic problems.

Inspired by the *Pyramid Match Kernel* [70] for multiple scale matching with different granularity, Lazebnik *et al.* proposed the *Spatial Pyramid Match Kernel* (SPMK) to match images at different grid resolutions of image partitions. Rather than only considering the highest resolution of the image partition (i.e. grid-based approach), SPMK conducts image matching from the coarse level to the finest level. The match score is weighted by the pyramid level, as the matches made at various pyramid levels are of different importance. Generally, the matches made at fine grid resolutions are supposed to gain more weight. The similarity between two models are measured by histogram intersection.

Finally, we review the *Spatial Envelope* proposed by Oliva *et al.* [7] to depict the shape of the nature scene. Traditionally, we are apt to think that a scene consists of multiple objects or some elementary patterns. It is also consistent to the observations that scene recognition is a progressive reconstruction process of the input from local elements. This is a bottom-up method. In contrast, we also can achieve the same goal using a top-down approach, which attempts to capture the scene information globally. This is supported by research stating that humans can recognize a scene without knowing the detailed object information in the scene. Based on these observations, Oliva *et al.* treated a scene as an object and attempted to describe the shapes of scenes by a set of perceptual dimensions (i.e., naturalness, openness, roughness, expansion, ruggedness).

## 2.2 Action Recognition

For action recognition, we can trace back to Johansson's research in 1973 on point-light displays (PLD) attached to the moving human body [38]. From a psychophysics point of view, his research proved that humans are able to recognize actions solely from the global body motion. Inspired by this research, many techniques have been developed to model the human body by capturing the global human motion via attached markers. Just like recognizing 2-D objects using 3-D models, action recognition also has one stream to reconstruct a 3-D model of human body, and then it is projected into 2-D space to match the observation [26] [73]. More works using markers [80], rectangular patches [28] and blob models [85] have been explored to recognize actions. Using attached physical markers to capture human body motion is useful for some practical problems such as films, games or automatic control systems, but for other scenarios such as surveillance systems and video search, this is difficult because the human body parts (or joints) are required to be automatically detected.

In this section, we address the approaches for action recognition using mainly vision techniques on videos. These approaches can be classified into two categories according to the extracted features. If the extracted features are global shapes, contours, or trajectories, we refer to them as holistic approaches, otherwise, they are part-based approaches.

### 2.2.1 Holistic Approaches

The most straightforward holistic approach is to correlate the action template (i.e. video clip) to the queried video. The correlation can be computed from various spatiotemporal volume properties such as intensity, gradient, optical flow, Gabor filters, etc. Shechtman *et al.* [34] proposed to measure the degree of consistency by computing the correlation using the local intensity variance. Similarly, Efros *et al.* [5] extracted an optical flow field as a descriptor from the stabilized object spatiotemporal volume and computed the cross correlation between the model and the input optical flow descriptors.

The shape of the subjects is one of the most reliable and frequently used pieces of global information, which can be described by silhouettes and contours. Cheung *et al.* [37] used *Shape-From-Silhouette* (SFS, or *Visual Hull*) to reconstruct the 3-D model of moving dynamic articulated subjects using multiple camera views. With the reconstructed 3-D model, the acquired shape and joint information was used to recognize the human motion. Instead of reconstructing the 3-D model of the unknown action in the recognition phase, Lv and Nevatia [35] proposed to match the 2-D silhouettes of the test sequence to the 2-D projections of the selected 3-D action exemplars. These 3-D exemplars captured the key poses of each type of actions and they can be shared by different actions. Weinland *et al.* [30] proposed a very similar idea, which is illustrated in Figure 2.3. Both of them can recognize an action from an arbitrary view.

Rather than reconstructing a 3-D model using multiple cameras, we can directly integrate the silhouettes into a 3-D space-time shape volume. Unlike the reconstructed 3-D subject model at every unit time, the 3-D space-time shape volume has a time dimension. Therefore, it is able to capture the shape volume changes resulting from the pose changing. Then the action recognition problem is successfully converted into 3-D object recognition. Yilmaz and Shah used differential geometry features extracted from the surfaces of the action volumes,

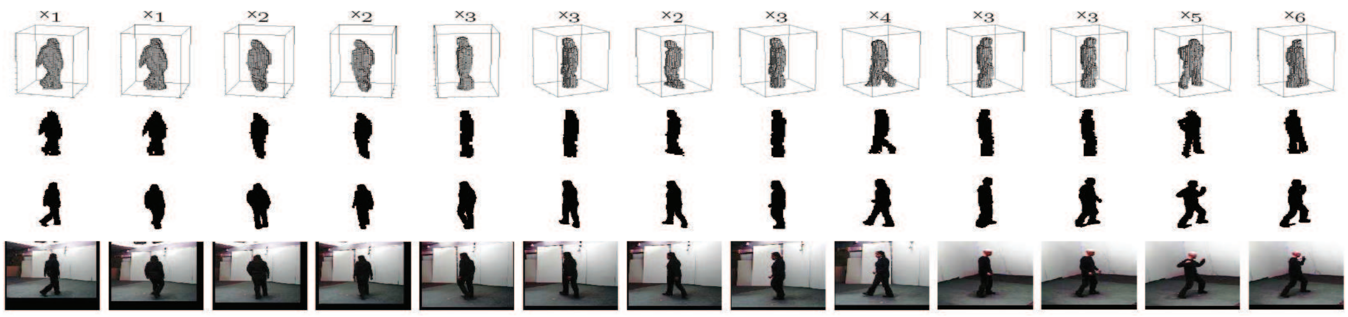


Figure 2.3: Examples showing arbitrary view action recognition (this figure is from [30]).The fourth and third row are the observed image sequences and their corresponding silhouettes. The second and first row are the matched silhouettes and their corresponding 3-D exemplars.

and achieved good performance. Blank *et al.* applied a 2-D shape analyzer to compute shape features with each frame of shape; however, the recognition is not view invariant.

Another way to integrate the silhouettes is through the use of the *motion history image* (MHI) and *motion energy images* (MEI) proposed by Bobick and Davis [3]. The basic idea is to record the moving path of motion into one single image. The time information is integrated into the image by assigning darker values to earlier motions. Figure 2.4 (A) illustrates this idea. This idea is extended to Hierarchical Motion History Images in papers by Davis [50] and by Meng *et al.* [41]. By combining both 3-D reconstruction and MHI, Weinland *et al.* [31] proposed an view independent *motion history volume* (MHV), which records the motion changes at any view over time.

It is obvious that silhouettes can provide sufficient discriminative information for action recognition when they are available. Nevertheless, its performance really depends on the results of background subtraction, which may be sensitive to illumination, color, and texture changes. Also, it is unable to handle self collusion. Since the requirements of silhouettes are too strict, this is relaxed to figure-centric schema, which only needs to detect the coarse location of the figure. With the detected figure, either optical flow features [5] or other filter-based features can be extracted from the block. For example, Schindler *et al.* [74] used linear Gabor filters to capture shape information and optical flow to depict motion information.

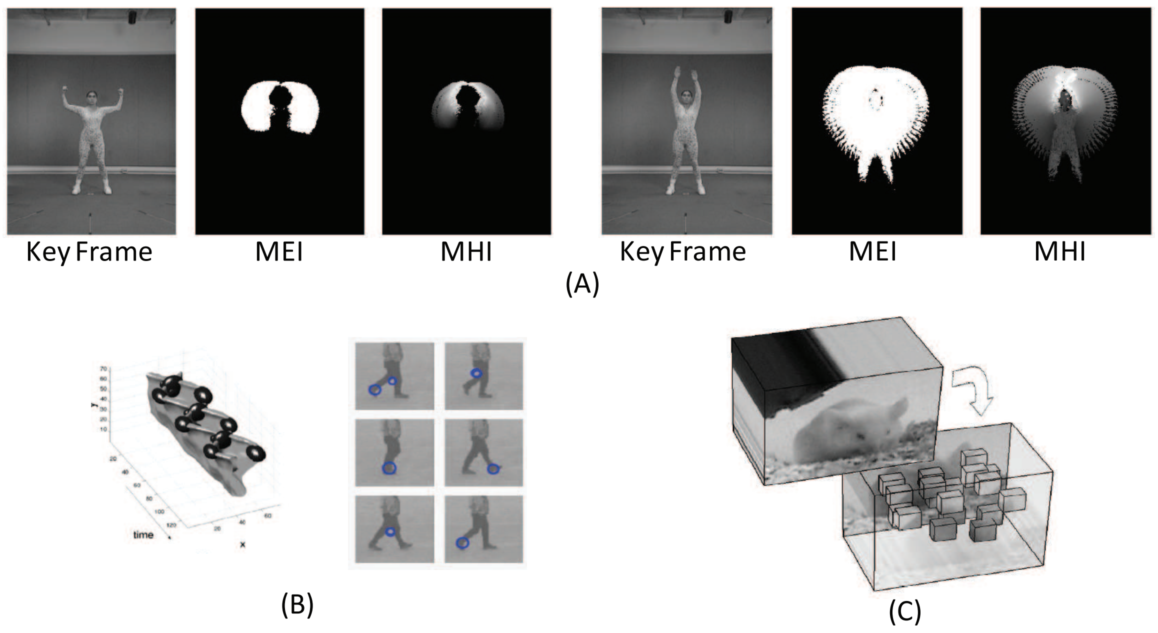


Figure 2.4: (A) Motion energy images (MEI) and motion history image (MHI)(this figure is taken from [3]); (B) Space-time interest points are detected by 3-D Harris-corner detector (this figure is taken from [46]); (C)Space-time interest points are detected by 1D Gabor detector in time direction (this figure is taken from [93]).

As discussed above, global approaches usually have to store 3-D templates or sequences of 2-D images. The recognition is achieved by various matching mechanisms like shape matching. To construct the templates, background subtraction might be a pre-condition. Yet, current background models are sensitive to camera motion and illumination changes, so the templates are not very reliable. What is more, template matching focuses on instance recognition, which makes it hard to generalize the templates for category-level recognition. This is because the model has to cope with many variances, such as deformations, view changes, and scale changes.

### 2.2.2 Part-based Approaches

Due to the limitation of holistic models in solving some practical problems, recently part-based models have received more attention. Unlike the holistic-based method, this approach extracts “bag of interesting parts”. Hence, it is possible to overcome some limitations like background subtraction and tracking. The local appearance features include local motion features and static features.

**Local motion feature** Fanti *et al.* [18] and Song *et al.* [126] proposed a triangulated graph to model the actions. Multiple features such as velocity, position, and appearance were extracted from the human body parts in a frame-by-frame manner. Ke *et al.* [123] extended the 2-D Harr-like filters, originally applied in face detection, into 3-D to capture volumetric features.

Inspired by the success of interesting patches used for object recogntion [79] [100] [71] [99], the so-called bag of *space-time interest points* [45] [93] based approaches are obtaining increasing attention. Typically, spatiotemporal interest points are first detected either by a 3-D Harris corner detector [45] in videos or 1D Gabor filters [93] in the temporal direction of the videos (Figure 2.4 (B) and (C) demonstrate some examples). Supposing a video is in $xyt$ space, detection of interest points from direction $x$, $y$, and $t$ were explored in [74]. Ning *et al.* [44] proposed 3-D Gabor filters to detect interesting points. Then various feature descriptors are extracted from the cuboids surrounding the interesting points. The descriptors can be flattened intensity, gradient and optical flow vectors [93], or a histogram of gradients and a histogram of optical flows [47].

With a bag of local motion feature descriptors, we can index them using efficient trees such as Sphere-Rectangle tree [90] and Kd-tree [68]. Then action retrieval or recognition can be done by querying each local motion feature detected from the unknown and simply voting to label the unknown action. Another popular mechanism is to represent an action as a *bag of video words* by quantizing the descriptors into video-words and computing their statistical distributions (histogram). Then, the discriminative learning model such as SVM [93] and the generative model such as pLSA [59] [111] and Hierarchical Bayesian [60]can be used to build the model.

**Local static feature** It is well known that the human vision system can recognize many types of human actions from a sequence of instantaneous postures or poses of a person in still images without motion information. Therefore, we believe the static pose in a single image can be useful for action categorization. Recently, pose recognition using local shape features such as shape contexts [60] [127], histogram of gradients of the local patches [2], appearance, and position contexts [43] [44] have obtained good results. Since a single pose only provides instantaneous information at a single instant, it is important to select the right pose in order to determine an action correctly. Instead of using a single pose, we can employ a sequence of poses, in order to make up for the lack of motion information. This is particularly useful for realistic videos where the motion features are unreliable due to unpredictable and often unintended camera motion (camera shake).

**Hybrid of static and motion features** Little work has been reported on the combination of static and motion features for action recognition in realistic videos until recently. Fanti *et al.* [18] utilized a mixture of static features (local appearance) and dynamic features (simple velocity descriptors) for action recognition. Neibles *et al.* [60] proposed a generative model to learn a hierarchical model using both static and dynamic features for action recognition, and their results verified that the hybrid features are useful. In chapter 4, we propose *Fiedler Embedding* to combine local motion features and spin image features which capture the global pose information. However, these methods may not be applicable for realistic videos due to the difficulty in acquiring good features in unconstrained videos. Instead of detecting spatiotemporal interest points, Mikolajczyk *et al.* [72] detected local static features with associated motion vectors from every single frame, and used motion vectors as a filter in recognition. Their action recognition method is akin to object recognition, and requires extra training images and object bounding boxes. Schindler *et al.* [39] combine different types of ST (spatiotemporal) features by simply concatenating the feature vectors.

## 2.3 Semantic Visual Vocabulary

In the field of computer vision, bag of features (BOF) is receiving increasing attention due to its simplicity and surprisingly good performance on object, scene and action recognition problems. The underlying idea is that a variety of statistical cues are present in images and videos, such as color or edge patterns and local structural elements [20] [29] [45] [79] [88] [93], which can be effectively used for recognition. Inspired by the success of the bag of words (BOW) approach in text categorization [23] [114], computer vision re-searchers have recently discovered the connection between local patches in images/videos and words in documents. In the BOW text representation, a document is represented as a histogram of words. In order to employ the BOW to represent an image or video, we need to quantize the local patches into visual words. The $k$-means algorithm is commonly used to construct an initial visual vocabulary due to its simplicity. However, it has two major drawbacks. The first being that the quality of the visual vocabulary is sensitive to the vocabulary size [87]. In general, thousands of visual words are used to obtain better performance on a relatively large dataset. But this vocabulary may contain a large amount of information redundancy. On the other hand, since the clustering criterion is only based on the appearance similarity, $k$-means is unable to capture the semantic relation between the features. This semantic relationship is useful for image and video understanding.

Several attempts have been made to bring the semantic information into visual vocabularies. We can categorize these attempts into two major classes: the supervised and unsupervised approaches. The supervised approaches use either local patch annotation [65] or image/video annotation [13] [36] [66] [78] [122] to guide the construction of a semantic visual vocabulary. Specifically, Vogel *et al.* [65] construct a semantic vocabulary by manually associating the local patches to certain semantic concepts such as “stone”, “sky”, “grass”, etc. The obvious drawback is that this approach is infeasible due to the large amount of manual labor required. Yang *et al.* [78] proposed unifying the vocabulary construction with classifier training, and then encoding an image by a sequence of visual bits that constitute the semantic vocabulary. Another interesting work utilizes randomized clustering forests to train a visual semantic vocabulary [36]. The classification trees are built first, but instead of using them for classification, the authors assign a visual word label to each leaf, which is how a semantic visual vocabulary is constructed. In addition, several other works [13] [66] [105] [122] use mutual information (MI) between the features and class labels to create the semantic vocabulary from an initial and relatively larger vocabulary quantized by the $k$-means algorithm (Hereafter, we will call the visual words in the initial vocabulary mid-level features in order to distinguish them from the low level raw features and high level semantic vocabulary features).

Some unsupervised approaches [4] [64] [76] [95] [111] were inspired by the success of the textual topic models in text categorization, such as pLSA [114] and LDA [23]. Those models represent an image or video as the mixture distribution of hidden topics that can essentially be a semantic visual vocabulary. There is a soft mapping between the hidden topics and the mid-level features. In chapter 3, we proposed to use maximization of mutual information (MMI) to obtain the optimal size of the visual semantic vocabulary for action recognition. We observe that semantically similar features generally have a higher co-occurrence value in the data set. This is the intrinsic reason that both the topic and MMI model can be successfully used to construct a semantic vocabulary.

Both the supervised and unsupervised approaches obtained good performance on object, scene and action recognition. This is because the semantic visual vocabulary can capture not only the appearance similarity but also the semantic correlation between the mid-level features. We can explain this point clearly using an example in text categorization. For instance, "pitching", "score" and "team" can be correlated to each other by "baseball", while "biker", "wheel" and "ride" may be correlated to each other by "motorcycle". Hence, we conjecture that the mid-level features produced by similar sources are apt to lie on dynamic feature manifolds. In other words, there exist strong correlations between each dimension of the features, which means the features may have a limited number of degrees of freedom. Abundant dimension information is redundant.

However, very few attempts have been made to explicitly preserve the manifold geometry of the feature space when constructing the semantic visual vocabulary. We propose to use Diffusion Maps [102] to capture the manifold geometric structure of the features in the process of embedding. In fact, DM is one of the techniques used for manifold dimension reduction like PCA, ISOMAP [82], Laplacian Eigenmaps [83], etc. In many applications, the distances between feature points that are far apart are meaningless, so preserving the local structure is sufficient for the embedding. Unlike DM, PCA and ISOMAP are global techniques that do not preserve local geometric information of the feature space. In addition, PCA is unable to handle nonlinear manifold data points. Since the diffusion distance derived from DM uses all the paths between two points to compute the distance, it is more robust to noise than the geodesic distance (shortest path distance) used by ISOMAP. DM is very similar to Eigenmaps-based approaches. However, since the embedding coordinates are weighted eigenvectors of the graph Laplacian, DM has an explicit distance measure induced by a nonlinear embedding in the Euclidean space. Eigenmaps representation does not have any explicit metric in the embedding space. Additionally, DM can employ multi-scale analysis on the feature points by defining different time values of the random walk.

# CHAPTER 3: LEARNING OPTIMAL NUMBER OF VISUAL WORDS FOR ACTION RECOGNITION

## 3.1 Introduction

In this chapter, we propose an approach to automatically discover the optimal number of *video-words* clusters ($VWC$s) using the Maximization of Mutual Information (MMI) principle in an **unsupervised** manner. Our goal is to find compact and yet discriminative $VWC$s by grouping the redundant *video-words*. The benefits of a compact representation is twofold: more effective and efficient classification due to lower dimension, and effectively capturing the spatiotemporal correlation of the $VWC$s. Specifically, we maximize the Mutual Information (MI) when merging two $VWC$s, which is **unsupervised**. The maximization of Mutual Information (MMI) has several available mechanisms [48] [91] [92]; we adopt the Information Bottleneck [91]. The MMI based clustering has been successfully used for word clustering where the words are grouped into semantic concept clusters (e.g. "pitching", "score","teams" etc. can be clustered into "baseball" concept, and "biker", "wheel", and "ride" may be clustered into "motorcycle" concept). This is effective due to the fact that words related to a particular concept have higher co-occurrence in documents. Similarly, each cluster of *video-words* achieved by MMI method tends to correspond to a group of semantically related *video words*. For instance, one particular cluster may contain the cuboids related to the "raising the hands" motion in different actions.

The $VWC$s are somewhat analogous to *hidden topics* in pLSA. However, there are significant differences between them. First of all, pLSA is a generative model, which employs hidden variables, while MMI clustering does not use hidden variables. Secondly, pLSA assumes conditional independence (i.e., given the latent variable, the document and word are independent), which is not required in MMI clustering. Besides, pLSA is used as a clustering method in [59]. The number of topics (clusters) normally is set to be the number of categories. Wong *et al.* [111] set the number of topics to be three times the number of categories. Nevertheless, our approach aims to automatically discover the optimal number of $VWC$s, such that the action can be represented by a compact yet discriminative model.

Although BOV has achieved very good performance, it ignores any spatial or temporal information between the *video-words*. The cuboids representing motion of the parts of a human body have a strong correlation to each other due to the fact that they belong to the same body. Figure 3.4 shows some examples of spatial distribution of the cuboids. Wong *et al.* [111] extended some successful shape models from object recognition to their 3D action recognition. However, they still ignore the temporal information. [109] introduces a temporal subsequence mining method in order to capture the temporal correlation of the *video-words*. However, the performance is a little worse than their SVM baseline. In our work, we apply the correlogram, which has been successively applied for image and scene classification [110]. The modified correlogram is able to somewhat cope with the translation, rotation, and scale problem. Additionally, we explore the spatiotemporal pyramid approach in order to capture both spatial and temporal information.

The major steps of the training phase in our framework are described in Table 3.1. The videos are feed into the system, and an appropriate number of cuboids (3D interest points) are extracted from each video. K-means algorithm is applied to get the large number of *video-words*. Then MMI clustering automatically discovers a compact representation from the initial codebook of *video-words* and efficiently captures the correlation. Furthermore, we use spatial correlogram and spatiotemporal pyramid models for capturing structural information. Finally, we use an SVM as a classifier to train and test these models.

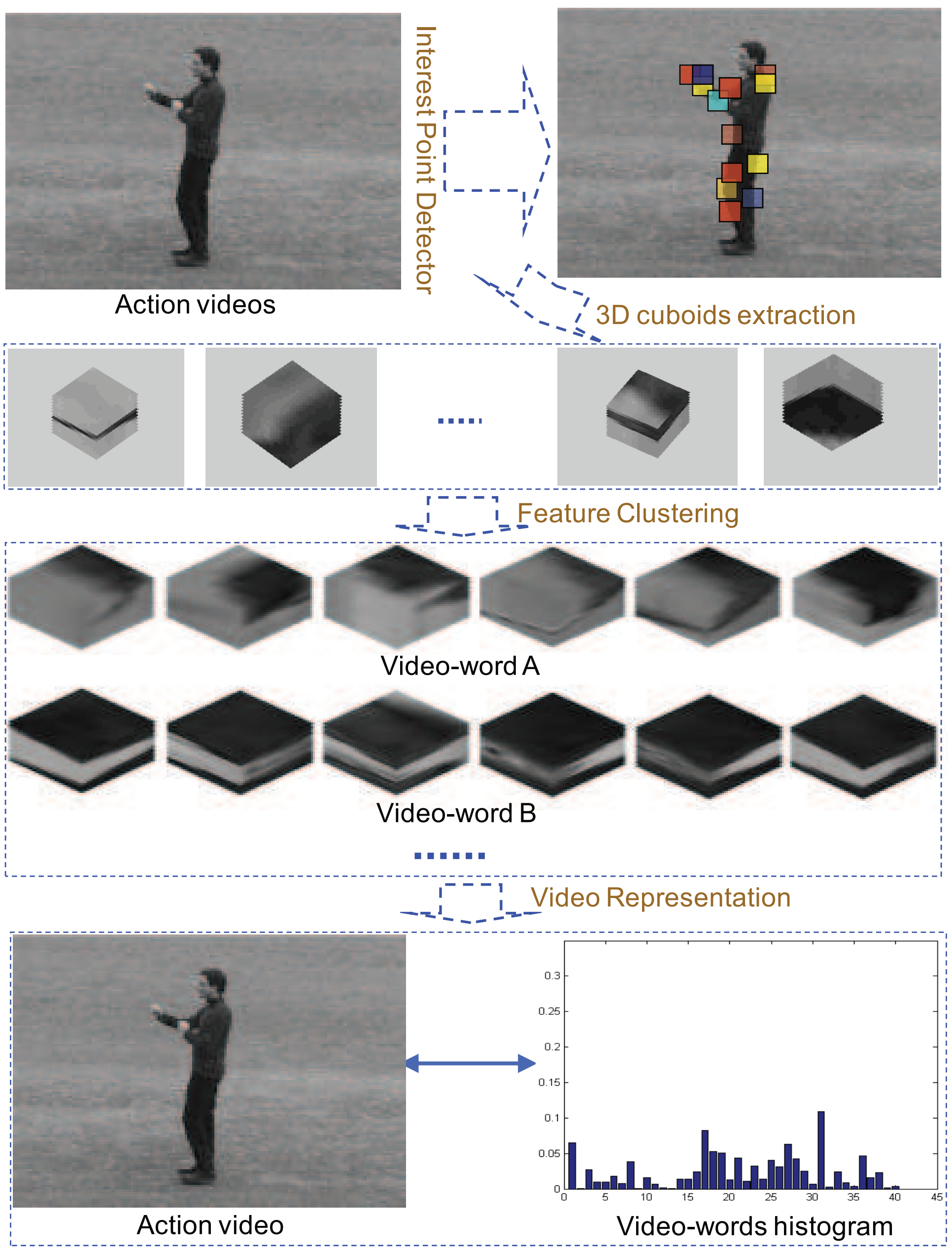


Figure 3.1: Illustration of the procedure of representing an action as a bag of video-words (histogram of bag of video-words).

**Objective:** Action recognition using the learnt optimal number of $VWC$s and their structural information.

- **Extracting Cuboids**. Apply separate linear filters in spatial and temporal directions, and select the local maxima.
- **Learning Codebook**. Quantize the cuboids into $N$ *video-words* using $k$-mean algorithm based on the appearance similarity.
- **Compressing Codebook**. Apply MMI clustering to find the optimal number of *video-word* clusters.
- **Capturing Structural Information**. Extract translation, rotation, and scale invariant spatial correlogram and spatial temporal pyramid.
- **Training SVM models**. Training using feature vectors extracted as described above.

Table 3.1: Major steps for the training phase of our framework

## 3.2 Bag of Video-words Model

Figure 3.1 illustrates the procedure of bag of video-word modelling. The procedure is very straightforward. First, detect interest points in the action video. Two widely used feature detectors are 3D Harris corner detector [45] and 1D Gabor filter (convoluting with time) [93]. Second, extract cuboids around each detected feature point, and then represent each one as a descriptor. Next, construct the vocabulary using $k$-means clustering. This is also called feature quantization. Finally, an action video is represented by the histogram of the video-words. With the histograms, action recognition or indexing can be conducted.

### 3.2.1 Feature Detection and Representation

As we can see from Figure 3.1, an action video is treated as a bag of features. In order to acquire discriminative features, an effective feature detector is a must. Many feature detectors are available for 2-D images; very few feature detectors, however, were designed for

3D video. Inspired by Harris-corner detector for spatial domain [19], Laptev and Lindeberg [46] extend it to the space-time domain to discover 3-D corners. In practice, it can effectively find the intensity changes in both space and time domains. In general, it generates very sparse features. However, there are two points that should get our attention. One is that rare features may affect the recognition performance as observed by Lowe [79]. Another one is that there is no argument to prove whether the 3D corners are discriminative for recognition. Therefore, we adopt another spatiotemporal interest points detector proposed by Dollar [93]. This detector produces dense feature points, and performs better on action recognition tasks [59, 93, 111].

Instead of using a 3D filter on the spatiotemporal domain, it applies two separate linear filters respectively to spatial and temporal dimensions. A response function can be represented as follows:

$$R = (I(x,y,t) * g_\sigma(x,y) * h_{ev}(t))^2 + (I(x,y,t) * g_\sigma(x,y) * h_{od}(t))^2, \qquad \text{(Eq. 3.1)}$$

where $g_\sigma(x,y)$ is the spatial Gaussian filter with kernel $\sigma$, which is the spatial smoothing kernel, and $h_{ev}$ and $h_{od}$ are a quadrature pair of 1D Gabor filters applied along the time dimension. They are defined as,

$$\begin{aligned} h_{ev}(t;\tau,\omega) &= -cos(2\pi t\omega)e^{-t^2/\tau^2} \\ h_{od}(t;\tau,\omega) &= -sin(2\pi t\omega)e^{-t^2/\tau^2}, \end{aligned}$$

where set $\omega = 4/\tau$. They give strong responses for temporal intensity changes. The interest points are detected at locations where response is locally maximal. This detector can detect features not only for periodic actions such as “running” and “walking”, but also for any other motions with spatially distinguishing image characteristics. Hence, the detected features may include space-time corners.

Around each interest point, a cuboid is extracted with spatiotemporal size covering the range within which the feature is detected. To measure the similarity between two cuboids, a descriptor is needed, such that the similarity can be measured by the Euclidean distance between the descriptors. Dollar et al. [93] propose several types of feature descriptors, such as the spatial and temporal gradients or their histograms, windowed optical flows or their histograms, etc. The dimensionality of the descriptor is reduced by PCA. In our work, the flattened gradient descriptor is used.

### 3.2.2 Action Descriptor

With the feature descriptors extracted from the action videos, the similarity between two action videos can be estimated by matching their descriptors. However, this must be computationally expensive and sensitive to occlusion. Instead of direct matching, we can quantize each feature descriptor $s, s \in R^d$ into the predefined discrete *video-word* $v$ in the vocabulary $\mathcal{V}$ by nearest neighbor. The predefined vocabulary is learned from a collection of training cuboids extracted from a subset of training videos by clustering like $k$-means algorithm. All the cuboids that are quantized into the same *video-word* are deemed to be matched. Hence, this is the pre-computed coarse matching. Now, an action video can be represented by the histogram of the *video-words* in it. This is an analogy to the bag of words idea in the area of document analysis [23] [114]. To compare two action videos, we can use Euclidean distance, KL-divergence, or Histogram Intersection to measure the dissimilarity or similarity.

However, there are still three unanswered questions for the bag of *video-words* problem. First, how large of a vocabulary is appropriate for recognition? This is the problem of vocabulary resolution. In the following section, we can answer this question to some extent. Second, can *video-words* be semantically meaningful? The MMI clustering can make it happened. Finally, how do we integrate structure information into the model? This is also explored in the following sections.

## 3.3 Clustering of Video-words by MMI

Consider two discrete jointly distributed random variables $X$ and $Y$, where $X \in \mathcal{X} = \{x_1, x_2, ..., x_n\}$ and $Y \in \mathcal{Y} = \{y_1, y_2, ..., y_m\}$. In our work, $\mathcal{X}$ represents a set of *video-words*, and $\mathcal{Y}$ is a set of action videos. We can simply build a histogram of the *video-words* to model each action video $y_i$. Once we normalize the histograms, the similarity of two action samples can be measured by their distance between two conditional distributions $p(x|y)$. However, the size of $\mathcal{X}$ is difficult to choose. If the vocabulary size is too small, it may cause over-clustering with higher intra-class distortion. Therefore, it is common to choose an appropriately large value for the vocabulary. But that may cause a sparse histogram and introduce noise for recognition. So, we seek to find a more discriminative and yet compact representation of $X$, say $\hat{X}$ which groups the *video-words* with higher co-occurrence relationships together, and also preserves the information about $Y$. Our criteria for $\hat{X}$ is to maximize the mutual information $I(\hat{X}; Y)$ under the constraint of lower value of the mutual information $I(\hat{X}; X)$. This is the idea of Information Bottleneck [92].

#### 3.3.0.1 Mutual Information

Given two discrete random variables $X$ and $Y$, the Mutual Information (MI) between them is defined as:

$$I(X; Y) = \sum_{y \in Y, x \in X} p(x, y) log \frac{p(x, y)}{p(x)p(y)}, \qquad \text{(Eq. 3.2)}$$

where $p(x, y)$ is the joint distribution of $X$ and $Y$, $p(x)$ and $p(y)$ are probability distributions of $X$ and $Y$ respectively. MI tells how much information of variable $X$ is contained in variable $Y$. Using Kullback-Leibler divergence, it also can be expressed as:

$$I(X, Y) = D_{KL}(p(x, y) \parallel p(x)p(y)), \qquad \text{(Eq. 3.3)}$$

where $D_{KL}$ computes the distance between two distributions. In the context of this paper, $X$ and $Y$ respectively represent *video-words* and actions.

#### 3.3.0.2 MMI clustering Algorithm

Our goal is to find an optimal mapping of the *video-words* $X$, say C(X) into a more compressed representation $\hat{X}$ such that the MI between $\hat{X}$ and $Y$, say $I(\hat{X};Y)$, is as high as possible, given the constraint on the MI between $X$ and $\hat{X}$, say $I(\hat{X};X)$. $I(\hat{X};X)$ signifies how compact the new representation $\hat{X}$ is. Obviously, a lower value gives a more compact representation, and the most compact representation will correspond to the merging of all *video-words* into one single cluster. However, that representation may not be discriminative, because it does not give any information regarding $Y$ from $\hat{X}$. Therefore, we also need to keep a higher value of $I(\hat{X};Y)$, which gives the discrimination of the new representation or quality of the clustering. There is a tradeoff between the compactness and discrimination. Given the mapping $p(\hat{x}|x)$, this problem can be mathematically expressed as:

$$max(I(\hat{X};Y) - \lambda^{-1} I(\hat{X};X)), \qquad \text{(Eq. 3.4)}$$

where $\lambda^{-1}$ is the Lagrange multiplier. The solution of formula Eq. 3.4 gives three self-consistent equations on $p(\hat{x}|x)$, $p(y|\hat{x})$, and $p(\hat{x})$. The details of the solution of this minimization problem are given in [92]. When $\lambda = 0$, the solution of Eq. 3.4 assigns all $x$ to one cluster, and when $\lambda \to \infty$, it gives a solution for hard clustering as follows: $p(\hat{x}|x) = 1$ if $x \in \hat{x}$, otherwise $p(\hat{x}|x) = 0$; $p(y|\hat{x}) = \frac{1}{p(\hat{x})} \sum_{i=1}^{|\hat{x}|} p(x_i, y)$ and $p(\hat{x}) = \sum_{i=1}^{|\hat{x}|} p(x_i)$. If one specified clustering $C(X)$ always has $I(C(X);Y) \geq I(C'(X);Y)$ where $C'(X)$ is an arbitrary mapping, $C(X)$ is one of the optimal solutions.

This problem can be solved by a greedy algorithm based on a bottom-up pair-wise merging procedure [91]. The algorithm starts with a trivial partition, where each element of $X$ is a singleton cluster. In order to keep $I(\hat{X};Y)$ as high as possible, at each step we greedily merge two components into one, which has minimal loss of mutual information $I(\hat{X};Y)$. Let $\hat{x}_1$ and $\hat{x}_2$ be two candidate clusters to be merged; the cost of this merge is defined as the loss of MI due to the merge, which is expressed as:

$$\Delta I(\hat{x}_1, \hat{x}_2) = I(\hat{X}_{bef};Y) - I(\hat{X}_{aft};X), \qquad \text{(Eq. 3.5)}$$

where $I(\hat{X}_{bef};Y)$ and $I(\hat{X}_{aft};Y)$ denote the MI before and after the merging step respectively. $x$ is a *video-word* which is represented by a normalized vector with its frequency in the training videos; specifically, it is a vector of $p(y|x)$ with $y \in Y$. Similarly, every cluster has a "prototype", say $p(y|\hat{x})$. Assume $\hat{x}_1$ and $\hat{x}_2$ are merged into $\hat{x}^*$; the new "prototype" is updated as:

$$p(y|\hat{x}^*) = \frac{p(\hat{x}_1)}{p(\hat{x}^*)}p(y|\hat{x}_1) + \frac{p(\hat{x}_2)}{p(\hat{x}^*)}p(y|\hat{x}_2), \qquad \text{(Eq. 3.6)}$$

where $p(\hat{x}^*) = p(\hat{x}_1) + p(\hat{x}_2)$. The prototype here is like the centroid of a cluster. Now the loss of MI can be derived from Eq. 3.5 and Eq. 3.6 as:

$$\begin{aligned}
\Delta I(\hat{x}_1, \hat{x}_2) &= I(\hat{X}_{bef};Y) - \sum_y p(\hat{x}^*)p(y|\hat{x}^*)log\frac{p(y|\hat{x}^*)}{p(y)} \\
&= I(\hat{X}_{bef};Y) - \sum_y \log\frac{p(y|\hat{x}^*)}{p(y)} \sum_{i=1,2} p(\hat{x}_i) \sum_{i=1,2} \frac{p(\hat{x}_i)}{p(\hat{x}^*)}p(y|\hat{x}_i) \\
&= I(\hat{X}_{bef};Y) - \sum_y log\frac{p(y|\hat{x}^*)}{p(y)} \sum_{j=1,2}(\frac{p(\hat{x}_i)}{p(\hat{x}^*)} \sum_{i=1,2} p(\hat{x}_i)) \qquad \text{(Eq. 3.7)} \\
&= \sum_{y,i=1,2} (p(\hat{x}_i)p(y|\hat{x}_i)log\frac{p(y|\hat{x}_i)}{p(y)} - p(\hat{x}_i)p(y|\hat{x}_i)log\frac{p(y|\hat{x}^*)}{p(y)}) \\
&= \sum_{y,i=1,2} p(\hat{x}_i)D_{KL}(p(y|\hat{x}_i)||p(y|\hat{x}^*)).
\end{aligned}$$

As we see, $\Delta I(\hat{x}_1, \hat{x}_2)$ is the weighted distance of two original "prototypes" to the merged "prototype". Also, we can consider the loss of MI due to the merging of clusters $\hat{x}_1$ and $\hat{x}_2$ as the distance between $\hat{x}_1$ and $\hat{x}_2$. At each step, we greedily merge the most closest ones. The algorithm is summarized as follows:

(i) Initiate $C(X) \equiv X$, which means regard each point as a singleton cluster.

(ii) At each step, compute the distance (actually $\Delta I(\hat{x}_1, \hat{x}_2)$) between each pair of elements using formula Eq. 3.7.

(iii) Pick the pair which gives the minimum loss of MI $\Delta I(\hat{x}_1, \hat{x}_2)$.

(iv) Continue the merging operation until the loss of MI $\Delta I(\hat{x}_1, \hat{x}_2)$ is larger than the predefined threshold $\epsilon$ or number of clusters.

In summary, the motivation to learn the optimal number of clusters of *video-words* is twofold. The compact features with lower dimensionality are efficient and effective to learn. Besides, compact features are easier to encode with spatiotemporal structure information. Here, we apply two steps to achieve this. We first use $k$-means algorithm to cluster the cuboids into *video-words*. Since the criterion for $k$-means is based on appearance similarity, cuboids belonging to one *video-word* are visually similar. Further, we group the *video-words* into some more compact but discriminative clusters via MMI clustering.

## 3.4 Spatiotemporal Structural Information

Bag of *video-words* approach ignores the spatial and temporal structural information of the features. In our work, we explore two approaches to capture this information, namely spatial correlogram and spatial temporal pyramid matching. This section describes the modified correlogram. It represents the correlation of two features at a certain relative distance, so it is translation and somewhat scale invariant.

Assume $n$ local cuboids, denoted as $\mathcal{P} = \{p_1, p_2, ..., p_n\}$, are extracted from an action video $\mathcal{A}$, and quantized into $m$ *video-words* or *VWC*s $\mathcal{V} = \{v_1, v_2, ..., v_m\}$. So a cuboid is represented by a triple $p_i = (x_i, y_i, v(p_i))$, where $(x_i, y_i)$ (the time dimension is ignored here) is the centroid of the cuboid and $v(p_i)$ is the function mapping a cuboid to a *video-word*. We also quantize the distance into $K$ distance levels $\mathcal{D} = \{D_1, D_2, ..., D_K\}$, where

$D_i = [d_{i1}\ d_{i2}]$ ($[x_1\ x_2]$ denotes an interval). By defining $D_i < D_j$ if $d_{i2} \leq d_{j1}$, we further assume $D_1 < D_2 < ... < D_K$. The distance between two cuboids $p_1$ and $p_2$ is defined as a function $d(p_1, p_2)$, which could be the $L_\infty$-norm or Euclidean distance. Consequently, the correlogram of two labels $v_i$ and $v_j$ with distance interval $D_k$ is defined as a probability $\mathcal{R}$,

$$\mathcal{R}(D_k, v_i, v_j) = \mathbf{Pr}\big(v(p_2) = v_j | v(p_1) = v_i, d(p_1, p_2) \in D_k\big), \qquad \text{(Eq. 3.8)}$$

where $p_1, p_2 \in \mathcal{P}$, $1 \leq i, j \leq m$ and $1 \leq k \leq K$. From the correlogram of two *video-words* $v_i$ and $v_j$, we can know the probability of finding a cuboid $p_2$ with label $v_j$ at $D_k$ distance away from the given cuboid $p_1$ with label $v_i$.

In practice, the computation of the correlogram of two labels is very straightforward. Consider a patch $p$: we define $T(D_k, p)$ as a local co-occurrence table of patch $p$ with $1 \times L$ dimensions, where each dimension denotes one label. This table captures the number of occurrence of each label $l_i$ at distance $D_k$ from patch $p$. Next, we normalize the co-occurrence table by sum of all the dimensions and get $\hat{T}(D_k, p)$. Finally, we can approximate the correlogram of label $v_i$ and all the labels at some distance $D_k$ as,

$$\hat{\mathcal{R}}(D_k, v_i) = \sum_{p \in \mathcal{S}_{v_i}}^{|\mathcal{S}_{v_i}|} \frac{\hat{T}(D_k, p)}{|\mathcal{S}_{v_i}|}, \qquad \text{(Eq. 3.9)}$$

where $\mathcal{S}_{v_i}$ denotes the set of cuboids with label $v_i$ and $|\mathcal{S}_{v_i}|$ is its cardinality. Actually, this is an averaging procedure. The $j$-th dimension of $\hat{\mathcal{R}}(D_k, v_i)$ gives the correlogram $\hat{\mathcal{R}}(D_k, v_i, v_j)$ of labels $v_i$ and $v_j$ at distance $D_k$. Hence, the correlogram $\hat{\mathcal{R}}(D_k)$ is an $L \times L$ matrix. Assume $\mathcal{R}_1$ and $\mathcal{R}_2$ respectively represent correlgorams of action videos $\mathcal{A}_1$ and $\mathcal{A}_2$, then the similarity between them is computed as,

$$S(\mathcal{R}_1, \mathcal{R}_2) = \sum_{k=1}^{K} \sum_{i,j=1}^{L} min(\mathcal{R}_1(D_k, v_i, v_j), \mathcal{R}_2(D_k, v_i, v_j)).$$

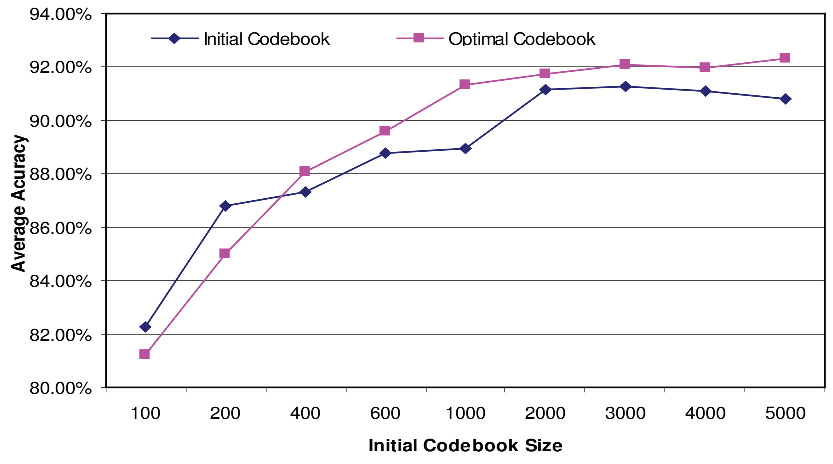
As the correlogram gives the local correlation of two $VWC$s , it is translation and rotation invariant. However, it may not be scale invariant due to the quantization of distance. Instead of using fixed absolute distance quantization, we use the relative distance quantization. Given a video, we get a bounding box around the object with diagonal length of $L_d^{sub}$. Then the relative distance quantization can be computed as:

$$D_k^{rel} = D_k^{abs} \frac{L_d^{sub}}{L_d^{frm}}, \qquad \text{(Eq. 3.10)}$$

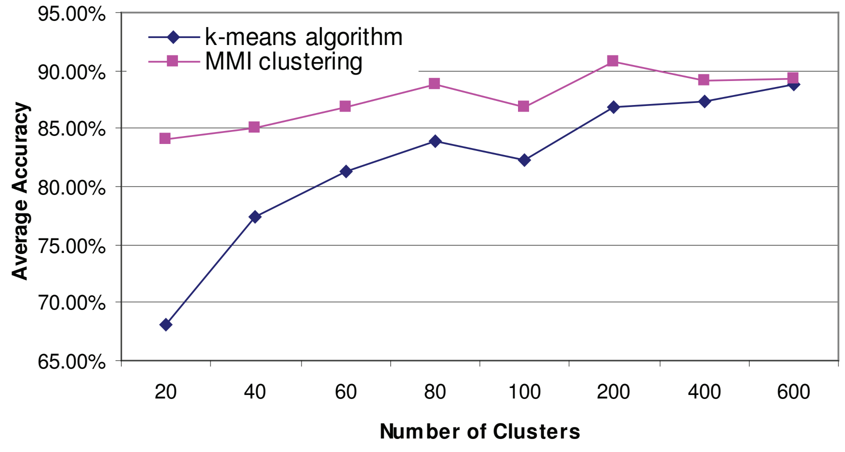
where $L_d^{frm}$ denotes the diagonal length of the frame.

## 3.5 Experiments and Discussion

We have applied our approach to two data sets: the KTH data set [45] and the IXMAS multi-view data set [30]. The default experiment settings are as follows. From each action video 200 cuboids are extracted. All the results reported in this chapter are obtained using the gradient-based feature descriptor. The initial vocabulary is generated by $k$-means algorithm, where 5 randomly selected videos of actors are used for training. We use SVM with Histogram Intersection kernel as the multi-classifier, and adopt the Leave One Out Cross Validation (LOOCV) and 6-fold cross validation (CV) strategies on the KTH and IXMAS data sets respectively. Specifically, we use 24 videos of actors as training and the rest of the videos as testing for the KTH data set, and 10 actors as training for the multi-view data set. The results are reported as the average accuracy of 25 runs on the KTH and 6 runs on IXMAS. In the following, the initial vocabulary and the optimal vocabulary refer to *video words* and $VWC$s respectively.

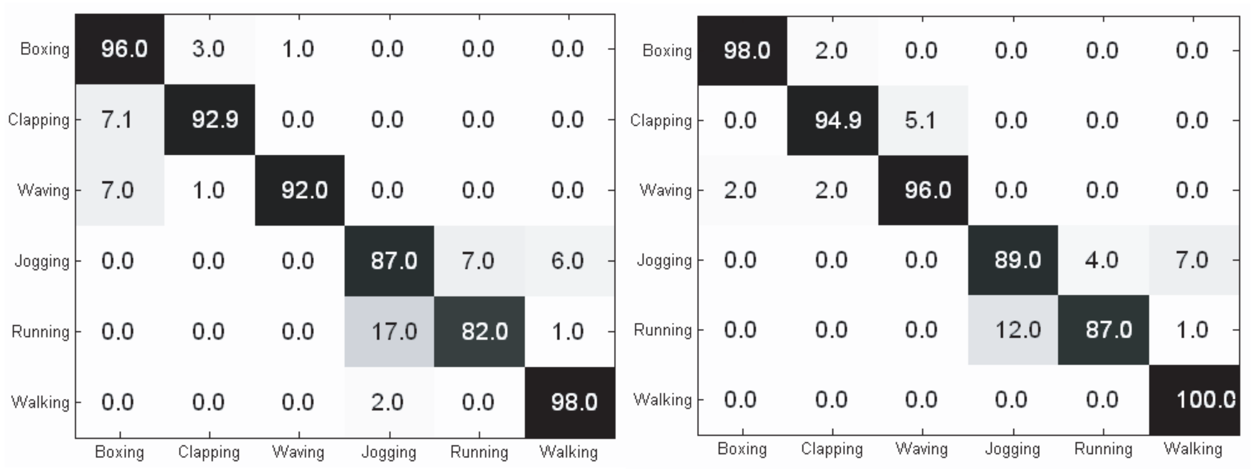


(a)



(b)

Figure 3.2: (a)The classification performance comparison between the initial vocabulary and the optimal vocabulary with different initial vocabulary sizes. (b) The performance comparison between using MMI clustering and directly applying k-means algorithm. MMI clustering reduces the initial dimension of 1,000 to the corresponding number.



|  | Boxing | Clapping | Waving | Jogging | Running | Walking |
|---|---|---|---|---|---|---|
| Boxing | 96.0 | 3.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| Clapping | 7.1 | 92.9 | 0.0 | 0.0 | 0.0 | 0.0 |
| Waving | 7.0 | 1.0 | 92.0 | 0.0 | 0.0 | 0.0 |
| Jogging | 0.0 | 0.0 | 0.0 | 87.0 | 7.0 | 6.0 |
| Running | 0.0 | 0.0 | 0.0 | 17.0 | 82.0 | 1.0 |
| Walking | 0.0 | 0.0 | 0.0 | 2.0 | 0.0 | 98.0 |

**(a)**

|  | Boxing | Clapping | Waving | Jogging | Running | Walking |
|---|---|---|---|---|---|---|
| Boxing | 98.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Clapping | 0.0 | 94.9 | 5.1 | 0.0 | 0.0 | 0.0 |
| Waving | 2.0 | 2.0 | 96.0 | 0.0 | 0.0 | 0.0 |
| Jogging | 0.0 | 0.0 | 0.0 | 89.0 | 4.0 | 7.0 |
| Running | 0.0 | 0.0 | 0.0 | 12.0 | 87.0 | 1.0 |
| Walking | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 100.0 |

**(b)**

Figure 3.3: (a) Confusion table for the classification using the optimal number of *VWC*s ($N_c$=177, average accuracy is 91.31%). (b) Confusion table for the classification using the *VWC* correlogram. The number of *VWC* is 60, and 3 quantized distances are used (average accuracy is 94.15%).

### 3.5.1 Experiments on KTH data set

The KTH data set contains six actions. They are performed by 25 actors under four different scenarios of illumination, appearance, and scale changes. In total, it contains 598 video sequences.

#### 3.5.1.1 Action recognition using orderless features

We investigate the gain of MMI clustering by comparing the classification performance before and after learning the optimal number of *VWC*s. Fig. 3.2(a) shows the performance comparison between the initial vocabulary (before learning) and the optimal vocabulary (after learning) with different sizes of initial vocabulary. As we see, the optimal vocabulary can consistently improve the performance when the size of the initial vocabulary is large. This improvement is very significant. SVM is a strong classifier which can cope with higher dimensional features. Hence, it is not easy to observe the gain of dimension reduction using

SVM. It also shows that by increasing the size of the initial vocabulary the performance decreases in the case of $k$-means clustering, while using $VWC$s the performance even increases slightly.

Instead of doing dimension reduction, we can directly get a lower dimension using $k$-means clustering. Here, we also investigate the gain of MMI clustering compared to directly applying $k$-means. We firstly create an initial vocabulary with size 1,000, which achieves 88.95% average accuracy. MMI clustering preserved 177 as the optimal number of $VWC$s, with average accuracy of 91.31%. Furthermore, we performed eight different clusterings with $\{20, 40, 60, 80, 100, 200, 400, 600\}$ clusters using MMI clustering and k-means algorithm respectively. Fig. 3.2 (b) shows the results. From the figure, we can see that MMI clustering can improve the performance significantly when the number of clusters ($N_c$) is small. This is due to better clustering or more compact data representation. $K$-means algorithm groups the cuboids into *video-words* based on the appearance of the cuboids. When $N_c$ is small, the intra-cluster variance is large, which hurts the performance. However, when MMI clustering groups the 1,000 *video-words* into new clusters, it tries to preserve the mutual information between the *video-words* and the actions, such that the *video-words* in the same cluster may have a strong correlation. Note that they are not necessarily similar in visual appearance. Although in MMI clustering intra-cluster variance of appearance may be large, it can preserve some meaningful concept correlations. Therefore, MMI clustering can still achieve better classification performance, even with small $N_c$. By increasing $N_c$, the performance lines in the figure will probably meet at some point. This phenomena makes sense, because MMI clustering starts with an initial vocabulary of size 1,000, which is the result of $k$-means algorithm. Hence, when $N_c$=1,000, there is no difference between them.

Another observation from our experiments is that the size of the training examples affects the performance very little. We try different $x$-fold CV, where $x$={3, 5, 8, 12, 25} in our experiments with $N_c = 200$, and we obtain the corresponding average accuracy {90.58, 89.97, 90.37, 90.67, 90.80}(%). Hence our LOOCV (25-fold CV) training scheme, which has about

| $x$-fold CV | 3 | 5 | 8 | 12 | 25 |
|---|---|---|---|---|---|
| Avg. Accu.(%) | 90.58 | 89.97 | 90.37 | 90.67 | 90.80 |

Table 3.2: The number of training examples vs. the average performance.

570 training videos, is reasonable. In other words, the performance is not affected much by changing of the number of training examples in our case.

In [59] and [111], the authors use pLSA to do unsupervised classification. More precisely, it is pLSA clustering, which groups the videos to the topics (clusters), and each cluster is assigned to one action. Given a test video, it will be assigned one major topic based on the probability. In order to check the unsupervised classification capability of our approach, we perform the double clustering scheme [91]. It has two phases. In the first phase, we use MMI clustering to get the optimal number of video-words clusters ($VWC$s). Then in the second phase, each video is represented by the $VWC$s, and we apply MMI clustering again on the new representations of the action videos. This time, however, we only group the videos. We pick the optimal number of $VWC$s of $C = 177$, and set the number of "topics"(action clusters) to 10, which is slightly larger than the number of actions (six). Our average accuracy is 84.13%, which is slightly better than 81.50% [59] (they use 6 topics) and 68.53% [111] (they choose 10 topics) by pLSA.

Fig.3.3 (a) shows the confusion table for the classification using the optimal number of $VWC$s ($N_c$=177). From this table, we can see the "hand" related actions ("boxing", "hand clapping", "hand waving") are easily confused with each other. The "leg" related actions (e.g. "jogging","running", and "walking") are also easy to get confused, especially for "jogging" and "running". In Fig. 3.5 we show two example testing videos from each category with their corresponding $VWC$ histograms to demonstrate discrimination of the distribution of the learnt $VWC$s. Actions from the same category share similar $VWC$ distribution. It is also clear to see from the peaks of these histograms that some $VWC$s are dominating in one action but not in the others. If we look into "jogging" and "running", they might have

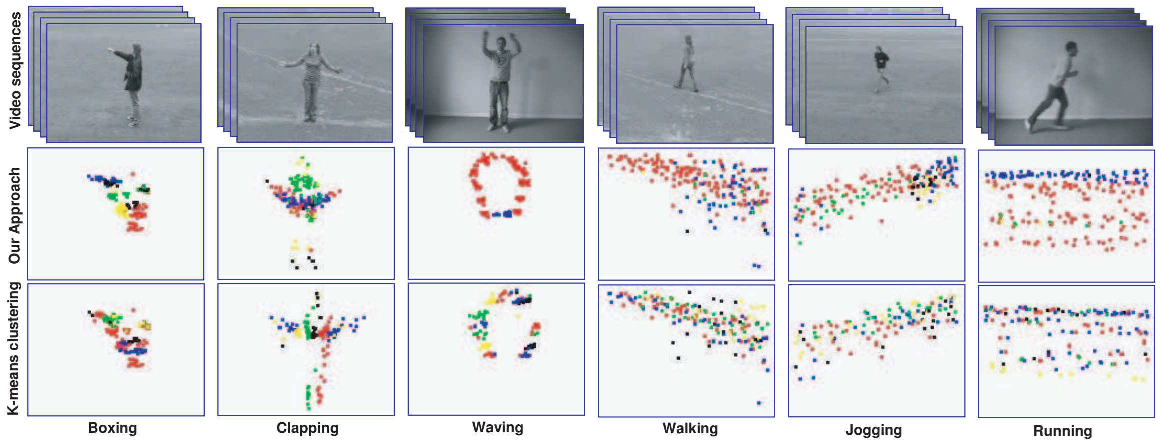


Figure 3.4: The first row shows the examples of six actions. The following two rows respectively demonstrate the distribution of the optimal 20 *video-words* clusters using our approach and 20 *video-words* using k-means. We superimpose the 3D interest points in all frames into one image. Different clusters are represented by different color codes. Note that our model is more compact, e.g. see "waving" and "running" actions (Best viewed in color).

some overlap bins (e.g. bin no. 3 and 20). This is why "running" is easier to confused with "jogging", which is consistent to the observation from the confusion table 3.3. Furthermore, from Fig. 3.4 we see the distribution of *VWC*s is more compact, while that of *video-words* is more dispersive.

#### 3.5.1.2 Classification using spatiotemporal structural information

To encode the spatiotemporal structural information of the cuboids, we have two options. One way is to encode the structural information into *video-words*, then use the learning tools ( e.g., pLSA ) to train. For instance, pLSA-ISM [111] performs pLSA clustering on the structural *video-words* by ISM model (actually, we can say that pLSA-ISM does dimension reduction on the ISM model). We encode the structural information in a more straightforward way. Specifically, our model captures the structural information of the optimal *VWCs* instead of the *video-words*. As we discussed, one benefit of performing MMI clustering on the *video-words* is that we can capture more complicated structural information using the compact and yet discriminative *VWC*s. When computing the correlgoram, we use a small

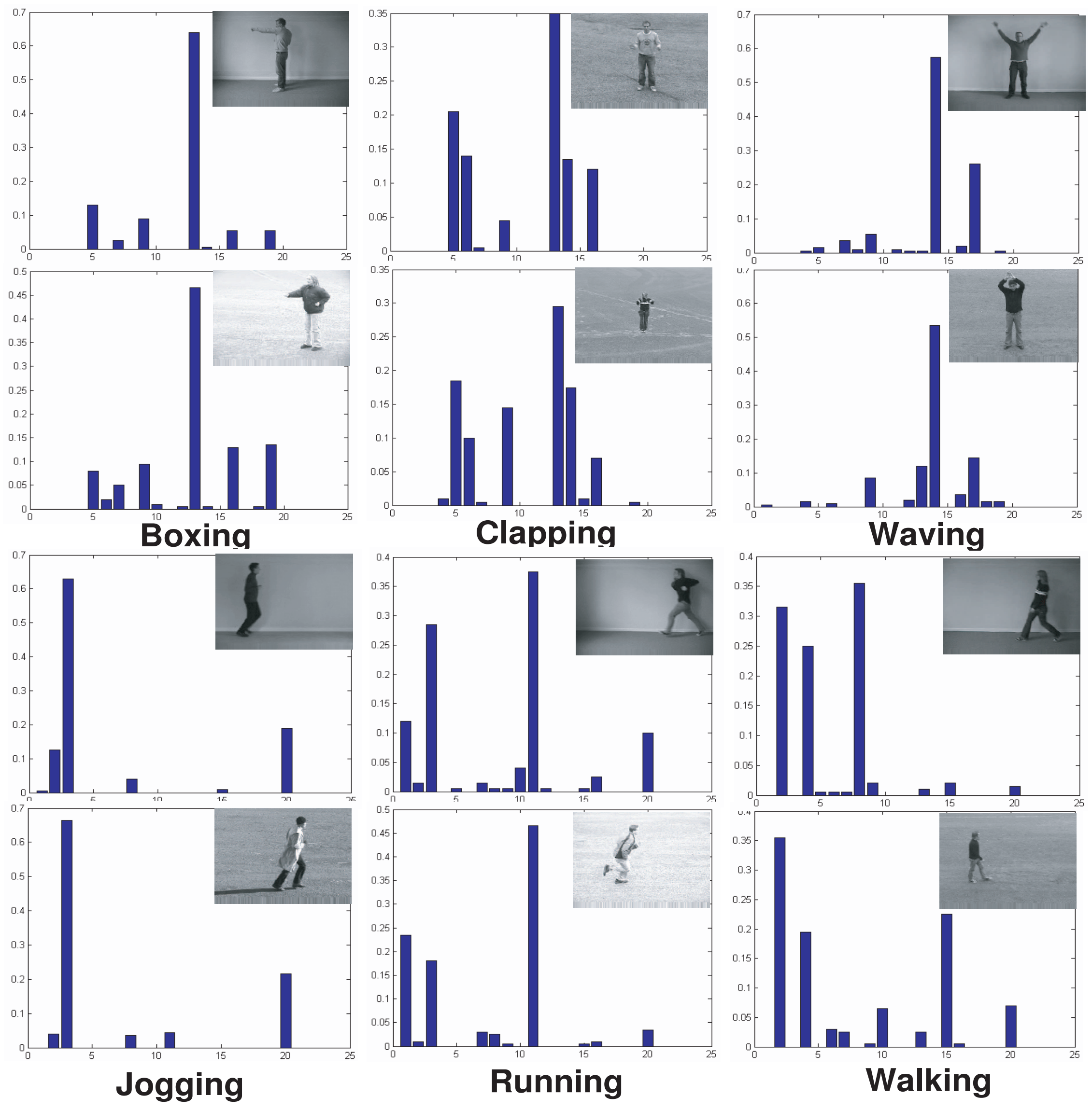


Figure 3.5: Example histograms of the $VWC$s ($N_c$=20) for two selected testing actions from each action category. These demonstrate that actions from the same category have similar $VWC$ distribution, which means each category has some dominating $VWC$s.

| *dimension* | 20 | 40 | 60 | 80 |
|---|---|---|---|---|
| VW(%) | 68.09 | 77.42 | 81.27 | 83.94 |
| VWC(%) | 84.11 | 85.13 | 86.79 | 88.80 |
| VW_Correl(%) | 82.28 | 84.92 | **86.61** | 85.45 |
| VWC_Correl(%) | 87.09 | 90.47 | **94.16** | 91.29 |
| STPM(%) | 88.21 | 92.90 | 93.79 | **93.81** |

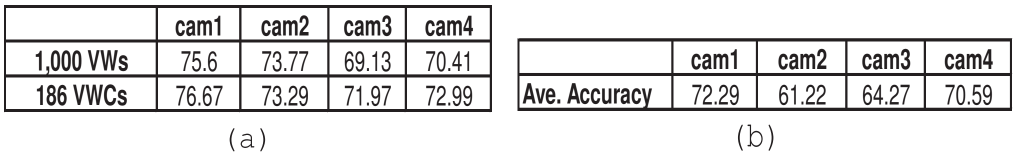
Table 3.3: The performance comparison between different models. *VW* and *VWC* respectively denote *video-words* and *video-word-clusters* based methods, and VW_Correl and VWC_Correl are their corresponding correlgoram models. *STPM* denotes the Spatiotemporal Pyramid Matching approach. The dimension denotes the number of *VW*s and *VWC*s.

number of *video words*(*VWs*) or *VWCs*, and adopt three absolute quantized distance intervals, say [8 16 32], from which we can estimate the relative distance by using Eq. 3.10. Table 3.3 shows the performance comparison between the *VW* correlogram (*VW_Correl*) and the *VWC* correlogram (*VWC_Correl*). As shown in the table, both *VW_Correl* and *VWC_Correl* achieve a large improvement compared to the corresponding orderless models, *VW* and *VWC* model. *VWC_Correl* also outperforms *VW_Correl*, which further verifies that *VWC*s are more discriminative. When the dimension (the number of *VWs* or *VWCs*) is larger than 60, we do not observe much performance improvement.

So far, very little work has been reported on exploiting temporal structure of the *video-words*. Nowozin *et. al.* [109] represent the action as overlapping sub-clips and perform subsequences mining and matching to capture the temporal information of *video-words*. We extend the Spatial Pyramid Matching [107] to the time dimension. Specifically, we perform action matching at multiple resolutions along the time dimension. We also quantize the position of the points. In our model, we have 15-bins quantization for spatial information and three pyramid levels in the time dimension. We show the results of Spatiotemporal Pyramid Matching (*STPM*) in Table 3.3. It also obtains better performance compared to the *VWC* model.

| *Methods* | Accuracy (%) | Structural Inf. |
|---|---|---|
| Our SVM VWCs | 91.31 | No |
| Our VWC Correl. | 94.16 | Yes |
| pLSA_ISM* [111] | 83.92 | Yes |
| WX_SVM [111] | 91.6 | Yes |
| pLSA [59] | 81.50 | No |
| Nowozin *et. al.* [109] | 84.72 | Yes |
| Dollar *et. al.* [93] | 80.66 | No |
| Schuldt *et. al.* [21] | 71.71 | No |

Table 3.4: The performance of the different bag of *video-words* related approaches. pLSA_ISM is the major contribution of [111].



|  | cam1 | cam2 | cam3 | cam4 |
|---|---|---|---|---|
| 1,000 VWs | 75.6 | 73.77 | 69.13 | 70.41 |
| 186 VWCs | 76.67 | 73.29 | 71.97 | 72.99 |

(a)

|  | cam1 | cam2 | cam3 | cam4 |
|---|---|---|---|---|
| Ave. Accuracy | 72.29 | 61.22 | 64.27 | 70.59 |

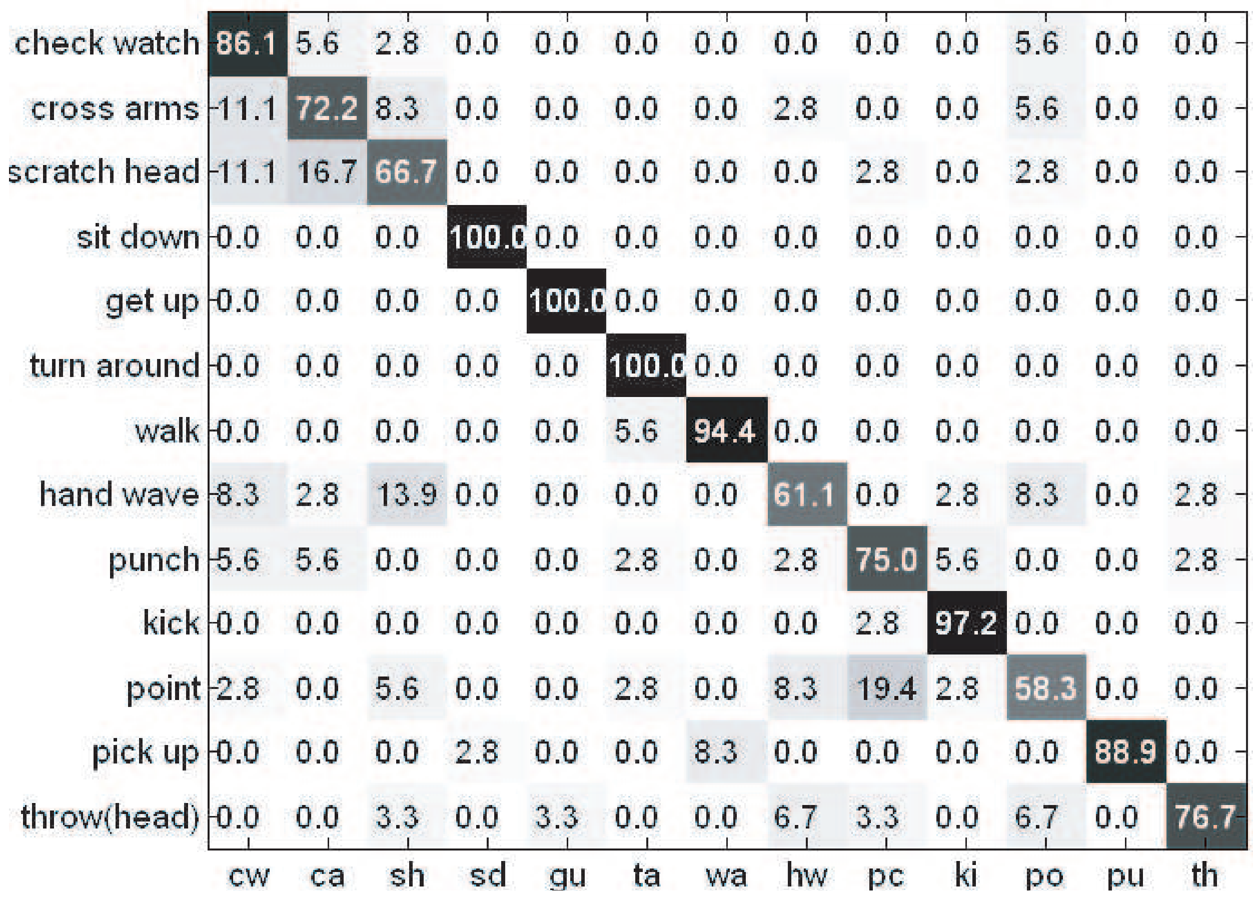
(b)

Figure 3.6: (a) Performance (%) comparison between the original 1,000 *video-words* and the optimal 189 *video-word-clusters*. (b) Average accuracy (%) using three views for training and single view for testing.

Although it is difficult to directly compare with other approaches due to different experiment settings, we summarize all the results of the bag of *video-words* related approaches in Table 3.4 as a reference.

### 3.5.2 IXMAS Multiview dataset

We also applied our approach to the IXMAS multi-view data set. It contains 14 daily-life actions performed three times each by 12 actors. Thirteen action videos are selected in our experiments. Most current approaches applied to this data set need some pre-processing such as background substraction, or 3D model construction. We are the first to use the data for the bag of *video-words* approach, which does not require background substraction. We select four views, excluding the top view. We generate the vocabulary with 1,000 *video-words* using $k$-means algorithm on four actors' actions. Though it is difficult to directly



| | cw | ca | sh | sd | gu | ta | wa | hw | pc | ki | po | pu | th |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| check watch | 86.1 | 5.6 | 2.8 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 5.6 | 0.0 | 0.0 |
| cross arms | 11.1 | 72.2 | 8.3 | 0.0 | 0.0 | 0.0 | 0.0 | 2.8 | 0.0 | 0.0 | 5.6 | 0.0 | 0.0 |
| scratch head | 11.1 | 16.7 | 66.7 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 2.8 | 0.0 | 2.8 | 0.0 | 0.0 |
| sit down | 0.0 | 0.0 | 0.0 | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| get up | 0.0 | 0.0 | 0.0 | 0.0 | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| turn around | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| walk | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 5.6 | 94.4 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| hand wave | 8.3 | 2.8 | 13.9 | 0.0 | 0.0 | 0.0 | 0.0 | 61.1 | 0.0 | 2.8 | 8.3 | 0.0 | 2.8 |
| punch | 5.6 | 5.6 | 0.0 | 0.0 | 0.0 | 2.8 | 0.0 | 2.8 | 75.0 | 5.6 | 0.0 | 0.0 | 2.8 |
| kick | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 2.8 | 97.2 | 0.0 | 0.0 | 0.0 |
| point | 2.8 | 0.0 | 5.6 | 0.0 | 0.0 | 2.8 | 0.0 | 8.3 | 19.4 | 2.8 | 58.3 | 0.0 | 0.0 |
| pick up | 0.0 | 0.0 | 0.0 | 2.8 | 0.0 | 0.0 | 8.3 | 0.0 | 0.0 | 0.0 | 0.0 | 88.9 | 0.0 |
| throw(head) | 0.0 | 0.0 | 3.3 | 0.0 | 3.3 | 0.0 | 0.0 | 6.7 | 3.3 | 0.0 | 6.7 | 0.0 | 76.7 |

Figure 3.7: The recognition performance when four views are used for training and a single view is used for testing. The average accuracy is 82.8%

compare our approach with [30] and [35], we obtained competitive performance, noting that our approach does not require 3D model construction.

**Learning from four views:** We adopt a 6-fold CV scheme, namely using 10 videos of actors for learning and the rest for testing. In the testing phase, we designed two testing schemes: recognition using single view and using multi-views. Our experimental setting is similar to that of [30]. Fig. 3.6(a) gives the single view recognition accuracy comparison between models learnt from the original 1,000 *video-words* and 189 *VWC*s. It shows that the VWC achieves better results than the original *video-words*. In the following, all reported results are achieved by using the optimal *VWC*s. Our average performance for each view

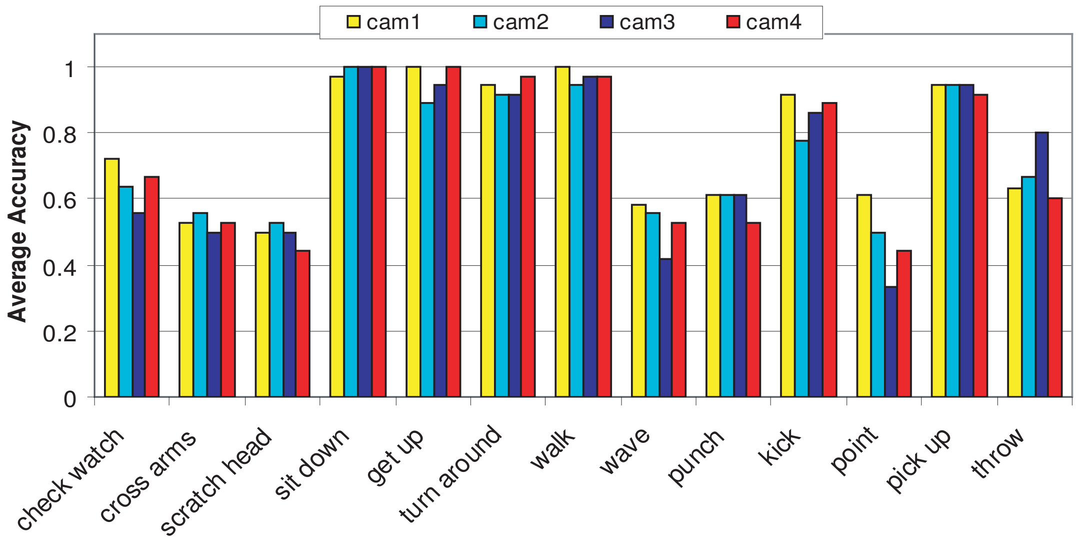


Figure 3.8: The recognition performance when four views are used for training and single view is used for testing.

outperforms that of [30], where {65.4, 70.0, 54.3, 66.0}(%) were reported as average accuracy for four views, and they only tested on 11 actions. Fig. 3.8 plots the details of recognition accuracy for each action.

In the recognition from multi-views, we adopt a simple voting method. Fig. 3.7 shows the confusion table of the recognition using voting from four views. The average rate is 82.8%, which is slightly better than the one reported in [30] (81.27%) and [35] (80.6%). It is quite interesting to note that our approach works much better on big motions, like "walk", "pick up" and "turn around", yet it somewhat gets confused with small hand motions such as "point", "cross arms", "wave hand", and "scratch head". A possible reason is that our features are orderless, which means they do not have any view constraints between them. The hand related actions share some basic motions which are difficult to distinguish by only using the orderless features. For instance, "wave hand" is easily confused with "scratch head".

**Learning from three views:** We trained actions from three selected views, and tested on the fourth view. Hence, there is no information from the fourth view when learning the models, which includes the vocabulary generation. Fig. 3.6 (b) lists the average accuracy of this experiment. The results are still satisfactory. Furthermore, we tried one more complicated experiments. We still train the models using three views, but when testing the model, from the fourth view we only select the subjects which are not included in the training phase. In this experiment, the learning process is totally blind to testing examples. The average accuracy is $\{42.6, 38.08, 58.3, 62.48\}$. The third and fourth views get better results, which means the other three views can provide enough information when testing on this view. To increase the performance, we conjecture more views are necessary.

## 3.6 Conclusion

In this chapter, we propose the MMI clustering approach to find the compact yet discriminative *VWC*s. Since the bag of *video-words* ignores the spatial and temporal structural information, we further use spatial correlogram and temporal pyramid match to make it up. Our approach has been extensively tested on two public data sets: KTH and IXMAS multiview data sets, and we obtain very impressive performance on both data sets. In particular, we are the first to apply the bag of *video-words* related approach on the multi-view data set, and we get competitive results.

# CHAPTER 4: LEARNING SEMANTIC VISUAL-WORDS BY CO-CLUSTERING FOR SCENE CLASSIFICATION

## 4.1 Introduction

In the previous chapter, we proposed a greedy algorithm to discover the optimal number of *visual-words* (i.e., *video-words* in action recognition) by merging the initial *visual-words* in pairwise. At each iteration, it picks up the merge that maximizes the mutual information between *video-word* clusters $\hat{X}$ and action videos $Y$. It works very well in practice. However, it has no guarantees on the global loss function since it only minimizes the loss step by step. On the other hand, it directly clusters *visual-words* based on their distribution on the videos firstly, and then classifies the action videos based on their distribution on the new *visual-word* clusters. One question is whether we can simultaneously cluster the action videos and the *visual-words* such that the two clusterings can boost each other. We believe the information of the action category can help solve the ambiguity of *visual-words* and vice versa. The essence of the simultaneous clustering is that *visual-words* clustering induces the action video clustering while the action clustering induces *visual-words*. In this chapter, we propose to utilize the Maximization of Mutual Information (MMI) co-clustering approach [48] to discover the cluster of *visual-words* for scene recognition. This provides us another option to discover the semantic *visual-word* clusters, which are also called *intermediate concepts* in this chapter.

In general, we can model a scene from the hierarchical viewpoint. On the bottom level, a scene can be modelled as a statistical distribution of color of pixels or interest patches. Yet beyond the low level, we can also describe a scene by the composition of objects such as cars, buildings, and persons. The objects can be further described in terms of parts, e.g., a wheel of a car, a window of a building, or a face of a person. The highest level could be the scene

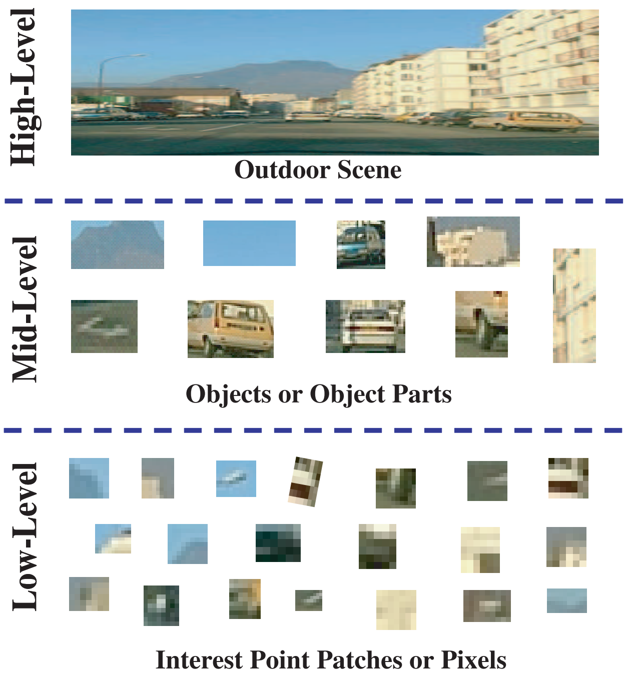


Figure 4.1: An illustration of hierarchical scene understanding.

as a whole. Figure 4.1 illustrates this idea with a scene image. Based on this interpretation of a scene, one strategy to recognize a scene is to model it using the statistical information from low level features. An alternate strategy to recognize a scene is to detect the objects in the scene. However, detecting objects in complex scenes is difficult. Instead of object detection, scene recognition can be done by checking the co-occurrence of large numbers of visual parts. These visual part groups might represent a certain semantic concept (e.g., water, rock, sky) [65]. For instance, the occurrence of a collection of image patches of "eye", "chin", and "forehead" may indicate a "face" or "person" in the image. Those highly correlated image patches can be linked to some semantic concept by manual annotation as paper [65] does. This can also be achieved automatically, such as the extraction of "hidden concepts" or "hidden topics" in [4] [64] [76] [95]. In contrast to low-level feature modelling, humans can obtain more semantic information from an image represented by intermediate concepts.

The proposed framework can also discover the semantic concepts by grouping the *visual words* based on their co-occurrence. Those *visual words* clusters are called high-level features, which are somewhat analogous to the *hidden concepts*. However, there are significant differences between them. Firstly, pLSA is a generative hidden variable model, while MMI co-clustering is not a hidden variable model. Secondly, pLSA assumes conditional independence, i.e., given the latent variables the image and *visual words* are independent, which is not required in MMI co-clustering. Besides, MMI co-clustering performs hard clustering, and it simultaneously clusters both words and documents. Moreover, in practice we observed that pLSA needs a considerable number of EM iterations to reach convergence.

Figure 6.1 shows the workflow of our framework for both learning and classification. Other than using MMI co-clustering technique to automatically discover *intermediate concepts*, we also investigate some spatial models to capture the spatial information of the features. We form a vocabulary from a collection of local patches sampled from the training images using $k$-means algorithm, which can efficiently group visually similar patches into one cluster (*visual word*). Then we use MMI co-clustering to further cluster the *visual words* into *intermediate concepts* in a unsupervised way. In order to capture the spatial information of the semantic concepts in the scene, we exploit the Spatial Pyramid Matching (SPM) [107] and weighted Spatial Concept Corellogram (SCC). Finally, we use SVM as a classifier to train and test these models.

## 4.2 Co-clustering by Maximization of Mutual Information

In this section, we present details on how co-clustering of visual-words and images is performed by maximizing the mutual information. Consider two discrete, jointly distributed random variables $X$ and $Y$, where $X \in \mathcal{X} = \{x_1, x_2, ..., x_n\}$ and $Y \in \mathcal{Y} = \{y_1, y_2, ..., y_m\}$. Here, $\mathcal{X}$ represents a set of *visual-words* in image classification, and $\mathcal{Y}$ is a set of images.

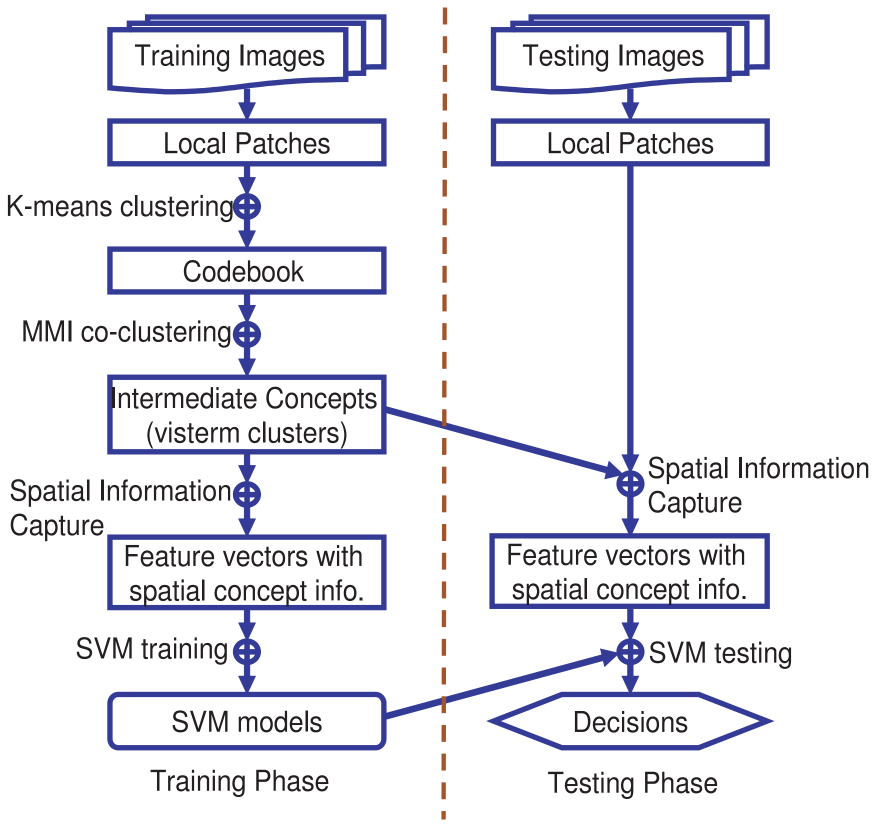

Training Images
Local Patches
K-means clustering
Codebook
MMI co-clustering
Intermediate Concepts (visterm clusters)
Spatial Information Capture
Feature vectors with spatial concept info.
SVM training
SVM models
Training Phase
Testing Images
Local Patches
Spatial Information Capture
Feature vectors with spatial concept info.
SVM testing
Decisions
Testing Phase

Figure 4.2: Work flow of the proposed scene classification framework.

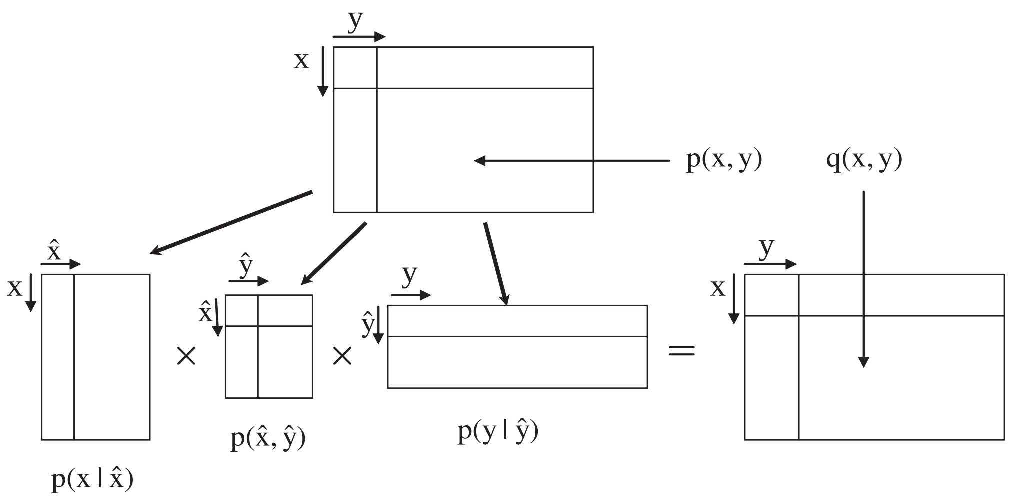


Figure 4.3: The graphical explanation of MMI co-clustering. The goal of MMI co-clustering is to find one clustering of X and Y that minimizes the distance between the distribution matrices p(x,y) and q(x,y).

In scene classification based on BOV modelling, the similarity of two images can be measured by their *visual-words* conditional distributions $p(x|y)$. One critical procedure for BOV modelling is to form vocabulary $X$ via vector quantization using $k$-means algorithm, which groups the local patches by their appearance similarity. If the vocabulary size is small, it may cause over-clustering with higher intra-class distortion. Therefore, it is common to choose an appropriate larger value for vocabulary size. However, this large size may introduce information redundancy in the co-occurrence matrix.

So, we seek to find a more compact representation of $X$, say $\hat{X}$, which is able to capture the "semantic concepts". This procedure is called "word clustering" in text classification. One criteria for $\hat{X}$ is to maximize the mutual information $I(\hat{X};Y)$. Since our original goal is to cluster $Y$, we can simultaneously perform clustering on $X$ and $Y$ by maximization $I(\hat{X};\hat{Y})$.

### 4.2.1 Co-clustering Algorithm

Consider a training image data set $\mathcal{Y}$ with $c$ categories, and its associated vocabulary $\mathcal{X}$ with $n$ *visual-words*; we seek to simultaneously cluster $Y$ into $c$ categories $\hat{\mathcal{Y}} = \{\hat{y_1}, \hat{y_2}, ..., \hat{y_c}\}$, and $X$ into $w$ disjoint clusters $\hat{\mathcal{X}} = \{\hat{x_1}, \hat{x_2}, ..., \hat{x_w}\}$. Actually, we can consider the clustering as two mapping functions $\hat{X} = C_X(X)$ and $\hat{Y} = C_Y(Y)$. In order to evaluate the quality of clustering, we utilize the following mutual information loss:

$$\Delta MI = I(X;Y) - I(\hat{X};\hat{Y}). \qquad \text{(Eq. 4.1)}$$

Because $I(X;Y)$ is fixed for specified data collections, the optimal co-clustering actually attempts to maximize $I(\hat{X};\hat{Y})$, given the number of clusters $c$ for $Y$, and $w$ for $X$ respectively. It is straightforward to verify that the MI loss also can be expressed in the following form (

please refer to [48] for details) :

$$\Delta MI = D_{KL}\big(p(x,y) \parallel q(x,y)\big), \tag{Eq. 4.2}$$

where $q(x,y) = p(\hat{x},\hat{y})p(x|\hat{x})p(y|\hat{y})$. This is the objective function when performing co-clustering. The input to the co-clustering algorithm is the joint distribution $p(x,y)$, which records the probability of occurrence of a particular *visual-words* $x$ in a given image $y$. The aim is to determine clusters with distribution $q(x,y)$, which is as close as possible to $p(x,y)$. The process is pictorially shown in Figure 4.3. For each new clustering $\hat{X}$ and $\hat{Y}$, we first compute the joint distribution matrix $p(\hat{x},\hat{y})$ as follows:

$$p(\hat{x},\hat{y}) = \sum_{x\in\hat{x},y\in\hat{y}} p(x,y). \tag{Eq. 4.3}$$

Then for $x \in \hat{x}$ we compute the conditional distribution $p(x|\hat{x})$,

$$p(x|\hat{x}) = \frac{p(x)}{p(\hat{x})}, \tag{Eq. 4.4}$$

where the marginal distribution $p(x) = \sum_{y\in Y} p(x,y)$ and $p(\hat{x}) = \sum_{\hat{y}\in\hat{Y}} p(\hat{x},\hat{y})$. For $x \notin \hat{x}$, $p(x|\hat{x}) = 0$. Similarly, we can get the conditional distribution $p(y|\hat{y})$. Consequently, the quality of this specified clustering is evaluated by $D_{KL}\big(p(x,y) \parallel q(x,y)\big)$.

The algorithm starts with randomly initial partitions $C_X^0$ and $C_Y^0$. The number of clusters for $X$ and $Y$ are specified as $w$ and $c$ respectively. At each iteration $t$ of the algorithm, two phases are involved:

(i) Clustering of $X$ while keeping $Y$ fixed. For each $x$, assign it to its new cluster, which means $C_X^{t+1} = argmin_{\hat{x}} D_{KL}\big(p(y|x) \parallel q(y|\hat{x})\big)$ where $q(y|\hat{x}) = p(y|\hat{y})p(\hat{y}|\hat{x})$. Update the probabilities based on the new $X$ cluster.

(ii) Clustering of $Y$ while keeping $X$ fixed. For each $y$, find its new cluster such that $C_Y^{t+2} = argmin_{\hat{y}} D_{KL}\big(p(x|y) \parallel q(x|\hat{y})\big)$, where $q(x|\hat{y}) = p(x|\hat{x})p(\hat{x}|\hat{y})$. Update the probabilities based on the new $Y$ cluster.

The iterations of the co-clustering stops when

$$\Delta^t MI - \Delta^{t+2} MI < \epsilon \qquad \text{(Eq. 4.5)}$$

where $\epsilon$ is the threshold.

In summary, in order to assign *intermediate concepts* to each image patch, we apply two steps. We first use $k$-means algorithm to cluster the image patches into *visual-words*. Since the criterion for $k$-means is based on appearance similarity, patches belonging to one *visual-words* are visually similar. Further, we group the *visual-words* into semantic clusters (*intermediate concepts*) via MMI co-clustering. The number of *intermediate concepts* is much less than that of *visual-words*. Our experiments show that we can do better scene classification using *intermediate concepts* than using *visual-words*.

## 4.3 Spatial Correlogram Kernel Matching

Bag of features model represents an image $\mathcal{I}$ as a collection of local features. Generally, we select image patches as the local feature, which are represented by SIFT descriptor. This is a mapping procedure: $\mathcal{I} \mapsto \{(d_k, x_k)\}_{k=1,\ldots,m}$, where $d_k \in \mathbb{R}^D$ (i.e. D=128 for SIFT) is the feature descriptor, and $x_k$ is the feature location. To compare two images, we can directly perform feature matching [79], or we can integrate spatial constraints to remove the outlier matched features [86]. Furthermore, bag of *visual-words* is introduced for fast image matching. Instead of computing the distance between two descriptors in $\mathbb{R}^D$, it quantized the space of $\mathbb{R}^D$ by clustering a set of training image patches, such that each patch is labelled with one of the vocabulary entries $\mathcal{V} = \{v_1, v_2, ..., v_N\}$. Then, image $\mathcal{I}$ is a collection of triples $\{l_k, x_k, y_k\}_{k=1,...,m}$, $l_k \in \mathbb{V}$, and $(x_k, y_k)$ is the patch location.

Bag of *visual-words* directly matches two images by conducting histogram intersection between two image histograms of *visual-words*, which is treated as fast coarse image matching. Similarly, we also can integrate spatial information to the *visual-words.* For instance, without spatial location information, the patches from "sky", "lake", and "road" may be assigned into the same *visual-word* since they have similar appearance (i.e., they are flattened surface in gray images). However, we notice that the patches from different categories may have different spatial distributions, such as "sky", which is generally on the top of an image. Spatial Pyramid Match (SPM) [107] exactly makes use of this property of the scene images. It further integrate coarse location information into the *visual-words*, resulting in a finer vocabulary.

The spatial information integrated in SPM, however, is not flexible. The resulting models are not invariant to any transform, such as rotation or translation transform. In this section, we propose the spatial correlgoram to integrate the relative spatial relationship between the features. Therefore, it is invariant to some transforms such as translation and rotation.

### 4.3.1 Spatial correlgram

Given the image $\mathcal{I} \mapsto \mathcal{P} = \{l_k, x_k, y_k\}_{k=1,...,m}$, each patch $p$ is triple assigned with a label and location. We also quantize the distance into $K$ distance levels $\mathcal{D} = \{D_1, D_2, ..., D_K\}$, where $D_i = [d_{i1}\ d_{i2}]$ (In this chapter, $[x_1\ x_2]$ denotes an interval). We define $D_i < D_j$ if $d_{i2} \leq d_{j1}$ for any $1 \leq i, j \leq K$. Then we assume $D_1 < D_2 < ... < D_K$. The distance between two patches $p_1$ and $p_2$ is defined as a function $d(p_1, p_2)$, which could be the $L_\infty$-norm distance or Euclidean distance. Consequently the spatial correlogram of two labels $v_i$ and $v_j$ with distance interval $D_k$ can be defined as a probability $\mathcal{R}$,

$$\mathcal{R}_{D_k}(v_i, v_j) = \mathbf{Pr}\big(p_1(l) = v_i, p_2(l) = v_j | d(p_1, p_2) \in D_k\big), \qquad \text{(Eq. 4.6)}$$

where $p_1, p_2 \in \mathcal{P}$, $1 \leq i, j \leq N$ and $1 \leq k \leq K$. From the correlogram of two labels $l_i$ and $l_j$, we can know the probability of finding a patch $p_2$ with label $l_j$ at $D_k$ distance away from the given patch $p_1$ with label $l_i$. Therefore, the correlogram represents the spatial distribution of one label with respect to other labels. It provides a discriminative pattern of an image.

In practice, the computation of spatial correlogram of two labels is very straightforward. Consider an image patch $p$, we define $T(D_k, p)$ as a local co-occurrence table of patch $p$ with $1 \times N$ dimensions, where each dimension denotes one label. This table captures the number of occurrence of each label $v_i$ at distance $D_k$ from patch $p$. Next we normalize the co-occurrence table by sum of all the dimensions and get $\hat{T}(D_k, p)$. Finally, we are able to approximate the correlogram of label $v_i$ and all the labels at some distance $D_k$ as,

$$\hat{\mathcal{R}}(D_k, v_i) = \sum_{p \in \mathcal{S}_{v_i}}^{|\mathcal{S}_{v_i}|} \frac{\hat{T}(D_k, p)}{|\mathcal{S}_{v_i}|}, \qquad \text{(Eq. 4.7)}$$

where $\mathcal{S}_{v_i}$ denotes the set of patches with label $v_i$ and $|\mathcal{S}_{v_i}|$ is its cardinality, $1 \leq i \leq N$, $1 \leq k \leq K$. Actually, this is an averaging procedure. The $j$-th dimension of $\hat{\mathcal{R}}(D_k, v_i)$ gives the correlogram $\hat{\mathcal{R}}(D_k, v_i, v_j)$ of label $v_i$ and $v_j$ at distance $D_k$. Hence, the correlogram $\hat{\mathcal{R}}(D_k)$ is an $N \times N$ matrix. Specially, when $i = j$, we call it autocorrelogram of label $l_j$ with distance interval $D_k$. Considering all the $K$ distance intervals, the dimension of the correlogram $\hat{\mathcal{R}}(\mathcal{I})$ is $N \times N \times K$, and the autocorrelogram is an $K \times K$ matrix.

### 4.3.2 Spatial Correlogram Kernel

From the computation of the spatial correlogram, we can see the correlogram is a sort of normalized local histogram ($T(D_i, p)$ is an occurrence table). So we can use *histogram intersection* function to match two images. Suppose $\mathcal{I}^1$ and $\mathcal{I}^2$ are two images with correlograms

$\mathcal{R}^1$ and $\mathcal{R}^2$ respectively. Then the Spatial Correlogram Kernel (SCK) is defined as,

$$K(\mathcal{R}_1, \mathcal{R}_2) = \sum_{k=1}^{K} \sum_{i,j=1}^{L} w_k \times min(\mathcal{R}^1_{D_k}(v_i, v_j), \mathcal{R}^2_{D_k}(v_i, v_j)), \qquad \text{(Eq. 4.8)}$$

where $w_k$ is the weight assigned to the matches made at distance interval $D_k$. Thus, we can assign higher weights to the matches found at smaller distances. Because $c \times min(x, y) = min(cx, cy)$ if $c, x, y > 0$, we can add these weights while computing the spatial correlogram. Since SCK is positive and semi-definite, it satisfies the Mercers condition. This results from the fact that min(:; :) is a Mercer kernel and that the set of Mercer kernels is closed under summation.

We show an example to demonstrate the SCC modeling is more discriminative than BOC modeling. Fig. 4.4 shows three synthetic images, with 100 patches and two concepts $A$ and $B$. The number of $A$ patches is identical to that of $B$. Obviously, some models (e.g., BOV, pLSA) can not distinguish the three images from each other, since they do not employ spatial statistical information. However, using autocorrelogram of concept $A$ or $B$, we are able to identify them. From the autocorrelogram plot, the similarity between Image I and Image II is higher than that of Image I and Image III.

## 4.4 Experiments

We have extensively applied our proposed approach to two diverse data sets: fifteen scene categories [107] and the LSCOM (Large Scale Concept Ontology for Multimedia) data set [77]. For both data sets, only gray level images are used. The default experiment setting is listed as follows. We utilize dense features sampled using a regular grid with space $M$=8 pixels. The patch size is randomly sampled between scales of 10 to 30 pixels. SIFT descriptor [79] is computed for each patch. We use a support vector machine (SVM) with Histogram Intersection kernel as a classifier. For 15 scene categories, we choose the one-versus-all

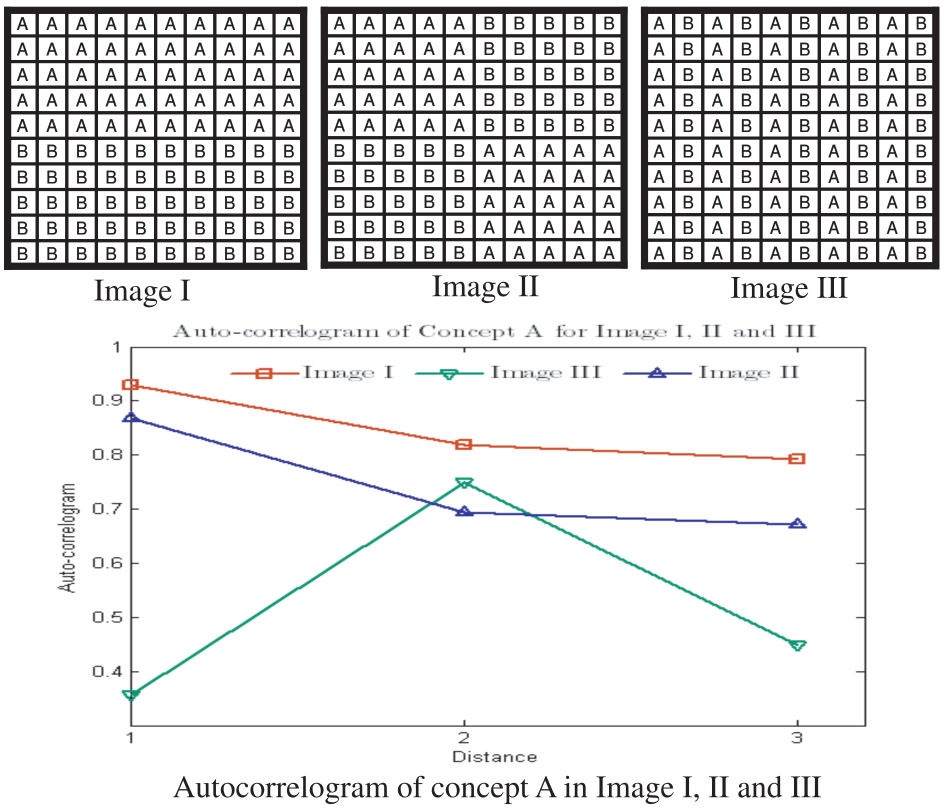


Figure 4.4: An example to show autocorrelogram of three synthetic images.

methodology for multi-class classification. The binary SVM classification is applied to the experiments on LSCOM. All the experiments on 15 scene categories are repeated 5 times with a different training data set, and the final results are reported as the average accuracy.

### 4.4.1 Classification of Fifteen Scene Categories

The fifteen scene categories are the same used by [107], which is union of the 13 scenes reported in [76] and two additional scenes added by Lazebnik et al. In fact, the thirteen categories contain 8 scenes originally reported in [7]. Each category has 212∼410 images. The average image size is about 250 $\times$ 300. We use 50 randomly selected images from each category to form the vocabulary of size $N_v$. Furthermore, we use MMI co-clustering to discover $N_c$ *intermediate concepts* from the vocabulary. We try several $N_v$, and finally fix

$N_v = 1,500$ which gives better performance. Then an image can be represented by *visual-words* histogram (BOV model) or an *intermediate concepts* histogram (BOC model). In the SVM classification phase, 100 images are randomly selected from each category as a training set, and the rest are used for testing.

#### 4.4.1.1 Classification using orderless features

We investigate the gain of MMI co-clustering (BOC model) compared to the $k$-means approach (BOV model) in two ways. One is to compare them using the same number of clusters($N_c = N_v$), and the other is to compare BOC with the original BOV with $N_v$=1,500. (*Original BOV means directly representing an image using bag of visterms from which the* intermediate concepts *are created.*) We conduct classification on the 15 scene categories using both BOV and BOC models by using different values of $N_v$ or $N_c$ from a set: $\{20, 60, 80, 100, 200, 300\}$. Table 4.1 shows the results. Overall, BOC is able to improve the performance between 3.16% to 16.07% compared to BOV; it works especially well when the number of clusters is small. This is due to better clustering. $K$-means algorithm groups the image patches into *visual-words* based on the appearance of the patches. When $N_v$ is small, the intra-cluster variance is larger, which hurts the performance. However, when grouping the 1,500 *visual-words* into semantic intermediate concept clusters, MMI co-clustering tries to preserve the mutual information between *visual-words* and images, such that the *visual-words* in the same cluster share certain common *intermediate concepts*, so they are not necessarily similar in visual appearance. Although in MMI co-clustering, intra-cluster variance of appearance may be large, it can preserve some meaningful concepts. Therefore, MMI co-clustering can still achieve better classification performance even with small $N_c$. The best performance for BOC is achieved when $N_c = 200$.

The classification accuracy is 76.38% when using BOV model with $N_v = 1,500$, which is slightly better than the best performance of BOC model. This is consistent with the

| $N_c/N_v$ | 20% | 60 | 80 | 100 | 200 | 300 |
|---|---|---|---|---|---|---|
| BOV | 47.25 | 61.69 | 65.34 | 67.72 | 70.81 | 71.46 |
| BOC | 63.32 | 68.53 | 70.25 | 73.01 | 75.16 | 74.62 |

Table 4.1: The average accuracy (%) achieved using strong and weak classifiers.

results of Lazebnik *et al.* [107] and Quelhas *et al.* [95]. We conjecture that it may be due to the dimension reduction achieved by the MMI co-clustering technique. While Bosch *et al.* claimed in their paper [4] that compared to original BOV, pLSA, which is another dimension reduction technique similar to our MMI co-clustering, performs better, we feel that the gain in performance due to the dimension reduction depends on classifier type and the performance of original BOV. The performance of BOV can vary with the patch sampling [33] and the number of *visual-words* [87]. If the patch sampling and $N_v$ have been optimized, it is not easy to achieve higher accuracy with any dimension reduction techniques because BOV representation does not contain much noise. Another reason may be due to the performance of the specific classifier. Some weak classifiers like $k$-Nearest Neighborhoods (KNN) perform poorly with high dimensional features. Therefore, when the dimension is reduced, they are able to achieve better performance. However, some strong classifiers (i.e. SVM) which are good at classification of high dimensional features, may not be able to achieve better performance with dimension reduction because of certain information loss.

In order to verify our conjecture, we conducted two groups of experiments. Table 4.2 shows the results using different sampling space $M$. Here, multi-class SVM is used as a classifier. The first row lists the results using BOV model with $N_v$=1,500, and the second row shows the results using BOC model where the *intermediate concepts* are extracted from the corresponding BOV codebook. When we increase the sampling space, the difference between the performance of BOV and BOC decreases from about 3.6% to -1.33%. In particular, the sampling setting in the third column is similar to the sampling in [4], and the performance of BOC is better than BOV. In fact, large sampling space generates fewer sampling features.

|  | M = 4(%) | M = 8(%) | M = 10(%) |
|---|---|---|---|
| BOV($N_v$=1,500) | 78.32 | 76.38 | 69.81 |
| BOC($N_c$=200) | 74.69 | 75.16 | 71.14 |

Table 4.2: The results achieved under different sampling space.

| $N_c \setminus N_v$ | 1500 | 40 | 60 | 80 | 100 |
|---|---|---|---|---|---|
| SVM | 76.38 | 64.60 | 68.53 | 70.25 | 73.01 |
| KNN (K=12) | 58.17 | 61.22 | 63.76 | 64.54 | 66.37 |

Table 4.3: The average accuracy (%) achieved using strong and weak classifiers.

Space with $M$=4 corresponds to more than 4,000 patches for each image, while space with $M$=10 corresponds to only about 500 patches. Therefore, we feel that with a large number of sampling patches, the BOV performs better. Our further experiments verified this. For $M$=8 (each image has more than 1,000 patches), we randomly select about 200 patches from each image to evaluate the performance. Then the results for BOV and BOC are 67.29% and 69.21% respectively. Therefore, the number of patches sampled from the image affects the comparison between BOC and original BOV.

We further investigated the performance of classification using different classifiers. Table 4.3 demonstrates the performance comparison of the SVM classifier and the KNN classifier with Euclidian distance. In both cases, 100 randomly selected images from each category were used for training. The first column with $N_v$=1,500 shows the BOV baseline, and the following column shows the result of BOC with different $N_c$. It is very clear that the KNN classifier does not work well for high dimensional data. Hence, the dimension reduction technique can improve the performance very much. However, SVM is a strong classifier which is able to handle high dimensional data. With reasonable $N_c$, the SVM can still achieve competitive results. However, low dimensional features provides us much better computational efficiency, which is very important for learning/classification of a large data set like LSCOM.

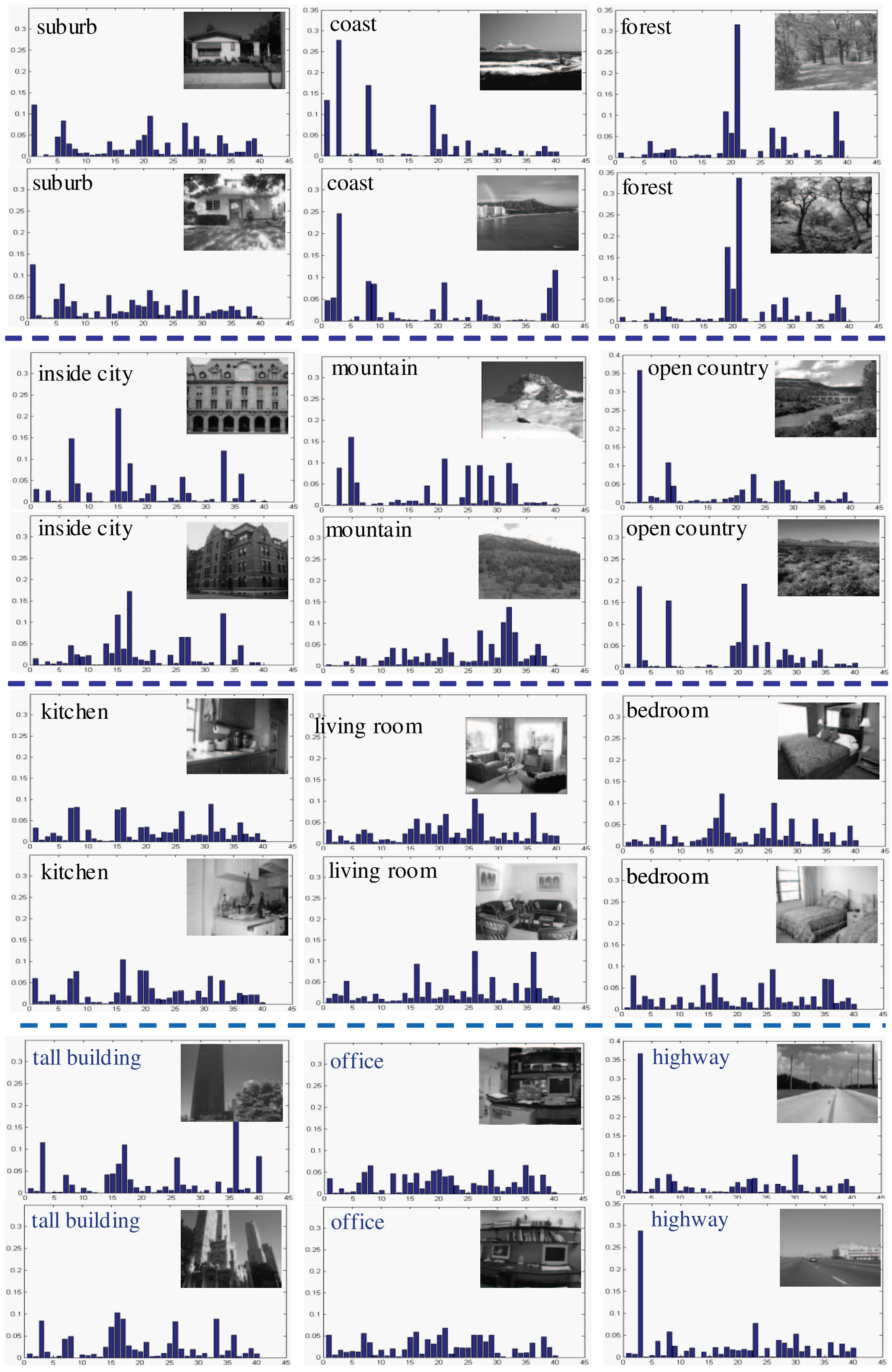


Figure 4.5: Example histograms of *intermediate concepts* for 2 selected testing images from each scene categories.

Finally we compared the performance of MMI co-clustering, pLSA, and IB. For all of them, we use the default experiment setting. pLSA achieves the best performance of 71.24% at $N_c = 80$. The best performance of IB is 72.49% when $N_c = 150$, while MMI co-clustering can achieve 75.16% at $N_c = 200$. Also, in the experiments we observed that pLSA converged after about 100 iterations, while MMI co-clustering can converge in less than 40 iterations. This is consistent with the claim in [6] that in practice it takes a considerable number of of EM iterations for pLSA to converge. The time complexity for co-clustering, pLSA, and IB are $O(t \cdot R \cdot (c + k))$, $O(t \cdot R \cdot k)$, and $O(|I|^3)$ respectively (where $t$ is number of iterations, $R$ is number of nonzero entries, $c$ is number of categories, $k$ is number of *intermediate concepts*, and $I$ is number of training images). Therefore, IB is not suitable for large datasets [91].

In Fig. 4.5 we show two example testing images in some categories with their corresponding concept histograms to demonstrate discrimination of BOC, and also demonstrate the meaningfulness of *intermediate concepts*. It is clear to see from the peaks of these histograms that some concepts are dominating in one scene category but not in the others.

#### 4.4.1.2 Classification using intermediate concepts and their spatial information

In order to capture the spatial information, we implement two models: Spatial Pyramid Matching (SPM) [107] and Spatial Concept Correlogram (SCC). For SPM, we repeatedly divide an image into sub-blocks and compute local histogram of *intermediate concepts* for each block. Finally, an image is represented by combining the local histograms from the sub-blocks of the pyramid. The representative vector has high dimensions of $\frac{1}{3}(4^L - 1)N_c$, where $L$ is the number of pyramid levels. In our experiments, we set $L = 3$. From Table 4.4, we can see that thanks to *intermediate concepts*, the SPM_IC (SPM using *intermediate concepts*) can improve the performance from about 2.79% to 4.28%, especially when the number of clusters is smaller. Interestingly, we notice that when $N_c = 80$, the SPM_IC can achieve competitive performance to SPM_V (SPM using *visual-words*) at $N_v = 400$ , while

| | 40 | 50 | 80 | 100 | 200 | 400 |
|---|---|---|---|---|---|---|
| SPM_V | 75.24 | 76.14 | 77.62 | 77.81 | 80.27 | 80.46 |
| SPM_IC | 79.52 | 80.19 | 80.93 | 81.33 | 83.19 | 83.25 |

Table 4.4: The performance (average accuracy %) of SPM using *visual-words* and *intermediate concepts*. SPM_IC and SPM_V denote SPM using *intermediate concepts* and *visual-words* respectively.

| | 40 | 60 | 80 | 100 | 200 | 400 |
|---|---|---|---|---|---|---|
| BOC | 65.48 | 68.53 | 70.25 | 73.01 | 75.16 | 74.21 |
| SCC | 65.97 | 69.71 | 72.40 | 74.39 | 77.76 | 78.15 |
| BOC+SCC | 71.10 | 73.06 | 75.18 | 78.33 | 81.49 | 81.72 |

Table 4.5: The average classification accuracy (%) obtained by various models (SCC, BOC, and SCC+BOC).

the dimension is reduced by 5 times. The best performance achieved by SPM_V is 80.46% [1] and 83.25% for SPM_IC.

In our experiments, we consider autocorrelogram. When computing the SCC, we divide the image into 2 by 2 blocks, and for each block, we compute its SCC. We set $D_1 = [1\ 64]$ and $D_2 = [64\ 128]$ in term of pixels in x and y directions. Table 4.5 shows the classification results. We can see the combination of SCC and BOC can achieve better performance over SCC and BOC. Interestingly, *correlatons* reported in [110] performs much worse than BOV. However, our SCC performs better. This might be due to the fact that our *intermediate concepts* correlogram is computed on patches, and weighted with a different quantized distance. The best performance for SCC+BOC is 81.72%, which is little worse than SPM_IC, but better than SPM_V. The number of dimensions is $9N_c$ which is much lower than that of SPM_IC. Fig. 4.6 shows the confusion table for the 15 sceene categories using SCC+BOC approach. It is clear from the confusion table that it is easy to distinguish “suburb”, “forest”, “street”, “tall building”, and “coast” scenes. However, the performance for “bedroom”, “living room”, and “kitchen” scenes is not as good.

[1]In [107] the best performance for SPM_visterm is quoted as 81.4%.

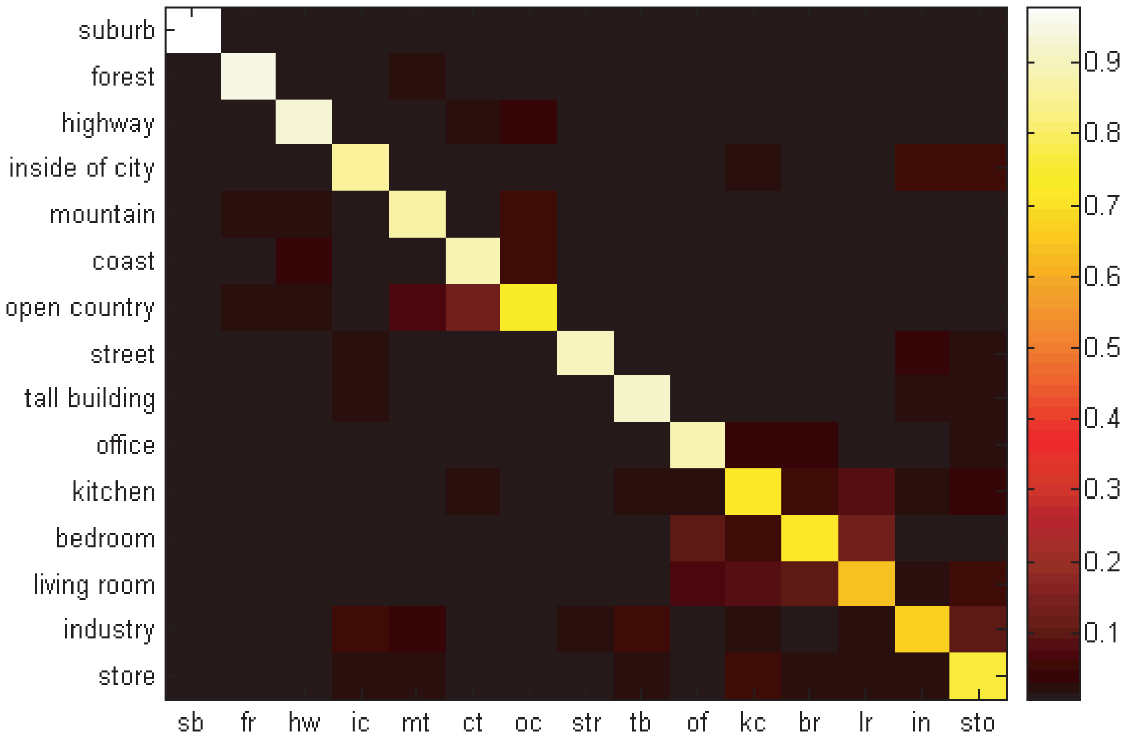


Figure 4.6: Confusion table of the best performance for the SCC+BOC model. The average performance is 81.72%.

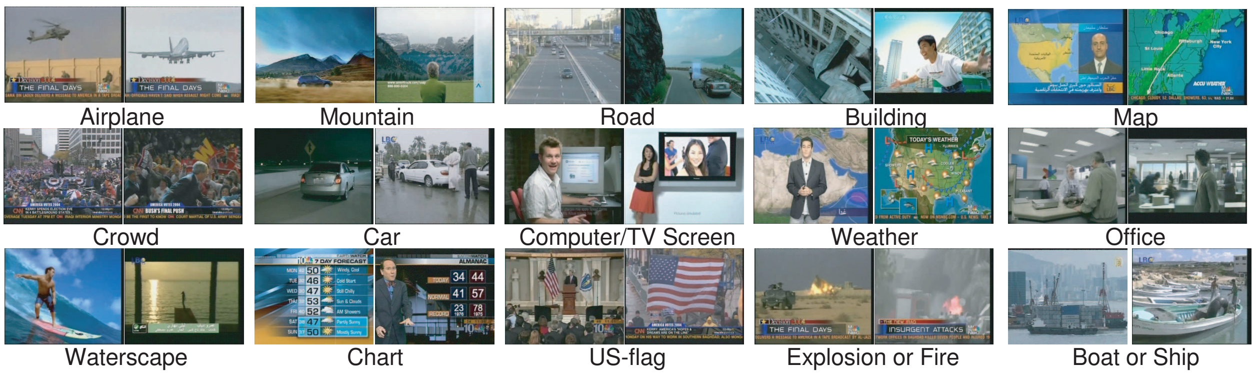


Figure 4.7: Example key frames selected from LSCOM Data set

### 4.4.2 Classification of LSCOM Dataset

The LSCOM data set [77], which includes more than 400 annotated categories, is a very challenging data set and has has been explored by the TRECVID community for several years [1]. This data set is more challenging, and it contains 61,901 key frames extracted from a variety of real TV news programs. Figure 4.7 shows some category examples. The size of the key frame is fixed to 240 $\times$ 352. In our experiments, the following 28 categories including scenes and objects are evaluated: airplane(522), animal(486), basketball(314), boat or ship(358), building(6,059), charts(511), clouds(876), weather(2,376), crowd(724), desert(564), flag-US(480), maps(940), meeting(3,025), military(2,419), mountain(827), road(3,836), studio(6,400), tennis(202), trees(3,462), urban(5,977), waterscape(1,584), computer_TV-screen(2,376), explosion or fire(1,020), industrial setting(239), car(4,116), fields(259), office(1,364), and vegetation(5,150). In this experiment, we want to demonstrate how the different classification approaches perform. Unlike the 15 scene categories, the key frames may contain several overlapping high level concepts. For example, in one key frame, you probably can see crowd, buildings, cars, or roads. Therefore, each key frame may be classified into multiple categories. We use binary SVM as the classifier (the key frames from one category are positive, all the rest are negative). The average precision (AP) is adopted as the performance measure. Assume that $D$ retrieved key frames are ranked, $R$ of them are relevant ($R < D$), and then we can define the AP as follows:

$$AP = \frac{1}{R}\sum_{j=1}^{D}\frac{R_j}{j} * I_j, \qquad \text{(Eq. 4.9)}$$

where $I_j = 1$ if the $jth$ shot is relevant, otherwise 0. $R_j$ is the number of relevant key frames in the top $j$ retrieved key frames.

To form the vocabulary, we randomly selected 50 key frames from each category of the 28 categories and 500 key frames from categories other than the 28 categories. Finally, an $N_v =$

|  | BOV-O | CC-BOC | BOV-D | pLSA-BOC |
|---|---|---|---|---|
| MAP | 61.91% | 59.48% | 45.07% | 55.77% |

Table 4.6: The MAP for the 28 LSCOM categories achieved by different approaches. BOV-O and BOV-D represent the BOV models with $N_v = 3,000$ and $N_v = 250$ respectively. CC-BOC and pLSA-BOC denotes the BOC model created by co-clustering and pLSA respectively.

$3,000$ vocabulary is achieved. Further, the "intermediate concepts" using MMI co-clustering and pLSA are generated from the $N_v = 3,000$ vocabulary. We tried different values of $N_c$, and chose the value of $N_c$ which gave us the best results. In the SVM learning/classification phase, we randomly divided the data set into three parts: one half for training, 1/4 for validation, and 1/4 for testing. Fig 4.8 shows the AP of each category. pLSA (pLSA-BOC) only performs better than the MMI co-clustering (CC-BOC) for 3 categories. Compared to the BOV with reduced dimension ($N_v = 250$), the CC-BOC always performs much better. Besides, for most cases, the CC-BOC can achieve competitive results compared to original BOV ($N_v = 3,000$). However, the gain of CC-BOC is computational efficiency with lower dimension when performing SVM learning and classification on a large data set. (e.g. it takes about 23 hours to learn and test the 28 categories for BOV with $N_v = 3,000$, while it only takes about 6 hours for BOC with $N_c = 250$ on a 2.99GHz machine.) The advantage of MMI co-clustering can be clearly noticed in Table 4.6, which demonstrates the Mean Average Precision (MAP) of the 28 LSCOM categories using different approaches. Compared to BOV-D, the benefit of MMI co-clustering is about 14.4% in terms of MAP, which further verifies that MMI co-clustering can get more meaningful clusters. Even compared to pLSA, MMI co-clustering performs about 4% better.

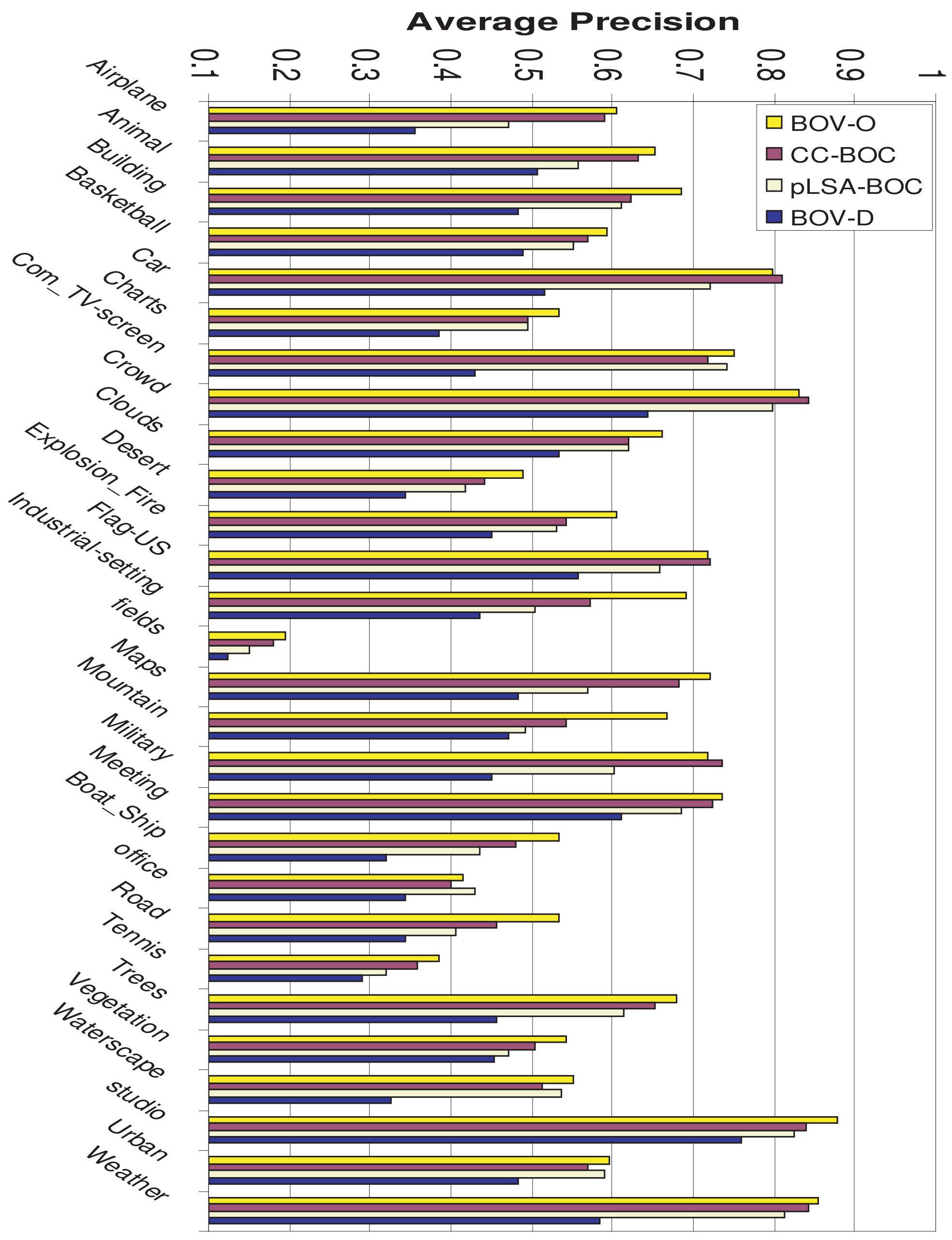


Figure 4.8: The AP for the 28 categories. BOV-O and BOV-D represents the BOV models with $N_v = 3,000$ and $N_v = 250$ respectively. CC-BOC and pLSA-BOC denotes the BOC model created by co-clustering and pLSA.

## 4.5 Conclusion

In this chapter, we propose a novel approach for scene modelling. The proposed method first extracts *intermediate concepts* from *visual-words* by using MMI co-clustering. Unlike $k$-means clustering, MMI co-clustering can preserve the mutual information of *visual-words* and images when clustering. Therefore, the more compact image representation can significantly improve the performance of classification. Also, in order to capture the spatial information of the *intermediate concepts*, the framework uses two spatial models, SPM and SCC. Experiment results show that both of these models can improve the classification accuracy significantly.

# CHAPTER 5: VISUAL RECOGNITION USING MULTIPLE HETEROGENOUS FEATURES

## 5.1 Introduction

In the previous two chapters we proposed two approaches to learn the semantic vocabularies for visual recognition by clustering in terms of minimizing the information loss function, and very promising results were obtained. Both of them only explore the clustering of homogeneous entities (e.g. clustering either *visual words* or images). Many problems in the field of computer vision, however, require analysis of entities that convey inherently different information, but are tied together due to explicit and implicit relationships between them. Such relationships, including homogenous and heterogenous, if properly identified, can play a crucial role in solving the problem at hand.

On the other hand, most existing approaches advocate the use of single features for recognition. However, we believe that though theoretically sound, the notion that a single feature will capture the range of complexity of visual recognition is pragmatically weak. Therefore, we address the specific issue of using multiple features for object and action recognition and propose a general framework for fusing information from complementary features.

In order to optimally combine these features, we develop a framework that allows for learning of explicit and implicit relationships between different classes of features in a principled manner. The framework is based on the concept of *Fiedler Embedding* [14], which is an algebraic method that explicitly optimizes the closeness criteria and is similar to the Laplacian Eigenmap [83]. It embeds diverse entities into a common Euclidian space, and thus enables the use of simple Euclidian distances for discovering relationships between features. For this purpose, the algorithm treats different entities as nodes in a graph, where weighted edges between the nodes represent the strength of the relationship between entities. The graph is then embedded into a $k$-dimensional space, subject to the criteria that similar nodes have Euclidean coordinates which are closer to each other. This is achieved by converting this constraint into a minimization problem whose solution is the eigenvectors of the graph Laplacian matrix.

The applicability of the framework is demonstrated on two tasks, namely *object recognition* and *action recognition*. In the object classification task, the entities consist of images, interest points, contours, and region segments. By embedding features and images into a common Euclidian space, we are able to discover homogenous and heterogenous relationships among them which are not evident otherwise, and we have used these relationships for improved object classification. The results are reported on the benchmark data set of Caltech-6. In the action task, we propose the use of two types of features: The first feature is a quantized vocabulary of spatiotemporal (ST) volumes (or cuboids) that are centered around 3D interest points in the video. The ST volumes are inherently local in nature, and therefore capture the local appearance and motion information. The second feature is a quantized vocabulary of spin-images [69], which aims to capture the spatiotemporal information of the actor by considering actions as 3D objects $(x, y, t)$ [12]. The 3D object itself is carved out by the contours of the individual performing the action. Note that the spin-image based features have not been used for action recognition before, and ours is the first method to explore their utility for this task. By embedding these entities into a common space, again we are able to discover explicit and implicit relationships between these entities that can be used for action recognition. The results are reported on the Weizmann data set [84] and IXMAS data set [30].

## 5.2 Fiedler Embedding

In this section, we present the details of the embedding procedure which is called Fielder Embedding. It was first proposed in [14] for the task of information retrieval from a document corpus. In this chapter we adapt this technique for image classification and learning of scene semantics, and show that how it can be used for discovering the relationships between disparate entities in a principled way. For image classification, this basically translates into discovering relationships between images and features. We start by describing the mathematical derivation of the embedding procedure, and for this purpose we use the nomenclature used by [14]. Later on we will briefly discuss its connection with the LSA technique.

Let $G = (V, E)$, where $V$ is a set of vertices and $E$ is a set of edges, represents a graph consisting of nodes representing different classes of features as illustrated in Figure 5.1 (or Figure 5.3). If two features $i$ and $j$ are related to each other, then we have an edge $(i, j)$ with a non-negative weight, $w_{i,j}$, between them. The more similar the features are to each other, the higher the weight. Our goal is to embed this graph into a low-dimensional Euclidian space, so that vertices with a high weight between them become closer to each other in this space. As a result, spatial proximity in this space can be used as a way to identify vertices that are similar to each other even if they do not have a direct edge between them (the implicit relationship). In posing this geometric embedding problem as an algebraic minimization problem, we seek points in a $k$-dimensional space that minimize the weighted sum of the square of the edge lengths. If $p_r$ and $p_s$ are locations of vertices $r$ and $s$, then the function can be written as,

$$\text{Minimize} \sum_{(r,s)\in E} w_{r,s} \mid p_r - p_s \mid^2, \qquad \text{(Eq. 5.1)}$$

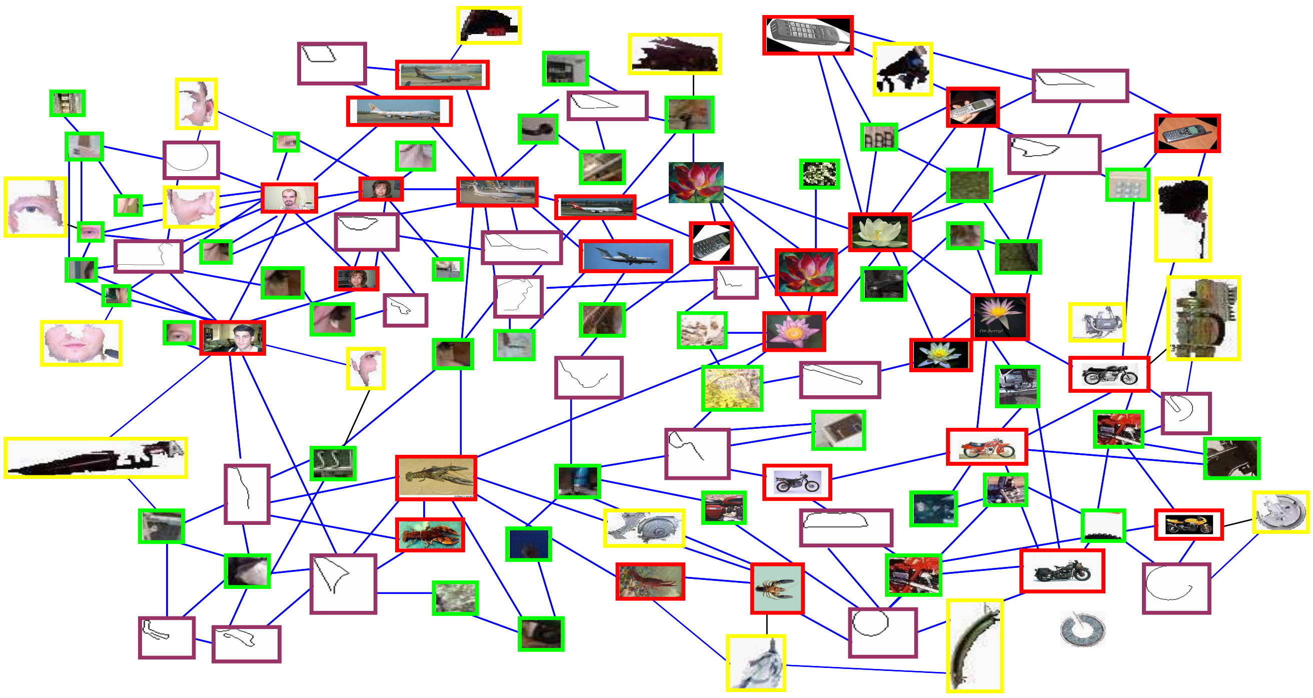


Figure 5.1: An illustration of the graph containing multiple entities as nodes. This includes images (red), SIFT descriptors (green), contours (purple) and regions (yellow). The goal of our algorithm is to embed this graph in a $k$-dimensional space so that semantically related nodes have geometric coordinates which are closer to each other. *(Please print in color)*

where $w_{r,s}$ represents the weight between the nodes $r$ and $s$. Expanding the above we get

$$\text{Minimize} \sum_{(r,s)\in E} w_{r,s}|p_r - p_s|^2 = \sum_{(r,s)\in E} w_{r,s} < p_r - p_s, p_r - p_s > \qquad \text{(Eq. 5.2)}$$

$$= \sum_{(r,s)\in E} w_{r,s}(p_r p_r^T + p_s p_s^T - p_r p_s^T - p_s p_r^T). \qquad \text{(Eq. 5.3)}$$

If the graph has $n$ vertices, and the target space has dimensionality $k$, then the positions of the vertices can be represented by an $n$ x $k$ matrix $X$. The Laplacian matrix $L$ of this graph can be described as:

$$L(i,j) = \begin{cases} -w_{i,j} & \text{if } e_{ij} \in E \\ \sum_k w(i,k) & \text{if } i = j \\ 0 & \text{otherwise,} \end{cases} \qquad \text{(Eq. 5.4)}$$

where $L$ is symmetric and positive semi-definite. Note that $L$ is nothing but the negative of the matrix of weights with diagonal values chosen to make the row-sums zero.

This will imply that $p_r$ or $p_s$ ($r = 1, 2, ..., n$ and $s = 1, 2, ..., n$) is a $k$-dimensional vector indicating the coordinate of the vertex in the $k$-dimensional space. Further expanding Equation 3, we get

$$\begin{aligned} \text{Minimize} \quad & (\sum_{s=2}^{s=n} w_{1,s}p_1p_1^T - \sum_{t=2}^{t=n} w_{1,t}p_tp_1^T) + (\sum_{s=1,s\neq 2}^{s=n} w_{2,s}p_2p_2^T - \sum_{t=1,t\neq 2}^{t=n} w_{2,t}p_rp_2^T) \\ \ldots & \\ + \quad & (\sum_{s=1,s\neq n-1}^{s=n} w_{n-1,s}p_{n-1}p_{n-1}^T - \sum_{t=1,t\neq n-1}^{t=n} w_{n-1,t}p_tp_{n-1}^T) \\ + \quad & (\sum_{s=1}^{s=n-1} w_{n,s}p_np_n^T - \sum_{t=1}^{t=n-1} w_{n,t}p_tp_n^T), \end{aligned}$$

where each term within a bracket belongs to the diagonal of the matrix $LXX^T$. This means that minimizing the above function is equivalent to minimizing the trace of $LXX^T$, which in itself is equivalent to minimizing the trace of $X^TLX$ due to the cyclic property of the trace of square matrices. Thus, our final minimization problem in terms of matrices $L$ and $X$ is

$$\text{Minimize} \quad Trace(X^TLX), \qquad \text{(Eq. 5.5)}$$

$$\text{(i)} \quad \text{for} \quad i = 1, ..., k \quad X_i^T1^n = 0, \qquad \text{(ii)} \quad X^TX = \Delta. \qquad \text{(Eq. 5.6)}$$

The first constraint makes the median of point sets in the embedding space to be at the origin, while the second constraint avoids the trivial solution of placing all the vertices at the origin. In the above equation $1^n$ is a vector of $n$ ones while $\Delta$ is a diagonal matrix of $\delta_i$ which are some positive values. As shown in [14], the solution to the above minimization is $X = \Delta^{1/2}[Q_2, ..., Q_{k+1}]$, where $Q$ is a matrix of normalized eigenvectors of $L$ sorted in a non-decreasing order based on the eigenvalues $\lambda_i$ of $L$. This implies that the coordinates of vertex $i$ are just the *ith* entries of eigenvectors $2, ..., k+1$ of $L$. This solution is referred to as the *Fiedler embedding* of the graph.

### 5.2.1 Fiedler Embedding and LSA

The Fiedler embedding can be related to the popular technique of Latent Semantic Analysis (LSA), which is commonly used for exploring relationships between two types of entities, for example, interest points and images. This relationship can be described in a graph theoretic manner as follows: The LSA technique operates on a bi-partite graph where one partition consists of interest points and the other partition consists of images. Let $A$ to refer to the Interest point-image matrix which is of size $m \times n$, where $m$ is the number of interest points or visual words and $n$ is the number of images. Then, LSA is performed by computing the singular values decomposition (SVD) of $A$ ($A = U\Sigma V^T$). By retaining the first $k$ eigenvalues of $\Sigma$ we generate a $k$ rank approximation of $A$. This truncated SVD can be considered as generating a $k$-dimensional embedding of SIFT descriptors and images. However, in this case interest points and images have their separate $k$-dimensional spaces, which implies that the interest point and the image coordinates are not related to each other. In other words, interest points and images live in their own $k$-dimensional space; therefore, Euclidian distances cannot be used to measure the strength of relationships between them. The other difference to observe is that LSA can only handle a very limited type of graph (i.e., bi-partite graphs), and therefore, is not extendable to the problems where more than two entities are involved. It also limits the edges to be between the partitions and not among the nodes in the partition, and therefore cannot handle a general graph with arbitrary edges (or relationships). On the other hand, Fiedler embedding does not impose such restrictions and allows us to capture relationships between a richer set of features in a seamless fashion. Furthermore, by embedding everything into a common $k$-dimensional space, it allows us to compare features of different dimensions and types using Euclidian distances, which is an added advantage over LSA.

## 5.3 Object Classification Framework

In this section, we present technical details of the object classification framework that is built upon the *Fiedler embedding* procedure. As we know, extraction of rich features in an image and determining the relationships is a challenging task. There can be numerous relationships among the features, and some of them can be explicit while others are implicit. Our aim is to integrate the image data with multiple features involving explicit and implicit relationships into a common space.

The main steps of the framework are: i) Learning visual vocabularies of interest points, contours, and region segments, ii) Construction of the Laplacian matrix from the training corpus, and iii) Embedding and computation of entity relationships using the nearest neighbor search. The algorithmic steps of the framework are described in the following table.

**Objective** Given $N_I$ training images, embed entities (images, interest points, contours, and regions) into a $k$-dimensional space such that the similarity between any two entities can be measured by the Euclidean distance.

**Algorithm**

(i) Quantize visual information by learning visual vocabularies:

- Learn interest point vocabulary of size $N_P$ by using SIFT descriptors of patches around interest points.
- Learn shape vocabulary of size $N_C$ by using contours.
- Learn region vocabulary of size $N_R$ by using region segmentation.

(ii) Construct an $(N_P + N_C + N_R)$ x $N_I$ Feature-Image co-occurrence matrix $\mathcal{S}$ by counting the corresponding feature frequency in each training image.

(iii) Weight $\mathcal{S}$ by *tf-idf* to compute a weighted co-occurrence matrix $\mathcal{S}'$

(iv) Compute Laplacian matrix $L$:

- Image-Feature blocks of $L$ are populated by directly using corresponding values from $\mathcal{S}'$.
- Image-Image and Feature-Feature similarity blocks are populated by computing Inner Product on matrix $\mathcal{S}'$

(v) Perform eigen-decomposition $L = \mathbb{V}^T\mathbb{D}\mathbb{V}$ where $\mathbb{V}$ are the eigenvectors and $\mathbb{D}$ are the eigenvalues sorted in the ascending order.

(vi) Construct a $k$-dimensional space: Pick $k$ eigenvectors corresponding to $k$ smallest eigenvalues excluding zeros $\mathbb{U}=\{u_1, u_2, ..., u_K\}$. These eigenvectors act as the basis of the $k$-dimensional space.

(vii) Entity Mapping: Position of an entity $\mathbf{q}$ in the $k$-dimensional space is computed as $\mathbb{D}^{1/2}\mathbb{U}^T\mathbf{q}/\|\mathbf{q}\|$.

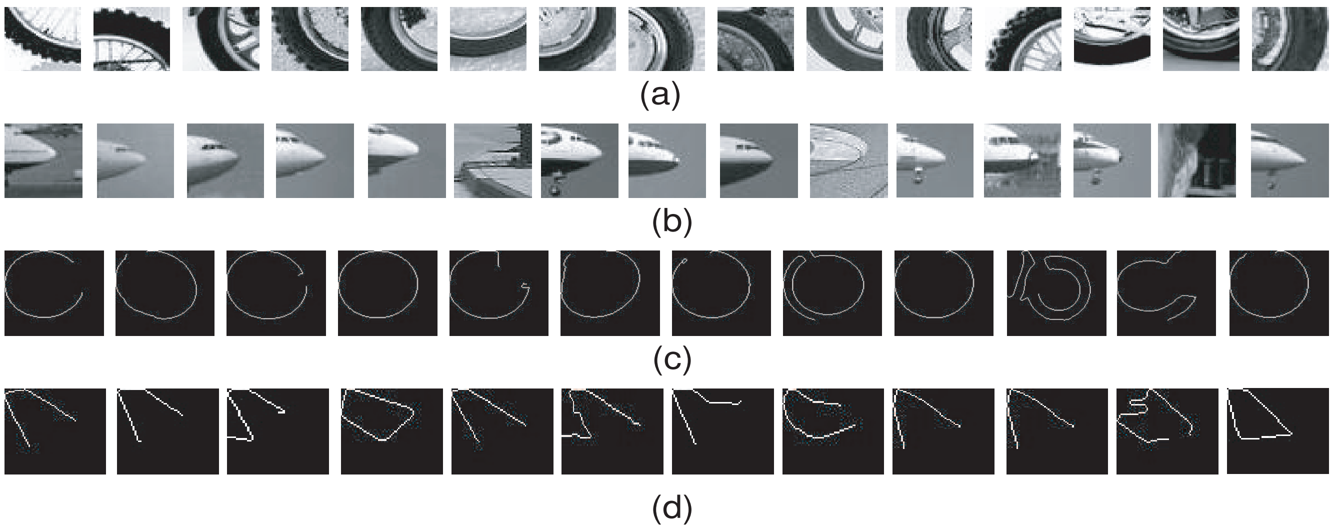


Figure 5.2: The figure shows two visual words each from the interest point, contour and region vocabularies. (a)-(b) Two words belonging to the interest point vocabulary. (c)-(d) Two words belonging to the contour vocabulary. (e)-(f) Two words belonging to the region vocabulary.

**Interest Point Vocabulary:** The interest point vocabulary is constructed by extracting patches around the selected interest points and representing them using 128-dimensional SIFT descriptor. The initial interest point detection is based on the curve based representation proposed by [101]. We start by applying the Canny edge detector to the images. Interest points are selected by randomly sampling the edge pixels. The circular patches are extracted from around the pixels which are then mapped to ellipses to make them affine invariant. The patches are sampled at different scales uniformly from the interval specified by the minimum and maximum allowable scale. Next, a 128 dimensional SIFT descriptor [79] is computed for each patch. The SIFT descriptors are quantized into $N_P$ clusters using $k$-means clustering, where the mean SIFT descriptor of each cluster is used as the representative interest point for that cluster. In this chapter, we set $N_P$ = 2,000. Figure 5.2(a)-(b) shows two visual words learned by this procedure.

**Learning Contour Vocabulary:** Extraction and clustering of contours is performed in a semi-supervised manner by assuming that the bounding boxes of the target objects in the training corpus are known. We first detect image contours using the Canny edge detector, where a contour segment $\mathcal{C}$ is considered as an edge chain which is an ordered sequence of $e$ edgels, i.e., $\mathcal{C} = \{\mathbf{c}(c_1, c_2)\}$. The similarity between two contours is computed using the Chamfer Distance [9], which is considered a robust measure for contour similarity. Formally, given two contours $\mathcal{C} = \mathbf{c}$, $\mathcal{C}' = \mathbf{c'}$, and the current position $\mathbf{y}$ at which similarity is to be computed, the Chamfer distance between them can be evaluated as:

$$distance(\mathcal{C}, \mathcal{C}', \mathbf{y}) = \frac{1}{|\mathcal{C}|} \sum_{\mathbf{c} \in \mathcal{C}} min_{\mathbf{c'} \in \mathcal{C}'} \parallel (\mathbf{c} + \mathbf{y}) - \mathbf{c'} \parallel, \qquad \text{(Eq. 5.7)}$$

where $|\mathcal{C}|$ is the cardinality of the set $\mathcal{C}$. Intuitively, this measure can be considered as a correlation between two contours, where one contour is shifted over the other to compute all possible correlations.

Next, contour clustering is used to learn the representative contours of objects. For this purpose we use agglomerative clustering to group contours into $N_C$ clusters. During this procedure, we merge two clusters whenever the distance between them is less then a predefined threshold of $Th_c = 0.2$. Each cluster is represented by a mean contour called the visual shape word. Finally, we map each contour segment in a given image to one of the visual shape words. We compute the distance of one visual word to the image by substituting the item $\mathcal{C}'$ in Equation Eq. 5.7 with $\mathcal{I}$, which represents all the edge chains in the image $I$. This step can be efficiently performed by first computing the distance transform of the edge map of the image [8]. A visual word is considered to be present in the image when the distance is under some threshold, $Th_I = 0.8$. Figure 5.2(c)-(d) shows two visual shape (contour) words learned by this procedure.

**Learning Region Vocabulary:** Region segmentation for images in the training corpus is obtained by using the mean-shift segmentation algorithm. Each region segment is then represented by an 18-dimensional feature vector, which includes a 9 dimensional feature vector representing the first three color moments in HSV space. The other 8 dimensions represent an 8-dimensional quantized edge orientation histogram, while the last dimension represents the normalized area of the region segment. The region vocabulary is then constructed by clustering the 18-dimensional feature vectors for all of the region segments into $N_R$ clusters using a $k$-means clustering algorithm. For our experiments, we generated a region vocabulary of size $N_R = 500$. Figure 5.2(c)-(d) shows two region words learned by this procedure.

### 5.3.1 Constructing Feature Laplacian Matrix

For embedding, the input data is represented as a Laplacian matrix $L$, which is a symmetric matrix of the weights between entities as described in Equation Eq. 5.4. For the scenario at hand, we have four types of entities, namely images, interest points (represented by the interest point vocabulary), contours (represented by the contour vocabulary), and region segments (represented by the region vocabulary). Therefore the Laplacian matrix will have the following block structure:

$$L = \begin{pmatrix} D_1 & P^T & F^T & B^T \\ P & D_2 & H^T & J^T \\ F & H & D_3 & Q^T \\ B & J & Q & D_4 \end{pmatrix}, \tag{Eq. 5.8}$$

where $D_1$, $D_2$, $D_3$, $D_4$, $P$, $F$, $B$, $H$, $J$, and $Q$, are the block matrices of image-image, interest point-interest point, contour-contour, region-region, image-interest point, image-contour, image-region, interest point-contour, interest point-region, and contour-region similarity, respectively. The dimensions of these sub-matrices depend on the values of $N_I$, $N_P$, $N_C$ and $N_R$.

Note that ideally the entities (images, interest points, contours, regions) representing the same semantic concept will come closer to each other in the $k$-dimensional space only if the strength of relationship between them is evaluated using the same similarity measurement. In practice, however, we measure the similarity between different types of entities using different similarity measures. For instance, the image-image similarity can be evaluated in term of the histogram intersection of patches or contours. However, histogram intersection cannot be used to measure the similarity between an image and a patch. The occurrence frequency of a patch in an image might be a better similarity measure in this case. Thus, the bottom line is that we can not compare how similar one image $I_1$ is to an image $I_2$ and a patch $P_1$ using a single type of the similarity measure. This raises the concern that since different similarity measures have different scales, it will become difficult to embed, say images $I_1, I_2$, and interest point $P_1$ into a common $k$-dimensional space since we will not be able to know which of them should be closer to $I_1$. The solution lies in the proper normalization of these different similarity measures.

In our implementation, we use "Term Frequency-Inverse Document Frequency" ($tf$-$idf$) to measure the similarities between images and features (interest points, contours, regions) i.e., the values in matrices $P$, $F$ and $B$. Suppose the size of a visual vocabulary is $N_\mathrm{x}$ (subscript $x$ can take on values $P$, $C$, and $R$), then an image can be represented by a $N_\mathrm{x}$-dimensional vector $(t_1, t_2, ..., t_i, ...t_{N_\mathrm{x}})$, where each element $t_i$ is this vector is computed as

$$t_i = \frac{s_{id}}{s_d} log \frac{N_I}{s_i}. \qquad \text{(Eq. 5.9)}$$

Here, $s_{id}$ is the frequency of feature $i$ in image $d$, $s_d$ is the total number of features in image $d$, $s_i$ is the number of features $s_i$ in the entire data set, and $N_I$ is total number of images.

For image-image or feature-feature similarity (i.e., matrices $D_1$, $D_2$, $D_3$, and $D_4$), different similarities were tried such as histogram intersection, cosine similarity, and inner product. We observed that although histogram intersection and cosine similarity were able to group similar images together, they were not able to group semantically similar images and features together. On the other hand, the Inner Product method was able to map similar images and features closer to each other in addition to mapping similar images together. We believe the reason was that the histogram intersection and the cosine similarity gave disproportional weights to the image-image or the feature-feature similarity as compared to the image-feature similarity. In order to verify the observation, we checked the mean value of image-image blocks ($D_1$), feature-feature blocks ($D_2$, $D_3$, $D_4$) and the image-feature blocks ($P$, $F$, $B$, $J$, and $Q$) in the $L$ matrix, and observed that the Inner Product achieved similar mean values for all the blocks, implying that the weights in Image-Image, Image-Feature and Feature-Feature similarity blocks are equal to each other. Therefore, using inner-product, the images and features can be mapped to the targeted $k$-dimensional space in a proper fashion.

### 5.3.2 Embedding

Once the Laplacian matrix $L$ is constructed, the Euclidian coordinates of each image and feature are computed by performing eigen-decomposition of the Laplacian matrix. First, $k$ eigenvectors corresponding to $k$ smallest eigen-values excluding zeros are selected. These eigenvectors act as the basis of the $k$-dimensional space. The $k$-dimensional coordinate $X$ of each entity (image or feature) is computed as described in Step 7 of the algorithm described in Section 5.3. Now, the semantic relationships between images and features can be computed by performing clustering on the coordinate values $X$ of each entity. We can do this because now $X$ is of the same dimension for all entities, irrespective of their input representation. The discovered relationships are then used for improved object classification.

## 5.4 Action Recognition Framework

In this section, we describe our action recognition framework which utilizes the *Fiedler Embedding.* In our work, we choose two types of features, *Spatial-Temporal* (ST) features and *Spin-Images.* These features normally capture the strong variation of the data in spatial and temporal directions that are caused by the motion of the actor. However, ST features contain only the local appearance and motion information, and therefore ignore the shapes of actors. To capture the holistic shape information, we consider actions as 3D objects in $(x, y, t)$ and compute their spin-images. Once the features are computed for the given action

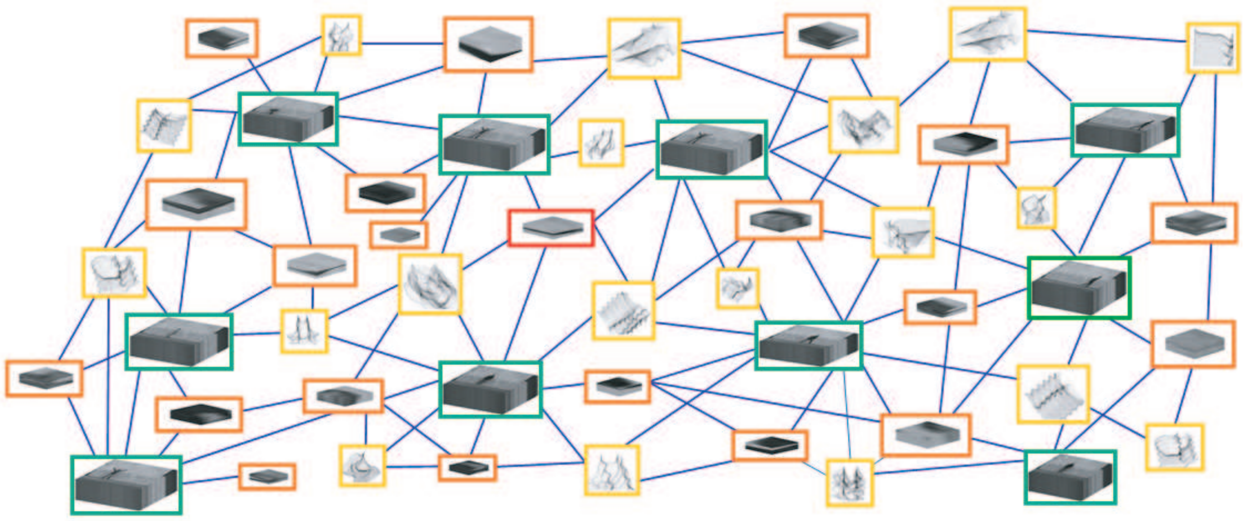


Figure 5.3: An illustration of the graph containing multiple entities as nodes. This includes ST features (red), Spin-Image features (yellow) and action videos (green). The goal of our algorithm is to embed this graph in a $k$-dimensional space so that similar nodes have geometric coordinates which are closer to each other.

corpus, we can construct a graph whose nodes consist of the above mentioned entities, and edges between the nodes capture the strength of relationship between the entities. The graph is pictorially described in Figure 5.3. we use the *Fiedler Embedding* to discover the relationships among features by projecting them into a common Euclidian space.

The framework is similar to that of object recognition. The main steps are: i) Learning of visual vocabularies from ST features and Spin-Images, ii) Construction of Laplacian matrix from the training videos, iii) Embedding and grouping of features. The algorithmic description of these steps is provided in Table 6.1.

### 5.4.1 Feature Extraction and Representation

We extract two types of features from the action videos: spatiotemporal features and spin-image features. In this section, we focus on the spin-image features. As for the spatiotemporal features, please refer to chapter 3.

**Objective:**Given $N_v$ training action videos, embed all entities (ST features, Spin-Image features, Videos) into a common $k$-dimensional space.

(i) Quantize visual information by learning visual vocabularies:
- Learn vocabulary of ST features of size $N_{st}$.
- Learn shape vocabulary based on spin-images of size $N_{si}$.

(ii) Construct a $(N_{st}+N_{si})*N_v$ Feature-Action co-occurrence matrix, $\mathcal{S}$, by counting the frequency of features in each action video.

(iii) Weigh $\mathcal{S}$ by *tf-idf* to obtain a weighted co-occurrence matrix $\mathcal{S}'$.

(iv) Construct Laplacian matrix $L$ as :
- Video-Feature similarity blocks of $L$ are computed directly from the corresponding value from $\mathcal{S}'$.
- Video-Video and Feature-Feature similarity blocks are computed by using the Inner Product of rows of matrix $\mathcal{S}'$.

(v) Perform eigen-decomposition of $L$ such that $\mathcal{L} = \mathbb{V}^T\mathbb{D}\mathbb{V}$ where $\mathbb{V}$ is a set of eigenvectors and $\mathbb{D}$ contains the eigenvalues sorted in ascending order.

(vi) Construct a $k$-dimensional space. Select $k$ eigenvectors corresponding to the $k$ smallest eigenvalues, excluding the zeros, say $\mathbb{U}=\{u_1, u_2, ..., u_k\}$, which is the basis of the $k$-dimensional space.

(vii) Map entities: A video $\mathbf{q}$ is mapped to the $k$ space as: $\mathbb{D}^{1/2}\mathbb{U}^T\mathbf{q}/\|\mathbf{q}\|$.

Table 5.1: Main steps of the action recognition framework.

**Spin-Image Features:** Spin-Images have been successfully used for 3D object recognition [69]. However, it has not been used for action recognition before. For actions, the spin-images can provide a more richer representation of how the local shape of the actor is changing with-respect to different reference points. These reference points may correspond to different limbs of the human body. For extracting the spin-images, we consider an action video as a 3D object. There are two main steps of the procedure: 1) Generating Action Volume, and 2) Spin-Image Extraction.

*Generating the Action Volume:* We create a 3D action volume by stacking the sequence of contours of the actor. For this purpose, we first apply the background substraction algorithm to get the contour $\mathcal{C}^t$ at frame $t$. To generate the 3D action volume, we find the

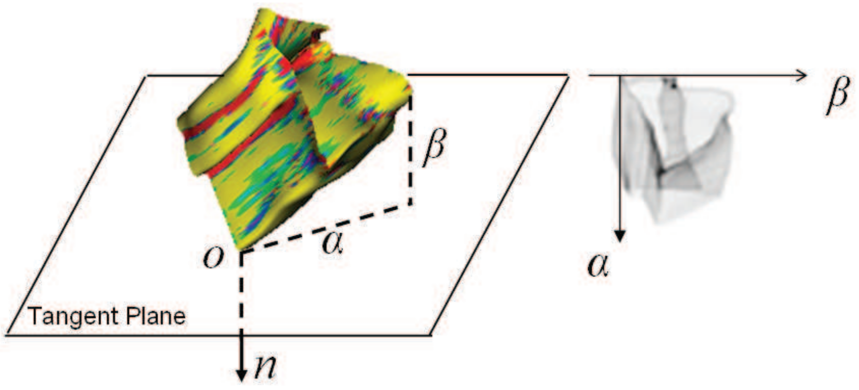


Figure 5.4: Left: the $(\alpha, \beta)$ coordinates of a surface point relative to the *orientated point* $O$. Right: the spin-image centered at $O$.

correspondences between points of two consecutive contours $\mathcal{C}^t$ and $\mathcal{C}^{t+1}$ using the graph theoretic approach proposed in [12]. Suppose $L$ and $R$ are two point sets corresponding to $\mathcal{C}^t$ and $\mathcal{C}^{t+1}$ respectively, and we create a bipartite graph with $|L|+|R|$ vertices. The weights of the edges connecting $L$ and $R$ are estimated from three items: proximity, orientation similarity, and shape similarity. Assume $c_i$ and $c_j$ are two vertices from $L$ and $R$ respectively. The proximity $d_{ij}$ between them is the $L2$ norm of their 3D coordinates $(x, y, t)$. The orientation similarity $\alpha_{ij}$ is the angle between the spatial norms of the vertices, and the shape similarity $\xi_{ij}$ is estimated from the neighbors. Then the weights are computed as

$$w_{ij} = exp(-\frac{d_{ij}^2}{\sigma_d^2})exp(-\frac{\alpha_{ij}^2}{\sigma_\alpha^2})exp(-\frac{\xi_{ij}^2}{\sigma_\xi^2}). \qquad \text{(Eq. 5.10)}$$

*Extracting Spin-Images:* Johnson et al. [69] introduced the spin-images for recognizing objects in complex 3D scenes. A spin-image (SI) is generated by projecting the mesh vertices onto a tangent plane with respect to a reference surface point, called the *orientated point*. The spin-image is an object centered feature, hence it is scale, rotation, and pose invariant.

Figure 5.4 illustrates the process of projecting a surface point onto a tangent plane with respect to the *orientated point* $O$. All the surface points are indexed by the radius $(\alpha)$ from

$o$ and the depth ($\beta$) from the tangent plane of $O$. The projection function is expressed as,

$$\alpha = \sqrt{\|x-o\|^2 - (n\cdot(x-o))^2}, \quad \beta = n\cdot(x-o),$$

where $x$ and $o$ are the 3D coordinates of the surface point and the *orientated point* $O$. Hence, all the 3D surface points are projected onto 2D plane. To produce a spin-image, we need to quantize ($\alpha$,$\beta$) and build a 2D histogram, which is called the spin-image. There are several important parameters controlling the generation of the spin-image. The support length, which defines the size of the spin-image, determines the locality of the spin-image. With a larger support length, spin-image can capture the entire object shape, and a smaller support length provides local shape information. Another important parameter is bin size, which determines the discrimination of the spin-image. Larger bin size will cause all points to fall into the same bin, while small size will separate the neighboring points. In this chapter, the bin size is set as the average length of the mesh resolution. Besides, uniform sampling is necessary for matching two shapes. Fig. 5.5 shows the action volumes and their selected corresponding spin-images. Instead of attempting pairwise matching of spin-images to match two actions, we use the bag of spin-images strategy. For this purpose, we first apply PCA to compress the dimensionality of the spin-image, and then use $K$-means to quantize them. We call the group of spin-images as a *video-word*. Finally, the action is represented by the bag of *video-words* model.

### 5.4.2 Construction of the Laplacian Matrix

The input for *Fiedler Embedding* is the Laplacian Matrix $\mathcal{L}$, which is a symmetric matrix constructed according to equation Eq. 5.4. In our case, we have three types of entities which are: ST features (ST), Spin-Image features (SI) and videos of actions. Therefore, $\mathcal{L}$ has the

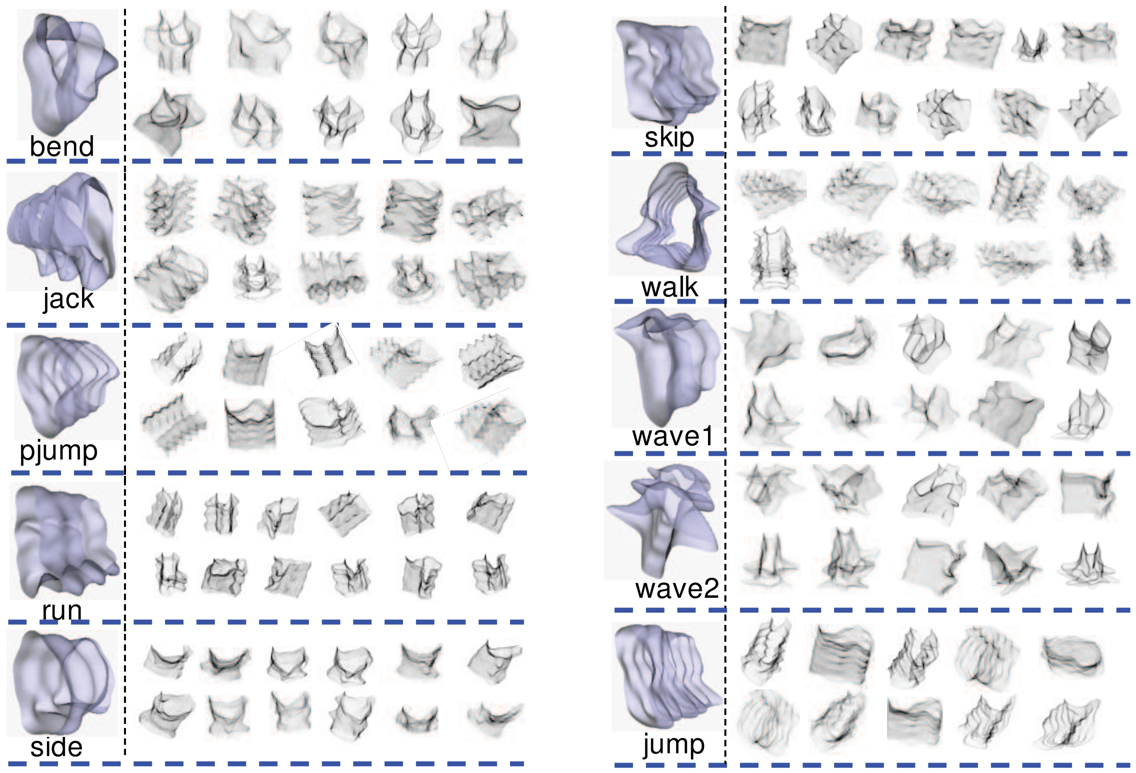


Figure 5.5: Some 3D (x,y,t) action volumes (the first column) with some of their sampled spin-images (red points are the *orientated points.*).

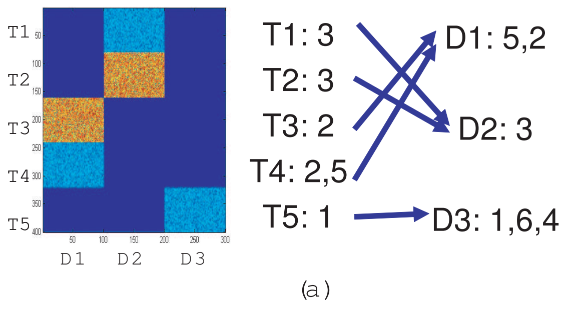
following block structure:

$$\begin{pmatrix} D_1 & E^T & F^T \\ E & D_1 & H^T \\ F & H & D_3 \end{pmatrix}, \qquad \text{(Eq. 5.11)}$$

where $D_1$,$D_2$,$D_3$,$E$, $F$, $H$, respectively, denote the block matrix of Video-Video, ST-ST, SI-SI, Video-ST, Video-SI, and ST-SI. In principle, the relationship matrix can be expressed by any measurement (e.g. similarity, co-occurrence).

Theoretically, *Fiedler Embedding* maps the entities into a common $k$-dimensional space by evaluating the relationship values between the entities. Therefore, the entities which have a stronger relationship might be placed at closer positions if the strength of relationship is evaluated using the common measurement. In other words, semantically similar entities stay closer to each other in the $k$-dimensional space. In practice, however, we must measure the similarity between different types of entities. For instance, the Video-Video similarity can be evaluated in term of the histogram intersection of features. Compared to histogram intersection, the frequency of occurrence is a much better measure for Video-ST or Video-SI similarity. Therefore, we can not directly compare how similar one action video $V_1$ is to another action video $V_2$ and Spin-Image $SI_1$ using the same type of measurement. This is because different measurements have different ranges. The solution lies in the proper normalization of different measurements. Similar to object recognition, we also measure the similarities between actions and features (ST features or Spin-Image features) by *Term Frequency-Inverse Document Frequency*$(tf - idf)$.

For video-video or feature-feature similarities (such as the blocks $D_1$, $D_2$, and $D_3$), we tried several different approaches e.g. Histogram Intersection, Cosine similarity, and Inner Product. We observed that Histogram Intersection and Cosine can group similar actions together, but they were unable to group similar actions and features together. However, the Inner Product method can group the similar actions and features into the same cluster. We conjecture that this may be due to Histogram Intersection and Cosine similarity assigning disproportional weights to the video-video or feature-feature similarity, as compared to the video-feature similarity. In order to verify our conjecture, we checked the mean value of video-video block ($D_1$), feature-feature blocks ($D_2$ and $D_3$), and video-feature blocks ($E$,$F$,$H$) of the $\mathcal{L}$ matrix. Inner Product achieved close mean value for the three types of blocks, so it can treat the video-video, video-feature and feature-feature equally when assigning the weights. Thus, the action videos and features are more accurately mapped to the $k$-dimensional space using the Inner Product.



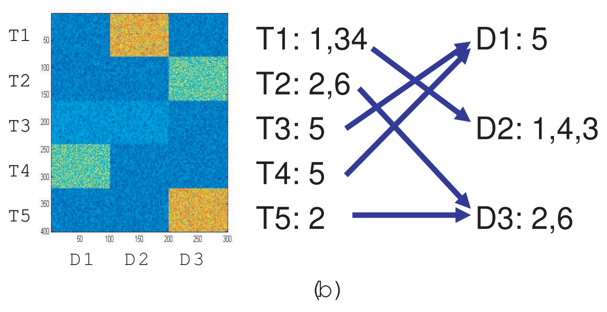


Figure 5.6: Clustering of entities in the $k$-dimensional embedding space. The entities are three images categories $D1$, $D2$, and $D3$, and five feature types $T1$, $T2$, $T3$, $T4$, and $T5$. The synthetically generated co-occurrence table between features and images is shown on the left side. While the graph represents the assignments of feature types and image category to the clusters in the 3−dimensional embedding space.

## 5.5 Experiments and Discussion

In this section, we present systematic performance analysis of the proposed framework along three lines: First, the algorithm is verified on a synthetically generated data set. Second, we apply the proposed approach on the widely used Caltech-6 data set and demonstrate the qualitative and quantitative results against a number of baselines. Third, we apply our algorithm on the task of action recognition.

### 5.5.1 Synthetic Data Set

The goal of performing experiments on a synthetically generated data set is to show that our proposed framework is capable of grouping semantically similar entities into common clusters in the $k$-dimensional embedding space. The synthetically generated data set has three image categories, represented by $D1$, $D2$, and $D3$; and five types of features, represented by $T1$, $T2$, $T3$, $T4$, and $T5$. Each feature type has a different distribution for each image category. In addition, each feature type has a vocabulary of size 80, which means the total number of features in the data set are 400, while each image category consists of 100 images. To carry out the embedding, the first step is to construct Feature-Image co-occurrence table as follows:

To populate the co-occurrence table, each feature type is considered dominant in a particular image category, where the frequency of features is considered to be generated by a Gaussian distribution. The mean and variance of the distribution are kept the same for all feature types. This step returns a co-occurrence table which looks like the one shown in Figure 5.6(a). Next, we construct the Laplacian matrix $L$ based on this co-occurrence table. For this data set, cosine similarity is used to measure the image-image relationship, by using the corresponding columns of the co-occurrence table as vectors in the cosine similarity measure. The feature-feature similarity is computed in the same way, except that the corresponding rows of the co-occurrence table are used as vectors in the cosine similarity measure. However, the image-feature similarity is evaluated by normalizing the co-occurrence values themselves.

Next, the proposed framework is applied to map all entities (images and features) into a common $k$-dimensional space by computing the eigenvectors of $L$, and computing the Euclidian coordinates of each entity from these eigenvectors. The value of $k$ was set to 3 for this purpose. Clustering is then performed on the $3-$dimensional coordinates of all mapped entities, and they are grouped into 6 clusters. The interpretation of the clustering results can be explained through the use of Figure 5.6. Here, Figure 5.6(a) shows the initial 400 x 300 feature-image co-occurrence table. It can be observed that the dominant types of features for categories $D1$,$D2$, and $D3$ are $\{T3, T4\}$, $\{T1, T2\}$, and $\{T5\}$, respectively. The corresponding graph in Figure 5.6(a) shows the mapping of feature types and image categories to one of the 6 clusters in the 3-dimensional embedding space. A symbol $(T1 : 3)$ means that all the features in $T1$ are assigned to cluster 3, while a symbol $(D1 : 5, 2)$ means that images from category $D1$ are either assigned to cluster 5 or cluster 2. A glance at these mappings shows that the results are exactly the same as predicted by the ground truth. For example, we have a mapping $(T4 : 2, 5)$, which means the features of type $T4$ are either assigned to cluster 2 or cluster 5. Since $T4$ is dominant in the image category $D1$, the images from $D1$ should also either fall into cluster 2 or cluster 5 for the embedding space to be semantically meaningful. This is exactly what is done by our algorithm. Similarly, feature types $\{T1, T2\}$ and images $D1$ are related to each other as reflected by the co-occurrence table; therefore, the embedding procedure should bring them closer to each other. Again this is reflected by the same cluster label, that is, cluster 3, assigned to all three entities. This emphasizes the fact that the Euclidian space that we are constructing is meaningful and can be used to discover homogenous and heterogenous relationships.

### 5.5.2 Caltech Data Set: Object Recognition

The second set of experiments is conducted on the widely used Caltech-6 data set. The aim is to show the efficacy of our framework against a number of well chosen baselines, which will emphasize the importance of discovering homogenous and heterogenous relationships between entities for improved classification. We have used four object categories for our experiments: faces (450 images), car rear (526 images), airplanes (1074 images), and motorbikes (826 images). We have learnt an interest point vocabulary of $N_P = 2,000$ visual words by randomly selecting 100 images from each category. The contour vocabulary is learnt by randomly selecting 50 images from each category with known bounding boxes around the target object. Only the contours that have a length larger than 60 pixels are selected for clustering using the procedure described in Section 5.3. This resulted in $N_C = 158$ contour clusters. For each cluster, we picked the contour which has the smallest distance to all the members of the cluster as the representative contour of the cluster. The region vocabulary of size $N_R = 500$ is constructed by using the same images that are employed for learning the interest point vocabulary. Next, 100 images are selected randomly from each category as training images for constructing the co-occurrence table. Using this table we computed the Laplacian matrix and eventually the $k$-dimensional embedding coordinates of all images and features.

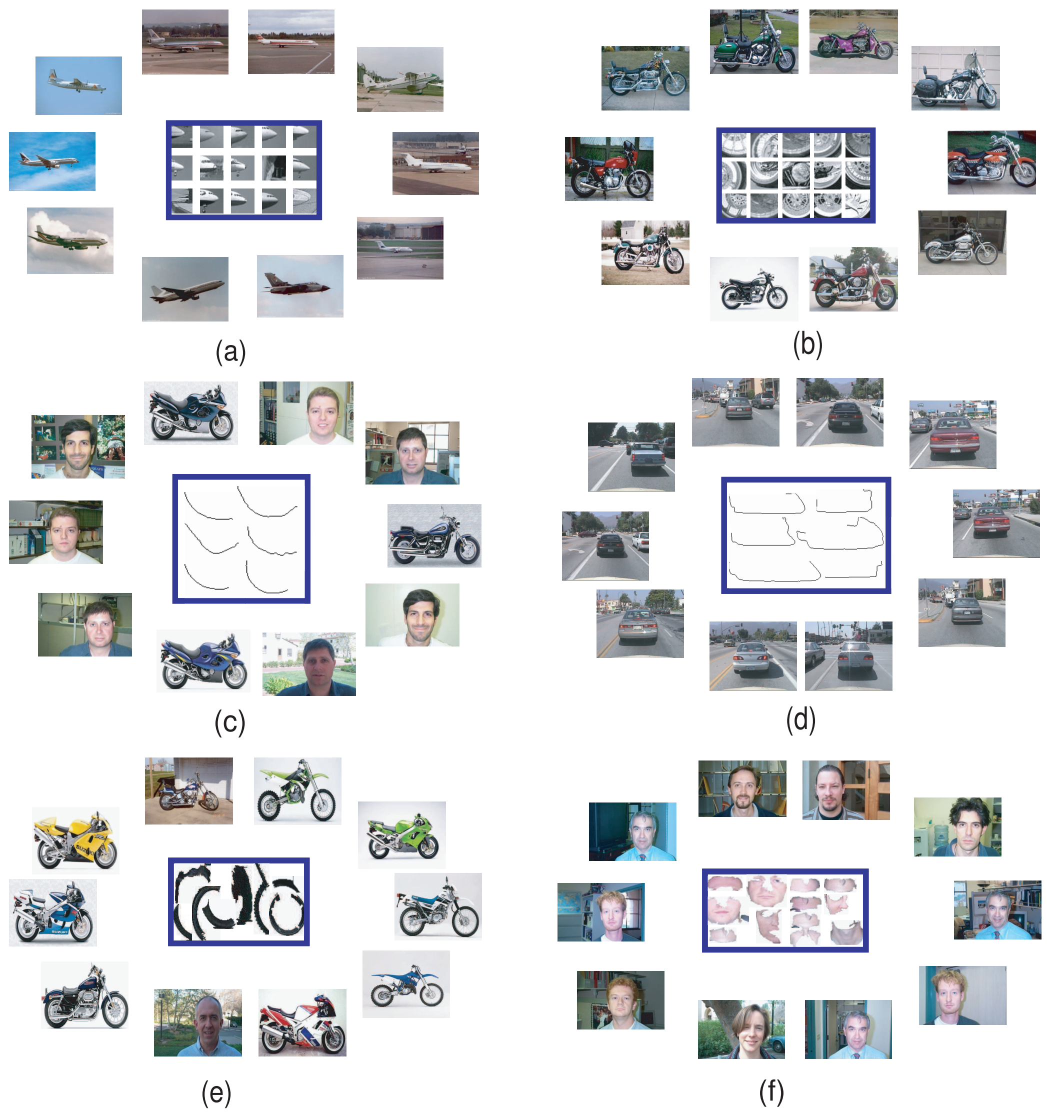


Figure 5.7: Qualitative results when the query is a feature and the output is a set of images. (a)-(b) Query: Interest point, Output: Ten nearest images. (c)-(d) Query: Contour, Output: Ten nearest images. (e)-(f) Query: Region, Output: Ten nearest images.

#### 5.5.2.1 Qualitative Results

The purpose of this experiment is to verify the semantic meaningfulness of the constructed embedding space. In addition, we demonstrate the unique strength of our proposed framework: it can map semantically similar entities to the same cluster in the $k$-dimensional embedding space, irrespective of the type of the entity. This is done by selecting a *query* entity and then returning nearby entities from the $k$-dimensional embedding space using Euclidian distances. If the embedding space is meaningful, then the returned nearby entities should have a relationship with each other.

A number of output combinations over entities are possible for each query entity. In the first result shown in Figure 5.7, we only return the nearby images. For instance, in Figure 5.7(a), the query entity is an interest point (from the learnt visual vocabulary) shown in the center rectangle. This interest point is representing the tip of the nose of the airplane. The 10 nearest images to this interest point from the $k$-dimensional space are shown around the interest point. It can be observed that all of the images are airplane images, although the distance was computed for other objects (motorbike, car rear, faces) as well. This tells us that in the embedding space we are able to map semantically related images and interest points to nearby regions. Therefore, we can use Euclidian distances to compute strength of relationships between them. Since all the retrieved images belong to the airplane class, it also means that the query interest point is only dominant in the airplane class. Another result of interest point based query is shown in Figure 5.7(b) with very similar interpretation of the results.

Next, we used contour as a query entity and again returned the 10 nearest images from the embedding space as shown in Figure 5.7(c) and (d). In Figure 5.7, the retrieved images belong to the face and motorcycle classes, which means this type of contour is shared by these two classes of objects. This is evident from the structure of the contour which is capturing the shape of the face around the chin area as well as wheel of a motorcycle. However, the query contour in Figure 5.7(d) is representing the shape of the rear of a car; therefore, all the retrieved images are from the car rear class. Both of these results further show the strength of the framework, where one only has to use a single contour or an interest point to retrieve relevant images based on Euclidian distances. This is very useful in the presence of occlusion or partial object views. Moving on, in Figure 5.7(e) and (f) we used regions as a query entity

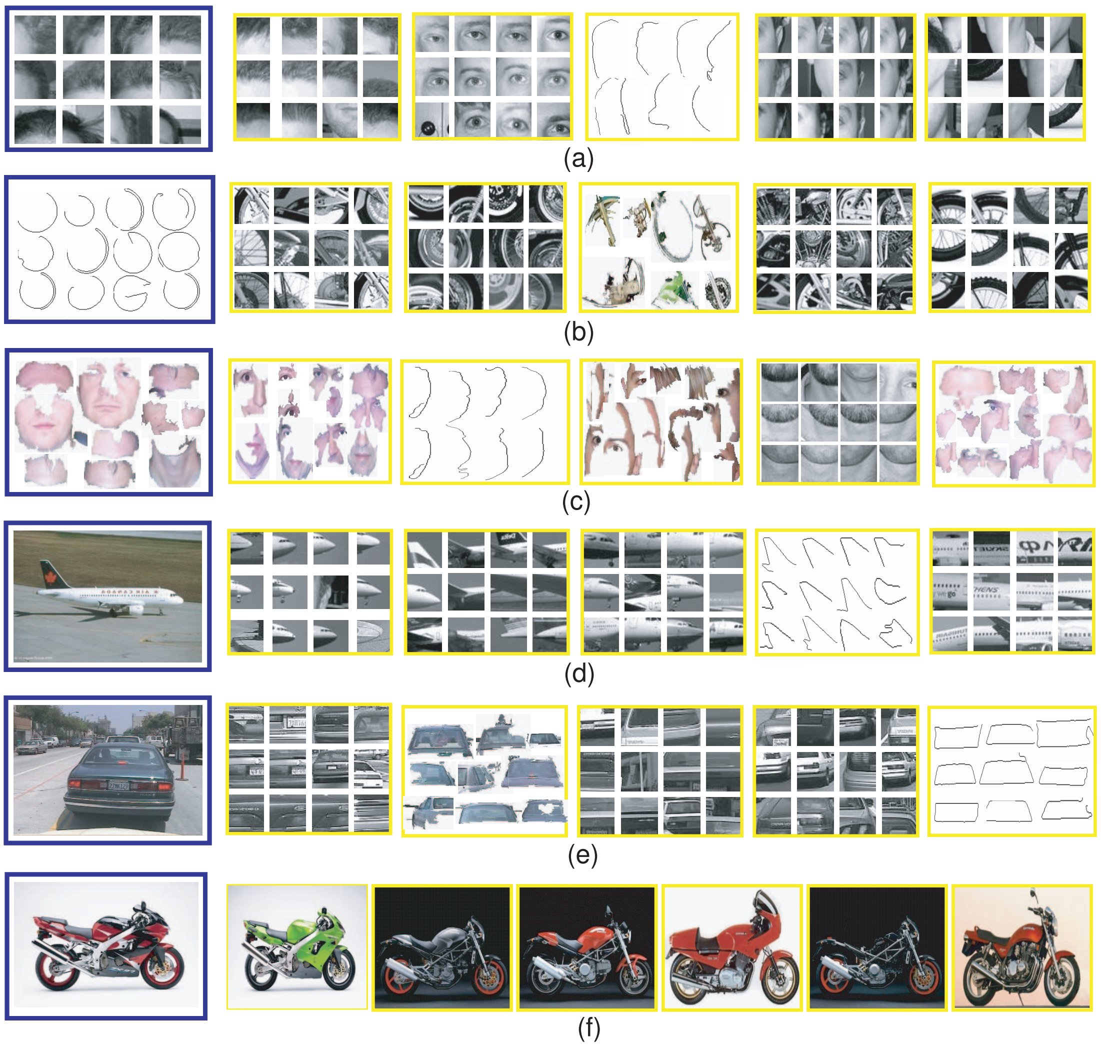


Figure 5.8: The results of different combinations of the query-result entities. (a) Query: Interest Point, Output: Five nearest interest points and contours. (b) Query: Contour, Output: Five nearest interest points and regions. (c) Query: Regions, Output: Five nearest interest points, contours, and regions. (d) Query: Image, Output: Five nearest interest points and contours. (e) Query: Image, Output: Five nearest interest points, contours, and regions. (f) Query: Image, Output: Five nearest images.

and returned nearest images from the embedding space. The query region in Figure 5.7(e) is mostly representing the area around the rim of the wheel or on the side of the human face; therefore, the nearest images in the $k$-dimensional space belong to the face and motorbike classes. In Figure 5.7(f), the query region clearly belongs to face; therefore, all the retrieved images are from the face class.

Since after embedding we have all entities (interest points, contours, regions, images) in the same space, we can retrieve any combination of these entities for the given query entity. The next result, shown in Figure 5.8, is aimed at demonstrating this strength, where for each query we retrieve the five nearest entities (interest points, contours, regions). In each result (Figure 5.8(a)-(f)), the query entity is shown on the left side with a blue rectangle around it, while the retrieved entities are shown within a yellow rectangle. In Figure 5.8(a), we used an interest point from a face as a query and returned the 5 nearest interest points and contours from the $k$-dimensional space. It can be visually observed that the features returned by our algorithm are very meaningful for the face class. We are able to return this heterogenous combination of features, because we are embedding everything into a common, semantically meaningful space. Figure 5.8(b)-(c) shows some other possible combinations of query-result entity types. In Figure 5.8(d)-(f), we used image as a query entity and returned different feature types as the output entities, using the same procedure. In all cases, the returned features and images are very meaningful. In summary, the qualitative results demonstrated in this section prove that the semantically similar objects are closer to each other in the constructed $k$-dimensional space.

#### 5.5.2.2 Quantitative Results

In this set of experiments, we tested the quantitative performance of our proposed framework on the task of object classification. We used a number of baselines to quantify the

| Interest Points | Face | Car Rear | Airplane | Motorbike | Average |
|---|---|---|---|---|---|
| Original Bag of Words | 96.8% | 100.0% | 35.9% | 82.6% | 78.8% |
| Our Approach | 97.2% | 100.0% | 71.5% | 92.3% | 90.2% |

(a)

| IPs+Contours | Face | Car Rear | Airplane | Motorbike | Average |
|---|---|---|---|---|---|
| Original Bag of Words | 69.5% | 80.4% | 76.7% | 91.9% | 79.6% |
| Our Approach | 85.7% | 98.2% | 92.0% | 90.2% | 91.5% |

(b)

| IPs+Contours+Regions | Face | Car Rear | Airplane | Motorbike | Average |
|---|---|---|---|---|---|
| Original Bag of Words | 90.4% | 78.2% | 97.4% | 85.9% | 88.0% |
| Our Approach | 85.7% | 96.9% | 96.2% | 89.3% | 92.0% |

(c)

| IPs+Contours+Regions | Face | Car Rear | Airplane | Motorbike | Average |
|---|---|---|---|---|---|
| LSA | 91.0% | 100.0% | 96.0% | 96.0% | 95.7% |
| Fiedler Embedding | 99.0% | 100.0% | 95.3% | 96.0% | 97.6% |

(d)

| Embedding Dimension K | Face | Car Rear | Airplane | Motorbike | Average |
|---|---|---|---|---|---|
| 100 | 94.5% | 98.5% | 84.0% | 98.0% | 93.8% |
| 50 | 88.4% | 98.0% | 85.5% | 92.0% | 91.0% |
| 4 | 83.4% | 78.5% | 39.5% | 49.0% | 62.6% |

(e)

| | Face | Car Rear | Airplane | Motorbike | Average |
|---|---|---|---|---|---|
| Interest Points (IPs) | 94.5% | 98.5% | 84.0% | 98.0% | 93.8% |
| IPs+Contours | 96.5% | 100.0% | 88.5% | 96.0% | 95.3% |
| IPs + Contours + Regions | 99.0% | 100.0% | 95.3% | 96.0% | 97.6% |

(f)

Figure 5.9: Figure summarizes results of different experiments. (a) A comparison of BOW approach with our method by using only the interest point features. (b) A comparison of BOW approach with our method by using both interest point and contour features together. (c) A comparison of BOW approach with our method by using all three features. (d) A comparison of Fielder embedding with LSA by using all three feature types. (e) A comparison of performance of our framework for different values of embedding dimension using only the interest point features. (f) A comparison of contributions of different features towards classification.

improvements obtained by our algorithm. In general, the experiments are conducted with different configurations of feature types and embedding dimensions.

In the first experiment, we compared the classification performance of the original bag of words (BOW) method with our approach. In the original BOW, raw frequency of features (words) is used as the image representation, while our approach constructs a weighted BOW by using the the meaningful groupings returned by the embedding procedure as follows: Suppose we have a feature frequency representation of an image $\mathbf{t} = (t_1, t_2, ..., t_i, ...t_{N_\mathrm{x}})$, where $N_\mathrm{x}$ is the size of the feature vocabulary. We define a function $f(i, j)$ which returns the $j$-th nearest neighbor of the $i$-th feature from the $k$-dimensional space using the Euclidian distance. Using this function, we compute the weight for the $i$-th bin of $\mathbf{t}$ (which is the frequency of $i$-th feature) as follows:

$$t'_i = \frac{1}{K} \sum_{j=1:K} t_{f(i,j)}, \tag{Eq. 5.12}$$

where we use $K$=5 (here $K$ is the total number of nearest neighbors). The process is repeated for all bins to get the complete weighted histogram for the image. Once the weighted histogram is constructed for each image, we train a Nearest Neighbor classifier by using 100 images as the training images from all 4 categories. A separate Nearest Nearest Neighbor classifier is learnt using the un-weighted feature frequency histograms by using the same 100 images as the training images. The performance of the two classifiers is compared by using all the remaining images as the test images. A detail analysis of the results is presented next.

The first result shown in Figure 5.9(a) is obtained by using only the interest point features, which means we have a $2,000$ dimensional histogram for each image. For discovery of meaningful groupings, interest points and images are embedded into a $k = 100$ dimensional space. The average accuracy of 90.2% obtained by our method is much better than the accuracy of original BOW, which is 78.8% in this case. This is indicative of the fact that the constructed embedding space is semantically meaningful, and therefore, helpful in discovering groupings, which eventually make the weighted histogram more discriminative. In this particular result, the improvement for the airplane category is remarkable (from 35.9% to 71.5%). The reason is that original frequency histograms have a lot of noise in them due to extensive clutter present in the airplane images. The source of the clutter is the airport background which is present in most of the images. However, our method is able to overcome this noise because it takes into consideration the semantic similarity of each interest point with all the airplane images, before deciding on its embedding coordinate. An interest point generated from clutter will not be able to have a high semantic similarity with all or most of the airplane images; therefore, its embedding coordinate will be far off from the airplane images and true interest points belonging to the airplane category. Thus, it will not be able to effect the weighting of the histogram. This observation is further vindicated by the result in Figure 5.9(b), where we used interest points as well as contours. Since contours are extracted by using the bounding boxes around the target category, the clutter does not have a strong influence anymore on the original BOW histogram. Therefore, the difference in the improvement is small as compared to Figure 5.9(a). Figure 5.9(c) shows the results obtained by using all three features at the same time. Again, our approach is able to outperform the original BOW approach. It can be observed that in all three cases (Figure 5.9(a)-(c)) our results are much better than the original BOW approach, which is indicative of the fact that the weighting based on the semantically meaningful neighborhood in the embedding space is helping.

In order to further verify the contribution of homogenous and heterogenous relationships among features, we performed an experiment where the $k$-dimensional embedding coordinates of the images are used for constructing the Nearest Neighbor classifier. We used the value of $k = 100$, while 100 images are used as training images for learning the nearest neighbor classifier. The results are shown in Figure 5.9(f). It can be observed that the use of more features is helpful in getting better accuracy, thus verifying our initial intuition that the use of homogenous and heterogenous relationships between complementary features should be exploited. Since our framework provides a principled way of combining feature, the addition of more features can be carried out in an elegant fashion.

Next, we compared the performance of our framework with LSA. We used all the features to construct a 100-dimensional embedding space, and then constructed a nearest neighbor classifier by using the Euclidian coordinates of the training images. The first 100 eigenvectors are kept for LSA, and the nearest neighbor classifier is learnt by using the projections of training images onto these 100 eigenvectors. The results are shown in Figure 5.9(d), where our framework improved over the LSA based classification by 1.9%. We believe the reason for improvement is that our framework considers a richer set of relationships between images in terms of features when placing them in the embedding space. Note that, although the difference in terms of percentage seems small, the matter of fact is that this improvement is obtained by correctly classifying the hardest examples. Moreover, our framework has more to offer in terms of the relationships that one can discover among diverse sets of features in an elegant manner since, everything now lies in the same space. Another useful aspect of our framework is that one can use only a single interest point (contour or region) for object detection by computing its distance to the images of the training class that lie in the same space. This is very useful, in presence of occlusion or partial object views. On the other hand, LSA will not be able to directly carry out this partial matching, as features and images lie in their separate spaces. Finally, we studied the effect of different values of $k$ on the classification performance using all the feature types. The same experimental setup is used as described for the previous experiments. The results are shown in Figure 5.9(e). The classification performance increases as we increase the value of $k$. We believe the reason for this is that a higher dimensional space allows a more flexibility to the minimization procedure when deciding about the coordinates of each entity. The best performance is achieved at $k$=100, beyond which we do not observe significant increase in the performance.

### 5.5.3 Weizmann Data Set: Action Recognition

This data set contains 10 actions performed by 9 different persons. There are total of 92 video sequences. On this data set, our approach shows improvement over the baseline method which uses a single type of feature. The default parameters for feature extraction are as follows. For ST features, we extracted 200 interest cuboids from each video with $\sigma = 1$ and $\tau = 2$. Then, we used $k$-means algorithm to generate the code books with sizes 200 and 1,000. For Spin-Image features, we uniformly sample 1,000 *orientated points* from each action volume. The $k$-means algorithm is applied again to quantize the feature vectors. We created two vocabularies with 600 and 1,000 *video-words* each. Finally, all the classification experiments were carried out by using K-Nearest Neighborhood (KNN) classifier with $K$=5. We used leave-one-out cross-validation scheme, so we had 10 runs for each experiments, and the average accuracy is reported.

**Qualitative Results.** The qualitative results shows the strength of *Fiedler Embedding* which has the capability to embed semantically similar entities (e.g. action videos, spin-image features, ST features) to one close cluster in the $k$-dimensional space. Figure 5.10 visually shows the results of the queries using different types of entities. In Figure 5.10 (a)-(c), the queries consisted of features (*video-words*, shown by blue rectangles), and results of the query consisted of the four nearest action videos (nearest in the Euclidian sense) from the embedding space. For each *video-word*, the percentages of the feature source (called the purity of the *video-word*) are also listed, which expresses the category specific property of the *video-word*. For instance, "wave2:99%, wave1:1%" means most of the features in this *video-word* are from action "wave2", therefore semantically it is related to "wave2". Fig. 5.10 (d)-(f) demonstrate three queries using features, where the returned results consist of the other nearest features from the embedding space. From Fig. 5.10 (f), we can note that the features of the action "run" are confused with the action "jump". In Fig. 5.10 (g) and (h), we performed queries using action videos. From the results, we infer that the action

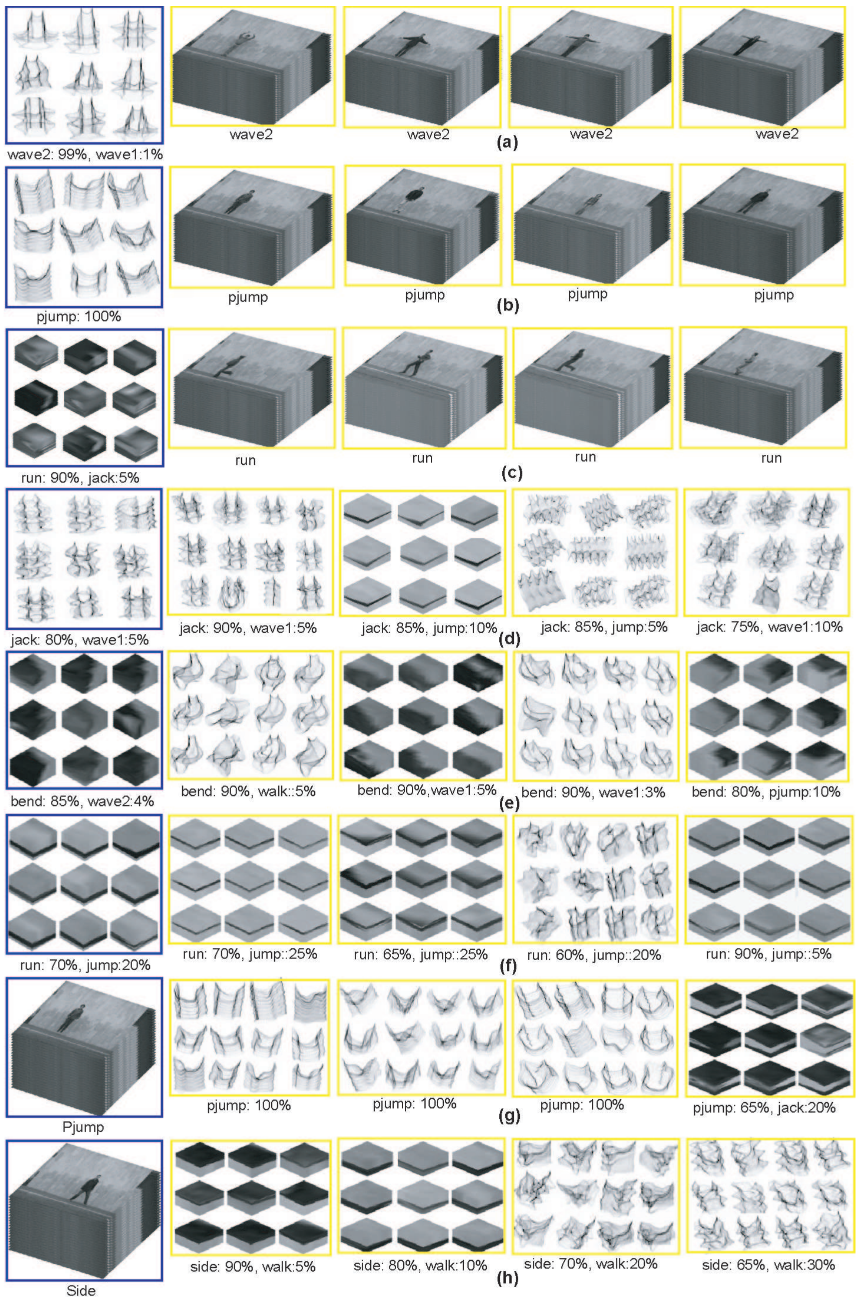


Figure 5.10: Figure shows different combinations of query-result that we used for qualitative verification of the constructed $k$-dimensional space. Each rectangle represents one entity (e.g. action video or a *video-word* (a group of features)). In (a)-(c), the features in blue which are from one *video-word* are used as query, and the 4 nearest videos in yellow from the $k$-dimensional space are returned. Under each *video-word*, the category component percentage is also shown (e.g. "wave2: 99%, wave1:1%" means 99% of features in this *video-word* are from "wave2" action). In (d) and (e), we respectively used ST features and Spin-Image features as query, and retrieved the nearest features in the $k$-dimensional space. In (f) and (g), two action videos are used as query, and the nearest features are returned.

| | Bend | Jack | Jump | Pjump | Run | Side | Walk | Wave1 | Wave2 | Average |
|---|---|---|---|---|---|---|---|---|---|---|
| Original Bag of Words | 100.0 | 100.0 | 44.4 | 88.9 | 80.0 | 66.7 | 100.0 | 88.9 | 88.9 | 84.2 |
| Weighted Bag of Words | 100.0 | 100.0 | 55.6 | 100.0 | 80.0 | 100.0 | 100.0 | 88.9 | 88.9 | 90.4 |

(a)

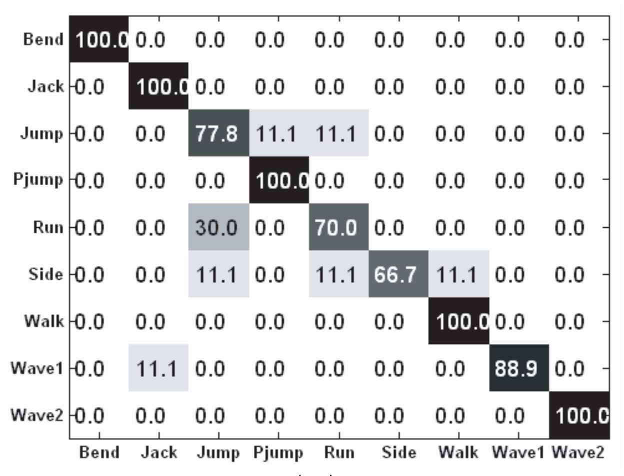
| | Bend | Jack | Jump | Pjump | Run | Side | Walk | Wave1 | Wave2 | Average |
|---|---|---|---|---|---|---|---|---|---|---|
| Original Bag of Words | 100.0 | 100.0 | 11.1 | 88.9 | 80.0 | 11.1 | 66.7 | 88.9 | 100.0 | 71.9 |
| Weighted Bag of Words | 100.0 | 100.0 | 44.4 | 88.9 | 90.0 | 55.6 | 100.0 | 100.0 | 100.0 | 86.5 |

(b)

Figure 5.11: The comparison of the BOW approach with our weighted BOW method.

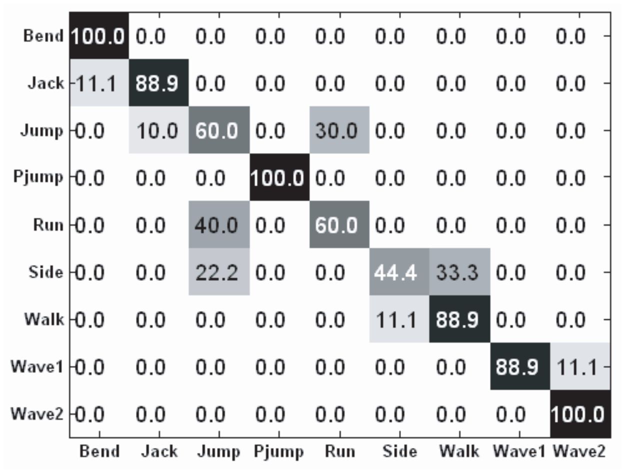
"pjump" is easier to recognize than the action "side". This is mainly because the *video-words* for action "pjump" have higher "purity" than those of "side". On the other hand, the action "side" might be confused with the action "run". The quantitative results in Fig. 5.12 further verifies this observation. In short, the qualitative results demonstrate that the constructed embedding space is meaningful, and provides us an insight into the relationships of the features.

**Quantitative Results.** In the following set of experiments, we show the quantitative results of our action recognition and compare them with the baselines. In order to show that the feature grouping is meaningful in the $k$-dimensional space, we compared the performance of classification using the original bag of *video-words* (term frequency) to that of weighted bag of *video-words* (weighted term frequency) representation of the video. The weighted bag of *video-words* is constructed by using the meaningful feature groupings returned by our embedding. Equation Eq. 5.12 gives the details for re-weighting the term frequency. Figure 5.11 shows the performance comparison between the original BOW model and our weighted BOW. In (a) and (c) we show the performance obtained using vocabulary of size $N_{si} = N_{ip} = 200$ and $N_{si} = N_{ip} = 1,000$ respectively. For the 1,000 dimensions case, our approach can improve 12% over the baseline. This is due to the fact that the constructed embedding space is semantically meaningful, and therefore it helps us discover the feature grouping that will eventually make the weighted BOW more discriminative.



| | Bend | Jack | Jump | Pjump | Run | Side | Walk | Wave1 | Wave2 |
|---|---|---|---|---|---|---|---|---|---|
| Bend | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Jack | 0.0 | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Jump | 0.0 | 0.0 | 77.8 | 11.1 | 11.1 | 0.0 | 0.0 | 0.0 | 0.0 |
| Pjump | 0.0 | 0.0 | 0.0 | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Run | 0.0 | 0.0 | 30.0 | 0.0 | 70.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Side | 0.0 | 0.0 | 11.1 | 0.0 | 11.1 | 66.7 | 11.1 | 0.0 | 0.0 |
| Walk | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 100.0 | 0.0 | 0.0 |
| Wave1 | 0.0 | 11.1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 88.9 | 0.0 |
| Wave2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 100.0 |

(a)



| | Bend | Jack | Jump | Pjump | Run | Side | Walk | Wave1 | Wave2 |
|---|---|---|---|---|---|---|---|---|---|
| Bend | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Jack | 11.1 | 88.9 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Jump | 0.0 | 10.0 | 60.0 | 0.0 | 30.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Pjump | 0.0 | 0.0 | 0.0 | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Run | 0.0 | 0.0 | 40.0 | 0.0 | 60.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Side | 0.0 | 0.0 | 22.2 | 0.0 | 0.0 | 44.4 | 33.3 | 0.0 | 0.0 |
| Walk | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 11.1 | 88.9 | 0.0 | 0.0 |
| Wave1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 88.9 | 11.1 |
| Wave2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 100.0 |

(b)

Figure 5.12: (a) Confusion table for *Fiedler embedding* with $k$=20. (b) Confusion table for LSA with $k$=25.

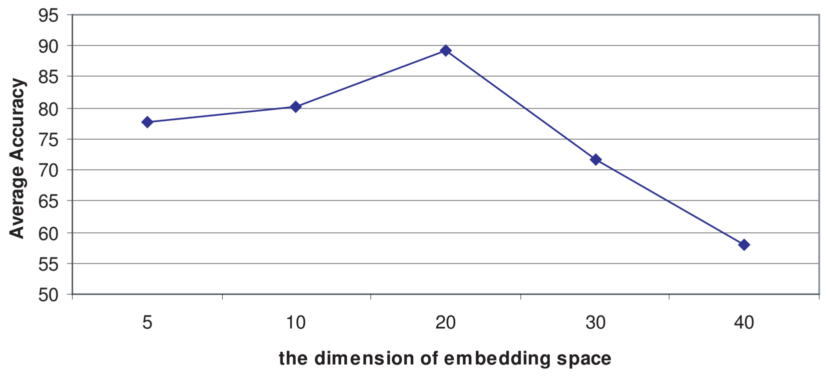


Figure 5.13: The variation of embedding dimension affects the performance. All experiments are carried out on $N_{si} = N_{ip} = 1,000$.

In order to further verify the contribution of the relationships of different types of features, we embed "ST features" and "Spin-Image features" into two separated spaces, and also embed both into the common space. Fig.5.14 (a) and (b) show two sets of experiments carried on $N_{si} = N_{ip} = 200$ and $N_{si} = N_{ip} = 1,000$ respectively. It is obvious that multiple features can improve the performance by about 12%-24%. The reason is because the embedding procedure is discovering the explicit or implicit relationship between different type of features.

|  | Bend | Jack | Jump | Pjump | Run | Side | Walk | Wave1 | Wave2 | Average |
|---|---|---|---|---|---|---|---|---|---|---|
| ST features | 100 | 88.9 | 55.6 | 77.8 | 60 | 44.4 | 44.4 | 44.4 | 44.4 | 62.2 |
| Spin-Image features | 88.9 | 55.6 | 44.4 | 88.9 | 90 | 33.3 | 88.9 | 77.8 | 100 | 74.2 |
| ST + Spin-Image features | 100 | 100 | 88.9 | 100 | 70 | 44.4 | 88.9 | 88.9 | 100 | 86.8 |

(a)

|  | Bend | Jack | Jump | Pjump | Run | Side | Walk | Wave1 | Wave2 | Average |
|---|---|---|---|---|---|---|---|---|---|---|
| ST features | 100 | 100 | 44.4 | 100 | 60 | 11.1 | 44.4 | 88.9 | 66.7 | 68.4 |
| Spin-Image features | 88.9 | 100 | 33.3 | 77.8 | 100 | 33.3 | 88.9 | 55.6 | 77.8 | 72.8 |
| ST + Spin-Image features | 100 | 100 | 77.8 | 100 | 70 | 66.7 | 100 | 88.9 | 100 | 89.3 |

(b)

Figure 5.14: The contributions of different features to the classification. (a)$N_{si} = N_{ip} = 200$, $k$=20,30 and 20 for ST features, Spin-Image features and the combination respectively. (b)$N_{si} = N_{ip} = 1,000$, $k$=20,70 and 20 for ST features, Spin-Image features and the combination respectively.

Next, we compared the performance of *Fiedler Embedding* based classification with the LSA. We used both features with vocabulary size of $N_{si} = N_{ip} = 1,000$. *Fiedler Embedding* achieves the best average accuracy of 89.26% when $k$=20, and LSA obtains 85.11% at $k = 25$. Fig.5.12 (a) and (b) show the confusion tables of *Fiedler embedding* and LSA respectively. Compared to the performance (71.9%) of directly concatenating two types of features (the original BOW is shown in Fig5.11(b)) , the advantage of *Fiedler embedding* is obvious. In our experiments, we also observed that the variation of dimension $k$ of the embedding space affects the performance. Fig.5.13 plots the performance variation against various choices of the dimension.

## 5.6 Conclusion

In this chapter we have proposed a unique way of looking at the relationships between different types of entities that one comes across in number of computer vision applications. This is done by treating different entities as nodes in a graph, where weighted edges between the nodes represent the strength of the relationship between entities. The graph is embedded into a $k$-dimensional space, subject to the criteria that similar nodes should have Euclidian coordinates which are closer to each other. Once all the entities are embedded into the same space, the relationships between them can be discovered by using simple Euclidian distances. We demonstrated the applicability of the proposed framework on problems of object classification and action recognition. The unique strength of the our framework is that it allows one to discover relationships between a wide range of entities of different dimensions and types, and it is general enough to handle a variety of computer vision problems.

# CHAPTER 6: LEARNING SEMANTIC VOCABULARIES USING DIFFUSION DISTANCE

## 6.1 Introduction

In the previous chapter, we verify that *Fiedler Embedding* is able to disclose the semantic relationships amongst varied entities. This results from the fact that it embeds the entity graph into one *common* $k$-dimensional space by minimizing the geometric distance amongst the entities. However, it is not able to explicitly measure the similarity or dissimilarity between two vertices (features or entities) on the graph. Moreover, it is not capable of conducting multi-scale analysis on the feature graph. In this chapter, therefore, we define the *diffusion distance* as an explicit metric to measure the similarity between two features. This diffusion distance is defined as a Markov random walk on the graph. Since it uses multiple walk paths between two vertices to measure the distance instead of the shortest path, it is able to reflect the connectivity between any pair of vertices. To some extent, the connectivity relationship indicates the geometric structure of the graph. Diffusion distance is derived from diffusion maps (DM) [102], which embeds the manifold points into a lower-dimensional space while preserving the intrinsic local geometric data structure.

The diffusion process begins by organizing the data points into a weighted graph (where the weight between two feature points is the feature similarity), which is a good way to represent the complex relationships between the feature points. Once we normalize the weight matrix and also make it symmetric and positive, we can further interpret the pairwise similarities as edge flows in a Markov random walk on the graph. In this case, the similarity is analogous to the transition probability on the edge. Then, utilizing the spectral analysis on the Markov matrix of the graph, we can find the dominant $k$ eigenvectors as the coordinates of the embedding space and map the feature points to the low-dimensional space while preserving their local geometric structures. In addition, by adjusting the time of the Markov chain, DM can be used to employ multi-scale analysis on the data. This multi-scale analysis is similar to Pyramid Match Kernel (PMK) [70], which performs matching under different resolutions of the feature space. If we consider the embedding process as clustering, DM embeds semantically similar features into the same cluster (i.e., some concept). The size of the cluster or the range of the concept is defined by the diffusion time. A larger diffusion time corresponds to a bigger cluster, which means a larger range of concept. For instance, "sport" is on a larger scale than "baseball" and "football", and "baseball" is on a larger scale than "team". With the multi-scale data analysis, we can match the data under different scales.

In fact, DM is one of the techniques used for manifold dimension reduction like PCA, ISOMAP [82], Laplacian Eigenmaps [83], etc. In many applications, the distances between feature points that are far apart are meaningless, so preserving the local structure is sufficient for the embedding. Unlike DM, PCA and ISOMAP are global techniques that do not preserve local geometric information of the feature space. In addition, PCA is unable to handle nonlinear manifold data points. Since the diffusion distance derived from DM uses all the paths between two points to compute the distance, it is more robust to noise than the geodesic distance (shortest path distance) used by ISOMAP. DM is very similar to Eigenmaps-based approaches. However, since the embedding coordinates are weighted eigenvectors of the graph Laplacian, DM has an explicit distance measure induced by a nonlinear embed-ding in the Euclidean space. Eigenmaps representation does not have any explicit metric in the embedding space. Additionally, DM can employ multi-scale analysis on the feature points by defining different time values of the random walk. In our work, we represent the mid-level features using Pointwise Mutual Information (PMI) [94], which is employed to measure the correlation of two variables, i.e. features and images or videos. We can consider mutual information as the expectation of PMI.

Figure 6.1 depicts the major steps for constructing a semantic visual vocabulary using diffusion maps. There are four components: extracting raw features, quantizing raw features

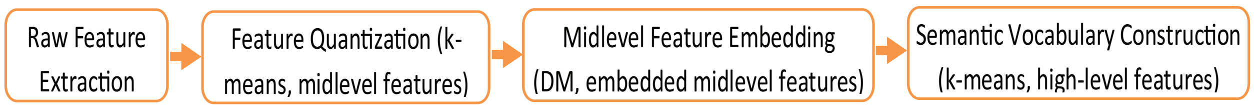

Raw Feature Extraction
Feature Quantization (k-means, midlevel features)
Midlevel Feature Embedding (DM, embedded midlevel features)
Semantic Vocabulary Construction (k-means, high-level features)

Figure 6.1: Flowchart of learning semantic visual vocabulary

into mid-level features using k-means, embedding mid-level features, and constructing visual vocabulary using k-means. We extract local patches from images or videos and represent them with the corresponding descriptors. These raw features are quantized into an initial visual vocabulary using k-means based on their appearance similarity. We call these quantized features mid-level features. Then each mid-level feature is represented by a vector, where each element corresponds to PMI of the feature with a particular image or a video. Next, the mid-level features are embedded into a lower-dimensional semantic space using DM. Since the distance between any two feature points is measured by the diffusion distance, we can further apply k-means with diffusion distance to construct the final semantic visual vocabulary. We have tested the diffusion framework on the KTH action data set [21], our own YouTube action data set [54] and the fifteen scene data set [107]. Very inspiring results have been obtained on them.

## 6.2 Diffusion Maps

In this section, we cover the background material related to diffusion maps embedding which follows the description in [102] and [104].

### 6.2.1 Diffusion distances in a graph

Graph-based data representation is an effective way to capture the structure information of the data. The distance between two nodes is often defined as the shortest path separating them, which is also called geodesic distance. However, it is sensitive to noise and lack of structure information of the data. The diffusion distances defined in this section can overcome these shortcomings.

We can construct a graph $\mathbf{G(\Omega, W)}$ with n nodes on the mid-level feature set $\Omega$, where $\mathbf{W} = w_{ij}(x_i, x_j)$ is its weighted adjacent matrix, which is symmetric and positive. The definition of $w_{ij}$ $(x_i, x_j)$ is totally application-driven, but it needs to represent the degree of similarity or affinity of two feature points. In our case, suppose that the mid-level features are on a manifold. We can start with a Gaussian kernel function, leading to a matrix with entries,

$$w_{ij}(x_i, x_j) = e^{(-\|x_i - x_j\|^2/(2\sigma^2))}, \quad \text{(Eq. 6.1)}$$

where $\sigma^2$ indicates the variance of the Gaussian. This graph $\mathbf{G}$ with $\mathbf{W}$ represents our knowledge of the local geometric relationships between the mid-level features. Then, we define a Markov random walk on the feature graph $\mathbf{G}$ using the dynamical systems theory. It is intuitive that if two nodes are closer (more similar), they are more likely transmitted to each other. Therefore, we can treat the normalized edge weight as the transition probability between them. As a result, we form Matrix $\mathbf{P}^{(1)} = p_{ij}^{(1)}$ by normalizing matrix $\mathbf{W}$ such that its rows add up to 1:

$$p_{ij}^{(1)} = w_{ij}/(\sum_k w_{ik}). \quad \text{(Eq. 6.2)}$$

Therefore, each entry of $\mathbf{P}$ can be considered as the transition kernel of the Markov chain on $\mathbf{G}$. In other words, $p_{ij}^{(1)}$ defines the transition probability from node $i$ to $j$ in a single time step and $\mathbb{P}$ defines the entire Markov chain. $\mathbf{P}^{(1)}$ reflects the first-order neighborhood geometry of the data. We could run random walk forward in time to capture information on larger neighborhoods by taking powers of the matrix $\mathbf{P}$. The forward probability matrix for $t$ time steps $\mathbf{P}^{(t)}$ is given by $[\mathbf{P}^{(1)}]^t$. The entries in $\mathbf{P}^{(t)}$ represent the probability of going from $i$ to $j$ in $t$ time steps.

In such a framework, a cluster is a region in which the probability of escaping this region is low. The higher the t, the higher the probability weight can be diffused to other points which are further away. This means the quantities in $\mathbf{P}^{(t)}$ reflect the intrinsic geometry of the data set defined via the connectivity of the graph in a diffusion process and the diffusion time t plays the role of a scale parameter in the data analysis. Generally, larger diffusion time means lower data resolution representation.

Subsequently, we define the diffusion distance D using the random walk forward probabilities $p_{ij}^{(t)}$ to relate the spectral properties of a Markov chain (its matrix and its eigen values and eigenvectors) to the geometry of the data.

$$[\mathcal{D}^{(t)}(x_i, x_j)]^2 = \sum_{q \in \mathbf{\Omega}} \frac{(p_{iq}^{(t)} - p_{jq}^{(t)})^2}{\varphi(x_q)^{(0)}}, \qquad \text{(Eq. 6.3)}$$

where $\varphi(x_q)^{(0)}$ is the unique stationary distribution which measures the density of the data points. It is defined by $\varphi(x_q)^{(0)} = \frac{d_q}{\sum_j d_j}$, where $d_q$ is the degree of node $x_q$ defined by $d_q = \sum_j p_{qj}$. Note that pairs of data points with high forward transition probability have a small diffusion distance. In other words, the diffusion distances will be small between two points if they are connected by many paths in the graph. This notion of proximity of points in the graph reflects the intrinsic geometry of the set in terms of connectivity of the data points in a diffusion process. The idea behind the diffusion distance is that it is computed on many paths through the graph. Compared to the shortest path method, diffusion distance take into account all the evidence relating $x_i$ to $x_j$ , so it is more robust to noise.

### 6.2.2 Diffusion Maps Embedding

Using the spectral theory of the random walk, matrix $\mathbf{P}$ has $n$ eigenvectors and eigenvalues such that:

$$\mathbf{P}^{(t)} v_s = \lambda_s v_s (s = 0, 1, ..., n - 1). \qquad \text{(Eq. 6.4)}$$

It can be proved that the diffusion distance can be computed using eigenvectors $v$ and eigenvalues $\lambda$ of $\mathbf{P}$,

$$[\mathcal{D}^{(t)}(x_i, x_j)]^2 = \sum_{s=1}^{n-1} (\lambda_s^t)^2 (v_s(x_i) - v_s(x_j))^2. \qquad \text{(Eq. 6.5)}$$

The distance can be approximated with the first $k$ eigenvectors. We only need a few terms in the above sum for certain accuracy because of the decay of the eigenvalues. The diffusion distance can then be approximated with relative precision $\delta$ using the first $k$ nontrivial eigenvectors and eigenvalues according to,

$$[\mathcal{D}^{(t)}(x_i, x_j)]^2 \approx \sum_{s=1}^{k} (\lambda_s^t)^2 (v_s(x_i) - v_s(x_j))^2, \qquad \text{(Eq. 6.6)}$$

where $\lambda_k^t > \delta \lambda_1^t$. If we use the eigenvectors weighted with $\lambda$ as coordinates on the data, $\mathcal{D}^{(t)}$ could be interpreted as the Euclidean distance in the low-dimensional space. Hence, we introduce diffusion map embedding, and the low-dimensional representation is given by

$$\Pi_t : x_i \mapsto \{\lambda_1^t v_1(x_i) \; \lambda_2^t v_2(x_i) \; ... \; \lambda_k^t v_k(x_i)\}^T. \qquad \text{(Eq. 6.7)}$$

The diffusion map $\Pi_t$ embeds the data into a Euclidean space in which the distance is approximately the diffusion distance,

$$[\mathcal{D}^{(t)}(x_i, x_j)]^2 \backsimeq ||\Pi_t(x_i) - \Pi_t(x_j)||^2. \qquad \text{(Eq. 6.8)}$$

The scaling of each eigenvector by its corresponding eigenvalue leads to a smoother mapping in the final embedding, since higher eigenvectors are attenuated. The mapping provides a realization of the graph $\mathbf{G}$ as a cloud of points in a lower-dimensional space, where the re-scaled eigenvectors are the coordinates. The dimensionality reduction and the weighting of the relevant eigenvectors are dictated by both the diffusion time t of the random

**Objective:** Given n points $\{x_i\}_{i=1}^{n}$ in a high dimensional space $\mathbf{\Omega}$, embed all points into a $k$-dimensional space.

(i) Construct a graph $\mathbf{G}$ with n nodes: add an edge between nodes $i$ and $j$ if $i$ is one of the N nearest neighbors of $j$.

(ii) Construct the weight matrix $\mathbf{W}$: if nodes $i$ and $j$ are connected, the edge weight $w_{ij}$ is computed using Equation Eq. 6.1.

(iii) Create Markov transition matrix $\mathbf{P}$: normalize matrix $\mathbf{W}$ using Equation Eq. 6.2 such that its rows add up to 1.

(iv) Compute Markov transition matrix $\mathbf{P}^{(t)}$ at diffusion time t.

(v) Perform eigen-decomposition on $\mathbf{P}^{(t)}$, and obtain eigenvalues $\lambda_s$ and eigenvectors $v_s$, such that $\mathbf{P}^{(t)} v_s = \lambda_s v_s$.

(vi) Embedding data by DM as Eq. 6.7.

Table 6.1: Procedure of diffusion maps embedding.

walk and the spectral fall-off of the eigenvalues. Diffusion maps embed the entire data set in a low-dimensional space in such a way that the Euclidean distance is an approximation of the diffusion distance. We summarize the procedure of DM in Algorithm 6.1.

### 6.2.3 Robustness to Noise

As aforementioned, the diffusion distance is robust to noise and small perturbations of the data. This results from the fact that the diffusion distance reflects the connectivity of nodes in the graph. In other words, the distance is computed from all the paths between two nodes s.t. all the “evidences” are considered. Although one of the paths may be affected by the noise, it has little weight on the computation of diffusion distance. However, the geodesic distance that is used in ISOMAP only considers the shortest path between two points, so it is sensitive to noise. Therefore, diffusion distance is more robust than geodesic distance to noise. In the following paragraphs, we want to verify this fact on a synthetic spiral and the real action data set. We generated 1,000 instances of two-dimensional spiral with Gaussian noise (see Figure 6.2 (a)). As for each instance of spiral, we construct a graph by connecting

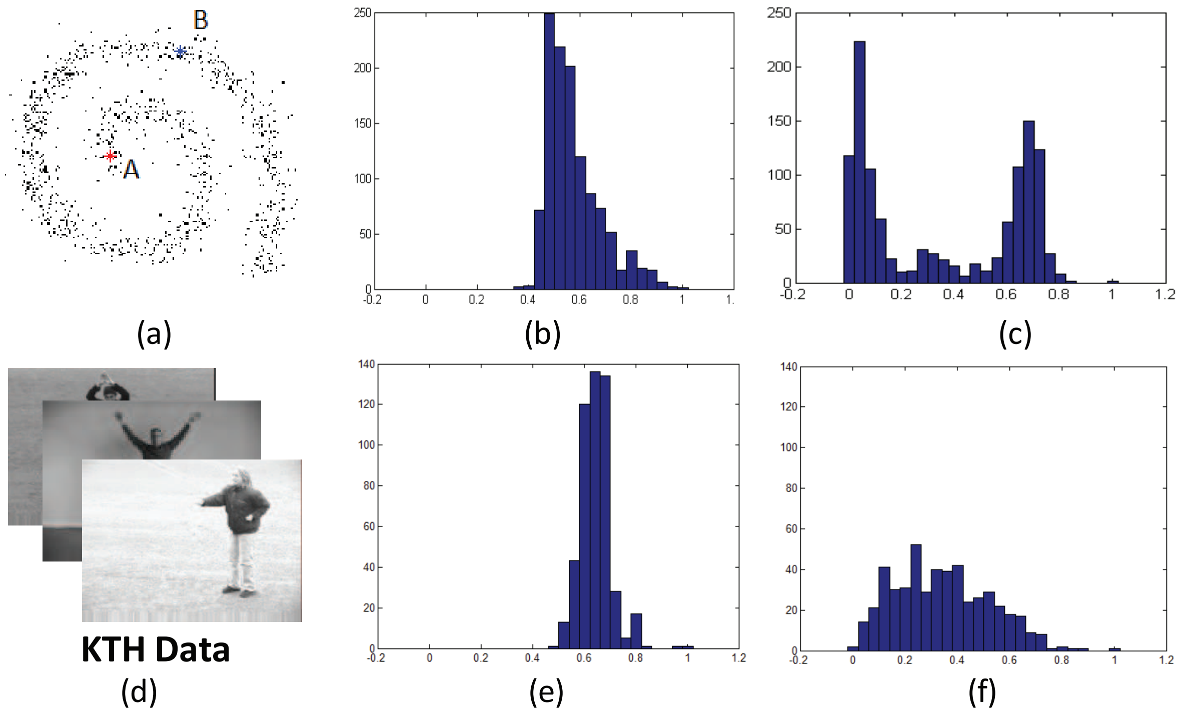


Figure 6.2: Demonstration of robustness to noise. (a) Two dimensional spiral points. (b-c) The distribution of the diffusion distance and geodesic distance between points A and B. (d) KTH data set. (e-f) The distribution of the diffusion distance and geodesic distance between two points on KTH data set.

any two points whose Euclidean distance is less than a threshold $\sigma$. When constructing the adjacency matrix $\mathbf{W}$, $w_{ij}$ is computed using Equation Eq. 6.1 for the connected points. In order to ensure the connectivity of the graph, $w_{ij}$ is set to a small number for any two non-connected points. We picked two fixed points A and B from all instances, and computed the diffusion distance and geodesic distance between them. Figure 6.2 (b) and (c) shows the distribution of the distances on all the trials. From it we can easily see the geodesic distance has much larger standard deviation than the diffusion distance. This shows that geodesic distance is more sensitive to the noise as compared to the diffusion distance. We further verified the robustness of diffusion distance on the KTH data set. We selected two mid-level features (A and B) that have the maximum Euclidean distance in an initial visual vocabulary with 1,000 visual words (mid-level features). Then we added Gaussian noise to the rest of features, and repeated this procedure 500 times. For each trial, we constructed a graph as we described in section 6.2. The distributions of the diffusion distances and geodesic distances between mid-level features A and B are shown in Figure 6.2 (e) and (f). Although the distribution of geodesic distance is better than that of the synthetic spiral, we can still see that diffusion distance has smaller standard deviation than geodesic distance, which further verifies that diffusion distance is more robust.

### 6.2.4 Feature Extraction

In this section, we briefly describe the methods to extract raw features (i.e., motion features for action recognition and SIFT features for scene classification), and then how to generate and represent the mid-level features.

**Motion features for action recognition:** We use the spatiotemporal interest point detector proposed by Dollar et al. [93]. Compared to the 3D Harris-Corner detector [46], it produces dense features that can improve the recognition performance in most cases. It utilizes 2-D Gaussian filter and 1-D Gabor filters in spatial and temporal directions respectively. A response value is given at every position (x, y, t). It produces high responses to the temporal intensity change points. The interest points are selected at the locations of local maximal responses, and 3D cuboids are extracted around them. For simplicity, we use the flat gradient vectors to describe the cuboids with PCA being utilized to reduce the descriptor dimension (e.g. 100 dimensions in our paper), which we call the gradient PCA (gPCA) descriptor.

**SIFT features for scene classification:** It has been shown that the dense features can achieve a better classification rate than sparse interest point features for the scene classification [33] [76] problem. In this chapter, we utilize dense features sampled using a regular grid with space M=8 pixels. The patch size is randomly sampled between scales of 10 to 30 pixels. SIFT descriptor [79] is computed for each patch.

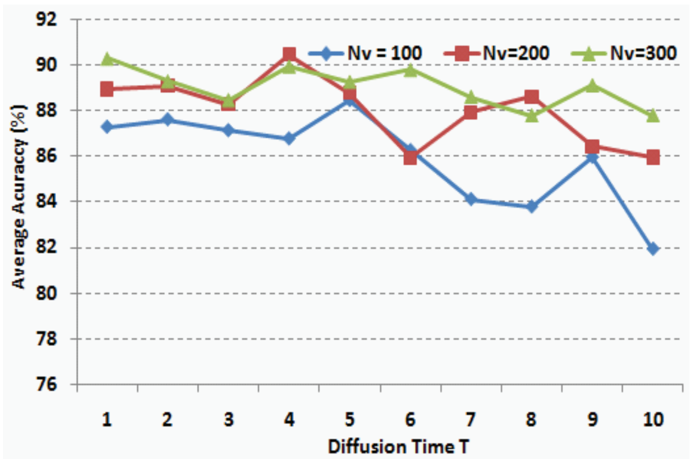
**Mid-level feature representation:** Once we extract the raw features (low-level features), we use $k$-means clustering to quantize these gPCA features or SIFT features into C clusters, which are the mid-level features forming the initial vocabulary. In general, a larger C value can obtain better performance. We choose C equals to 1,000 and 2,000 for the action data set and scene data set respectively. In order to construct the semantic vocabularies based on the mid-level features, we use PMI to represent the mid-level features. Suppose we have Nt number of training images or videos, we compute the PMI between a training image/video x and mid-level feature y as,

$$pmi(x; y) = log(\frac{f_{xy}}{\sum_{\varepsilon} f_{\varepsilon y} \sum_{\omega} f_{x\omega}}), \qquad \text{(Eq. 6.9)}$$

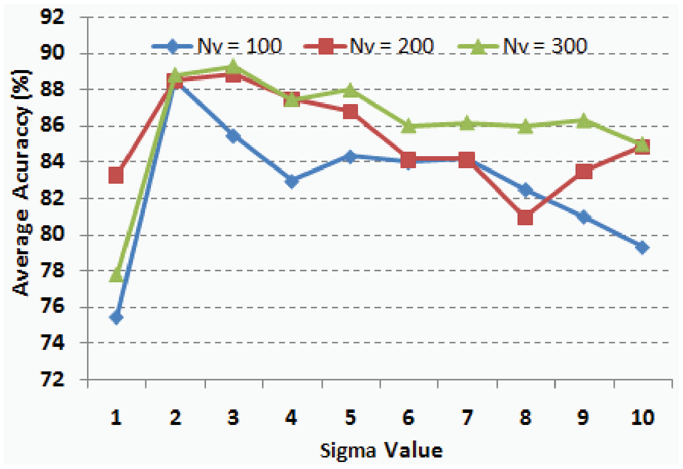
where $f_{xy} = \frac{c_{xy}}{N_t}$, $c_{xy}$ is the number of times feature y appears in image or video x. Then we can represent the mid-level feature y in terms of an Nt dimensional feature vector, and the distance between any two features $y_1$ and $y_2$ can be computed using equation Eq. 6.1.

## 6.3 Experiments and Discussion

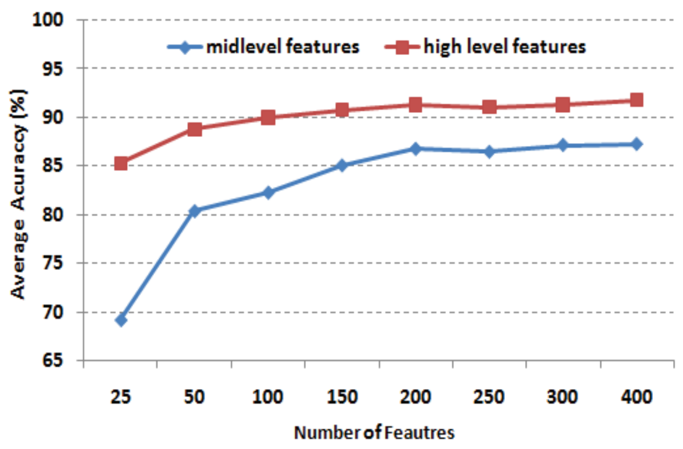
We tested our approach on the KTH action data set, our own YouTube action data set, and the fifteen scene data set. SVM with Histogram Intersection kernel is chosen as the default classifier. For the action data set, we perform the leave one out cross validation (LOOCV) scheme, which means 24 actors or groups are used for training and the rest for testing. For the fifteen scene data set, we randomly selected 100 images from each category for training, and the rest for testing.



(a)

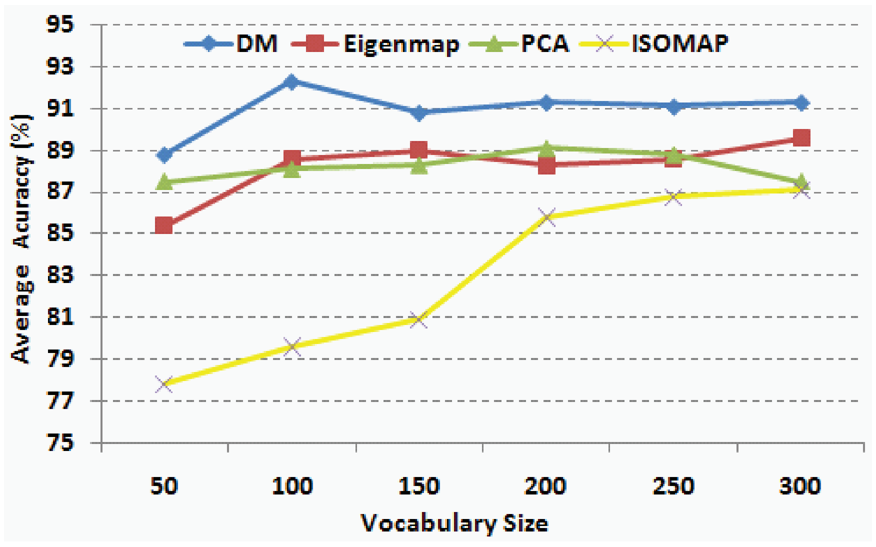


(b)

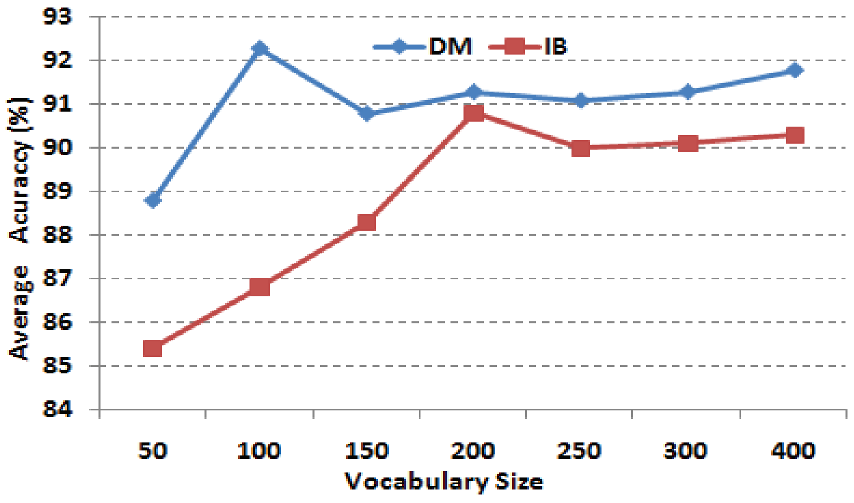


(c)

Figure 6.3: (a) and (b) shows the influence of diffusion time and sigma value, respectively, on the recognition performance. The three curves correspond to three visual vocabularies of size 100, 200, and 300 respectively. The sigma value is 3 in (a) and the diffusion time is 5 in (b); (c) The comparison of recognition rate between mid-level and high-level features.

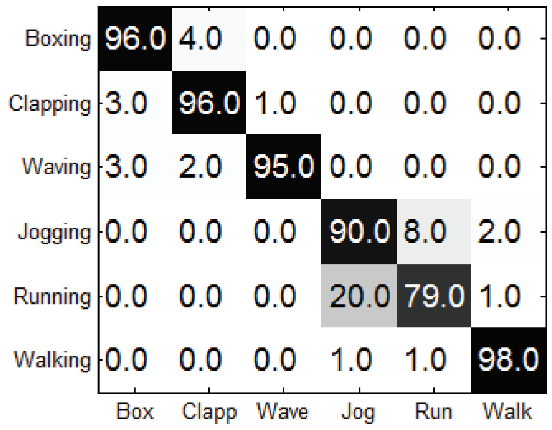


(a)



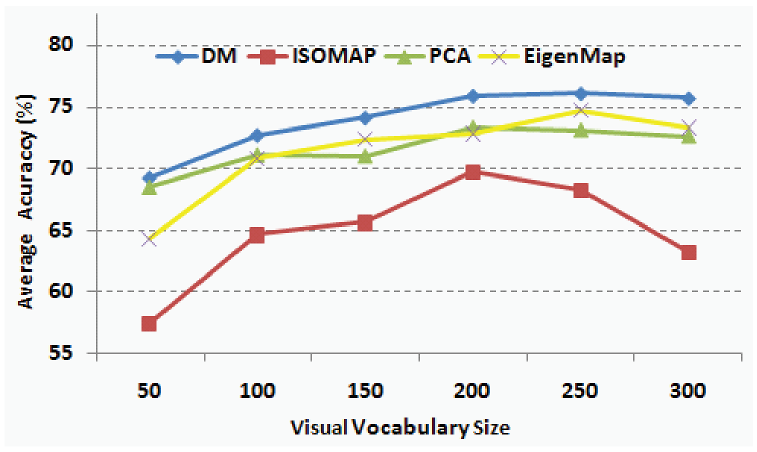
(b)

Figure 6.4: (a) Comparison of performance between different manifold learning schemes. (b) Comparison of performance between DM and IB

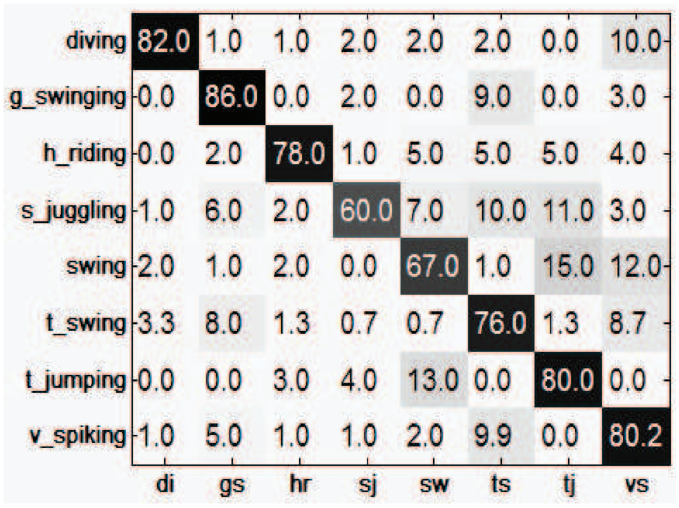


| | Box | Clapp | Wave | Jog | Run | Walk |
|---|---|---|---|---|---|---|
| Boxing | 96.0 | 4.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Clapping | 3.0 | 96.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| Waving | 3.0 | 2.0 | 95.0 | 0.0 | 0.0 | 0.0 |
| Jogging | 0.0 | 0.0 | 0.0 | 90.0 | 8.0 | 2.0 |
| Running | 0.0 | 0.0 | 0.0 | 20.0 | 79.0 | 1.0 |
| Walking | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 98.0 |

(a)



(b)



| | di | gs | hr | sj | sw | ts | tj | vs |
|---|---|---|---|---|---|---|---|---|
| diving | 82.0 | 1.0 | 1.0 | 2.0 | 2.0 | 2.0 | 0.0 | 10.0 |
| g_swinging | 0.0 | 86.0 | 0.0 | 2.0 | 0.0 | 9.0 | 0.0 | 3.0 |
| h_riding | 0.0 | 2.0 | 78.0 | 1.0 | 5.0 | 5.0 | 5.0 | 4.0 |
| s_juggling | 1.0 | 6.0 | 2.0 | 60.0 | 7.0 | 10.0 | 11.0 | 3.0 |
| swing | 2.0 | 1.0 | 2.0 | 0.0 | 67.0 | 1.0 | 15.0 | 12.0 |
| t_swing | 3.3 | 8.0 | 1.3 | 0.7 | 0.7 | 76.0 | 1.3 | 8.7 |
| t_jumping | 0.0 | 0.0 | 3.0 | 4.0 | 13.0 | 0.0 | 80.0 | 0.0 |
| v_spiking | 1.0 | 5.0 | 1.0 | 1.0 | 2.0 | 9.9 | 0.0 | 80.2 |

(c)

Figure 6.5: (a) Confusion table of KTH data set when the size of the semantic visual vocabulary is 100. The average accuracy is 92.3%. (b) Performance comparison between DM and other manifold learning schemes on the YouTube action data set. (c) Confusion table of the YouTube data set when the size of semantic visual vocabulary is 250. The average accuracy is 76.1%.

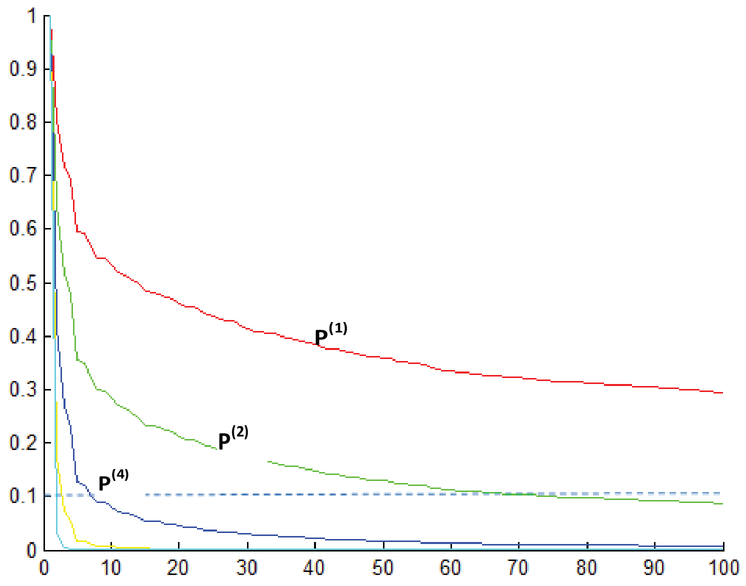


Figure 6.6: The decay of the eigenvalues of $\mathbf{P}^t$ on YouTube data set when sigma is 14.

### 6.3.1 Experiments on KTH data set

The KTH data set contains six actions: boxing, clapping, waving, jogging, walking, and running. They are performed by 25 actors under four different scenarios. In total it contains 598 video sequences. All the following experiments are conducted on 1,000 mid-level features. As we discussed, the DMs provide a method to represent the data at different resolutions by using varied diffusion times. Generally, high data resolution can be obtained at smaller diffusion times. Therefore, the diffusion time $t$ can affect the performance of the visual vocabulary. The three curves in Figure 6.3 (a) illustrate the influence of $t$ on the action recognition rates when the size of the semantic visual vocabulary ($N_v$) is 100, 200, and 300 respectively (here, the sigma value is 3 for all of them). It seems that higher recognition accuracy is obtained at a smaller $t$ value when the sigma is fixed. In fact, when $t$ is larger, the data resolution is lower, which may decrease the quality of the visual vocabulary. Additionally, the sigma value of Equation Eq. 6.1 also affects the recognition rate. Figure 6.3 (b) shows its influence on the recognition performance when fixing the diffusion time $t$=5. The sigma value affects the recognition accuracy by influencing the decay speed of

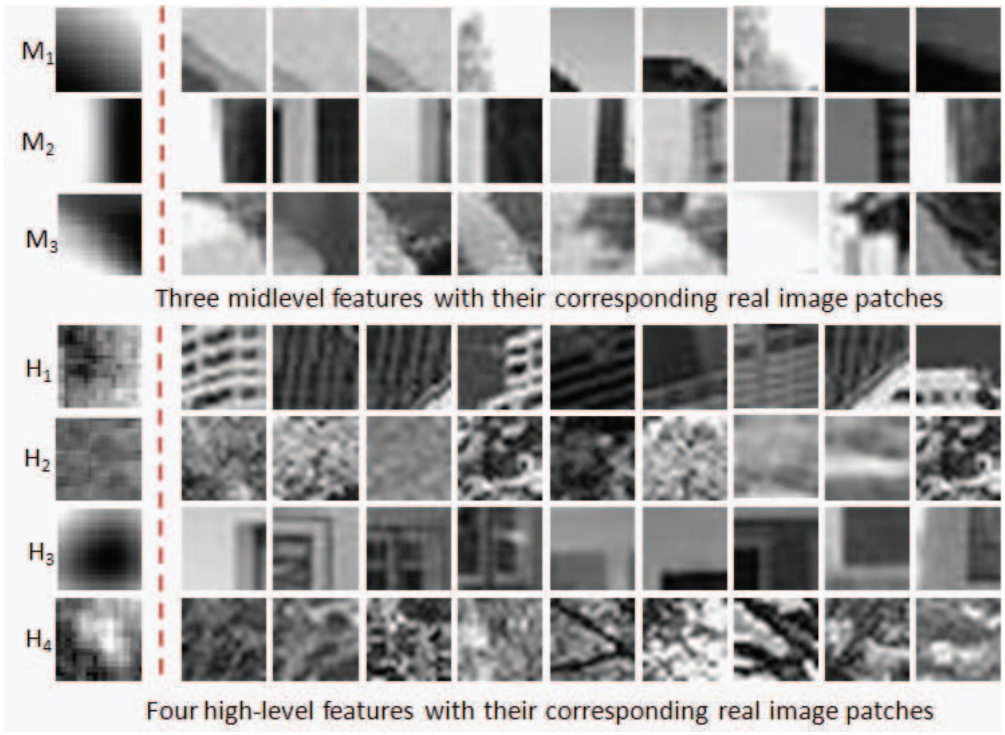

M1
M2
M3
Three midlevel features with their corresponding real image patches
H1
H2
H3
H4
Four high-level features with their corresponding real image patches

Figure 6.7: Some examples of mid-level and high-level features with their corresponding real image patches. Each row lists one mid-level or high-level feature followed by its image patches. The three mid-level features are selected from 40 mid-level features. The four high-level features are selected from 40 high-level features generated by DM from 1,000 mid-level features.

the eigenvalues of matrix $mathbf P^{(t)}$. In general, larger sigma values perform worse when diffusion time is fixed. In the following experiments, all the results are reported with the tuned (better) parameters.

In order to verify that our learnt semantic visual vocabulary (high-level features) is more discriminative than the mid-level features, we compared the recognition rate obtained by using high-level and mid-level features under the same size. The high-level features are learnt from the 1,000 mid-level features using DM. The reported recognition rates are the best ones achieved with different diffusion times and sigma values. Figure 6.3 (c) shows the comparison. It is clear that high-level features can achieve much better performance than mid-level features. Particularly, the recognition rate (88.9%) with 50 features is comparable to that of 400 mid-level features. In addition, when the number of features is larger than 100, the recognition rate is over 90%, and the increase is slow with the growing number of features. It means the recognition rate is not sensitive to the number of features, which is not the case with the mid-level features. This verified the aforementioned fact that the learnt high-level features are semantically meaningful. They can largely improve the recognition efficiency without decreasing the performance for a large data set.

We believe the features lie on some manifolds, therefore we can apply the manifold learning technique to embed them into a low-dimensional space while maintaining the data structure. We conducted a group of experiments to compare some other manifold techniques (e.g. PCA, ISOMAP, Eigenmap) to DM. We have briefly discussed the difference between them in the introduction. All of them firstly embed the mid-level features into a 100-dimensional space, and then apply $k$-means to the mid-level features to obtain $N$ clusters (high-level features). The results are shown in Figure 6.4 (a) (all the techniques have been tuned to have better parameters). We can see the DM can achieve varied improvements from about 2% to 5% in terms of recognition rate, as compared to others. Both DM and ISOMAP define an explicit metric in the embedding space (i.e., diffusion distance and geodesic distance respectively). The experiments further confirm that diffusion distance is more robust than geodesic distance.

As we know, the semantic high-level features are learnt by applying $k$-means clustering on the embedded mid-level features. Another way to show the effectiveness of DM embedding is to compare the recognition rate of high-level features learnt by embedded mid-level features to that of original mid-level features without embedding ($k$-means is used as a clustering for both). The results are shown in Table 6.2. The improvements are varied from 2.7% to 4.0%.

We believe PMI can capture the relationship between a particular mid-level feature and videos as well as other mid-level features. This is further verified by the experiments shown in Table 6.3. We conducted two groups of experiments. Both of them use DM to embed

| Vocabulary Size ($N_v$) | 50 | 100 | 200 | 300 |
|---|---|---|---|---|
| Embedded features | 88.8% | 92.3% | 91.3% | 91.3% |
| Original features | 84.8% | 88.3% | 88.6% | 88.3% |

Table 6.2: Performance comparisons between two vocabularies learnt from mid-level features with and without DM. embedding.

| Vocabulary Size ($N_v$) | 50 | 100 | 150 | 200 | 250 |
|---|---|---|---|---|---|
| PMI | 88.8% | 92.3% | 90.8% | 91.3% | 91.1% |
| Frequency | 85.8% | 88.3% | 88.6% | 89.8% | 88.3% |

Table 6.3: Performance comparison between two different midlevel feature representations: PMI vs. Frequency. embedding.

features into a lower-dimensional space. The difference is that one of them uses PMI to represent the mid-level features and the other directly uses frequency to represent them.

It is very interesting to check the confusion table when the best average accuracy is obtained; see Figure 6.5 (a). “Jogging” obtains a 90% recognition rate, which is better than most existing approaches [72]. However, “running” is easily misclassified as “jogging”. The overall average accuracy of 92.3% is much better than the average accuracy of 89.3% obtained by directly using the 1,000 mid-level features for classification. It is also a little bit better than some existing BOF-based approaches [47] [111].

### 6.3.2 Experiments on YouTube data set

Since the KTH data set is relatively simple, we collected a more complex and challenging data set based on YouTube videos and our personal video collections. Since we do not have control over the video capturing process, the data set has the following properties: 1) a mix

| | DM | ISOMAP | PCA | EigenMap |
|---|---|---|---|---|
| Average Accuracy | 74.9% | 73.5% | 73.3% | 73.1% |

Table 6.4: Best results of different manifold learning techniques.

of still cameras and moving cameras, 2) cluttered background, 3) variation in object scale, 4) varied viewpoint, 5) varied illumination, and 6) low resolution. This action data set contains 8 categories: volleyball spiking (v_spiking), trampoline jumping (t_jumping), soccer juggling (s_juggling), horseback-riding (h_riding), diving, swinging, golf-swinging (g_swinging), and tennis-swinging (t_swinging). Most of them share some common motions such as "jumping" and "swinging". We organize the video sequences into 25 relatively independent groups, where separate groups are either taken in different environments or by different photographers. The data set contains 800 video sequences in total. Figure 1.6 shows some examples of the YouTube data set.

We extracted about 200 to 400 cuboids from each video, and then used k-means to obtain 1,000 mid-level features. All the experiments were conducted on these features. Figure 6.5 (b) demonstrates the performance comparison between DM and other manifold learning methods. It shows the DM gives a more stable recognition rate than other approaches with varied vocabulary sizes. The best result obtained was 76.1% in accuracy, which is at least about 2.4% higher than the best results obtained by others. We show its details in the confusion table in Figure 6.5 (c) for the best results. We can see that lots of actions are misclassified as "t_jumping" and "v_spiking". The reason may be that these two actions are not uniform and share many action units with other action categories. We also noticed the best result of 76.1% is competitive to the result of 75.1% obtained by directly using the 1,000 mid-level features for recognition.

Figure 6.6 shows the decay of the eigenvalues of $\mathbf{P}^{(t)}$ when the sigma value is 14. For diffusion time $t$=2, the top 70 eigenvectors can the most significant ones, and for $t$=4, the top 10 are the most significant ones. We noticed when $t$ is larger, very few (i.e., 20) eigenvectors can achieve good performance.

### 6.3.3 Experiments on Scene data set

We further verified our framework on the fifteen scene data set [107]. We learnt 2,000 mid-level features using $k$-means. Table 6.4 lists all the best results we obtained using different manifold learning. It is quite interesting to look at some visualized mid-level features and high level features shown in Figure 6.7. The visualized center patches are achieved by averaging all the patches belonging to the mid-level or high-level features. In our experiments, we noticed that all the mid-level features obtained by $k$-means are visually smooth like M1, M2, and M3 in the figure. This is due to the patches of a given mid-level feature having similar appearance. While looking at the high-level features, they appear to be more meaningful. For instance, in Figure 6.7 H1 might represent mostly parts of the buildings, H2 might represent foliage in forest, suburb, and open country scenes, and H3 might represent parts of the windows or doors in “living room”, “kitchen”, and “bedroom” scenes.

## 6.4 Conclusion

In this chapter, we propose a novel approach for generic visual vocabulary learning. We first learnt the mid-level features (the initial visual vocabulary) using k-means, then use the DM to embedding the mid-level features into low-dimensional space while maintaining the local relationships of the features. These embedded mid-level features are further clustered to reconstruct a semantically meaningful visual vocabulary. We tested our approach on three complicated data sets. The results verify that the learnt semantic visual vocabularies obtained stable performance compared to the mid-level features learnt by k-means. In addition, we also compared DM with other manifold learning techniques. In most cases, the DM can perform better, especially for the action data set.

# CHAPTER 7: CONCLUSION AND FUTURE WORK

In this work, we have explored the problem of learning semantic features from the raw features for visual recognition, including object/scene recognition and action recognition. A bag of raw features is extracted from the examples, and further quantized into *visual words*. Then any instance can be represented by the histogram of the *visual words*, which is the so-called bag of *visual words* model for visual representation. Although it is successful in visual recognition, this visual representation still suffers several drawbacks. First, its performance is sensitive to the visual vocabulary size. Computationally expensive cross-validation is a must to dig out the appropriate quantization granularity. This limitation is partially due to its second drawback, which is that the *visual words* are not semantically meaningful. Finally, it loses the spatial distribution of the features, which is critical for further improvement on recognition. In order to overcome these limitations, we proposed four approaches to acquire effective semantic features from the a vast amount of raw features. We analyzed the problems from two aspects, i.e., feature clustering via maximization of mutual information and feature projection using manifold learning. We summarize the major contributions in the following section.

## 7.1 Summary of Contributions

(i) Action recognition via maximization of mutual information.

- A criterion of achieving better tradeoff between compactness and discrimination.
- An algorithm to automatically discover the optimal number of semantic features (cluster of *visual words*).
- Application of the bag of *visual words* to multi-view action recognition for the first time.

- Representation of actions in terms of bag of semantic features.
- Representation of actions in terms of spatiotemporal correlgoram to capture the structure of feature geometric distribution.

(ii) Scene recognition using co-clustering.

- An algorithm to simultaneously cluster images and *visual words* to obtain semantic features.
- An introduction of the novel concept of *intermediate concepts.*
- Representation of a scene in terms of bag of concepts. (BoC).
- An effective scene matching approach of spatial correlgoram match kernel (SCMK) to capture spatial information on features.

(iii) Object/action recognition using *Fiedler Embedding.*

- A novel framework to embed all features into the same semantic space.
- A novel action representation in terms of spin-images to capture the shape of actions.
- A new approach to use the feature's relationship to improve visual recognition in terms of weighted bag of *visual words.*

(iv) Learning semantic vocabulary using diffusion distance.

- Introduction of the novel diffusion distance to measure the similarity between two features.
- An novel framework to learn semantic visual vocabulary.

## 7.2 Future Work

The approaches proposed in this work can be improved in many ways. We describes some attempts in the following subsections.

### 7.2.1 Refine the output of information bottleneck

The algorithm we applied in this work is an agglomerative information bottleneck. Although it works very well in our experiments, it still suffers some limitations. For example, it is computationally expensive. In addition, as it is a greedy method, it can not guarantee the optimal solution. However, its clustering results can be further improved by the sequential information bottleneck. The basic idea of this variation is similar to $k$-means algorithm. It is able to reach a "stable" solution. So we can use agglomerative IB to obtain a better number of clusters, and then use sequential IB to refine the clusters.

### 7.2.2 Semi-supervised method

Whether using feature clustering via maximization of mutual information or feature projection by manifold learning, video (image)-VisualWords co-occurrence matrix is a critical data structure providing the semantic information of the visual words. Neither of them use the supervised information explicitly. Since in the classifier training phase we have labelled training instances, we can use these labels to improve the semantic features. One example is co-clustering. We can provide labels to the co-clustering method, and it can use the labels as an initial clusters of videos. It can also compare the video cluster with the true labels at each iteration.

Another example is diffusion maps. The visual words similarity is computed on the distribution of visual words on videos. However, when we construct the initial vocabulary using $k$-means, we can preserve the label information of each cuboids in one visual words. Hence, it is easy to obtain the category percentage in each single visual words, which can provide semantic information of the visual words from another aspect. Then the similarity between two visual words can also computed on the category percentage of the visual words. Gaussian Kernel is the right tool to combine different measurements.

### 7.2.3 multi-scale matching

As we can see, diffusion maps has the capability to perform multi-scale analysis on the data by defining varied time steps. However, we only use one level of information to construct the vocabulary. It is worth to exploit how to combine the information provided by different time steps. Thus, we can match instances in multiple scales.

### 7.2.4 Efficient shape model

Although we have proposed to use spatial correlogram match kernel and spatiotemporal pyramid match kernel, there is still lots of room for us to explore the geometric relationship between the features in location. One potential solution is to apply conditional random field (CRF), which able to capture the feature relationship in space.

# REFERENCES

[1] http://www-nlpir.nist.gov/projects/trecvid/.

[2] A. Bissacco, M. H. Yang and S. Soatto. Detecting humans via their pose. In *Proceedings of the Neural Information Processing Systems Conference*, 2007.

[3] A. Bobick and J. Davis. Recognition of human movement using temporal templates. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(3):257–267, 2001.

[4] A. Bosch, A. Zisserman and X. Munoz. Scene classification via plsa. In *Proceedings of the European Conference on Computer Vision*, 2006.

[5] A. Efros, A. Berg, G. Mori and J. Malik. Recognizing action at a distance. In *Proceedings of the International Conference on Computer Vision*, 2003.

[6] A. Kaban and M. A. Girolami. Fast extraction of semantic features from a latent semantic indexed text corpus. *Neural Processing Letters*, 15(1):31–43, 2002.

[7] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *Inernational Journal of Computer Vision*, 42(3):145–175, 2001.

[8] A. Opelt et al. A boundary fragment model for object detection. In *Proceedings of European Conference on Computer Vision*, 2006.

[9] A. Opelt et al. Incremental learning of object detectors using a visual alphabet. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2006.

[10] A. Smeulders, M. Worring, S. Santini, A. Gupta and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22:1349–1380, 2000.

[11] A. Torralba, K. P. Murphy, W. T. Freeman and M. A. Rubin. Context-based vision system for place and object recognition. In *Proceedings of the International Conference on Computer Vision*, 2003.

[12] A. Yilmaz and M. Shah. Action sketch: A novel action represetantion. In *Proceddings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[13] B. Fulkerson, A. Vedaldi and S. Soatto. Localizing objects with smart dictionaries. In *Proceedings of European Conference on Computer Vision*, 2008.

[14] B. Hendrickson. Latent semantic analysis and fiedlder retrieval. *SIAM Linear Algebra and its Application*, 421(1):345–355, 2007.

[15] B.A. Olshausen and D.J. Field. Sparse coding with an over-complete basis set: A strategy employed by v1? *Vision Research*, 37:3311–3325, 1997.

[16] B.S. Manjunath and W.Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842, 1996.

[17] B.V. Funt and G.D. Finlayson. Color constant color indexing. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(5):522–529, 1995.

[18] C. Fanti, L. Zelnik-Manor and P. Perona. Hybrid models for human recognition. In *Proceedings of International Conference on Computer Vision*, 2005.

[19] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the Aley Vision Conference, pp. 147–151*, 1988.

[20] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(5):530–536, 1997.

[21] C. Schuldt and I. Laptev. Recognizing human actions: A local svm approach. In *Proceedings of the International Conference on Pattern Reconition*, 2004.

[22] C.P. Papageorgiou, M. Oren and T. Poggio. A general framework for object detection. In *Proceedings of the International Conference on Computer Vision*, 1998.

[23] D. Blei, A. Ng and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(1):993–1022, 2003.

[24] D. G. Lowe. The viewpoint consistency constraint. *International Journal of Computer Vision*, 1(1):57C72, 1987.

[25] D. G. Lowe. Recognizing solid objects by alignment with an image. *International Journal of Computer Vision*, 5(2):195C212, 1990.

[26] D. Hogg. Model-based vision: A program to see a walking person. *Image and Vision Computing*, 1(1):5–20, 1983.

[27] D. Kersten. Predictability and redundancy of natural images. *Journal Optical Soc. Am. A*, 4(2):2395–2400, 1987.

[28] D. Ramanan and D. Forsyth. Automatic annotation of everyday movements. In *Technical Report UCB/CSD-03-1262, EECS Department, University of California, Berkeley.*, 2003.

[29] D. Slater and G. Healey. The illumination-invariant recognition of 3d objects using color invariants. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(2):206–210, 1996.

[30] D. Weinland, E. Boyer and R. Ronfard. Action recognition from arbitrary views using 3d examplars. In *Proceedings of the International Conference on Computer Vision*, 2007.

[31] D. Weinland, R. Ronfard and E. Boyer. Free viewpoint action recognition using motion history volumes. *Computer Vision and Image Understanding*, 104(2-3):249–257, 2006.

[32] D. Ziou, and S. Tabbone. Edge detection techniques an overview. *International Journal of Pattern Recognition and Image Analysis*, 8(4):537–559, 1998.

[33] E. Nowak, B. Triggs and F. Jurie. Sampling strategies for bag-of-features image classification. In *Proceedings of the European Conference on Computer Vision*, 2006.

[34] E. Shechtman and M. Irani. Space-time behavior based correlation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[35] F. Lv and R. Nevatia. Single view human action recognition using key pose matching and viterbi path searching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[36] F. Moosmann, B. Triggs and F. Jurie. Fast discriminative visual codebooks using randomized clustering forests. In *Proceedings of the Neural Information Processing Systems Conference*, 2006.

[37] G. Cheung, S. Baker and T. Kanade. Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[38] G. Johansson. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics*, 14(1):201–211, 1973.

[39] G. Schindler, L. Zitnick and M. Brown. Internet video category recognition. In *Proceedings of the workshop of Internet Vision*, 2008.

[40] H. J. Wolfson and I. Rigoutsos. Geometric hashing: An overview. *IEEE Computer Science Engeering*, 4(4):10C21, 1997.

[41] H. Meng, N. Pears and C. Bailey. A human action recognition system for embedded computer vision application. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[42] H. Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. In *Tech Report CMU-RI-TR-3 Carnegie-Mellon University, Robotics Institute*, 1980.

[43] H. Ning, W. Xu, Y. Gong and T. S. Huang. Latent pose estimator for continuous action recognition. In *Proceedings of the European Conference on Computer Vision*, 2008.

[44] H. Ning, Y. Hu and T.S. Huang. Discriminative learning of visual words for 3d human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[45] I. Laptev. On space-time interst points. *International Journal of Computer Vision*, 64(2-3):107–123, 2005.

[46] I. Laptev and T. Lindeberg. Space-time interest points. In *Proceedings of the IEEE International Conference on Computer Vision*, 2003.

[47] I. Laptev, M. Marszalek, C. Schmid and B. Rozenfeld. Learning realistic human actions from movies. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[48] I. S. Dhillon, S. Mallela and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of the Special Interest Group on Knowledge Discovery and Data Mining*, 2003.

[49] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8:679–714, 1986.

[50] J. Davis. Hierarchical motion history images for recognizing human motion. In *Proceedings of the IEEE Workshop on Detection and Recognition of Events in Video*, 2001.

[51] J. Huang, S.R. Kumar, M. Mitra, W. Zhu and R. Zabih. Image indexing using color correlograms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 762-768*, 1997.

[52] J. Liu and M. Shah. Scene modeling using co-clustering. In *Proceedings of the IEEE International Conference on Computer Vision*, 2007.

[53] J. Liu and M. Shah. Learning human actions via information maximization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[54] J. Liu, J. Luo and M. Shah. Recongizing realistic actions from videos 'in the wild'. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[55] J. Liu, S. Ali and M. Shah. Recognizing human actions using multiple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[56] J. Liu, Y. Yang and M. Shah. Learning semantic visual vocabularies using diffusion distance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[57] J. Liu, Y. Zhai, A. Basharat, B. Orhan, S. Khan, H. Noor, P. Berkowitz and M. Shah. University of central florida at trecvid 2006: High-level feature extraction and video search. In *Proceedings of the TREC Video Retrieval Evaluation Workshop (TRECVID), Gaithersburg*, 2006.

[58] M. Urban J. Matas, O. Chum and T. Pajdla. Robust wide baseline stereo from maximally stable extremum regions. In *Proceedings of British Machine Vision Conference*, 2002.

[59] J. Niebles and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. In *Proceedings of British Machine Vision Conference*, 2006.

[60] J. Niebles and L. Fei-Fei. A hierarchical model of shapes and appearance for human action classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[61] J. Ponce and J. Brady. Towards a surface primal sketch. *Three Dimensional Machine Vision*, pages 195–240, 1987.

[62] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[63] J. Shotton et al. Contour based learning for object detection. In *Proceedings of the International Conference on Computer Vision*, 2005.

[64] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman and W. T. Freeman. Discovering objects and their location in images. In *Proceedings of the IEEE International Conference on Computer Vision*, 2005.

[65] J. Vogel and B. Schiele. Natural scene retrieval based on a semantic modeling step. In *Proceedings of the IEEE Conference on Image and Video Retrieval*, 2004.

[66] J. Winn, A. Criminisi and T. Minka. Object categorization by learned universal visual dictionary. In *Proceedings of the International Conference on Computer Vision*, 2005.

[67] J.J. Koenderink and A.J. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, 55(1):267–375, 1987.

[68] J.L. Bentley. Multidimensional binary search trees in database applications. *IEEE Trans. Software Engineering*, SE-5(4):333–340, 1979.

[69] A. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

[70] K. Grauman and T. Darrell. Pyramid match kernels: Discriminative classification with sets of image features. In *Proceedings of the International Conference on Computer Vision*, 2005.

[71] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[72] K. Mikolajczyk and H. Uemura. Action recognition with motion-appearance vocabulary forest. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[73] K. Rohr. Towards model-based recognition of human movements in image sequences. *Graphical Model and Image Processing*, 59(1):94–115, 1994.

[74] K. Schindler and L. V. Gool. Action snippets: How many frames does human action recognition require? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[75] K. Yan and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Proceddings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 506–513, 2004.

[76] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[77] L. Kennedy, A. Hauptmann, S.F. Chang, J.R. Smith. Lscom lexicon definitions and annotations version 1.0. In *In DTO Challenge Workshop on Large Scale Concept Ontology for Multimedia*, 2006.

[78] L. Yang, R. Jin, R. Sukthankar and F. Jurie. Unifying discriminative visual codebook generation with classifier training for object category reorganization. In *Proceddings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[79] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(91):91–110, 2004.

[80] L.W. Campbell and A.F. Bobick. Recognition of human body motion using phase space constraints. In *Proceedings of the International Conference on Computer Vision*, page 624C630, 1995.

[81] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. Assoc. Comp. Mach.*, 24(6):381C395, 1981.

[82] M. Balasubramanian and E. Schwartz. The isomap algorithm and topological stability. *Science*, 295(5552):7–13, 2002.

[83] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[84] M. Blank, L. Gorelick, E. Shechtman, M. Irani and R. Basri. Actions as space-time shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[85] M. Brand, N. Oliver and A. Pentland. Coupled hidden markov models for complex action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[86] M. Leordeanu and M. Hebert. A spectral technique for cor-respondence problem using pairwise constraint. In *Proceedings of the IEEE International Conference on Computer Vision*, 2005.

[87] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision*, 62(1–2):61–81, 2005.

[88] M.J. Swain and D.H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[89] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[90] N. Katamaya and S. Satoh. The sr-tree: An index structure for high-dimensional nearest neighbor queries. In *ACM's Special Interest Group on Management Of Data*, 1997.

[91] N. Slonim and N. Tishby. Document clustering using word clusters via the information bottleneck method. In *Proceedings of the ACM Special Interest Group on Information Retrieval*, 2000.

[92] N. Tishby, F. Perira and W. Bialek. The information bottleneck method. In *Proceedings of the 37-th Annual Allerton Conference on Communication, Control and Computing*, 1998.

[93] P. Dollar, V. Rabaud, G. Cottrell and S. Belongie. Hierarchical motion history images for recognizing human motion. In *ICCV workshop: VS-PETS*, 2005.

[94] P. Pantel and D. Lin. Discovering word scenes from text. In *Proceedings of Special Interest Group on Knowledge Discovery and Data Mining*, 2002.

[95] P. Quelhas, F. Monay, J.-M Odobez, D. Gatica-Perez, T. Tuytelaars and L. Van Gool. Modeling scenes with local descriptors and latent aspects. In *Proceedings of the International Conference on Computer Vision*, 2005.

[96] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[97] Q. Zhu, S. Avidan, M. Ye and K-T Cheng. Fast human detection using a cascade of histograms of oriented gradients. In *Proceddings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[98] R. C. Bolles and R. Horaud. *3DPO: A Three-Dimensional Part Orientation System.* Kluwer Academic Publishers, 1987.

[99] R. Fergus, L. Fei-Fei, P. Perona and A. Zisserman. Learning object categories from google's image search. In *Proceedings of the 10th International Conference on Computer Vision, Beijing, China*, volume 2, pages 1816–1823, 2005.

[100] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[101] R. Fergus, P. Perona, and A. Zisserman. A visual category filter for google images. In *Proceedings of European Conference on Computer Vision*, 2004.

[102] R.R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational harmonic Analysis*, 21:5–23, 2006.

[103] S. Belongie, J. Malik and J. Puzicha. Shape matching and object recognition using shape context. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2(4):509–522, 2002.

[104] S. Lafon and A. B. Lee. Diffusion maps and coarse-graining: a unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):1393–1430, 2006.

[105] S. Lazebnik and M. Raginsky. Supervised learning of quantizer codebooks by information loss minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

[106] S. Lazebnik, C. Schmid and J. Ponce. Sparse texture representation using affine-invariant neighborhoods. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[107] S. Lazebnik, C. Schmid and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[108] S. M. Smith and J. M. Brady. Susan - a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

[109] S. Nowozin, G. Bakir and K. Tsuda. Discrinative subsequence mining for action classification. In *Proceedings of the International Conference on Computer Vision*, 2007.

[110] S. Savarese, J. Winn and A. Criminisi. Discriminative object class models of appearance and shape by correlatons. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[111] S. Wong, T. Kim and R. Cipolla. Learning motion categories using both semantics and structural information. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[112] S.C. Zhu. Statistical modeling and conceptualization of visual patterns. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(6):1–22, 2003.

[113] S.G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1985.

[114] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42:177–196, 2001.

[115] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43(1):29–44, 2001.

[116] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):117–154, 1998.

[117] T.O. Binford. Visual perception by computer. In *Proceedings of the IEEE Conference on Systems and Control*, 1971.

[118] V. Ferrari, L. Fevrier, F. Jurie and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1):36–51, 2008.

[119] V. Ferrari, T. Tuytelaars and L. Van Gool. Object detection by contour segment networks. In *Proceedings of the European Conference on Computer Vision (ECCV), Graz*, 2006.

[120] W. E. L. Grimson and T. Lozano-Perez. Localizing overlapping parts by searching the interpretation tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):469C482, 1987.

[121] W. Freeman and E. Adelson. The design and use of steerable filters. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[122] W.H. Hsu and S. Chang. Visual cue cluster construction via information bottleneck principle and kernel density estimation. In *Proceedings of the Conference on Image and Video Retrieval*, 2005.

[123] Y. Ke, R. Sukthankar and M. Hebert. Efficient visual event detection using volumetric features. In *Proceedings of the IEEE International Conference on Computer Vision*, 2005.

[124] Y. Lamdan, J. T. Schwartz, and H. J. Wolfson. Object recognition by affine invariant matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1988.

[125] Y. Rui, TS Huang, M. Ortega and S. Mehrotra. Relevance feedback: A power tool for interactive content-based image retrieval. *IEEE Transactions on Circuits and Video Technology*, pages 94–115, 1998.

[126] Y. Song, L. Goncalves and P. Perona. Unsupervised learning of human motion. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(25):1–14, 2003.

[127] Y. Wang, H. Jiang, M.S. Drew, Z. Li and G. Mori. Unsupervised discovery of action classes. In *Proceddings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.