VISUAL CLASSIFICATION BY MULTIPLE FEATURE FUSION AND LARGE-SCALE LEARNING

Dissertation

Submitted in partial fulfillment of
the requirement for the degree

Doctor of Philosophy in Electrical Engineering

at

The City College of New York

of the

City University of New York

by

Shizhi Chen

August 2013

Approved:

_____

Professor YingLi Tian, Dissertation Advisor

_____

Professor Kenneth M. Sobel, Deputy Chair
Department of Electrical Engineering

VISUAL CLASSIFICATION BY MULTIPLE FEATURE FUSION AND LARGE-SCALE LEARNING

by

Shizhi Chen

Submitted to the Department of Electrical Engineering
on August 5, 2013, in partial fulfillment of the
requirement for the degree of
Doctor of Philosophy in Electrical Engineering

Abstract

Robust visual classification can be applied to numerous practical applications, such as augmented reality, personal robotics and medical image analysis. The main challenges for visual classification are accuracy and efficiency in terms of both computation and memory. This dissertation will present efforts to address these challenges.

To improve accuracy, we propose a new feature representation, i.e. EigenMap, and a novel multiple-kernel learning framework, i.e. margin-constrained multiple-kernel learning. The EigenMap utilizes kernel density estimation to approximate the location probability of features in an image. Hence, the proposed feature representation incorporates both appearance and spatial information of local regions.

A popular methodology is to utilize the complementarity of multiple feature types, by either a simple concatenation or multiple-kernel learning (MKL). The method of the simple concatenation of feature vectors from different feature types requires manual feature selection and careful parameter tuning. On the other hand, MKL can automatically combine

different feature types to achieve better performance. However, the MKL tends to only select the most discriminative feature type and ignore other less discriminative feature types, which may provide complementary information for visual classification.

In order to better utilize complementary features with less discriminative power, we propose a margin-constrained multiple-kernel learning (MCMKL) method by extending MKL with margin constraints and dimensionally normalized kernel. The proposed MCMKL method learns weights of different feature types, i.e., base features, according to their discriminative power. Unlike the conventional MKL, MCMKL incorporates less discriminative base features by assigning smaller weights when constructing the optimal combined kernel, so that we can fully take the advantages of complementary multiple features to improve the accuracy of visual classification.

To improve efficiency of visual classification, especially on a large-scale classification with a large number of categories, a typical approach is to use one-vs-all linear Support Vector Machine (SVM). However, it has been criticized that the complexity increases linearly with the number of categories. On the other hand, nearest-neighbor (NN) based classifiers can naturally handle large numbers of categories and do not have learning step. But the inferior performance, as compared with learning based classifiers, as well as expensive computation and memory costs have hindered these classifiers on large numbers of classification categories.

We propose a novel classifier scheme, i.e., the Discriminative Hierarchical K-Means Tree (D-HKTree), which combines the advantages of both NN-based and learning-based classifiers for large-scale classification. It incorporates learning-based classifier into NN-based classifier, and extends

the NN-based classifier to large-scale dataset with significantly lower computational cost and memory usage. At the same time, the D-HKTree can still achieve the state-of-the-art classification accuracies on several challenging datasets.

The main contributions of this dissertation include a new spatially encoded object representation and novel classifier frameworks to improve both accuracy and efficiency of visual classification. The proposed EigenMap object representation incorporates spatial context into the popular bag of words model without manually partitioning an image into a set of sub-regions. We also extend the multiple-kernel learning framework, with margin constraints and dimensionally normalized kernel, in order to maximize joint discriminative power of multiple complementary features. Finally, we propose a novel classifier scheme, i.e., Discriminative Hierarchical K-Means Tree (D-HKTree), to take advantages of both learning based and nearest neighbor based classifiers for large-scale visual classification.

Acknowledgment

I would like to express my deepest appreciations to my advisor, Professor YingLi Tian, for her guidance, patience and understanding. She not only teaches me to become a good researcher in computer vision field, but also encourages me to apply the research to benefit our society and help people with special needs. This dissertation would not have been possible without her consistent support and encouragements.

I would like to thank my dissertation committee members, Professor Zhigang Zhu, Professor Fred Moshary and Dr. Rogerio Schmidt Feris for their valuable comments and suggestions on revising the final dissertation.

I would also like to thank my colleagues and friends in the CCNY Media Lab, including Xiaodong Yang, Chucai Yi, Chenyang Zhang, Yang Xian, Carol Mazuera, Ze Ye, Hangrong Pan, Long Tian, Baiyu Xi, Shuai Yuan, Shuihua Wang, Michael Quintian, Faiz Hasanuzzaman, and many others for their friendship and supports. Special thanks go to Xiaodong and Chucai for the discussions of many research topics and their constructive comments on revising this dissertation. I also enjoy the collaboration with Xiaodong for the large scale learning part of this dissertation.

Furthermore, I would like to acknowledge NOAA CREST with much appreciation, which has been providing financial support for my Phd study.

Finally, I cannot make it so far without the unbound love and continuous support from my family. I would like to express my deepest gratitude to my parents for their dedication and love. I also want to say thank you to my wife, Man Ting Wong, for her patience and understanding over the Phd study. I appreciate my wonderful children, Shuyi Chen and Shuhui Chen. They have given me so much fun and the experience as a parent.

List of Figures

Figure 1: Some sample images for objects and scenes 15

Figure 2: Box filter approximation of Gaussian function 25

Figure 3: Interest points detected by SURF 27

Figure 4: Construct SIFT descriptor 29

Figure 5: Haar wavelet filters 29

Figure 6: Reprentation of an image 31

Figure 7: Bag of Words representation for visual classification . . . 32

Figure 8: Construction of Bag of Words model 33

Figure 9: Spatial layout of Spatial Pyramid Matching 37

Figure 10: Hyper-plane of linear SVM 43

Figure 11: One-vs.-all SVM and hierarchical SVM 53

Figure 12: Class taxonomy with four levels 54

Figure 13: Relaxed hierarchy for four class data 56

Figure 14: Concept of image-to-class distance 63

Figure 15: Difference between NBNN and local NBNN 66

Figure 16: EigenMap representation for visual classification 71

Figure 17: Segmentation example 73

Figure 18: EigenMap Generation 75

Figure 19: Location likelihood using kernel density estimation . . . 76

Figure 20: Sample images of UIUC Sport Scene Database 77

Figure 21: Sample images of the Natural Scene database 78

Figure 22: EigenMap result on scene databases 79

Figure 23: Five-fold cross validation result of EigenMap 81

Figure 24: Sample confusion matrices of EigenMap 82

Figure 25: Compare EigenMap with state of the arts 83

Figure 26: The effect of number of eigenvectors 84

Figure 27: Feature discrimination from SVM margin 89

Figure 28: Multi-modal fusion for affect recognition 93

Figure 29: Facial features 94

Figure 30: Body gesture features 94

Figure 31: Temporal segmentation 96

Figure 32: Sample video in FABO database 98

Figure 33: The top 12 performances over fusion algorithms 98

Figure 34: Best performance of fusion algorithms 100

Figure 35: Feature weight distribution over base features 101

Figure 36: Optimal kernel parameter γ vs. feature dimension 102

Figure 37: Contamination from noisy feature 104

Figure 38: Labeled Hierarchical K-means Tree (L-HKTree) 107

Figure 39: The framework of D-HKTree framework 108

Figure 40: Illustration of unit step function 111

Figure 41: Comparing memory usage 120

Figure 42: Comparing relative computational complexity 122

List of Tables

Table 1: YouTube statistics in 2012 16

Table 2: Feature dimension of base features 99

Table 3: Comparison to NN-based classifiers 118

Table 4: Comparison to learning based classifiers 121

Table 5: Comparison to the hybrid classifiers 125

Contents

1 Introduction 14

1.1 Applications and Challenges of Visual Classification . . . 14

1.2 Improving Classification Accuracy 16

1.2.1 Spatially Encoded Object Representation 16

1.2.2 Multiple Features Fusion 17

1.3 Improving Classification Efficiency for Large-scale Learning 19

1.4 Dissertation Organization 20

2 Related Work 22

2.1 Local Features for Object Representation 22

2.1.1 Feature Extraction 23

(A) Interest Point Detector 23

(B) Interest Point Descriptor 28

2.1.2 Object Representation 30

(A) Bag of Words Representation 32

(B) Bag of Words in Spatial Context 36

2.2 Multiple-Feature Fusion 38

2.2.1 Direct Concatenation 39

2.2.2 Single-Kernel Learning 42

2.2.3 Multiple-Kernel Learning (MKL) 48

2.3 Large-scale Learning 51
2.3.1 Discriminative Learning 52
2.3.2 Nearest Neighbor based Classifier 61

3 Spatially Encoded EigenMap Representation 70

3.1 Summary 70
3.2 Method 72
3.2.1 Overview 70
3.2.2 Feature Extraction 72
(A) Region Features:
Texture, Shape, and Color 72
(B) Interest Point Features:
Uniform Grids and Harris Corners 73
3.2.3 Codebook Formation and Feature Quantization . . 74
3.2.4 EigenMap Generation 75
3.2.5 Classifier 77
3.3 Experiments 77
3.3.1 Databases 78
3.3.2 Experimental Setups 80
3.3.3 Experimental Results 80
(A) Compare to the Bag of Words Model
and the LDA model 80
(B) Compare to the State-of-the-art
Performance 83
(C) Select Number of Principal Components
for EigenMap 84
3.4 Discussion 84

4 Margin-Constrained Multiple Kernel Learning 86

4.1 Summary 86
4.2 Method 87
4.2.1 Multiple-Kernel Learning (MKL) 87
4.2.2 Margin Constraints 88
4.2.3 Dimensionally Normalized Kernel 90
4.3 Multi-Modal Fusion for Affect Recognition 91
4.3.1 Overview of MCMKL-based Affect Recognition . . 92
4.3.2 Facial Features 93
4.3.3 Body Gesture Features 94
4.3.4 Temporal Segmentation 95
4.3.5 Temporal Normalization 95
4.3.6 MCMKL Based Multi-Modal Feature Fusion . . . 96
4.4 Experiments 97
4.4.1 Experimental Setups 97
4.4.2 Comparison to Existing Work and MKL 98
4.4.3 Evaluate Feature Weight Distribution 101
4.4.4 Contamination from
Less Discriminative Features 104
4.5 Discussion 105

5 Discriminative Hierarchical K-Means Tree 106

5.1 Summary 106
5.2 Construction of Discriminative
Hierarchical K-means Tree 107
5.2.1 Algorithm Overview 108

5.2.2 Labeled Hierarchical K-means Tree 109
(A) Towards L-HKTree 109
(B) Building L-HKTree 113
(C) Pre-Classification with L-HKTree 114
5.2.3 Discriminatively Learned Histogram 115
5.2.4 Classification with D-HKTree 116
5.3 Experiments 117
5.3.1 Experimental Setup 118
5.3.2 Comparisons to NN-based Classifiers 118
5.3.3 Comparisons to Learning-based Classifiers 121
5.3.4 Comparisons to Hybrid Classifiers 124
5.4 Discussion 125

6 Conclusion and Future Work 127

6.1 Discussions and Conclusion 127
6.2 Limitations and Future Work 128

Bibliography 130

My Publication List 143

Chapter 1

Introduction

Visual classification is a process of recognizing an object or understanding a concept by analyzing visual appearance. It categorizes an object or a concept into a group, within which all objects or concepts share similar properties. For the convenience of discussion, we use object recognition as a general term for both physical object recognition and concept understanding in this dissertation.

1.1 Applications and Challenges of Visual Classification

Robust visual classification has been an active research area [22, 28, 40, 44] in computer vision field. It remains a driving force for many practical applications, e.g., human computer interaction (HIC), surveillance, augmented reality, personal robotics and medical applications. Nevertheless, two main challenges, i.e., accuracy and efficiency, have to be overcome, in order for visual classification to have a wide range of applications.

Figure 1: Some sample images for objects and scenes show large intra-class variations, which includes occlusion, viewing direction changes, and scale changes.

An object appears different under varying conditions, e.g., occlusion, scale change and viewing angle change. These variations adversely affect the accuracy of visual classification. Figure 1 shows some example images from several object classes, i.e., butterfly and killer whale, and from scene classes, i.e., golden gate bridge and beach. Each class exhibits large intra-class variations due to changes of viewing direction, scale, and occlusion etc. It is still an open question how to represent an object to maintain high

Table 1: YouTube statistics of video watched and uploaded per month in the year of 2012 [110].

unique users	hours of video watched	hours of video uploaded
1 billion	4 billion	3 million

discriminative power, while achieving intra-class invariance. It is generally believed that multiple features are necessary to accurately recognize an object. However, it is not clear what is the best way to combine these features to achieve a high accuracy.

On the other hand, efficiency in terms of both computation and memory has become ever more important when designing a visual classification system, as the size of available data increases exponentially. Table 1 shows YouTube statistics per month in the year of 2012, released by Google [110]. Each month, there are more than 3 million hours of video uploaded, and 4 billion hours of video watched by viewers. In this dissertation, we present an effort to address both challenges, i.e., accuracy and efficiency for visual classification.

1.2 Improving Classification Accuracy

To improve accuracy, the most straightforward approach is to construct a more discriminative object representation from features [26, 49]. The other approach is to combine multiple feature types by utilizing their complementarity [22, 40, 88].

1.2.1 Spatially Encoded Visual Representation

The bag of words (BOW) object representation has been adapted in many state-of-the-art visual classification systems [26, 53] due to its simplicity and excellent performance. However, it has ignored all spatial relationships among different object parts. Spatial Pyramid Matching (SPM) [49] incorporates such spatial relationship by manually partitioning an image into multiple sub-regions, and constructs a BOW histogram for each sub-region. Then all BOW histograms are concatenated together as the final object representation. Even though SPM models absolute spatial information of object parts based on image space, it has outperformed BOW in many challenging datasets [49].

However, SPM requires partition of an image into sub-regions. And the dimension of final object representation can be very large if a large codebook size is used. We propose a new spatially encoded object representation, i.e., EigenMap, which does not require manual partition of image and has much lower dimension as compared to SPM. The density distribution of object parts over the image space is captured by their EigenMaps. The collection of these EigenMaps incorporates both appearance and spatial relationship of object parts.

1.2.2 Multiple Features Fusion

Popular approaches to combine multiple feature types are simple concatenation [22, 40, 78, 108] and multiple kernels learning (MKL) method [2, 73, 88, 89]. Due to the complexity of visual classification, no single feature is able to provide the best tradeoff between discriminative power and invariance in all datasets. Some features might be scale invariant, while others might be rotation invariant. Depending on dataset, one feature might

perform better than another, or vice versa. Higher classification accuracy is generally obtained, by combining multiple feature types together.

A straightforward way to combine multiple feature types is simple concatenation, which results in a feature vector representation with large dimension [22, 40, 78, 108]. The simple concatenation is very easy to implement, and achieves higher accuracy if right features are combined. Nevertheless, the right features have to be manually selected, and they are different for different datasets. Hence, simple concatenation requires expert knowledge on both features and datasets, in order to obtain optimum performance. Furthermore, the simple concatenation fusion method is vulnerable to the contamination of less discriminative feature types, especially those with large feature dimensions.

The multiple-kernel learning (MKL) [2, 73, 88, 89] is able to partially eliminate some drawbacks of the simple concatenation fusion method. It can automatically select most discriminative feature types for a dataset. The MKL also provides shielding from the contamination of less discriminative base features by assigning very large weights to the most discriminative base features. It has recently shown the effectiveness to fuse multiple base features in object detection and recognition [88, 89]. However, MKL tends to select only the most discriminative base features and ignore other less discriminative base features. Therefore, MKL method cannot fully take the advantages of all types of base features from heterogeneous modalities, which usually provide complementary information.

In order to address these issues, we propose a margin-constrained multiple-kernel learning (MCMKL) by applying additional margin constraints and dimensionally normalized kernel. Unlike conventional MKL method, our proposed MCMKL is able to learn the most discriminative base

features while still considering other base features, which are less discriminative, but can potentially provide complementary information. We apply MCMKL method on affect recognition from multiple modalities (e.g. face and body gesture). The extensive experimental results on the FABO (Face and Body Gesture) facial expression database [41] demonstrate the effectiveness of the proposed method for multi-modal feature fusion.

1.3 Improving Classification Efficiency for Large-scale Learning

Efficiency of visual classification, in terms of computational cost and memory usage, has become ever more important as the available data size increases exponentially. Most learning-based algorithms, *e.g.*, one-versus-all linear SVM [19, 25], have computation cost increasing at least linearly with the number of object classes [14, 49, 56, 100]. At the testing phase for these methods, the computation cost becomes infeasible when scaling up to large numbers of classes. To improve efficiency of linear SVM for large-scale object classification, Hierarchical SVM-based methods [34, 38, 59] utilize hierarchical decision structure so that the computational complexity only increases sub-linearly with the total number of classes. However, these approaches improve efficiency through compromising classification accuracy to some extent.

On the other hand, non-parametric nearest neighbor (NN) based classifiers require no training phase and can naturally handle large numbers of classes. However they have to retain all the training examples in testing phase, which becomes infeasible on large-scale datasets due to the expensive computation cost and memory usage. For example, the total memory

required to store dense SIFT features [58] for the training data of SUN dataset [99] is around 100 GB, which far exceeds the memory of a desktop (typically 4G – 48G).

In order to facilitate large-scale object classification by taking advantages of both learning-based and NN-based methods, we incorporate discriminative learning algorithms into NN-based methods as a novel classification framework, i.e., Discriminative Hierarchical K-means Tree (D-HKTree). The complexity of the proposed D-HKTree only grows sub-linearly with the number of categories, which is much better than the recent hierarchical SVM based methods. The memory usage in the D-HKTree also benefits from precluding all training features, which is order of magnitude less than the NN-based methods. In the evaluations on several object recognition and scene understanding dataset, i.e., Caltech 101 [31], Caltech 256 [39] and SUN [99] dataset, D-HKTree obtains state-of-the-art accuracies, while with significantly lower computation cost and memory requirement.

1.4 Dissertation Organization

The dissertation is organized as the following. Chapter 2 introduces the related work on features, object representations, multi-feature fusion algorithms and large-scale learning. Chapter 3 presents a new object representation, EigenMap, for object or scene classification. We evaluate the EigenMap on several scene datasets. Chapter 4 describes Margin Constraint Multiple Kernel Learning (MCMKL) in details. The MCMKL is evaluated on the FABO dataset for affect recognition. Large-scale learning with D-HKTree is presented in Chapter 5, and is evaluated on several large-scale

object recognition and scene understanding datasets. Finally, Chapter 6 concludes the dissertation and points out the future directions for visual classification.

Chapter 2

Related Work

Significant efforts on visual classification have been made in recent years [15, 48, 102], to improve classification accuracy and efficiency. In this chapter, we discuss related work from object representation to classifier designs, which enhance the performance of visual classification, in terms of accuracy and efficiency.

2.1 Local Features for Object Representation

Features for object representation can be roughly divided into two categories, i.e., global and local features. Global feature captures the overall appearance of an image or an object [68, 71, 82, 84], e.g., color histogram [82], principal component analysis [68, 84]. This approach has worked surprisingly well on some applications, e.g., image retrieval [82], face recognition [84]. Nevertheless, the global feature is usually not robust with partial occlusion, lighting change and view angle change etc.

On the other hand, the local feature [6, 33, 45, 46, 50, 58, 76] can naturally handle the aforementioned issues, as it captures information from a local patch. By utilizing the Bag of Words (BOW) representation [26, 60, 70, 81, 86, 96, 97], the local features have shown striking performance on

object and scene recognition [64, 99]. By incorporating spatial information in the BOW [18, 49, 63, 75, 101], the classification accuracy is further improved.

In this section, we will discuss various methods extracting local features. After features extracted, some popular algorithms, which organize local features to represent an object, are also presented.

2.1.1 Feature Extraction

Two main components are usually involved in feature extraction, i.e., interest point detector [4, 42, 58, 61, 65] and descriptor [4, 27, 58, 66]. Interest point detector extracts repeatable and distinctive local salient regions in an image. Interest point descriptor summarizes the characteristics of the neighbor pixels of each interest point based on either intensity or gradient information.

(A) Interest Point Detector

One of the most popular interest point detector is Harris corner based detector [42, 79], which is based on second moment matrix

$$A(x, y) = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad , \tag{2-1}$$

where I_x and I_y denote first derivatives of image intensity I at position (x, y) in x and y directions. If the second moment matrix has two large eigenvalues, the pixel (x, y) is considered as a corner. To avoid the expensive computation of eigenvalues decomposition, Harris and Stephens

[42] propose a cornerness measure M_c, which only depends on the determine and trace of matrix A.

$$M_c = Det(A) - k \times Tr^2(A) \quad , \tag{2-2}$$

where k is an empirical constant to be determined. However, Harris corner based detector is not scale invariant.

The other common interest point detector is based on Laplacian of Gaussian (LoG) [55, 58]. Blurred image $L(x, y, \sigma)$ is obtained by the convolution of the original image $I(x, y)$ with Gaussian kernel $G(x, y, \sigma)$, as shown in Equation (2-3).

$$L(x, y, \sigma) = G(x, y, \sigma) \otimes I(x, y) \quad , \tag{2-3}$$

where σ is standard deviation of Gaussian kernel. To detect scale invariant interest points, scale normalized Laplacian $\nabla^2_{norm} L(x, y, \sigma)$ is computed as

$$\nabla^2_{norm} L(x, y, \sigma) = \sigma^2 (L_{xx} + L_{yy}) \quad , \tag{2-4}$$

where L_{xx} and L_{yy} are second derivatives of L at pixel (x, y) in x and y directions. The interest point at the detected scale can be defined from scale-space maximum or minimum response of the scale normalized Laplacian, as shown in Equation (2-5).

$$(\hat{x}, \hat{y}, \hat{\sigma}) = \underset{(x, y, \sigma)}{\operatorname{argmaxmin}\, local} \quad \nabla^2_{norm} L(x, y, \sigma) \quad . \tag{2-5}$$

Scale Invariant Feature Transform (SIFT) [58] approximates scale normalized Laplacian $\nabla^2_{norm} L(x, y, \sigma)$ by Difference of Gaussian (DOG),

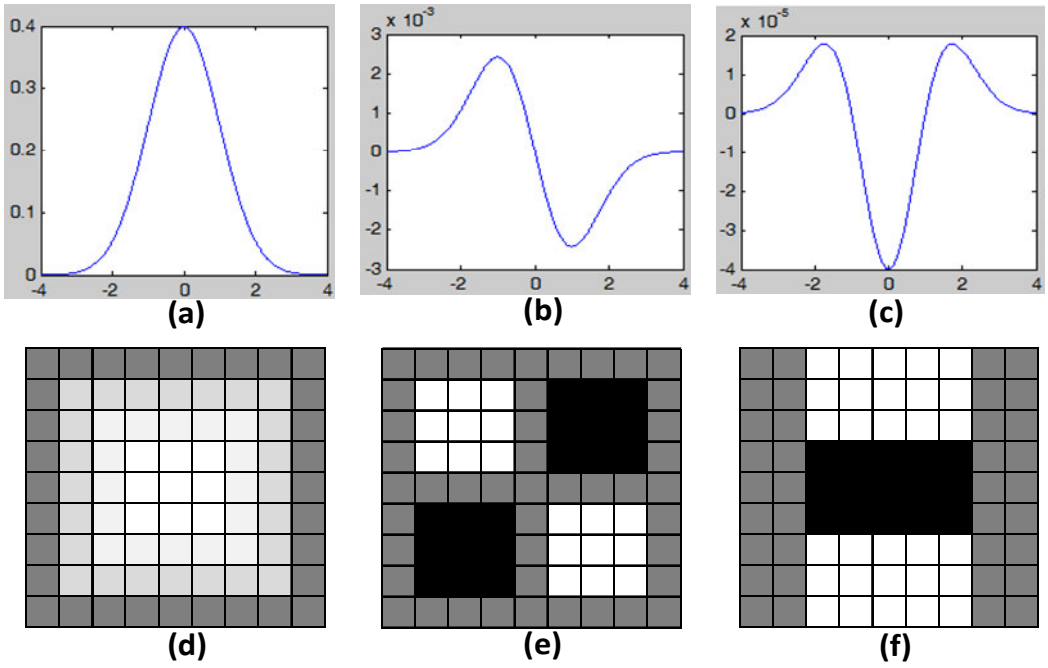


Figure 2: Box filter approximation of Gaussian function and their derivatives. (a) One dimensional (1-D) Gaussian function g(x) with standard deviation of 1; (b) First derivative of 1-D Gaussian function $\frac{dg}{dx}$; (c) Second derivative of 1-D Gaussian function $\frac{d^2g}{dx^2}$; (d) Approximated 2-D Gaussian function G(x,y) with standard deviation of 1.2; dark grey color indicate 0, and lighter color indicate larger positive value; (e) Box filter approximation of $G_{xy} = \frac{\partial^2 G}{\partial x \partial y}$; (f) Box filter approximation of $G_{yy} = \frac{\partial^2 G}{\partial y^2}$;

following the study [54]. Since Gaussian kernel satisfies diffusion equation over scale, the DOG approximation can be derived from Equations (2-6) and (2-7) below.

$$\sigma \nabla^2 L(x, y, \sigma) = \frac{\partial L(x, y, \sigma)}{\partial \sigma} = \frac{L(x, y, k\sigma) - L(x, y, \sigma)}{k\sigma - \sigma} \quad , \qquad (2\text{-}6)$$

$$\nabla^2_{norm} L(x, y, \sigma) = \sigma^2 \nabla^2 L(x, y, \sigma) \propto L(x, y, k\sigma) - L(x, y, \sigma) \quad , \qquad (2\text{-}7)$$

where k is a constant. Hence, the interest point can also be defined from scale-space maximum or minimum response of DOG in the right hand side of Equation (2-7). SIFT (Scale Invariant Feature Transform) detector has shown very robust performance, as it is invariant to scale, rotation and change in illumination etc.

To speed up the detection, Hessian matrix based Speeded Up Robust Feature (SURF) [4] detector simplify Gaussian filter using box filters as shown in Figure 2(e) and 2(f). Figure 2 shows box filter approximation of 2-dimensional (2-D) Gaussian function and its derivatives. Figure 2(a), (b) and (c) show 1-D Gaussian function, its first derivative and second derivative respectively.

Figure 2(d) shows the approximated 2-D Gaussian function G(x,y) with standard deviation of 1.2. Figure 2(e) and 2(f) show the box filter approximation of G_{xy} and G_{yy}respectively. The Hessian matrix is defined as

$$H(x, y, \sigma) = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{xy} & L_{yy} \end{bmatrix} \quad . \tag{2-8}$$

Therefore, the scale-normalized determinant of Hessian matrix will be

$$Det_{norm}(H) = \sigma^4 (L_{xx} L_{yy} - L_{xy}^2) \quad . \tag{2-9}$$

By using box filters to approximate Gaussian kernel, the determinant of the scale-normalized Hessian matrix [4] becomes

$$Det_{norm}(H_{approx}) = D_{xx} D_{yy} - (0.9 D_{xy})^2 \quad , \tag{2-10}$$

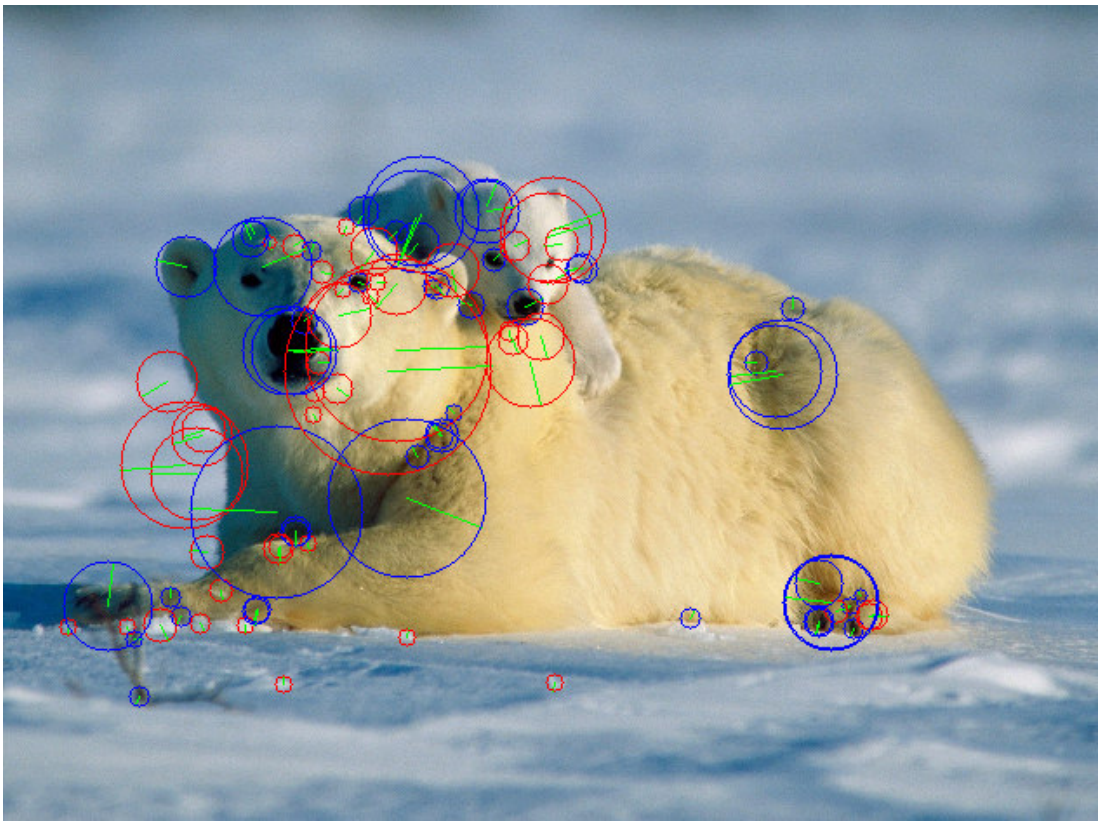


Figure 3: Interest points detected by SURF; The green line indicate the dominant orientation of an interest point. The size of a circle shows scale of interest point. The red circle indicates the intensity of an interest point is larger than its neighbor pixels, and the blue circle indicates the intensity of an interest point is smaller than its neighbor pixels.

where D_{yy} is the convolution of box filter in Figure 2(f), which is normalized by the filter size, with the original image $I(x, y)$. Similarly, D_{xx} and D_{xy} are the convolution of corresponding box filters with the original image. The convolution of box filter can be efficiently computed by integral image techniques [90]. Finally, interest point and its scale can be found by

$$(\hat{x}, \hat{y}, \hat{\sigma}) = \text{argmax}\, local_{(x,y,\sigma)}(Det_{norm}(H_{approx})) \quad . \qquad (2\text{-}11)$$

Figure 3 shows interest points detected by a popular implementation of SURF detector [29]. As shown in Figure 3, most detected interest points are on the two bears, instead of the snow background. This demonstrates that the extracted features can potentially form an excellent representation of bears.

For some applications, e.g., object recognition, scene understanding, uniform sampled pixels or dense sampling are also used as interest points [12, 13] in literatures. Better performance has been reported for those applications if dense sampling is used [12, 30, 51].

(B) Interest Point Descriptor

After an interest point detected, descriptor summarizes the characteristics of its neighborhood pixels by either intensity or gradient information. One of the most successful descriptors is SIFT descriptor, which is based on gradient [58].

Dominant orientation is first computed by accumulating a local orientation histogram of gradient directions over neighborhood patch centered at interest point. Then peaks are detected in the orientation histogram to find the dominant orientations.

Figure 4 illustrates the computation of a SIFT descriptor after the dominant orientation calculated. Along the dominant orientation, rectangular 4 x 4 grids are laid out on the image. In each grid, a local orientation histogram with 8 bins is accumulated by each pixel in the grid, weighted with their gradient magnitudes. By concatenating together orientation histograms from all grids, we have the final SIFT descriptor, which has 128 dimensions.

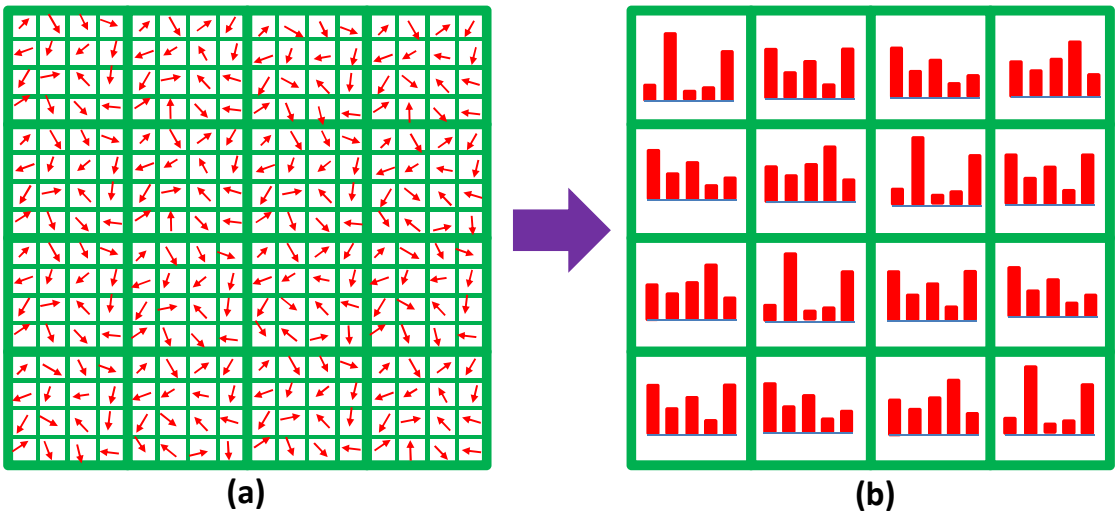


Figure 4: Construct SIFT descriptor. (a) Along the dominant orientation, 4 x 4 rectangular grids are laid out on the image, and the gradients are calculated for each pixel (small red arrow). (b) The corresponding local orientation histograms (8 bins) are calculated for each grid. Finally, SIFT descriptor is the concatenation of all oriental histograms.

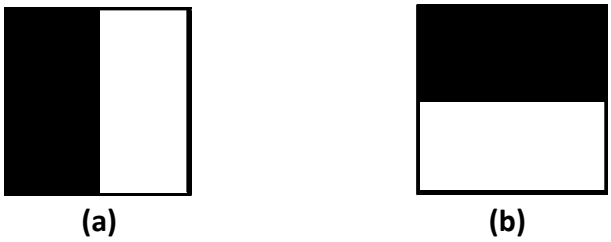


Figure 5: Haar wavelet filters used to compute horizontal and vertical response d_x and d_y, where light region is +1, and dark region is -1.

Motivated by the success of SIFT feature, extensions to the original SIFT feature are also proposed. Among them, SURF descriptor [4] is well known for speeding up the original SIFT descriptor by employing integral image technique. The dominant orientation is computed by detecting maximum response of Haar wavelet in a sliding orientation window.

Similar to SIFT descriptor, 4 x 4 rectangular grids are also laid out on the image along the dominant orientation. In each grid, the Haar wavelet responses d_x and d_y in the x and y directions are computed for each pixel within the grid. The Haar wavelet filters d_x and d_y are illustrated in Figure 5(a) and Figure 5(b) respectively. Then sum of wavelet responses ($\sum d_x$, dy) and their absolute responses (dx, dy) are computed for each grid. These four sums are used to represent a grid. The concatenation of all response sums in the 4 x 4 grids is the SURF descriptor.

PCA-SIFT [45] computes local maps of gradient magnitude over local patches surrounding interest points. Then principal component analysis (PCA) projects the gradient magnitude maps to low dimensional subspace. The resulted descriptor is much compact than the original SIFT descriptor, however with the price of losing distinctiveness. Color SIFT [12, 17, 95] extends regular SIFT with color information, and results in more distinctive descriptor.

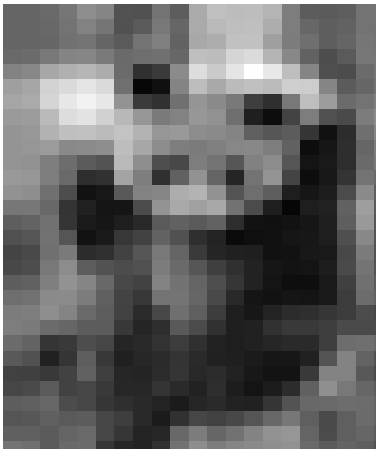
2.1.2 Object Representation

Object representation describes the represented object as a feature vector. The ideal object representation is discriminative enough to distinguish one object from another, while robust to variations, such as lighting and viewing direction etc.

To illustrate the importance of object representation, we use Figure 6 as an example, which shows an image of a panda with the size of 300 by 207 pixels. We convert the image to gray scale and down sample the image to smaller size, i.e., 30 by 21 pixels, as shown in Figure 6(b). With such small size, we can still see the panda from the image. However, if we represent the

(a)

(b)



97	106	119	127	131	85	82	119	83	169	190	176	199	177	134	99	95	92	110	121	97
97	104	114	118	119	91	93	137	80	183	172	191	200	174	126	112	105	91	115	124	114
104	110	119	118	113	97	107	135	85	181	158	196	198	170	128	101	84	91	126	114	115
123	135	149	157	143	113	91	94	105	197	206	226	230	214	168	111	81	82	109	114	112
148	185	207	214	208	127	23	15	140	220	203	209	229	236	230	175	101	75	100	116	110
170	211	230	233	234	141	18	15	94	160	142	147	140	105	157	213	168	93	121	121	120
161	196	220	231	227	177	126	85	125	137	137	122	24	7	49	168	138	91	122	126	122
145	187	203	211	202	176	188	149	143	149	150	135	70	37	92	99	30	43	114	142	120
152	173	165	173	160	188	179	145	143	133	135	135	122	116	108	25	24	35	94	130	108
148	128	117	124	89	196	159	125	102	122	125	119	128	139	99	20	25	39	95	133	115
149	131	108	71	53	182	129	36	50	140	100	50	118	150	91	22	24	42	106	116	108
145	125	94	39	25	126	150	80	133	164	131	33	108	140	38	22	26	49	134	100	95
136	111	71	26	19	32	108	160	189	182	183	116	102	65	18	24	28	67	166	90	92
102	114	60	18	21	31	33	103	124	115	151	84	37	17	19	24	30	75	166	82	97
83	117	85	16	29	55	39	87	90	67	51	22	16	20	20	22	33	70	154	85	91
76	108	119	20	36	58	67	123	93	52	33	30	22	18	19	21	30	55	134	88	88
72	107	138	56	76	73	77	125	102	77	55	46	30	21	20	21	28	54	127	85	84
86	116	149	138	92	88	65	127	109	95	75	57	37	23	19	20	24	68	123	80	87
101	127	144	130	98	84	58	124	117	99	79	48	31	26	19	19	22	49	105	95	96
120	141	147	109	94	81	33	89	117	94	78	33	20	31	22	20	29	52	110	90	76
124	130	131	100	98	75	22	58	107	92	65	27	34	35	58	70	73	64	63	58	68
113	76	65	78	82	54	31	54	83	56	32	42	39	33	61	48	47	87	101	98	110
97	41	45	88	63	33	44	37	51	38	38	42	25	25	33	30	61	128	109	93	93
82	45	53	113	61	31	36	36	56	47	37	31	22	21	23	31	103	136	90	77	88
70	62	68	92	55	34	43	51	52	38	39	35	24	22	29	63	138	132	99	83	89
88	102	82	56	61	42	44	42	32	34	41	33	27	29	56	116	126	119	113	93	87
86	101	91	77	69	59	38	34	32	48	53	48	47	70	105	98	87	98	103	101	97
93	90	83	121	64	59	38	31	79	93	83	100	110	130	148	98	67	96	107	95	94
110	96	90	108	57	53	38	31	111	148	119	116	145	146	153	109	75	99	99	86	81
114	82	113	118	45	44	38	40	97	128	130	144	148	127	135	115	102	96	87	78	78

(c)

Figure 6: Reprentation of an image; (a) original image (size of 300 x 207); (b) small-size image (size of 30 x 21); (c) The corresponding matrix (30 x 21) for the small-size image in (b);

down-sampled image in a matrix format with same size, as shown in Figure 6(c), there is no way we can tell where the panda is in the matrix, even though the matrix in Figure 6(c) contains identical information as that in Figure 6(b).

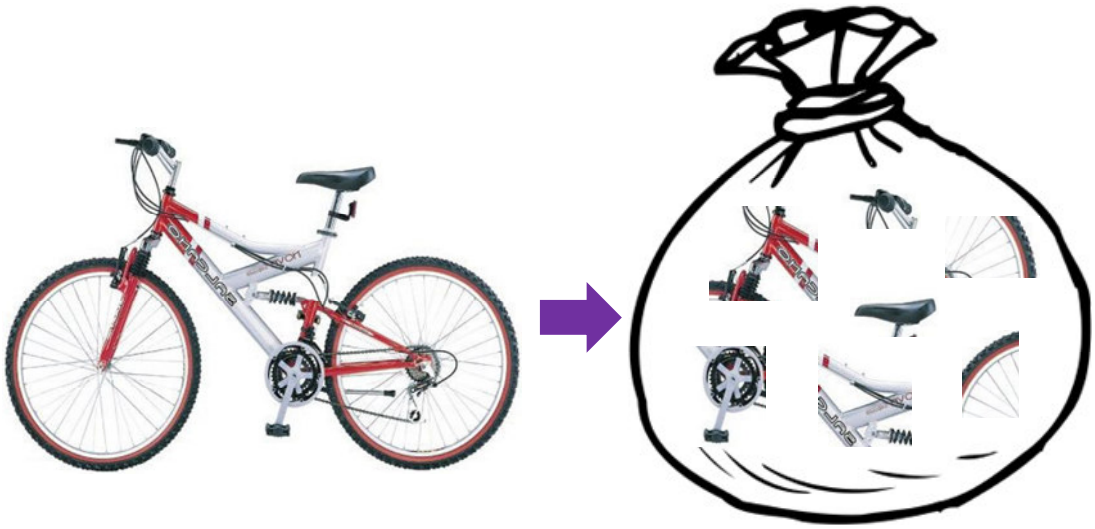


Figure 7: Illustrate Bag of Words representation for visual classification [111].

The example above demonstrates the challenges of object recognition in the computer vision field. On the other hand, it also shows the importance of object representation for accurate visual recognition. In this dissertation, we will focus on algorithms, which build object representation from local features. Some of the most popular approaches are Bag of Words (BOW) [26, 80] and its variants with spatial context [49, 56, 63, 100].

(A) Bag of Words Representation

The Bag of Words (BOW) is commonly used in document classification or text retrieval application [3]. A document is represented by a histogram of words, where each element in the histogram is frequency of occurrences of a corresponding word in the document.

Inspired by the success of the BOW model in text retrieval literature, computer vision community also adapts it to represent an image by treating local features as visual words. Figure 7 illustrates that a bicycle is represented by an orderless collection of local patches, i.e., components of

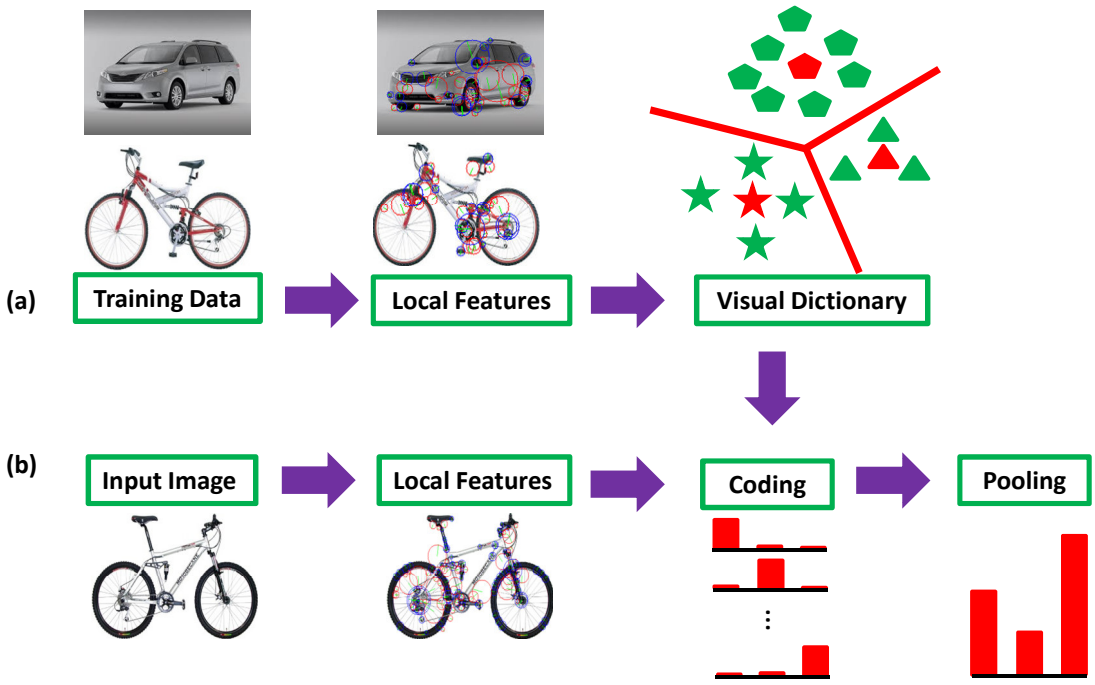


Figure 8: Pipeline to construct Bag of Words (BOW) model. (a) Visual dictionary construction; (b) image or object representation using BOW model [112].

the bicycle. The BOW model ignored spatial relationship between different components of the bicycle, which, however, is very important for human to recognize an object.

Despite the loss of spatial information, the BOW object representation has been successfully applied in computer vision applications, such as image retrieval and visual classification [26, 80, 96, 97, 106].

Different from words in a text, there are much more variations in local features. Hence nearby features in feature space are grouped together to form a cluster, which is represented by a prototype feature, i.e., a visual word. The collection of visual words from each cluster in training data forms a visual dictionary or codebook. Figure 8 shows a pipeline of BOW model including visual dictionary construction in Figure 8(a), and object representation in Figure 8(b).

We first extract local features, e.g., SURF features, from a set of training images or objects. Figure 8(a) shows local features, which are projected onto a two-dimensional space and marked by green color. Then similar local features are clustered together, and are represented by a prototype feature or visual word, marked with red color in Figure 8(a). Visual dictionary is formed by collecting all visual words from feature clusters, i.e., 3 visual words in Figure 8(a). A popular clustering algorithm is unsupervised K-Means clustering method [57].

To construct object representation, local features are also extracted from an input image. Then coding and pooling operators are used to generate the final BOW representation, as illustrated in Figure 8(b). In the coding step, local features are encoded using visual words in a dictionary. The simplest coding method is vector quantization, i.e., hard assignment. The hard assignment of a local feature is to assign the weight of 1 to the nearest neighbor visual word, while all other visual words are assigned with 0 weight. Equation (2-13) shows hard assignment code $\partial_i \in R^K$ for i^{th} local feature x_i in an image.

$$dist(x_i - d_k) = \|x_i - d_k\|_2^2 \quad . \tag{2-12}$$

$$\partial_{i,j} = \begin{cases} 1 & if \ \ j = \arg\min\limits_{k=1,\ldots,K} dist(x_i, d_k) \\ 0 & otherwise \end{cases} , \tag{2-13}$$

where $d_k$is k^{th} visual word in the codebook, and K is the codebook size, i.e., total number of visual words in the dictionary.

However, hard assignment of a local feature usually has large quantization error since it assigns weights to only one visual word. By considering soft probabilistic version of coding as shown in Equation (2-14),

soft assignment has normally achieved better performance in visual classification [36, 37].

$$\partial_{i,j} = \frac{e^{-\beta * dist(x_i, d_j)}}{\sum_{k=1}^{K} e^{-\beta * dist(x_i, d_k)}} \quad . \qquad (2\text{-}14)$$

Euclidean distance is only meaningful to approximate a geodesic distance within a local region in feature space [92, 103]. Instead of assigning weights to every visual word, local soft assignment [56] assigns weights to only n nearest neighbor visual words by modifying distance function of Equation (2-12) as shown in Equation (2-15).

$$dist(x_i, d_k) = \begin{cases} \|x_i - d_k\|_2^2 & if\ d_k \in NN_n(x_i) \\ \infty & otherwise \end{cases} \quad , \qquad (2\text{-}15)$$

where $NN_n(x_i)$ is n nearest neighbor visual words of the local feature x_i. The local soft assignment is a very simple coding method. Nevertheless, it achieves comparable performance with other more sophisticated coding scheme [56].

Sparse coding on the other hand reconstructs a local feature as a linear combination of a few visual words, as shown in Equation (2-16).

$$\partial_i = \underset{\partial_m}{\operatorname{argmin}} \quad \left\| x_i - \sum_{k=1}^{K} \partial_{m,k} * d_k \right\|_2^2 + \lambda \|\partial_m\|_1 \quad , \qquad (2\text{-}16)$$

where $\|\partial_m\|_1$ is L1 norm of ∂_m, λ is a parameter to control the sparsity of ∂_m.

Better coding techniques not only increase visual classification accuracy by eliminating some undesired noise, but also improve classification efficiency.

Pooling is to aggregate all codes of local features of an object into a single vector. The pooling vector, which summarizes the codes, is the BOW vector, i.e., the object representation. Two of the most popular pooling operators are average pooling (Equation (2-17)) and maximum pooling (Equation (2-18)).

$$p_k = \frac{1}{\|I\|} \sum_{i=1}^{\|I\|} \partial_{i,k} \quad , \qquad (2\text{-}17)$$

$$p_k = \max_{i=1,\ldots,\|I\|} \partial_{i,k} \quad , \qquad (2\text{-}18)$$

where $\|I\|$ is the total number of local features in an image, and p_k is k^{th} element of the BOW vector. Maximum pooling has achieved higher accuracy and is employed in most state-of-the-art systems for visual classification [15, 56].

(B) Bag of Words in Spatial Context

A major limitation, using the bag of words (BOW) model as an object representation is that it only models an object as a collection of local features without considering features' spatial information in the object. As proven by many researchers, knowing spatial relationship among different object parts can be very important in visual classification [18, 49, 75].

Both absolute spatial information [49, 63] and relative spatial information [75] have been incorporated into the bag of words representation. Absolute spatial context is the location information of local features with reference to the image coordinates. Despite the fact that absolute spatial information of local features is not invariant to translation or

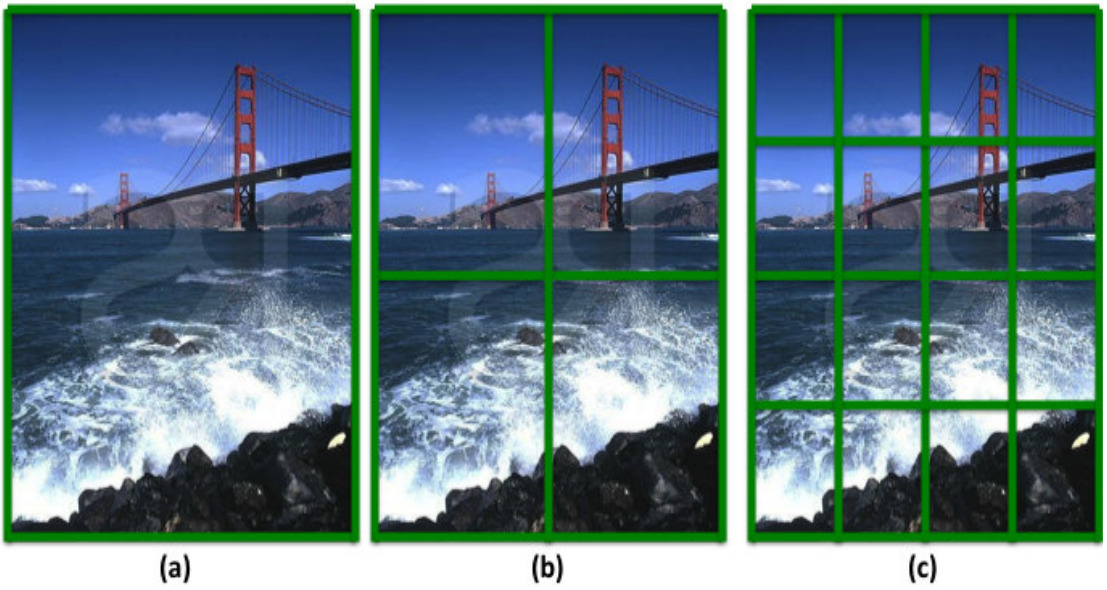


Figure 9: Spatial layout of Spatial Pyramid Matching (SPM) at (a) first level; (b) second level; (c) third level.

rotation of cameras, it has achieved the state of the art performances on several benchmark datasets for visual classification [49, 63].

Lazebnik et al. [49] proposed Spatial Pyramid Matching model (SPM) as feature representation, by partitioning an image into successively smaller sub-regions and calculating a BOW histogram over each sub-region. Then, the SPM model concatenates all BOW histograms together with appropriate weights.

Conventionally, SPM is calculated at three levels, as illustrated in Figure 9. At the first level, the whole image is a sub-region, and BOW is computed for the whole image. At second and third level, the image is divided into 2 x 2, and 4 x 4 sub-regions respectively. A BOW histogram is computed for each sub-region. As compared with the orderless BOW model, the SPM achieved better accuracy in several challenge datasets [30, 31, 39, 52, 99] with slightly increase of computational cost.

Recently, McCann and Lowe [63] propose Spatial Local Coding (SLC), which is also based on absolute spatial information of local features. SLC augments SIFT features with their absolute location (x, y) in the image coordinate system. Then a dictionary is constructed based on these augmented SIFT feature. Local soft assignment and maximum pooling are employed to build the final spatially encoded BOW representation. By augmenting SIFT with location, the location information has influenced both codebook construction and feature coding. During codebook construction, the local features in neighbor regions are more likely grouped together than features that are far away from each other.

Savarese et al. [75] borrowed the idea of color correlograms [43] to develop visual word correlograms, which incorporate relative spatial information. Correlograms capture spatial correlation between all possible pairs of visual words by forming a co-occurrence matrix of visual words as a function of distance. However, the correlograms matrix requires expensive computation cost even after utilizing the integral image techniques [75].

Link analysis models patterns of connections of different images based on extracted local features [127]. Kim *et al*. apply link analysis to learn visual model of object categories [127].

2.2 Multiple-Feature Fusion

Due to the complexity of visual classification, there is not a single feature, which can provide the best tradeoff between the discriminative power and invariance. Different feature types may be complementary to each other for visual classification task. However, it is still an open question how to combine these different feature types to achieve the best tradeoff.

In this section, we introduce two popular approaches in the literatures, i.e., direct concatenation [22, 40, 78, 108] and multiple-kernel learning (MKL) [2, 73, 88, 89] methods.

2.2.1 Direct Concatenation

The straightforward methodology at feature level fusion is the simple concatenation of feature vectors from different modalities to form a large feature vector [22, 40, 78, 108]. Given a set of feature types $\vec{f}_1$, $\vec{f}_2$, … , $\vec{f}_N$, the final concatenated feature $\vec{f}_c$ is

$$\vec{f}_c = [\vec{f}_1, \vec{f}_2, ..., \vec{f}_N] \tag{2-19}$$

The kernel matrix can be constructed by Equation (2-20), assuming Gaussian RBF kernel employed.

$$K(\vec{f}_c^{\,i}, \vec{f}_c^{\,j}) = e^{\frac{-\left\|\vec{f}_c^{\,i} - \vec{f}_c^{\,j}\right\|_2^2}{2\sigma^2}}, \tag{2-20}$$

where $\vec{f}_c^{\,i}$ and $\vec{f}_c^{\,j}$ are concatenated feature vector for i^{th} sample and j^{th} sample. In image classification, a sample is an image. By expanding both $\vec{f}_c^{\,i}$ and $\vec{f}_c^{\,j}$, we have

$$K(\vec{f}_c^{\,i}, \vec{f}_c^{\,j}) = e^{\frac{-\left\|[\vec{f}_1^{\,i}, \vec{f}_2^{\,i}, ..., \vec{f}_N^{\,i}] - [\vec{f}_1^{\,j}, \vec{f}_2^{\,j}, ..., \vec{f}_N^{\,j}]\right\|_2^2}{2\sigma^2}}. \tag{2-21}$$

Note that the square of L2 norm in Equation (2-21) can also be expressed as

$$\left\|[\vec{f}_1^{\,i},\vec{f}_2^{\,i},...,\vec{f}_N^{\,i}]-[\vec{f}_1^{\,j},\vec{f}_2^{\,j},...,\vec{f}_N^{\,j}]\right\|_2^2=\sum_{k=1}^{N}\left\|\vec{f}_k^{\,i}-\vec{f}_k^{\,j}\right\|_2^2 \quad . \tag{2-22}$$

Substituting Equation (2-22) to Equation (2-21), we have

$$K(\vec{f}_c^{\,i},\vec{f}_c^{\,j})=e^{\frac{-\sum_{k=1}^{N}\left\|\vec{f}_k^{\,i}-\vec{f}_k^{\,j}\right\|_2^2}{2\sigma^2}} \quad . \tag{2-23}$$

Reorganizing Equation (2-23), we have

$$K(\vec{f}_c^{\,i},\vec{f}_c^{\,j})=\prod_{k=1}^{N}e^{\frac{-\left\|\vec{f}_k^{\,i}-\vec{f}_k^{\,j}\right\|_2^2}{2\sigma^2}} \quad . \tag{2-24}$$

We can recognize that the product term in Equation (2-24) is the RBF kernel for k^{th} feature type $K(\vec{f}_k^{\,i},\vec{f}_k^{\,j})$. Then the kernel matrix of the concatenated feature vector $K(\vec{f}_c^{\,i},\vec{f}_c^{\,j})$ is the entrywise product of all kernel matrices of single feature types, as shown in Equation (2-25). Note that we have assumed same standard deviation σ used across all feature types in deriving Equation (2-25).

$$K(\vec{f}_c^{\,i},\vec{f}_c^{\,j})=\prod_{k=1}^{N}K(\vec{f}_k^{\,i},\vec{f}_k^{\,j}) \tag{2-25}$$

From Equations (2-24) and (2-25), we can see that all feature types are contributing to the final kernel matrix, i.e., the concatenated feature vector's kernel matrix. If some feature types are less discriminative, it will degrade the final kernel matrix $K(\vec{f}_c^{\,i},\vec{f}_c^{\,j})$, which in turn lower the final classification accuracy.

The relative weights of single feature types, contributing to the final kernel matrix, can be approximated based on Equation (2-23). If we assume

$\left(\vec{f}_{k,m}^{\,i} - \vec{f}_{k,m}^{\,j}\right)^2$ is statistically same for each dimension of every single feature type, the relative weights then is proportional to dimension of single features.

Hence, if one of feature types is noisy, and has very large feature dimension, the noisy feature will dominate the final kernel matrix. Therefore, direct concatenation fusion method requires a careful design for selections of features and parameters, such as feature dimension etc. This is essentially the manual feature selection.

One of applications for multiple features fusion is affect recognition. Human affective state is complicated and sometimes can be very subtle. It may not be detected just from the facial expressions. Fortunately, we observe affective state naturally through multiple modalities, such as facial expression, body gesture, audio signal etc. These observations through different modalities provide complementary information on the affective states e.g., face and body gesture modalities.

One of active research areas on affect recognition is how to combine features from different modalities. A popular approach is to design features based on discriminative power of each modality, and then combine them by direct concatenation [40, 78].

Shan et al. [78] apply the Canonical Correlation Analysis (CCA) to project facial features and body gesture features into a low dimensional space which maximizes their correlation. Then the authors concatenate the projected feature vectors together to train a Support Vector Machine (SVM) classifier for affect recognition. However, it is difficult to extend this method to more than two types of features, or base features, since it needs to find the correlated space between a pair of base features. In practice, it is very likely

to have more than two base features for affect recognition due to the problem complexity.

Gunes and Piccardi [40] select frames, which are the common apex frames from both face and body gesture modalities for affect recognition, and then perform a direct concatenation to combine base features from both modalities. However, the apex frame selection is based on the knowledge of temporal dynamics, which is usually very difficult to predict in advance.

2.2.2 Single-Kernel Learning

In this section, we review single-kernel learning in Support Vector Machine (SVM) [16, 25]. We first derive formula for training and testing with linear kernel and then generalize results to non-linear kernels.

The objective of SVM learning is to find a hyper-plane, which can separate features of two classes with maximum margin and minimum training error.

Figure 10 illustrates the hyper-plane, which is a solid purple line separating features of pentagon class from that of triangle class. The hyper-plane can be expressed as

$$\vec{w} \cdot \vec{x} + b = 0 \quad . \tag{2-26}$$

Two hyper-planes in Equations (2-27) and (2-28) are two support vector hyper-planes of class +1 and class -1 respectively. Two purple dash lines in Figure 10 illustrate the hyper-planes.

$$\vec{w} \cdot \vec{x} + b = +1 \quad . \tag{2-27}$$

$$\vec{w} \cdot \vec{x} + b = -1 \quad . \tag{2-28}$$

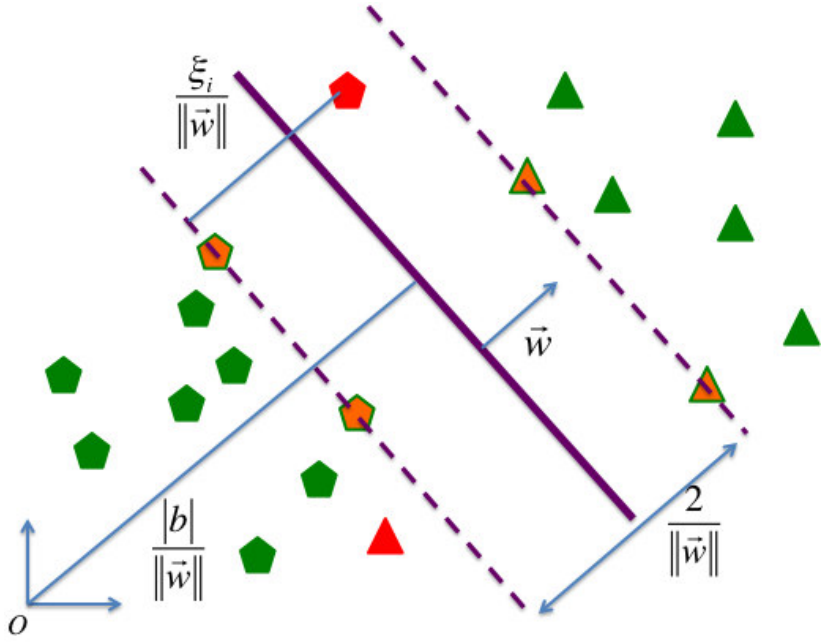


Figure 10: Hyper-plane of linear SVM. Features in orange color are support vectors with zero training error. Features in red color are support vectors with nonzero error.

The distance between these two hyper-planes are called margin, which is $\frac{2}{\|\vec{w}\|}$, as shown in Figure 10. The larger margin represents the better generalization of the learned model. The features with orange color are support vectors, which lie on the two support vector hyper-planes.

Any training features of a class, which lies beyond the class' support vector hyper-plane, toward the other class, introduce training error. The error is proportional to the distance between the feature and its support vector hyper-plane. As an example, the features with red color in Figure 10 are features, which have training error. To quantize training error, we define

$$\vec{w}\cdot\vec{x}_i+b\geq+1-\xi_i \quad \text{for} \quad y_i=+1 \quad , \tag{2-29}$$

$$\vec{w}\cdot\vec{x}_i+b\leq-1+\xi_i \quad \text{for} \quad y_i=-1 \quad , \tag{2-30}$$

$$\xi_i \geq 0 \qquad \forall i \qquad , \qquad (2\text{-}31)$$

where training feature $\vec{x}_i$ has corresponding class label y_i, which is either +1 or -1. The slack variable ξ_i is upper bound of training error for feature $\vec{x}_i$. The upper bound of total training error is then $\sum_{i=1}^{l} \xi_i$, where l is the total number of training features. Hence, the objective function, which maximizes margin and minimizes training error, is to minimize f

$$f = \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{l} \xi_i \qquad , \qquad (2\text{-}32)$$

where C is a constant. Now we have formulated an optimization problem with objective function of Equation (2-32) and constraints of Equations (2-29)-(2-31).

Combining Equation (2-29) and (2-30), we have

$$y_i(\vec{w} \cdot \vec{x}_i + b) - 1 + \xi_i \geq 0 \qquad . \qquad (2\text{-}33)$$

By introducing non-zero Lagrange multiplier ∂_i and μ_i for each training feature to enforce Equations (2-33) and (2-31) respectively, we formulate the primal Lagragian function of Equation (2-32) as

$$L_P = \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \partial_i [y_i(\vec{x}_i \cdot \vec{w} + b) - 1 + \xi_i] - \sum_{i=1}^{l} \mu_i \xi_i \qquad , \qquad (2\text{-}34)$$

where

$$\partial_i \geq 0 \qquad , \qquad (2\text{-}35)$$

$$\mu_i \geq 0 \qquad , \qquad (2\text{-}36)$$

$$\partial_i \left[y_i (\vec{x}_i \cdot \vec{w} + b) - 1 + \xi_i \right] = 0 \quad , \tag{2-37}$$

$$\mu_i \xi_i = 0 \quad . \tag{2-38}$$

Taking derivative of L_P with respect to $\vec{w}$, b and ξ_i, we have

$$\frac{\partial L_p}{\partial \vec{w}} = \vec{w} - \sum_{i=1}^{l} \partial_i y_i \vec{x}_i = 0 \quad , \tag{2-39}$$

$$\frac{\partial L_p}{\partial b} = -\sum_{i=1}^{l} \partial_i y_i = 0 \quad , \tag{2-40}$$

$$\frac{\partial L_p}{\partial \xi_i} = C - \partial_i - \mu_i = 0 \quad . \tag{2-41}$$

Reformulate Equation (2-34) as

$$L_P = \sum_{i=1}^{l} \partial_i + \frac{1}{2} \left\| \vec{w} \right\|^2 - \sum_{i=1}^{l} \partial_i y_i \vec{x}_i \cdot \vec{w} + \sum_{i=1}^{l} \xi_i (C - \partial_i - \mu_i) - b \sum_{i=1}^{l} \partial_i y_i \quad . \tag{2-42}$$

By substituting Equations (2-40) and (2-41), we can remove the last two terms in Equation (2-42).

$$L_P = \sum_{i=1}^{l} \partial_i + \frac{1}{2} \left\| \vec{w} \right\|^2 - \sum_{i=1}^{l} \partial_i y_i \vec{x}_i \cdot \vec{w} \quad . \tag{2-43}$$

Substitute Equation (2-39) to Equation (2-43), we have the dual form of Lagrangian function as shown in Equation (2-44) below.

$$L_D = \sum_{i=1}^{l} \partial_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \partial_i \partial_j y_i y_j \vec{x}_i \cdot \vec{x}_j \quad . \tag{2-44}$$

Now the optimization problem is to maximize Equation (2-44) subject to

$$0 \le \partial_i \le C \quad . \tag{2-45}$$

$$\sum_{i=1}^{l} \partial_i y_i = 0 \quad . \qquad (2\text{-}46)$$

Note that the constraint of Equation (2-45) can be obtained from Equations (2-35), (2-36) and (2-41). The Equation (2-46) is same as Equation (2-40).

We discuss ∂_i value in three different scenarios. The first one is when training feature $\vec{x}_i$does not lie on or beyond its support vector hyper-plane, i.e. $y_i(\vec{x}_i \cdot \vec{w} + b) - 1 > 0$. This scenario corresponds to the green features in Figure 10. There is no training error for the feature, i.e., $\xi_i = 0$. From Equation (2-37), we know $\partial_i = 0$.

The second scenario is when the training feature $\vec{x}_i$lies on its support vector hyper-plane, i.e. $y_i(\vec{x}_i \cdot \vec{w} + b) - 1 = 0$. This corresponds to the orange feature in Figure 10. There is no training error in this scenario, i.e., $\xi_i = 0$. Therefore, $y_i(\vec{x}_i \cdot \vec{w} + b) - 1 + \xi_i = 0$ and ∂_i is undefined, which means some training features on the support vector hyper-plane are non-zeros and some can be 0.

The last scenario is when the training feature $\vec{x}_i$lies beyond its support vector hyper-plane, i.e. $y_i(\vec{x}_i \cdot \vec{w} + b) - 1 < 0$. Hence, there is training error, i.e., $\xi_i > 0$. From Equations (2-38) and (2-41), we know that $\partial_i = C$.

After removing training features with $\partial_i = 0$, Equation (2-39) becomes

$$\vec{w} = \sum_{i=1}^{N_s} \partial_i y_i \vec{x}_i \quad . \qquad (2\text{-}47)$$

where $N_s$is the number of support vector used in training.

By selecting training features with $0 < \partial_i < C$, we know $\xi_i = 0$, and Equation (2-37) becomes $y_i(\vec{x}_i \cdot \vec{w} + b) - 1 = 0$. In other words, training features

with $0<\partial_i<C$ is on their support vector hyper-planes. Therefore, b can be computed by simply selecting one of such training features. Of course, it is numerically wiser to take the average of b over all such training features as shown in Equation (2-48).

$$b=\frac{1}{\left|\{i\,|\,0<\partial_i<C\}\right|}\sum_{\{i|0<\partial_i<C\}} y_i-\vec{x}_i\cdot\vec{w} \tag{2-48}$$

Substituting Equation (2-47) to Equation (2-48), we have

$$b=\frac{1}{\left|\{i\,|\,0<\partial_i<C\}\right|}\sum_{\{i|0<\partial_i<C\}}\left[y_i-\sum_{j=1}^{Ns}\partial_j y_j\vec{x}_i\cdot\vec{x}_j\right] \quad . \tag{2-49}$$

During testing phase, the predicted class label score y_t of the testing feature $\vec{x}_t$ is

$$y_t=\vec{w}\cdot\vec{x}_t+b=\sum_{i=1}^{N_s}\partial_i y_i\vec{x}_i\cdot\vec{x}_t+b \quad . \tag{2-50}$$

Then the predicted class label is sign(y_t).

To generalize SVM training and testing to other kernels, including non-linear kernels, we replace the linear kernel term $\vec{x}_i\cdot\vec{x}_j$ with $K(\vec{x}_i,\vec{x}_j)$. Therefore kernel training of Equations (2-44)-(2-46) becomes to maximize

$$L_D=\sum_{i=1}^{l}\partial_i-\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\partial_i\partial_j y_i y_j K(\vec{x}_i,\vec{x}_j) \quad . \tag{2-51}$$

subject to

$$0\le\partial_i\le C \quad , \tag{2-52}$$

$$\sum_{i=1}^{l}\partial_i y_i=0 \quad . \tag{2-53}$$

The testing phase of Equation (2-50) becomes

$$y_t = \sum_{i=1}^{N_s} \partial_i y_i K(\vec{x}_i, \vec{x}_t) + b \quad , \qquad (2\text{-}54)$$

where

$$b = \frac{1}{\left|\{i \mid 0 < \partial_i < C\}\right|} \sum_{\{i \mid 0 < \partial_i < C\}} \left[y_i - \sum_{j=1}^{Ns} \partial_j y_j K(\vec{x}_i, \vec{x}_j) \right] \quad . \qquad (2\text{-}55)$$

In literature, kernel function $K(\vec{x}_i, \vec{x}_j)$ is usually considered as linear kernel of transformed features of $\vec{x}_i$and $\vec{x}_j$, i.e., $\Phi(\vec{x}_i) \cdot \Phi(\vec{x}_j)$. The transformed feature $\Phi(\vec{x}_i)$can be in infinite dimension space.

2.2.3 Multiple-Kernel Learning (MKL)

Given a set of base features and their associated base kernels K_k, we want to find the optimal kernel combination $K_{opt} = \sum_k d_k K_k$, where d_k is the weight for the k^{th} base feature. The kernel combination K_{opt}approximates the best trade-off between the discriminative power and the invariance for a specific application.

Equations (2-56) to (2-58) show the objective cost function *f* and the constraints for the multiple-kernel learning (MKL) proposed in [88].

$$\underset{\mathbf{w},\, \xi_i, d_k}{Min} \quad f = \frac{1}{2}\left\|\vec{w}\right\|^2 + C\sum_i \xi_i + \sum_k \sigma_k d_k \quad , \qquad (2\text{-}56)$$

subject to

$$y_i(\vec{w} \cdot \Phi(\vec{x}_i) + b) - 1 + \xi_i \geq 0 \quad , \qquad (2\text{-}57)$$

$$\xi_i \geq 0 \ \ \forall i \ ; \ d_k \geq 0 \ \ \forall k \ ; \ \ A\vec{d} \geq \vec{p} \quad . \qquad (2\text{-}58)$$

where $\Phi(\vec{x}_i)$ is the combined features corresponding to K_{opt} in a high dimensional space for sample x_i, which is shown in Equation (2-59). Equivalently, K_{opt} can be expressed in Equation (2-60), where $\Phi_k(\vec{x}_i)\cdot\Phi_k(\vec{x}_j)$ forms k^{th} base kernel K_k.

$$K_{opt}(\vec{x}_i,\vec{x}_j)=\Phi(\vec{x}_i)\cdot\Phi(\vec{x}_j) \tag{2-59}$$

$$K_{opt}(\vec{x}_i,\vec{x}_j)=\sum_k d_k\Phi_k(\vec{x}_i)\cdot\Phi_k(\vec{x}_j) \tag{2-60}$$

The optimization can be carried out in a SVM framework subject to additional regularization term of weight d_k in the objective function.

In order to handle large-scale problems involving many base kernels, the minimax optimization strategy [20, 73, 88] is used with two iteration steps. In the first step, feature weight d_k is fixed, i.e., $K_{opt}=\sum_k d_k K_k$ is fixed. Then the optimization problem of Equation (2-56) can be solved by any standard SVM solver using its dual form of Equation (2-61), since the term $\sum_k \sigma_k d_k$ is simply a constant.

$$\underset{\partial_i}{Max}\ \mathrm{L}_D=\sum_i \partial_i-\frac{1}{2}\sum_{i,j}\partial_i\partial_j y_i y_j K_{opt}(\vec{x}_i,\vec{x}_j)+\sum_k \sigma_k d_k \quad , \tag{2-61}$$

subject to

$$0\le\partial_i\le C;\ \sum_i \partial_i y_i=0 \quad . \tag{2-62}$$

In the second iteration step with the fixed ∂_i, projected gradient descent is employed to find updated feature weights d_k as shown in Equations (2-63) and (2-64).

$$\frac{\partial L_D}{\partial d_k} = \sigma_k - \frac{1}{2}\sum_{i,j} \partial_i \partial_j y_i y_j K_k(\vec{x}_i, \vec{x}_j) \tag{2-63}$$

$$d_k^{new} = d_k^{old} - \frac{\partial L_D}{\partial d_k} \tag{2-64}$$

These two iteration steps are repeated until converge or the maximum number of iterations is reached. The final weights of base features can be determined.

Then we train SVM classifiers using the optimal combined kernel according the final weights of base features. A new sample x_t is assigned the class labels with the sign of Equation (2-65).

$$y_t = \sum_{i=1}^{Ns} \partial_i y_i \sum_k d_k K_k(x_i, x_t) + b \quad . \tag{2-65}$$

The multi-class problem can be solved by one-vs.-one or one-vs.-all strategy similar to SVM.

MKL method provides an elegant framework to fuse many base features by assigning larger feature weight to the most discriminative base feature. Compared to the direct concatenation method, MKL can avoid the contamination from less discriminative base features, especially when those features have large dimensions.

However, MKL method tends to select very few base features from the feature pool. It often only selects one or two base features, which are discriminative at a particular high dimensional space *H*. Moreover, the kernel parameter, e.g., σ in Gaussian RBF kernel, associated with the optimal high dimensional space *H* may be significantly different for the base features from different modalities. Therefore, traditional MKL cannot utilize

the maximum discriminative power of complementary base features at the same time.

2.3 Large-scale Learning

As the continuously increasing scale in both examples and classes of available datasets [28, 31, 39, 99], research work on visual classification has to take into account of many practical constraints, e.g., computation and memory costs, for both training and testing.

ImageNet [28] is currently one of the largest public available datasets for image classification task. It has 14 million images with more than 21,000 image categories. With a such large dataset, both computation and memory cost become crucial in algorithm design. To train 10,000 classifiers with linear SVM using one-vs-all framework, it takes one CPU year [28]. For nearest neighbor classifier, a brute force approach can also take one year to go through 4.5 million testing images [28]. With 1000 codebook size, it requires 18 GB memory capacities to simply store the Bag of Words histograms for 4.5 million training images.

Most learning-based algorithms, e.g., one-vs.-all linear SVM [19, 25], at best linearly increase with the number of image classes [14, 49, 56, 100]. During testing, it becomes computationally infeasible when scaling up to large numbers of classes.

On the other hand, non-parametric nearest neighbor (NN) based classifiers require no training phase and can naturally handle large numbers of classes. However, they have to retain all the training examples in testing phase, which becomes infeasible on large-scale datasets due to the expensive computation cost and memory usage. For example, the total memory

required to store dense SIFT features [58] for the SUN dataset [99] is around 100 GB, which far exceeds the memory of a desktop (typically 4G – 48G). Compared to learning-based classifiers, accuracies of NN-based classifiers are normally much lower, which also limits their applications for visual classification.

In this section, we will discuss some research effort to improve efficiency of discriminative learning algorithms for large-scale visual classification problem. We will also present related work on nearest neighbor (NN) based methods, and discuss limitations, which prevent NN-based methods from large-scale visual classification application. Some research work on combining both learning based and NN-based classifiers are also presented.

2.3.1 Discriminative Learning

To adapt linear SVM for large-scale visual classification, most research work are built on hierarchical decision tree structure [34, 38, 59]. Figure 11 shows the comparison between the conventional one-vs.-all classification scheme and hierarchical tree structure classification scheme.

As shown in Figure 11(a), one-vs.-all classification framework learns a model for each category. During testing phase, a test image is predicted by the models of all classes, with each of them assign a prediction score. Then the final predicted label is simply the class label, which has the maximum prediction score. Hence, one-vs.-all framework has the complexity of $O(N)$, where N is the total number of classes.

On the other hand, hierarchical structure classifiers partition class labels space hierarchically based on affinity between class labels. As

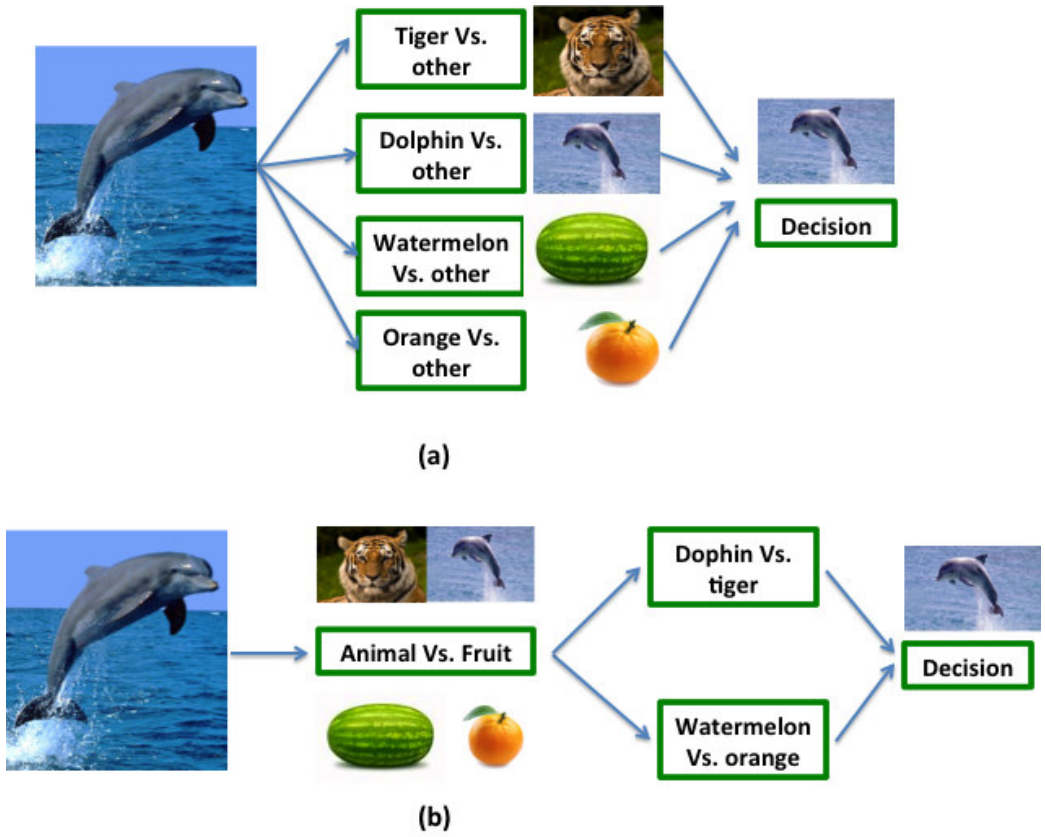


Figure 11: Illustrate the difference between one-vs.-all SVM and hierarchical SVM classification schemes; (a) One-vs.-all linear SVM classification scheme; (b) Hierarchical SVM classification scheme [113, 114, 115, 116, 117].

illustrated in Figure 11(b), tiger is more similar to dolphin than watermelon or orange. Hence at the first level of the hierarchy, tiger and dolphin belong to one group, i.e., animal group, while watermelon and orange belong to another group, i.e., fruit group. At the following levels, each group of class labels is further divided into subgroups recursively, until there is only one class label in each subgroup.

A model is learned for each partition in the hierarchical tree structure. During testing phase, a new image is classified at each level to either go to right branch or left branch of the hierarchical tree, starting from the top level. The final prediction result can be obtained at the leaf node of the

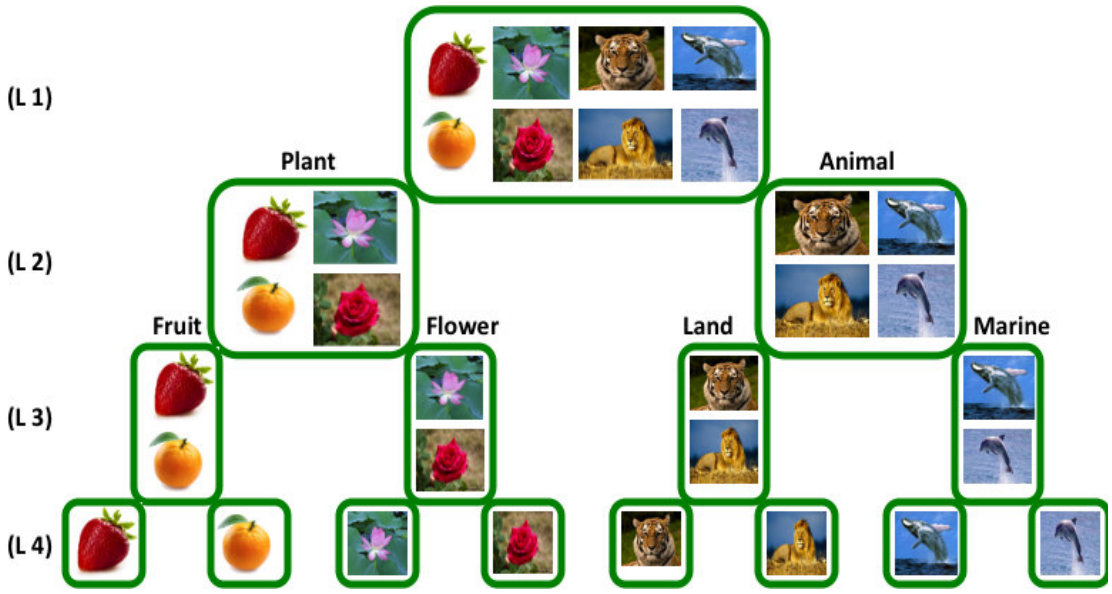


Figure 12: Illustrate class taxonomy with four levels [114, 115, 117, 118, 119, 120, 121, 122].

hierarchical structure. As illustrated in Figure 11(b), a new image is first predicted as either animal or fruit at the first level. After it is predicted as animal, then the image is predicted as either tiger or dolphin at the second level. The final predicted class label is obtained at this level. Due to the tree structure, the computation cost of the hierarchy based classifiers only increase logarithmically with the total number of classes.

Much research effort on hierarchical SVM based framework is on how to partition class label space [7, 21, 91, 98, 104, 109], and how to handle confused classes [34, 38, 59].

Griffin et al. [38] propose class taxonomies to represent the hierarchical partitions of class label space, as illustrated in Figure 12. The first level is super-group, which contains all class labels. At the second level, the class labels are partitioned as two groups, i.e., plant and animal. At the third level, plant group is further divided into fruit and flower group. Animal group is also divided into land animal and marine animal groups. Finally the

fourth level contains leaves of taxonomies, at which one group only has one class label.

The taxonomies are learned based on affinity between a pair of classes or a pair of class groups. The affinity is measured based on confusion matrix, which can be constructed using cross validation scheme during training phase.

Two different methods are employed to generate class taxonomies automatically. The first method splits the confusion matrix into two groups recursively using Self-Tuning Spectral Clustering [105] until only one category in all sub-groups. This is essentially a top-down approach.

The second method is a bottom-up approach. Initially, every class is an individual group. Two groups with maximum mutual confusion are joined into one group, while the confusion matrix is updated by averaging their rows and columns. The process is repeated until one group contains all class labels.

Similar to other hierarchical classifier framework, Griffin et al. [38] train a classifier for every class label partition in the taxonomies. In testing phase, they propose a termination node, at which the test image will stop going down the hierarchical taxonomies, and perform one-vs.-all classification for the group of class labels at the termination node.

Taking Figure 12 as an example, the classification framework is identical to one-vs.-all classification if we place the termination node at the first level. If the termination node is at the second level, and the test image is classified as plant group at the first level, then all four classes in the animal group are eliminated for further consideration. The test image will stop at the second level, and one-vs.-all classifier is employed to determine, to which class in the four plant classes the test image belong. The termination node at

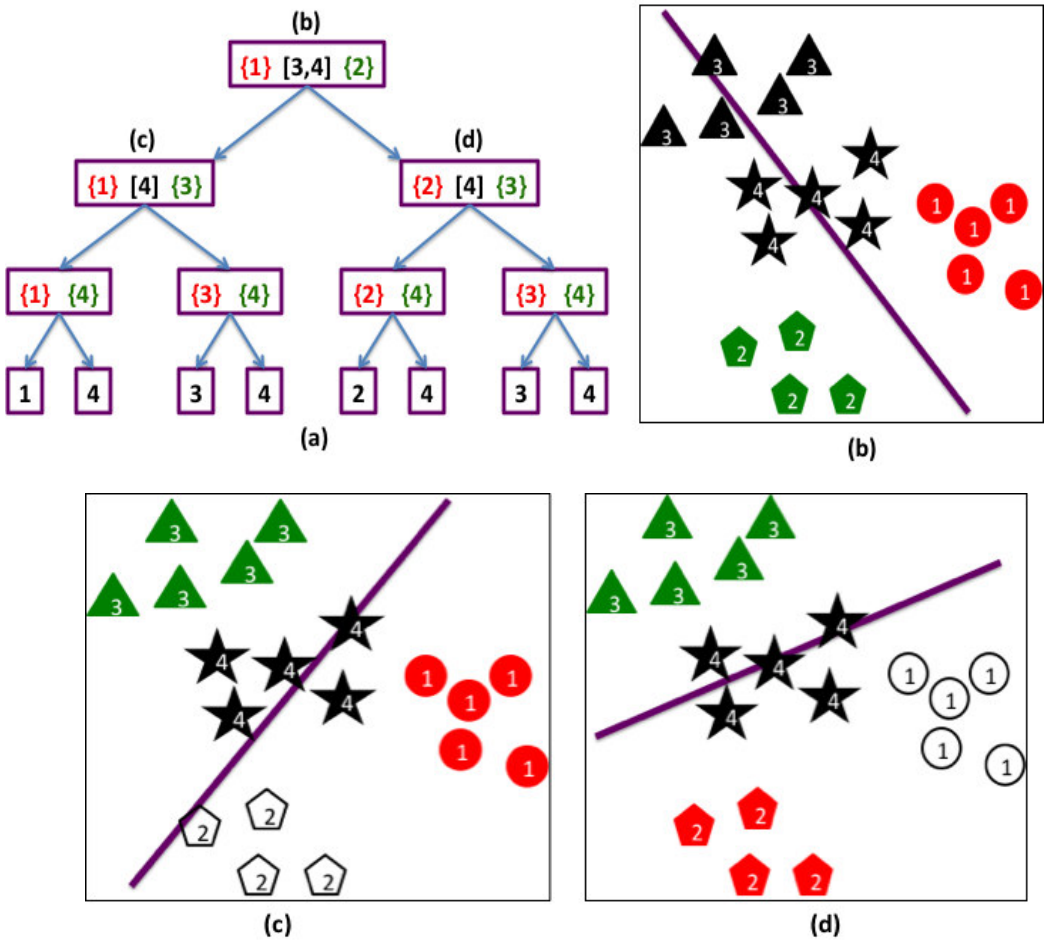


Figure 13: Illustrate relaxed hierarchy for four class data. (a) Relaxed hierarchy with red colored class labels as "+1" class, and green colored class labels as "-1" class. The black colored class labels in bracket are in ignored group. (b), (c) and (d) illustrate the coloring of class labels and the associated decision boundary for first level (top level) and second level respectively. Note that red color indicates "+1" class, green color indicates "-1" class, and black color indicates ignored class labels. Unfilled features in (c) and (d) are not considered during coloring process.

the last level results in a classifier, which is same as the conventional hierarchical based classifier presented before.

In order to achieve a better tradeoff between accuracy and efficiency, several papers [34, 59] utilize the relaxed hierarchy, which postpones the decision for some confusing classes in the hierarchical decision structure.

Figure 13(a) illustrates the concept of relaxed hierarchy. At each level, class labels are divided into three groups. The first group of class labels is colored as red, which is "+1" class. The second group is colored as green, which is "-1" class. The third group includes all confused class labels, which are colored as black. The ignored class labels are not participating in computing decision boundary at the node.

Figure 13(b) illustrates the class label coloring with decision boundary obtained by data from "+1" and "-1" groups at the first level of the hierarchy in Figure 13(a). In Figure 13(b), class label 1, which features are marked as red, belong to "+1" group, and class label 2, which features are marked as green, belong to "-1" group. The class labels 3 and 4 belong to the ignored group, and their features are marked as black. Similarly Figure 13(c) and Figure 13(d) shows coloring and associated decision boundary at the left node and right node of the second level in the hierarchy. Note that the class labels, which features are not marked with any color, participate in neither coloring process nor decision boundary calculation.

From intuition, the less ignored class labels is at a node of the hierarchy, the better efficiency of the hierarchical decision structure is. In other words, if there are more classes participating in computing decision boundary at a node, the more class labels can be eliminated at next level of the hierarchy.

The authors [34] formulate this tuition into a principle optimization problem. At each node, given l training samples ($\vec{x}_i$, y_i), where $y_i \in \Upsilon = \{1,2,...,N\}$, class labels Υ are partitioned into a positive subset S_y^+, a negative subset S_y^- and an ignored subset S_y^0 . A particular partition or coloring gives a different binary classification problem, where positive

samples are $S_x^+ = \{\vec{x}_i \mid y_i \in S_y^+\}$, and negative samples are $S_x^- = \{\vec{x}_i \mid y_i \in S_y^-\}$. Note that the samples from ignored subset S_y^0 are not participating in the binary classification problem. An additional coloring variable for each class label is also introduced, i.e., $\mu_n \in \{-1, 0, +1\}$, where *n* is from 1 to *N*, indicating which subset the n^{th} class label should belong to. Then, the optimization problem at each node becomes to minimize

$$f = \frac{1}{2}\|\vec{w}\|_2^2 + C\sum_{i=1}^{l}|\mu_{yi}|\xi_i - A\sum_{i=1}^{l}|\mu_{yi}| \quad . \tag{2-66}$$

subject to

$$\mu_{yi} \in \{-1, 0, +1\} \quad . \tag{2-67}$$

$$\mu_{yi}(\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i \quad . \tag{2-68}$$

$$\xi_i \geq 0 \quad . \tag{2-69}$$

$$-B \leq \sum_{n=1}^{N}\mu_n \leq B \quad . \tag{2-70}$$

$$\left|\{n \mid \mu_n > 0\}\right| \geq 1 \quad \text{and} \quad \left|\{n \mid \mu_n < 0\}\right| \geq 1 \quad . \tag{2-71}$$

The first two terms in objective function (Equation (2-66)) is to minimize training error and maximize SVM classifier margin. They are same as standard SVM objective function if only training samples from positive subset, i.e., "+1" group, and negative subset, i.e., "-1" group, are considered. The last term in objective function is to encourage all training samples participating to calculate binary decision boundary at a given node. That is to encourage small size in ignored subset S_y^0. The constraints of

Equations (2-70) and (2-71) ensure data balance of positive and negative group for the binary classification problem at the node.

To solve the optimization problem of Equations (2-66)-(2-71), the authors [34] propose a two-step alternating method, i.e., (1) fixing μ_n, and optimizing $\vec{w}, b$; (2) fixing $\vec{w}, b$, therefore ξ_i, optimize μ_n.

With fixing μ_n, the optimization problem reduces to standard SVM problem with positive samples from S_x^+, and negative samples from S_x^-. We can then find $\vec{w}, b$ with standard SVM solver [19].

With fixing $\vec{w}, b$, therefore ξ_i, the objective function of Equation (2-66) becomes

$$f = \frac{1}{2}\|\vec{w}\|_2^2 + C\sum_{i=1}^{l}[\delta(\mu_{yi}-1)\xi_i^+ + \delta(\mu_{yi}+1)\xi_i^-] - A\sum_{i=1}^{l}|\mu_{yi}| \quad , \qquad (2\text{-}72)$$

where $\delta(t-m)$ is delta function, which is 1 only when $t=m$. Otherwise, the delta function has value of 0. Since $\|\vec{w}\|_2^2$ is fixed at this step, hence we drop the first term of Equation (2-72). By recognizing μ_{yi} is same for samples of a given class label *n*, we reformulate Equation (2-72) over class labels.

$$f = C\sum_{n=1}^{N}\sum_{i=\{i\,|\,yi=n\}}[\delta(\mu_n-1)\xi_i^+ + \delta(\mu_n+1)\xi_i^-] - A\sum_{n=1}^{N}\sum_{i=\{i\,|\,yi=n\}}|\mu_n| \qquad (2\text{-}73)$$

Reorganizing Equation (2-73), we have

$$f = C\sum_{n=1}^{N}\sum_{i=\{i\,|\,yi=n\}}[\delta(\mu_n-1)\xi_i^+ + \delta(\mu_n+1)\xi_i^- - \frac{A}{C}|\mu_n|] \quad . \qquad (2\text{-}74)$$

Since C is a constant, it can be dropped out from the objective function, and let f_n represent cost function of a class label *n*, i.e.,

$$f_n = \sum_{i=\{i \mid yi=n\}} [\delta(\mu_n - 1)\xi_i^+ + \delta(\mu_n + 1)\xi_i^- - \frac{A}{C}|\mu_n|] \quad . \quad (2\text{-}75)$$

Equation (2-74) becomes

$$f = \sum_{n=1}^{N} f_n \quad . \quad (2\text{-}76)$$

Therefore, with fixing $\vec{w}, b$, the optimization is to minimize Equation (2-76), subject to

$$\mu_n \in \{-1, 0, +1\} \quad . \quad (2\text{-}77)$$

$$-B \le \sum_{n=1}^{N} \mu_n \le B \quad . \quad (2\text{-}78)$$

$$\left|\{n \mid \mu_n > 0\}\right| \ge 1 \quad \text{and} \quad \left|\{n \mid \mu_n < 0\}\right| \ge 1 \quad . \quad (2\text{-}79)$$

Note that

$$\xi_i^+ = \max\{0, 1 - \vec{w} \cdot \vec{x}_i - b\} \quad , \quad (2\text{-}80)$$

$$\xi_i^- = \max\{0, 1 + \vec{w} \cdot \vec{x}_i + b\} \quad . \quad (2\text{-}81)$$

Since μ_n can only take three discrete values as shown in Equation (2-77), a single class label's cost function f_n, which depends on μ_n, can only take three values too. To minimize Equation (2-76) is equivalent to select minimum f_n over three possible μ_n values for each class label. Let's say $\hat{\mu}_n$ gives minimum f_n, then $\sum_{n=1}^{N} \hat{\mu}_n$ need to satisfy the constraint of Equation (2-78).

If $\sum_{n=1}^{N} \hat{\mu}_n$ satisfy Equation (2-78), then $\hat{\mu}_n$ is the solution. If $\sum_{n=1}^{N} \hat{\mu}_n > B$, then we know there are more class labels in the positive subset,

i.e., S_y^+. Therefore, some class labels' $\hat{\mu}_n$ should be decreased. This can be done by calculating the delta increase of f_n, i.e., Δf_n, per unit decrease in μ_n for each class label. Then sort the Δf_n, and select minimum Δf_n, and decrease its corresponding μ_n. The process is repeated until $\sum_{n=1}^{N} \hat{\mu}_n$ satisfy Equation (2-78). If $\sum_{n=1}^{N} \hat{\mu}_n < -B$, same approach is adapted except to increase μ_n.

Hierarchical based SVM classifiers improve classification efficiency because of the hierarchical tree structure. However, the price is compromising classification accuracy to some extent.

Deep learning recently has successfully applied on large-scale image classification. Krizhevsky et. al. [48] has trained a deep convolutional neural network (CNN) to classify a million of images on a subset of ImageNet. It outperforms the state of the art performance by a significant percentage.

The CNN they applied has 8 layers, i.e. 5 convolutional layers and 3 fully connected layers with 60 million parameters and 650,000 neurons. Given such large network, the over fitting becomes a major issue. To avoid the over fitting problem, the authors propose several techniques including data augmentation and dropout etc. [48].

Dean et. al. [128] propose to replace the dot-product kernel operator with locality-sensitive hashing in order to accelerate the convolution with millions of filters. The technique achieves 20,000 times faster, and is used to classify 100,000 object classes.

2.3.2 Nearest Neighbor based Classifier

Nearest neighbor based classifiers can naturally handle large number of categories. However, their classification accuracy is usually much lower

than learning based classifier [62, 99]. Even though NN-based classifier is well known for its efficiency on training, i.e., no training required, its testing speed is extremely slow [62]. NN-based classifier usually needs to store all training data in test phase. Hence, memory usage can become prohibitively large when applying on large-scale datasets.

A popular approach to apply nearest neighbor on image classification is to form an image representation by Bag of Words (BOW) model for both training and testing images. The distances between a test image and each of training images, i.e., image-to-image distances, are computed based on the Euclidean distance of their BOW vectors. The predicted class label of the test image is assigned with the class label of the training image, which has the smallest image-to-image distance.

Boiman et al. [10] argue that two practices have severely degraded the performance of nearest neighbor classifier, i.e., (1) vector quantization step used in forming the BOW image representation, and (2) classification based on image-to-image distance.

BOW enjoys a compact representation of an image based a codebook, which size is usually small, e.g., a few hundred to a thousand visual words. Nevertheless, the compactness comes with the price of discriminative power loss for individual descriptors during vector quantization process. Without training phase, nearest neighbor based classifier cannot compensate for such loss as in learning-based classifiers. Hence, NN-based classifiers yield inferior performance.

The use of image-to-image distance cannot generalize well to dataset with large intra-class variation, especially when the training sample size in each class is small. On the other hand, image-to-class distance describes the

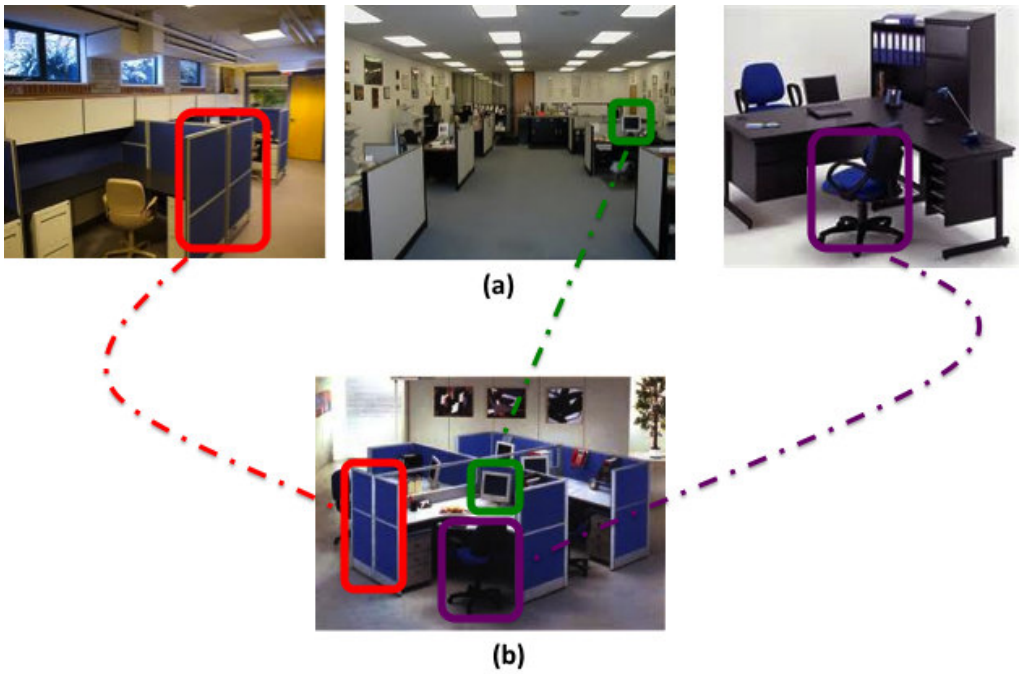


Figure 14: Illustrate the concept of image-to-class distance by searching similar patches of a query image over all training images in class "office"; (a) Training images of class "office"; (b) query image [123,124,125,126].

overall similarity between a query image and all training images in a class. It is expected to generalize better during testing.

Figure 14(b) shows a query image, and Figure 14(a) shows three training images of class "office". The query image is significantly different from any one of the three training images. Nevertheless, its patches, which are highlighted in different colors, can still find similar ones in different training images of class "office". To compute image-to-class distance between the query image and the class "office", we can simply accumulate these patch similarity. Since the search of similar patches is not limited to a single training image, the image-to-class distance generalize better in testing phase, especially with limited training samples in each class.

To avoid vector quantization and utilize image-to-class distance, Boiman et al. [10] propose Naïve-Bayes Nearest-Neighbor (NBNN)

classifier. The NBNN achieves comparable performance with the learning based classifiers. In the following, we will provide the overview of NBNN derivation.

Given a query image Q, the predicted class label $\hat{C}$ is

$$\hat{C} = \underset{C}{\operatorname{argmax}} \quad p(C \mid Q) \quad . \tag{2-82}$$

By Bayes's theorem,

$$p(C \mid Q) = \frac{p(Q \mid C)p(C)}{p(Q)} \quad . \tag{2-83}$$

Equation (2-82) becomes

$$\hat{C} = \underset{C}{\operatorname{argmax}} \quad \frac{p(Q \mid C)p(C)}{p(Q)} \quad . \tag{2-84}$$

Since $p(Q)$ is constant over all class label C, and we assume uniform prior, i.e., $p(C)$ is a constant, Equation (2-84) becomes

$$\hat{C} = \underset{C}{\operatorname{argmax}} \quad p(Q \mid C) \quad . \tag{2-85}$$

We further assume independence of descriptors $\vec{d}_i$, i.e.,

$$p(Q \mid C) = \prod_{i=1}^{N} p(\vec{d}_i \mid C) \quad , \tag{2-86}$$

where N is the total number of descriptors in the query image Q. Substituting Equation (2-86) into Equation (2-85), and take the logarithmic probability, we have

$$\hat{C} = \underset{C}{\operatorname{argmax}} \quad \log\left(\prod_{i=1}^{N} p(\vec{d}_i \mid C)\right) \quad . \tag{2-87}$$

Therefore,

$$\hat{C} = \underset{C}{\operatorname{argmax}} \quad \sum_{i=1}^{N} \log\left(p(\vec{d}_i \mid C)\right) \quad . \tag{2-88}$$

Next, approximating $p(\vec{d}_i \mid C)$ with Parzen window estimator [72, 74], gives

$$p(\vec{d}_i \mid C) = \frac{1}{L}\sum_{j=1}^{L} K(\vec{d}_i - \vec{d}_j^{\,C}) \quad , \tag{2-89}$$

where $\vec{d}_j^{\,C}$ is a training descriptor from class *C*. NBNN further assumes that the summation term in Equation (2-89) can be approximated by kernel value of $K(\vec{d}_i - NN_C(\vec{d}_i))$, where $NN_C(\vec{d}_i)$ is the nearest neighbor training descriptor in class *C*. Therefore,

$$p(\vec{d}_i \mid C) = \frac{1}{L} K(\vec{d}_i - NN_C(\vec{d}_i)) \quad , \tag{2-90}$$

By choosing Gaussian RBF kernel, we substitute Equation (2-90) to Equation (2-88).

$$\hat{C} = \underset{C}{\operatorname{argmax}} \sum_{i=1}^{N} \log\left(\frac{1}{L} e^{-\frac{\left\| \vec{d}_i - NN_C(\vec{d}_i) \right\|_2^2}{2\sigma^2}} \right) \quad . \tag{2-91}$$

Reorganizing Equation (2-91) gives

$$\hat{C} = \underset{C}{\operatorname{argmax}} \quad \sum_{i=1}^{N} -\log(L) - \frac{\left\| \vec{d}_i - NN_C(\vec{d}_i) \right\|_2^2}{2\sigma^2} \quad , \tag{2-92}$$

where σ is a kernel bandwidth, which is positive number.

Assuming L and σ are equal over classes, i.e., balance training data with equal kernel bandwidth for all class labels, we can have

$$\hat{C} = \underset{C}{\operatorname{argmax}} \quad \sum_{i=1}^{N} -\left\| \vec{d}_i - NN_C(\vec{d}_i) \right\|_2^2 \quad , \tag{2-93}$$

Equivalently,

$$\hat{C} = \underset{C}{\arg\min} \quad \sum_{i=1}^{N} \left\| \vec{d}_i - NN_C(\vec{d}_i) \right\|_2^2 \quad , \tag{2-94}$$

Equation (2-94) is the classification rule of NBNN classifier. For each descriptor in a query image, we find a nearest neighbor descriptor over all

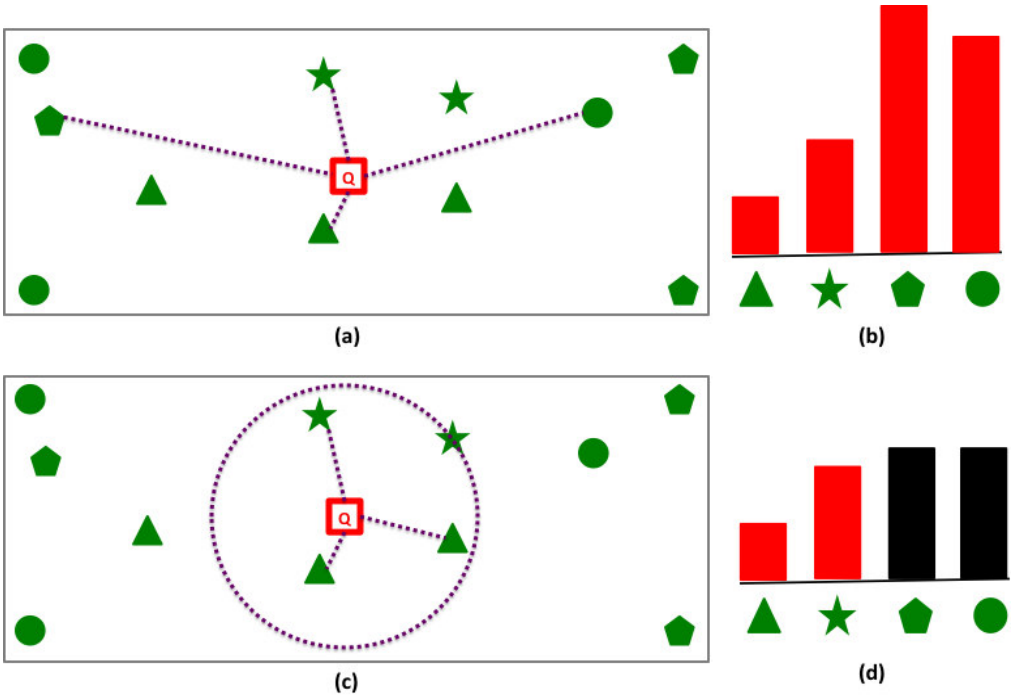


Figure 15: Illustrate the difference between NBNN and local NBNN. (a) NBNN calculate query feature (red square) to class distance for each of the four classes; (b) Bar representation of query feature to class distance over all four classes for NBNN method; (c) local NBNN calculates query feature to class distance only for classes within the circle; (d) Bar representation of query feature to class distance over all four classes. The black bars indicate background distance.

training images in each class. Then we compute the Euclidean distance between the query descriptor and nearest neighbor descriptor of each class. We call the distance as feature-to-class distance. These feature-to-class distances are accumulated over class labels for all query descriptors in the query image. Finally, the query image is assigned with the class label, which has the minimum accumulated feature-to-class distance, i.e., minimum image-to-class distance.

However, similar to other NN-based classifiers, NBNN requires expensive classification cost in both computation and memory. Even after

using some approximate nearest neighbor techniques [1, 67]. The computation cost of NBNN is still linearly proportional to the number of classes, since each query descriptor needs to compute the feature-to-class distance for all classes.

To reduce computational cost, McCann and Lowe [62] propose local NBNN, in which feature-to-class distance are accumulated only over the classes, which are presented in a local neighborhood of a query descriptor. They argue that the feature-to-class distance cannot approximate the likelihood of query descriptor $p(\vec{d}_i \mid C)$ well, if it gets too large.

Figure 15 illustrates the difference between NBNN and local NBNN method. Figure 15(a) illustrates the calculation of query feature to class distance for all the four classes, i.e., triangle, star, pentagon and circle. Note that query feature in Figure 15 is shown as a red square. Figure 15(b) is the bar representation of query feature to class distance over all four classes.

Figure 15(c) illustrates the local NBNN algorithm. Local NBNN finds $k+1$ nearest neighbors features, which are illustrated using the circle, i.e., the star feature on the circle is $(k+1)^{th}$ nearest neighbor. We then calculate query feature to class distance only for classes within the circle, i.e., the classes, in which some features are k nearest neighbor of the query feature; Note that radius of the circle is used as background distance, i.e., the distance between query feature and $(k+1)^{th}$ nearest neighbor (the star feature on the circle). Figure 15(d) shows the bar representation of query feature to class distance for local NBNN. For classes, in which no feature is the k nearest neighbor of query feature, i.e., not within the circle, background distance is assumed. They are indicated by black bars.

Local NBNN improves the computation cost of NBNN by only updating the classes found in a local neighborhood. Hence, the complexity only grows logarithmically with the number of categories. Nevertheless, the computation cost can still be expensive when the training data is large. For instance, we observe that it takes more than 100 seconds for local NBNN to classify one image on the Caltech 256 dataset using dense SIFT features.

Some research efforts have been made to combine learning-based and NN-based methods. Tuytellaars et al. [85] observe the complementarity between NBNN and Bag-of-Words (BOW). They propose to kernelize NBNN and combine with BOW kernel, using a discriminative learning framework, e.g. Multiple Kernel Learning (MKL) [35, 88]. In order to reduce the computation cost, they simply down-sample query features in a testing image, which adversely affects classification results. SVM-KNN [107] hybrids k nearest neighbors (K-NN) method with learning based classifier, i.e., SVM, to improve accuracy of the K-NN method. Despite these efforts [5, 10, 85, 88, 107] made to improve the performance of NN-based classifiers, very few work, if any, has tried to extend the NN-based classifiers to large-scale visual classification, to improve accuracy as well as reduce computation and memory costs simultaneously.

Another related work is vocabulary tree [69], which constructs hierarchical k-mean tree by recursively dividing training data into k groups. k is the branch factor of the vocabulary tree. The leaf nodes in the tree then form a large dictionary as the codebook. The vector quantization has the computation cost of $O(log(n))$ instead of $O(n)$ as in the conventional vector quantization method, where n is the codebook size. A very efficient hierarchical TF-IDF (Term Frequency and Inverse Document Frequency)

scoring scheme is proposed based on large codebook, which the hierarchical k-mean tree provides.

Chapter 3

Spatially Encoded EigenMap Representation

3.1 Summary

We propose a novel approach to describe and recognize visual categories. Inspired by the success of Bag of Words approach, we represent an object using a collection of EigenMaps, which incorporate both appearance and spatial information.

Each EigenMap captures the location likelihood of a visual word through the kernel density estimation method. By collecting EigenMaps of all visual words, our approach can effectively integrate both local features and their global correspondences.

Experimental results on scene datasets demonstrate significant performance improvement as compared with the standard Bag of Words approach and the Latent Dirichelet Allocation model, which also utilizes a codebook of visual words, over several feature types including both region features and interest point features.

3.2 Method

3.2.1 Overview

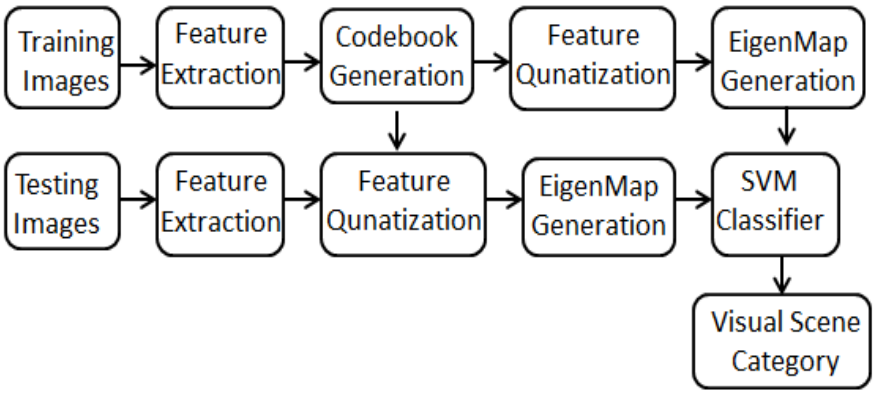


Figure 16: Flowchart of the proposed approach using EigenMap representation in visual scene classification.

The flowchart in Figure 16 illustrates our overall approach in visual scene classification using EigenMap representation.

We first extract features from images. A codebook is generated from the training image features by using an unsupervised clustering algorithm such as the K-Means method. The center feature vectors in the codebook are called visual words. Then each feature in both training and testing images is vector-quantized to one of the visual words in the codebook.

We then construct location map for each visual word in a scene image using the kernel density estimation method [8]. The EigenMap of a visual word is then generated by projecting the location map to the principal component space. The concatenation of every visual word's EigenMap in the scene image forms an input feature vector of a SVM (Support Vector Machine) classifier [19, 25]. Finally we can classify an unknown scene image to different scene categories. The proposed EigenMap representation of a scene image not only incorporates spatial information in the appearance features, but also effectively integrates both local features and their global interactions.

3.2.2 Feature Extraction

In order to verify the effectiveness of the proposed model, we extract five types of different features, which include both region features and interest point features. In other words, for each type of features, we evaluate the performance improvement of the EigenMap model. In our experiments, we extract three types of region features and two types of interest point features.

(A) Region Features: Texture, Shape, and Color

Three types of region features are extracted in our experiments: texture, shape, and color. Before generating any region features, we first perform segmentation on images using the algorithm proposed by Felzenszwalb and Huttenlocher [32]. As shown in Figure 17, connected pixels with same color are used to represent one segmented region. At each segmented region, the above three types of region features are extracted.

Texture features are generated by passing the original image with S filter bank [87]. S filter bank is rotationally invariant with 13 isotropic. There are 13 responses for each image. The means and standard deviations of each response are calculated for individual segmented region in the image. In other words, each segmented region has 13 means and standard deviations of the filter responses. These means and standard deviations are combined together as texture features of one segmented region.

Figure 17: Segmentation example of a Polo scene. Connected pixels with same color belong to the same segment.

A simple type of shape features is extracted in this experiment following the approach in [52]. The size of each segmented region is found by calculating the maximum length of a segmented region in x and y directions. Then, the shape feature of each segmented region is formed by combining the size and the number of pixels in each segmented region.

Color features are formed by calculating color histograms of each segmented region over the RGB color space. Each color space is divided into 10 bins. Therefore, the color feature vector of each segmented region has 1000 dimensions.

(B) Interest Point Features: Uniform Grids and Harris Corners

In addition to the region features above, two types of interest point features are evaluated in our experiments: the uniform grids and the Harris corners. In our evaluations, the uniform grid method is used to sample interest points every 10 pixels in x and y directions. The number of interest points generated for a typical image (resolution of 300 by 500) is around 1500.

Unlike the uniform grid method, the Harris corners utilize gradient information to detect more stable interest points in an image [42]. The average number of the Harris corners in one image is approximately 100 in our experiments, which is significantly less than the number of the uniform grid interest points.

The Scale Invariant Feature Transform (SIFT) descriptor [58] is used to describe all interest points regardless of their detection methods. A square patch window with each interest point at its center is extracted. The patch window size is 24 by 24 pixels. 4 by 4 center points are uniformly sampled from the patch window. For each center point, an 8-Bin orientation histograms of gradients within the patch window is constructed. The gradient magnitudes are further weighted by a Gaussian function with the mean corresponding to the center point. Then all histograms of the 16 center points are concatenated together to form an interest point descriptor, which has 128 dimensions.

3.2.3 Codebook Formation and Feature Quantization

After extracting feature vectors from the training images, the K-Means clustering algorithm is used to group the feature vectors together based on the Euclidean distance. As a result, the center feature vectors in all clusters are called visual words. The resulting visual words form the codebook vocabulary [26]. The codebook sizes of the texture, shape and color features are 120, 100 and 30 respectively. Both the uniform grid and the Harris corner features have the codebook size of 150.

The features in each image are then vector-quantized to one of visual words in the codebook. The vector quantization process of a feature is to

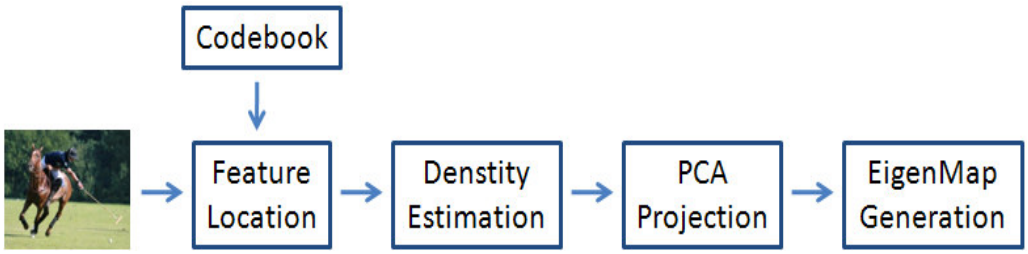


Figure 18: The flowchart of EigenMap Generation for each visual word from the codebook.

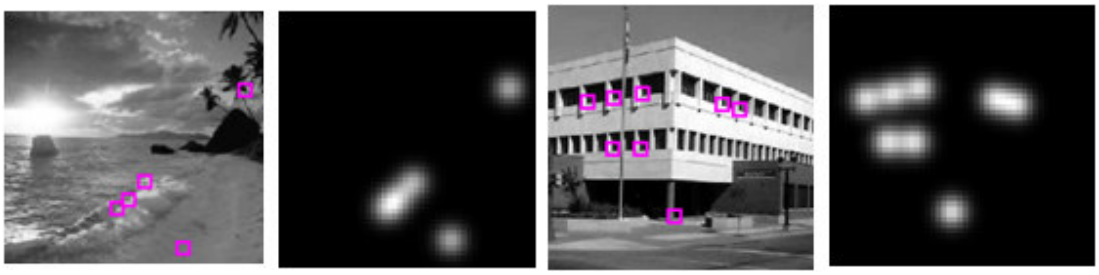
find a visual word in the codebook with the smallest Euclidean distance. Then the feature is represented by the closest visual word in the codebook.

3.2.4 EigenMap Generation

In order to effectively incorporate spatial information into these visual words and describe their global correspondence within a scene image, we generate an EigenMap for each visual word in the scene image. The flowchart of EigenMap generation for each visual word is shown in Figure 18.

Given an input image and a codebook of visual words generated from the K-Means clustering algorithm as described in the last section, we first locate a visual word V_i in the input image and mark the corresponding positions at the visual word V_i's location map. The location map has fixed size of 50 by 50 pixels. Then we use kernel density estimation [8] to model the location likelihood of the visual word V_i in its location map, as illustrated in the examples shown in Figure 19. The kernel we used is the normal distribution with the standard deviation of 2.

The next step is to project the constructed location map to a lower dimensional space using the principal component analysis, as shown in Equation (3-1).



(a) coast (b) inside city

(c) living room

Figure 19: A visual word Vi's locations in an input image and its location likelihood on the corresponding location map using the kernel density estimation analysis.

$$\vec{s} = \Phi^T * (\vec{m} - \bar{m}) \quad , \qquad (3\text{-}1)$$

where $\vec{s}$ is the location map projected in the eigen-space, $\vec{m}$ is the location map, and $\bar{m}$ is the average map of all training location map $\vec{m}$. Φ is a matrix, in which each column vector is an eigenvector of the location maps' covariance matrix, obtained from the training images, in the order of descending eigenvalues of the covariance matrix. Finally, the EigenMap $\vec{\eta}$ is constructed as the concatenation of $\vec{s}$ and μ, as shown in Equation (3-2).

$$\vec{\eta} = [\vec{s}, \mu] \quad , \qquad (3\text{-}2)$$

where $\mu$is the mean value of a visual word's location map $\vec{m}$. Typical dimension of EigenMap $\vec{\eta}$ is below 8, which results a very compact representation of a scene image, as compared with previous work [49, 75].

Figure 20: Sample images of the 8 scene categories from the UIUC Sport Scene Database.

After each visual word's EigenMap $\vec{\eta}$ of a scene image is constructed, we then concatenate the EigenMaps of all visual words together to represent the scene image. This concatenated feature vector is also an input to the SVM classifier.

3.2.5 Classifier

We employ the SVM with the RBF kernel as our multi-class classifier with one vs. one framework. The SVM is to find a set of hyper-planes, which separate each pair classes of data with the maximum margin. That is to assign a scene category to an unknown image based on the collections of visual words' EigenMaps.

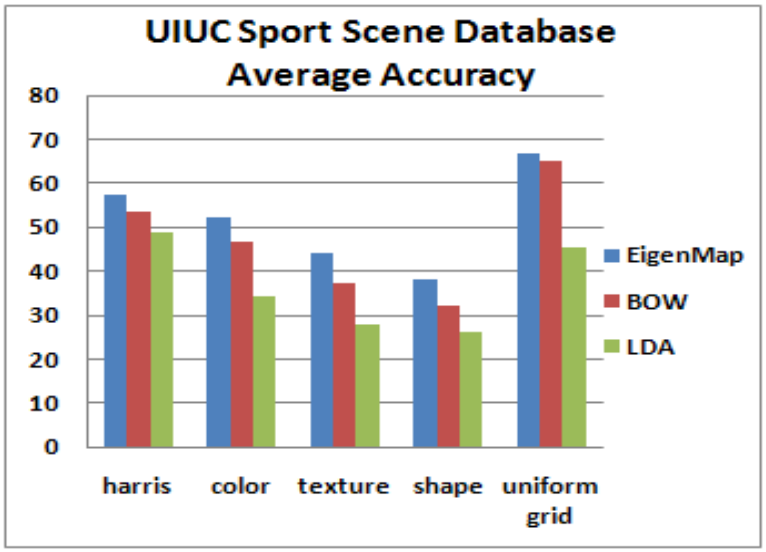
3.3 Experiments

Figure 21: sample images of the Natural Scene database, which has 13 categories.

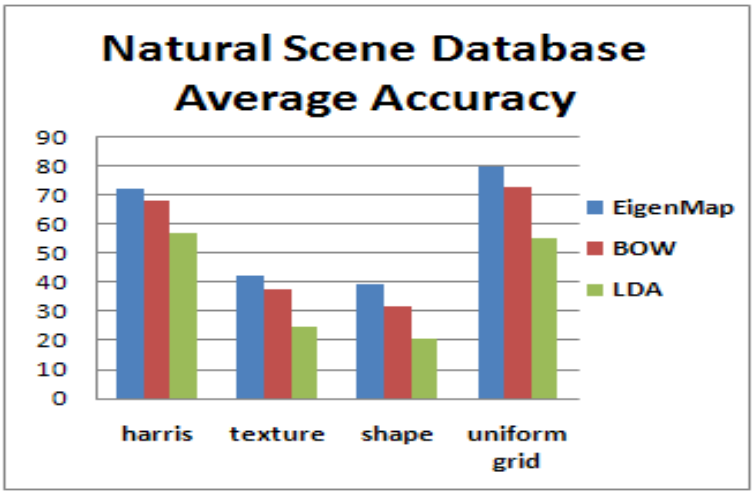
3.3.1 Databases

Experiments are performed over two databases: the UIUC Sport Scene database [52] and the Natural Scene database [30].

The UIUC Sport Scene database is a very challenging visual scene database with significant intra-class variations in the background, scale and lighting etc. As shown in Figure 20, the badminton scene can happen on the badminton court or at the backyard of a house. The lighting and scale can



(a)



(b)

Figure 22: Comparing to the Bag of Words (BOW) model and the Latent Dirichlet Allocation (LDA) model on (a) the UIUC Sport Scene database; and (b) the Natural Scene database.

also be very different. The database consists of 8 categories of sport scenes with 500 images in each category.

Natural Scene database consists of 13 categories, with 210 scene images in each category, as shown in Figure 21. Most of them are gray images. Therefore, we cannot evaluate the color feature on this dataset.

3.3.2 Experimental Setups

We divide the dataset of each category into five subsets. Then the images of one subset are used as testing set, while the images from the remaining four subsets are used as training set. The process is repeated five times with each of the five subsets used as the testing data once. All experimental results reported in the dissertation are the average accuracy of the five repeated testing.

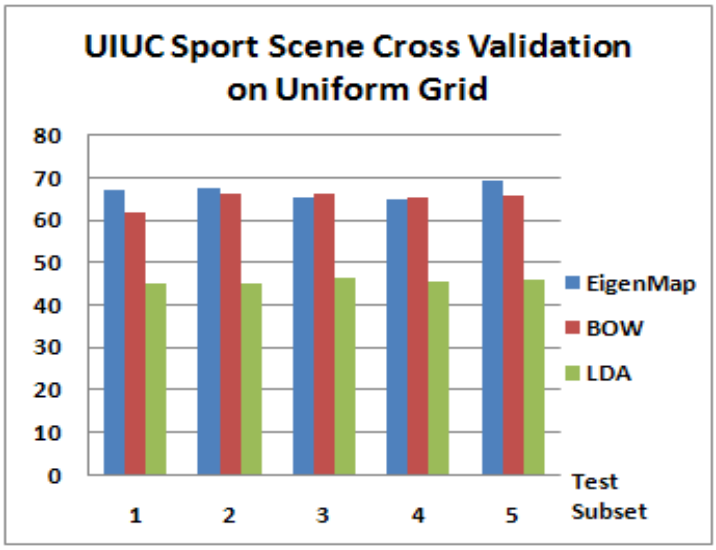
3.3.3 Experimental Results

(A) Compare to the Bag of Words Model and the LDA model

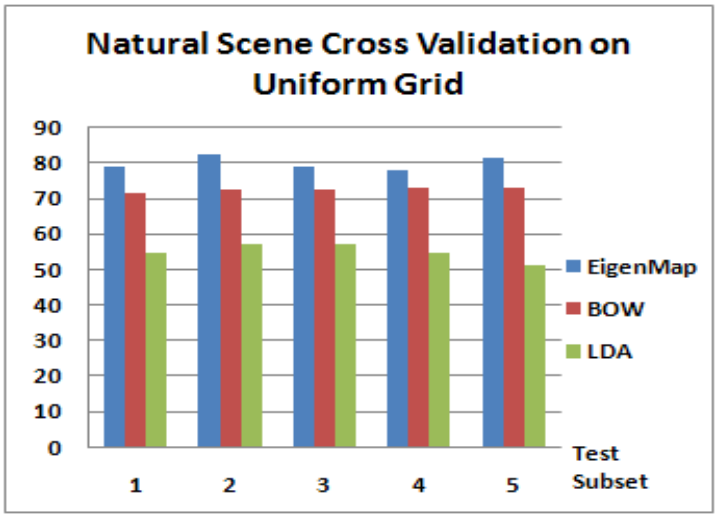
For each feature type, i.e., texture, shape, color, the Harris corner with the SIFT descriptor and the uniform grid interest point with the SIFT descriptor, we compare the proposed EigenMap approach with the Bag of Words model (BOW) [26] and the Topic Discovery model [52]. More specifically, we employ the Latent Dirichlet Allocation (LDA) [9] similar to the approach proposed by Li et al. [52]. They are all running under the same experimental setup.

The detailed comparisons over different feature types are shown in Figure 22(a) and Figure 22(b) for the UIUC Sport Scene database and the Natural Scene database respectively. On the UIUC Sport Scene database, the EigenMap model outperforms the standard Bag of Words model by the average of 4.8% over all different feature types. It also outperforms the LDA model by the average of 15.3%.

We observe larger performance improvement over the other two models on the Natural Scene database. The proposed approach improves



(a)



(b)

Figure 23: Comparing the BOW and the LDA models using the five-fold cross validation results of the uniform grid interest point features on (a) the UIUC Sport Scene database; and (b) the Natural Scene database.

classification accuracy by the average of 6% and 19% as compared with the BOW and the LDA models respectively.

The consistent performance improvement over every feature type verifies the effectiveness of the proposed model in the visual scene classification. The spatial correspondences among local features, which the

badminton	**79**	3	5	1	3	5	2	2	79%
bocce	2	**48**	11	9	7	6	5	12	48%
croquet	6	6	**65**	7	10	2	3	1	65%
polo	2	10	2	**75**	4	5	1	1	75%
rockclimbing	2	5	2	3	**70**	3	7	8	70%
rowing	2	7	2	2	6	**65**	8	8	65%
sailing	4	3	0	0	0	20	**67**	6	67%
snowboarding	2	18	1	0	14	3	8	**54**	54%

(a) EigenMap on Sport Scene

badminton	**79**	1	5	0	5	4	3	3	79%
bocce	4	**50**	11	7	6	5	8	9	50%
croquet	6	3	**69**	11	9	1	0	1	69%
polo	1	7	8	**66**	4	7	3	4	66%
rockclimbing	1	8	8	6	**70**	2	2	3	70%
rowing	6	10	3	3	2	**59**	10	7	59%
sailing	5	11	1	1	1	12	**62**	7	62%
snowboarding	5	21	2	4	13	1	6	**48**	48%

(b) BOW on Sport Scene

bedroom	**25**	0	6	7	0	0	0	3	0	0	0	0	1	60%
suburb	0	**39**	0	1	0	0	0	2	0	0	0	0	0	93%
kitchen	4	0	**26**	6	0	0	0	0	0	0	0	4	2	62%
living room	9	0	7	**22**	0	0	0	1	2	0	0	0	1	52%
coast	0	0	0	0	**36**	0	1	0	2	3	0	0	0	86%
forest	0	0	0	0	0	**41**	0	0	1	0	0	0	0	98%
highway	1	0	0	0	5	0	**34**	0	1	0	1	0	0	81%
inside city	1	0	2	0	1	1	0	**30**	1	0	1	5	0	71%
mountain	1	0	0	0	1	1	1	0	**36**	1	1	0	0	86%
open country	0	1	0	0	4	4	1	0	2	**30**	0	0	0	71%
street	0	0	0	1	0	0	1	1	0	0	**39**	0	0	93%
tall building	0	0	0	0	0	0	0	1	0	0	1	**40**	0	95%
office	4	0	3	2	0	0	0	0	0	0	0	0	**33**	79%

(c) EigenMap on Natural Scene

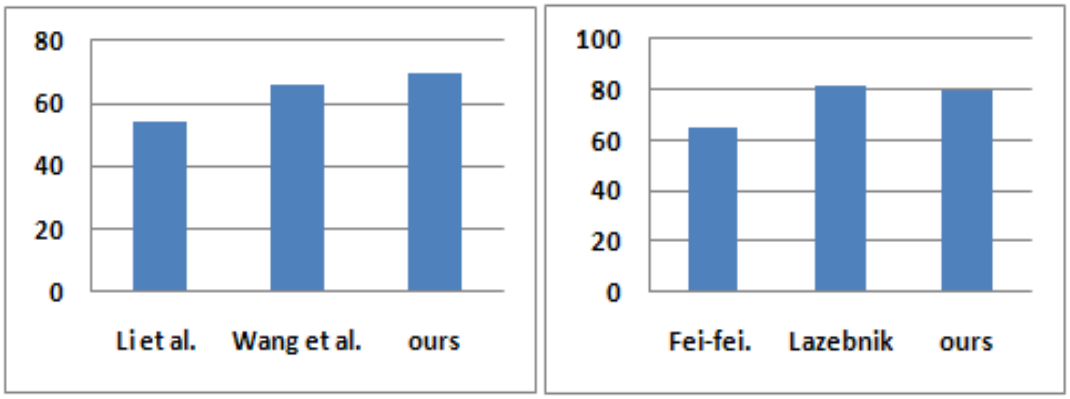
bedroom	**20**	0	10	8	0	0	0	1	0	0	0	0	3	48%
suburb	0	**38**	0	1	0	0	1	2	0	0	0	0	0	90%
kitchen	2	0	**26**	7	0	0	0	2	0	0	0	1	4	62%
living room	13	2	4	**16**	0	0	1	4	0	0	1	0	1	38%
coast	0	2	0	0	**32**	0	3	0	1	4	0	0	0	76%
forest	0	0	0	0	0	**41**	0	0	0	1	0	0	0	98%
highway	1	1	0	0	2	0	**31**	1	2	3	1	0	0	74%
inside city	0	1	4	0	1	1	0	**32**	0	1	1	1	0	76%
mountain	0	1	0	0	1	2	1	0	**34**	2	1	0	0	81%
open country	0	2	0	0	6	4	0	0	1	**29**	0	0	0	69%
street	1	1	0	1	0	0	0	3	0	0	**35**	1	0	83%
tall building	0	0	0	3	1	0	0	1	0	0	1	**36**	0	86%
office	3	0	6	7	0	0	0	0	0	0	0	0	**26**	62%

(d) BOW on Natural Scene

Figure 24: Sample confusion matrices of the EigenMap and the BOW models over both the UIUC Sport Scene and the Nature Scene database;

EigenMap model captures, contribute to the performance improvements. Figure 23 shows the detailed cross validation results as compared with the BOW and the LDA model.

The sample confusion matrices of the uniform grid interest point feature are also shown in Figure 24 for both the UIUC Sport Scene database



(a) (b)

Figure 25: Compare with the state of the art reported by Fei-Fei and Perona [30], Lazebnik et al. [49], Li et al. [52], and Wang et al. [93] on both (a) UIUC Sport Scene database and (b) Natural Scene database.

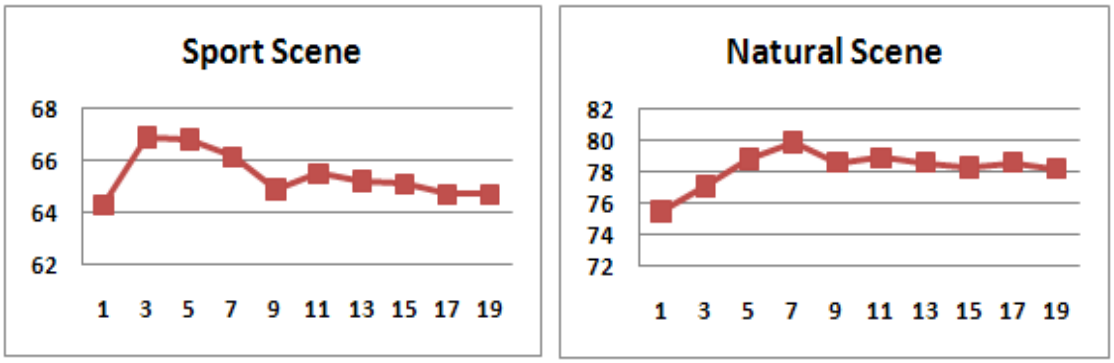
and the Natural Scene database. The true positive rate for each category is shown in the last column next to the corresponding confusion matrix.

All the four confusion matrices are generated using the uniform grid interest point with the SIFT descriptor, where the rows are the ground truth while the columns are the classified categories. As we can see from the confusion matrices of the EigenMap and the BOW, the EigenMap approach achieves higher performance on most of the scene categories.

In the UIUC Sport Scene dataset, the most confusion occurs between the "Rowing" and the "Sailing" categories since both sport scenes are very similar in the background, which contains water in the scene images. In the Natural Scene dataset, the most confusion occurs between the bedroom and the living room scene images.

(B) Compare to the State-of-the-art Performance

Figure 25(a) and 25(b) show the detailed comparison with the state-of-the-art performance on the UIUC Sport scene dataset [52, 93] and the



(a) (b)

Figure 26: The effect of number of eigenvectors used in the PCA projection on the classification performance over (a) the UIUC Sport Scene database; (b) the Natural Scene database; Note that we used the uniform grid interest point with the SIFT descriptor for both databases.

Natural Scene dataset [30, 49] respectively. The results are directly cited from their papers. From Figure 25, the proposed EigenMap model achieves a state-of-the-art performance on both the UIUC Sport Scene database and the Natural Scene database.

(C) Select Number of Principal Components for EigenMap

We also evaluate the effect of number of eigenvectors used in the construction of the EigenMap on the classification performance. As we can see from Figure 26, the number of eigenvectors used in the PCA projection achieves the best performance when it is around 5. As the number of eigenvectors continues increasing, the performance degrades slightly. That suggests that we only need a very small dimensional space to represent each visual word's EigenMap.

3.4 Discussion

We have proposed a novel EigenMap representation of a scene image, which can not only incorporates the spatial information with the appearance features, but also integrates both local features and their global correspondences effectively. The EigenMap model has been evaluated on two public databases for scene image classification and outperforms both the standard Bag of Words model and the LDA model. The proposed model also achieves a state-of-the-art performance on both datasets with small feature dimension.

Chapter 4

Margin-Constrained Multiple Kernel Learning

4.1 Summary

Recent advances in multiple-kernel learning (MKL) show the effectiveness to fuse multiple base features in object detection and recognition. However, MKL tends to select only the most discriminative base features but ignore other less discriminative base features which may provide complementary information. Moreover, MKL usually employ Gaussian RBF kernels to transform each base feature to its high dimensional space. Generally, base features from different modalities require different kernel parameters for obtaining the optimal performance. Therefore, MKL may fail to utilize the maximum discriminative power of all base features from multiple modalities at the same time. In order to address these issues, we propose margin-constrained multiple-kernel learning (MCMKL) method by extending MKL with margin constraints and applying dimensionally normalized RBF (DNRBF) kernels for application of multi-modal feature fusion. The proposed MCMKL method learns weights of different base features according to their discriminative power. Unlike the conventional MKL, MCMKL incorporates less discriminative base features by assigning smaller weights when constructing the optimal combined kernel, so that we

can fully take the advantages of the complementary features from different modalities. We validate the proposed MCMKL method for affect recognition from face and body gesture modalities on the FABO dataset. Our extensive experiments demonstrate favorable results as compared to the existing work, and MKL-based approach.

4.2 Method

4.2.1 Multiple-Kernel Learning (MKL)

Multiple-kernel learning (MKL) is to find the optimal combination of multiple base kernels K_k, i.e., $K_{opt} = \sum_k d_k K_k$, where d_k is the weight for the k^{th} base kernel. Its objective function is shown in Equations (2-56) to (2-58) in section 2.2.3. We copy the equations here for the convenience.

$$\underset{\mathbf{w},\, \xi_i, d_k}{Min} \quad f = \frac{1}{2}\|\vec{w}\|^2 + C\sum_i \xi_i + \sum_k \sigma_k d_k \quad , \qquad (4\text{-}1)$$

subject to

$$y_i(\vec{w}\cdot\Phi(\vec{x}_i)+b)-1+\xi_i \geq 0 \quad , \qquad (4\text{-}2)$$

$$\xi_i \geq 0 \;\; \forall i \;;\; d_k \geq 0 \;\; \forall k \;;\; A\vec{d} \geq \vec{p} \quad , \qquad (4\text{-}3)$$

where $\Phi(\vec{x}_i)$ satisfies

$$K_{opt}(\vec{x}_i, \vec{x}_j) = \Phi(\vec{x}_i)\cdot\Phi(\vec{x}_j) \quad . \qquad (4\text{-}4)$$

The dual form of Equation (4-1) is

$$\underset{\partial_i}{Max}\ \mathrm{L}_D = \sum_i \partial_i - \frac{1}{2}\sum_{i,j}\partial_i \partial_j y_i y_j K_{opt}(\vec{x}_i, \vec{x}_j) + \sum_k \sigma_k d_k \quad , \qquad (4\text{-}5)$$

subject to

$$0 \le \partial_i \le C;\ \sum_i \partial_i y_i = 0 \quad . \qquad (4\text{-}6)$$

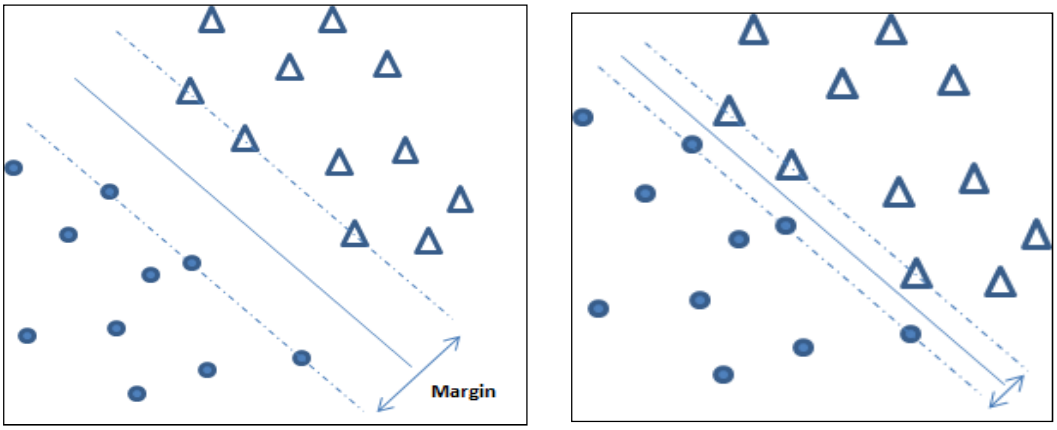
The optimization is carried out by two iteration steps: (1) fixing feature weight d_k, then solving Equation (4-5) with standard SVM solver; (2) fixing ∂_i, then updating feature weights d_k with projected gradient descent as shown in Equations (4-7) and (4-8).

$$\frac{\partial L_D}{\partial d_k} = \sigma_k - \frac{1}{2}\sum_{i,j}\partial_i \partial_j y_i y_j K_k(\vec{x}_i, \vec{x}_j) \qquad (4\text{-}7)$$

$$d_k^{new} = d_k^{old} - \frac{\partial L_D}{\partial d_k} \qquad (4\text{-}8)$$

4.2.2 Margin Constraints

To address these issues, we propose a Margin-Constrained Multiple Kernel Learning (MCMKL) method. This is motivated by the observations that base feature which is more discriminative usually finds a hyper-plane with larger margin to separate support vectors of opposite classes during training of SVM machines. A hyper-plane of base feature “a” in Figure 27(a) has a larger margin than that of base feature “b” in Figure 27(b) to separate the class of solid dot from the class of triangle. This suggests that the base feature “a” is more discriminative than the base feature “b” for the classification of the solid dot and the triangle class.



(a) (b)

Figure 27: (a) The hyper-plane of base feature "a" has a large separation margin to separate solid dot class and triangle class; (b) The hyper-plane of base feature "b" has a small margin to separate solid dot class and triangle class.

Therefore, the separation margin for each base feature in its high dimensional space provides a rough measurement on the base feature's discriminative power. Nevertheless, these rough measurements can effectively guide MKL when searching for the optimal feature combination. The separation margin for each base feature can be calculated using Equation (9) as the inversed square root of its own objective cost function.

$$m_k = \frac{2}{\|\vec{w}_k\|} \approx \frac{\sqrt{2}}{\sqrt{f_k}} = \frac{\sqrt{2}}{\sqrt{\frac{1}{2}\|\vec{w}_k\|^2 + C\sum_i \xi_i + \sigma_k d_k}} \quad . \qquad (4\text{-}9)$$

After obtaining the separation margin m_k for each base feature, we select one of base features as the reference base feature, which has the feature weight of d_s and the margin m_s. The weight d_k of k^{th} base feature is constrained in the range, which has the lower bound of LB_k and the upper bound of UB_k according to the margin ratio between m_s and m_k during training. LB_k and UB_k

can be calculated as in Equation (4-11). The additional weight constraints in Equation (4-10) are enforced during the multiple-kernel learning.

$$LB_k \le d_k \le UB_k \qquad \forall k \qquad . \tag{4-10}$$

$$LB_k = \left(\frac{m_k}{m_s}\right)^n * d_s \quad ; \quad UB_k = \left(\frac{m_k}{m_s}\right)^n * d_s * (1+\delta) \qquad . \tag{4-11}$$

where n is a parameter that controls the margin sensitivity on the feature weight ratio between d_k and d_s. As n increases, the values of LB_k and UB_k become more sensitive to the ratio of m_k and m_s. δ is a constant to control the range width of the feature weight d_k. In our experiments, we set n to 1.5 and δ to 1.

4.2.3 Dimensionally Normalized Kernel

Gaussian RBF kernel is one of the most popular non-linear kernels due to its excellent performance in numerous applications. It is defined in Equation (4-12).

$$K(\vec{x}_i, \vec{x}_j) = \exp(-\gamma \sum_{q=1}^{D} (x_{i,q} - x_{j,q})^2) \qquad . \tag{4-12}$$

where $\vec{x}_i$ and $\vec{x}_j$ are the i^{th} sample and the j^{th} sample along with $x_{i,q}$ and $x_{j,q}$ as the q^{th} element in a feature vector. D is the sample's feature dimension.

γ is the RBF kernel parameter, which determines the mapping from a low dimensional feature space L to a high dimensional space H.

Assuming that $(x_{i,q} - x_{j,q})^2$ is statistically same, the kernel value decreases when the feature dimension increases at a fixed γ as shown in Equation (4-12). Hence, Equation (4-12) suggests the inverse relationship between the

optimal γ and the feature dimension. This intuition is confirmed in our experiments.

In MKL fusion, base features from different modalities may have significantly different feature dimensions, which will result very different optimal γ values for each base feature. Therefore, MKL cannot utilize the maximum discriminative power of all base features from different modalities at the same time.

We can treat γ as a feature selection parameter in MKL, which select only few base features at a time. This intuition also explains the observations reported in [88] that MKL tends to select only very few most discriminative base features. Therefore, MKL cannot take the full advantages of all types of features from multiple modalities.

Based on these observations, we propose a dimensionally normalized RBF kernel (DNRBF), which is defined in Equation (4-13).

$$\mathbf{K}(x_i, x_j) = \exp(-\frac{\gamma}{D}\sum_{q=1}^{D}(x_{i,q} - x_{j,q})^2) \tag{4-13}$$

This normalization step is essential to eliminate the effect of feature dimension on γ selection, so that all base features have a similar optimal γ. Therefore, MCMKL can utilize the maximum discriminative power of all base features from multiple modalities.

4.3 Multi-Modal Fusion for Affect Recognition

Affect recognition from multiple modalities is a challenging problem. Our study focuses on fusion of features from visual modalities, i.e., face and body gesture modality.

Different from conventional approaches to fuse features from multiple modalities, which simply concatenate all feature vectors from different sources together and feed the concatenated feature vector into a classifier, such as SVM, we apply a margin-constrained multiple-kernel learning (MCMKL) method to fuse features from both face and body gesture modalities. MCMKL can effectively combine all types of features for affect recognition by assigning an appropriate feature weight to each type of features and calculate the optimal kernel for affect recognition.

4.3.1 Overview of MCMKL-based Affect Recognition

Figure 28 shows an overview of our affect recognition system, which consists of five major parts, i.e., facial feature extraction, body gesture feature extraction, expression temporal segmentation, temporal normalization, and MCMKL-based classification.

Two types of facial features, i.e., Image-HOG and MHI-HOG [23] are extracted in our experiments. Here, HOG stands for Histogram of Gradients [27], and MHI stands for Motion History Image [9, 83]. Image-HOG features capture facial appearance changes, while MHI-HOG features represent facial motion information.

Four types of gesture features are extracted, which include location features, motion area features, Image-HOG features, and MHI-HOG features around both hands.

Each expression in video sequences can be first temporally segmented into onset, apex, offset and neutral phases [23]. Then, we perform a temporal normalization procedure to handle different temporal resolutions of

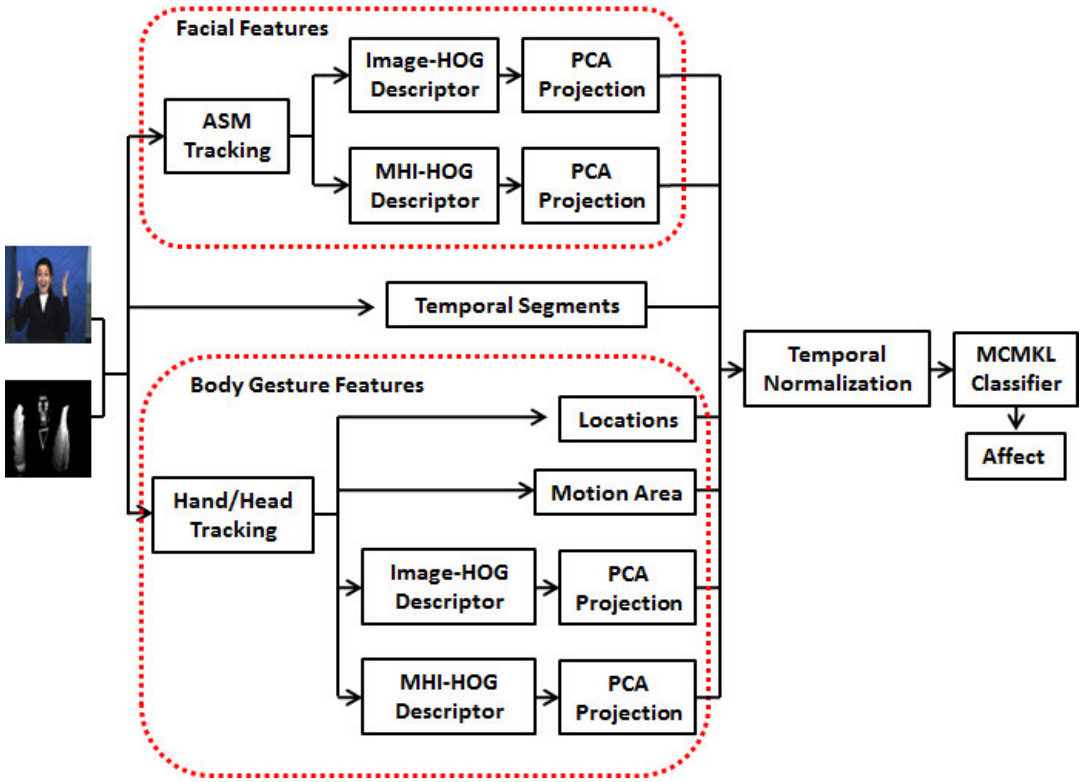


Figure 28: The overview of our proposed MCMKL-based multi-modal fusion for affect recognition through both face and body gestures.

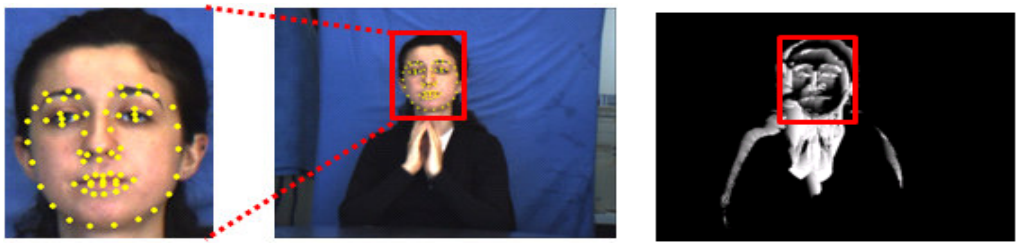
expressions. Finally MCMKL method is employed to find the optimal feature combination and recognize affects.

4.3.2 Facial Features

Active Shape Model [24, 94] is first applied to track 53 facial landmark points including brows, eyes, nose, mouth, and face contour, as shown in Figure 29(a). Then we locate the corresponding positions of the facial points in the Motion History Image (MHI), as shown in Figure 29(b).

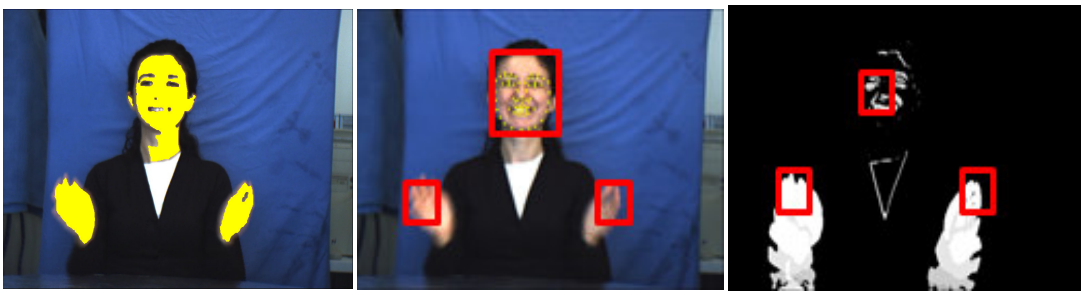
The next step is to extract Image-HOG and MHI-HOG features on original video frames and the corresponding MHI images respectively.

We use 48 by 48 pixels patches with the number of orientation bin equals to 6 and 8 for the Image-HOG and the MHI-HOG features



(a) (b)

Figure 29: (a) Facial landmark points tracking; (b) Motion History Image.



(a) (b) (c)

Figure 30: (a) skin color detection; (b) head and hand position in the original video frame; (c) head and hand positions in the MHI image.

respectively. The MHI image captures motion information of each selected facial point, while the original video frame conveys the appearance information. Finally, we concatenate the Image-HOG descriptor of all the 53 facial points and apply Principal Component Analysis (PCA) to reduce the feature dimension of the concatenated Image-HOG feature from 2862 to 40. Similarly, we can obtain the MHI-HOG descriptor for the corresponding frame and reduce the feature dimension of the concatenated MHI-HOG from 3816 down to 40 for each frame.

4.3.3 Body Gesture Features

To extract body gesture features, we first track both hands and head in an expression video. The head position is simply the center point of the

facial points from the ASM model (see Figure 30(b)). To track hands, we apply a skin color detection [47] followed by the removal of the face regions, which has already been tracked by the ASM model as shown in Figure 30(a) and 30(b).

In addition to the positions of head and hands, we also calculate the motion areas (e.g. the numbers of motion pixels in MHI image) within the detected regions of head and hands. Figure 30(c) shows the head and hand regions in a MHI image.

We further extract Image-HOG and MHI-HOG features in hand regions by uniformly sampling interest points. Then a bag of words representation with the codebook size of 80 is used to describe the distribution of Image-HOG and MHI-HOG features of hand regions. Finally, we perform PCA to reduce their feature dimensions.

4.3.4 Temporal Segmentation

An expression is a sequence of facial movements, which can be roughly described by neutral, onset, apex and offset temporal segments.

Figure 31 shows a sample of the ground truth temporal segmentation of an expression video. The temporal segmentation procedure is necessary to accurately model the expression dynamics, which has been proven crucial for facial behavior interpretation [77]. In our experiments, we simply use the ground truth temporal segmentation and the affect recognition is performed on the complete expression cycle, i.e., onset, apex and offset.

4.3.5 Temporal Normalization

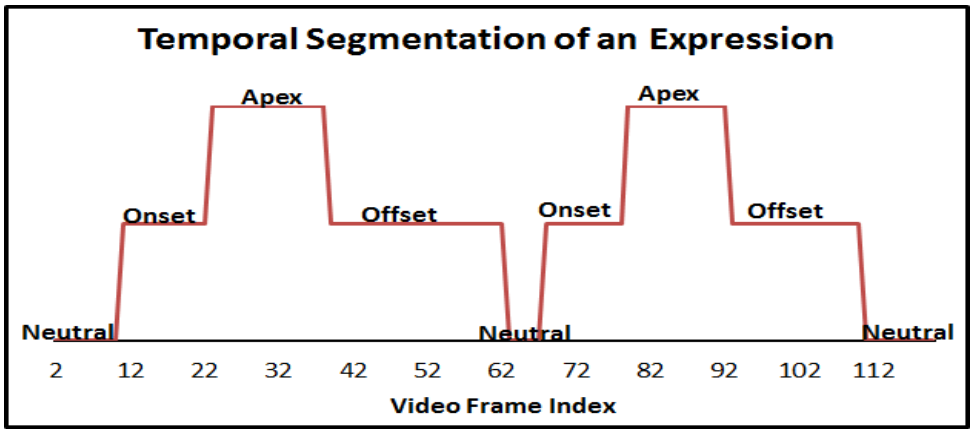


Figure 31: temporal segmentation of an expression video.

In general, the temporal resolution of an expression is generally different when performed by different people. Even same expression performed by same person at a different time, the temporal resolution may not be the same. In order to resolve this time resolution issue in expression videos, we adopt the temporal normalization approach by normalizing all types of features over a complete expression cycle.

The temporal normalization over an expression cycle can be easily implemented by linear interpolation over frame's feature vector along the temporal direction.

4.3.6 MCMKL Based Multi-Modal Feature Fusion

Features from multiple modalities may have different forms. Therefore dimensions of different types of features may vary significantly. Our proposed margin-constrained multiple kernel learning (MCMKL) method can effectively fuse all base features from different modalities, i.e., face and body gesture modality, by assigning a feature weight to each base feature.

We concatenate the Image-HOG and the MHI-HOG of facial points as one base feature, i.e., the face feature. The other four base features are from the gesture channel, i.e., location, motion area, and both hands' Image-HOG and MHI-HOG features. Using the margin of each individual base feature as a guide, along with the DNRBF to synchronize the optimal kernel parameter, the MCMKL learns the optimal combined kernel by selecting a proper weight for each base feature during the fusion.

For our multi-classes application, we choose one vs. one classification, and then using the maximum voting scheme to label testing samples.

4.4 Experiments

4.4.1 Experimental Setups

We use a bi-modal face and body gesture database, i.e., FABO database in our experiments [9]. The database is collected using two cameras, i.e., one for face and one for body gesture in a laboratory environment. A sample video is shown in Figure 32. However, we only employ the videos captured by the body camera to extract features for both modalities, since the videos from the body camera already contain both face and body gesture information.

After removing the categories in the database with very small number of samples, there are 8 expression categories, i.e., "Anger", "Anxiety", "Boredom", "Disgust", "Fear", "Happiness", "Puzzlement", and "Uncertainty". The total number of videos used in our experiment is 255 and each video has 2 to 4 complete expression cycles.

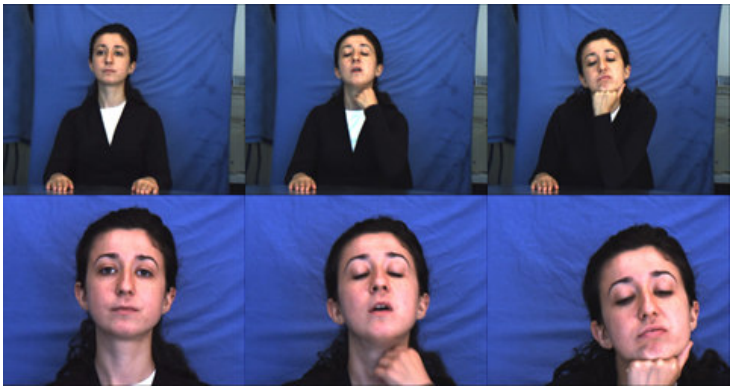


Figure 32: sample video in FABO database recorded by body (top) and face (bottom) camera;

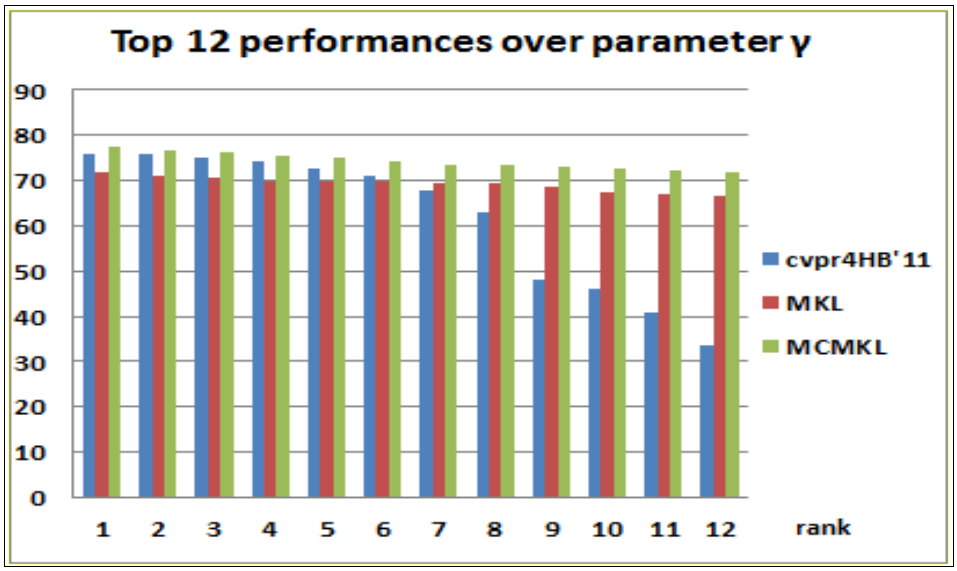


Figure 33: The average performance of the top 12 ranks by sweeping kernel parameter log2(γ) from -15 to 8 for each of the three methods, i.e., concatenation (cvpr4HB'11), MKL, and MCMKL.

We randomly divide the videos into three subsets. Two subsets are used in training and the remaining subset is used in testing. No same video appears in both training and testing. But same subject may appear in both training and testing due to the random selection process.

4.4.2 Comparison to Existing Work and MKL

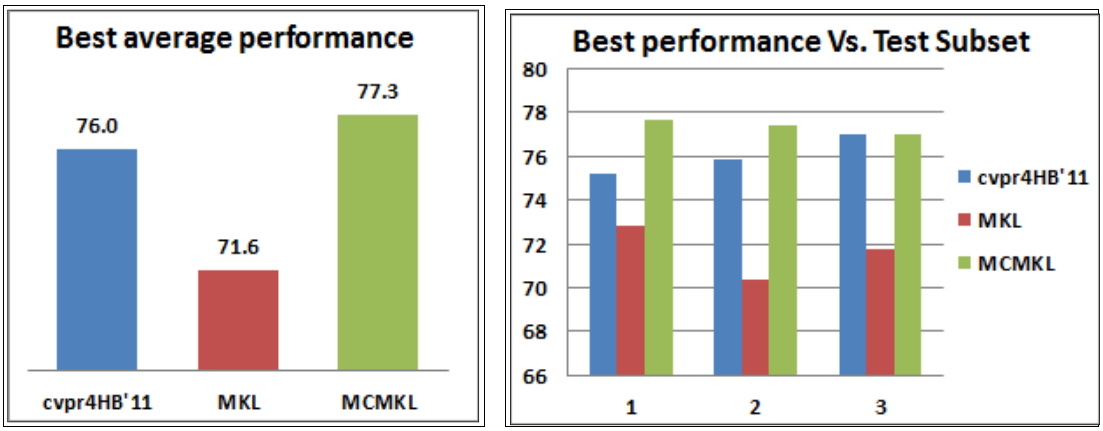
Table 2: The feature dimension for each base feature, i.e., Face, Loc (location), MA (motion area), Img-HOG (Image-HOG from gesture), and MHI-HOG (from gesture).

Base Feature	Face	Loc	MA	Img-HOG	MHI-HOG
Dimension	2400	180	90	120	30

In order to evaluate the effectiveness of the proposed MCMKL fusion method, we compare it to the most recent work on the FABO database [41] by using same features, and same training and testing dataset. We further compare MCMKL with MKL method. The performance of the comparison is displayed in Figure 33.

The five base features are used in our experiments, which include face feature, location feature, motion area feature, Image-HOG and MHI-HOG feature of both hands. The face feature is the concatenation of the Image-HOG and the MHI-HOG from the face modality. The Table 2 shows the corresponding feature dimension for each base feature. These base features are fused through the concatenation, MKL, and MCMKL methods.

To make a fair comparison, we sweep kernel parameter $\log_2(\gamma)$ from -15 to 8, and select the top 12 performances for each fusion method. Then we rank these 12 performances by a descending order of their accuracy. We repeat same experiment for three different subsets and the average performances are reported in Figure 33. Figure 34 shows more details of the rank 1 comparison, i.e., the comparison of the best performances for the three fusion methods: MCMKL, MKL, and direct feature concatenation (cvpr4HB'11 [23]).



(a) (b)

Figure 34: (a) The best average performance by sweeping kernel parameter log2(γ) from -15 to 8 for each fusion method, i.e., cvpr4HB'11, MKL, and MCMKL; (b) The best performance of the three fusion method in three different testing subsets.

MCMKL outperforms the other two methods over all the top 12 ranks, as shown in Figure 33. If we look at the rank 1 comparison in Figure 34, we can see that the proposed MCMKL achieves better performance than the concatenation method. Note that the five base features have been carefully selected, and the parameters, e.g., PCA projection dimensions etc. are also carefully chosen for the concatenation fusion method in [23]. On the other hand, MCMKL effectively select those feature vectors, and it can still outperform the concatenation fusion method by the average of 1.3%.

Our proposed MCMKL outperforms traditional MKL method by an average recognition rate of 5.7% on these base features, as shown in Figure 34(a). Figure 34(b) shows the performance comparison over three different testing subsets.

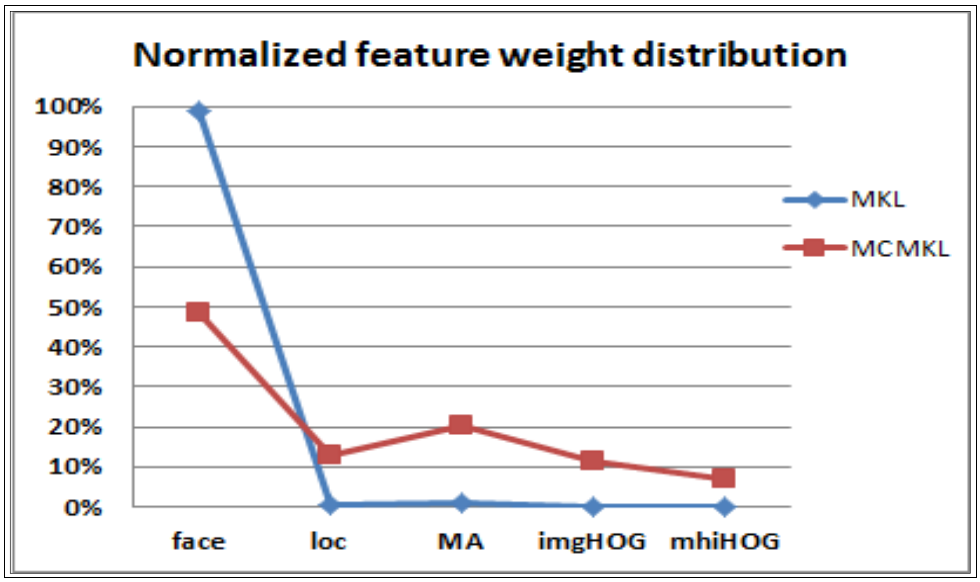


Figure 35: Comparing the average feature weight distribution of the 5 base features, i.e., face, location (loc), motion area (MA), and both hands' Image-HOG (imgHOG) and MHI-HOG (mhiHOG) features, for MKL and MCMKL methods.

4.4.3 Evaluate Feature Weight Distribution

In this section we verify that our proposed MCMKL is more effective than MKL to incorporate less discriminative features, which provide complementary information to the base features with the maximum discriminative power. We select the kernel parameter γ of 2^{-15} for MKL and 2^{-1} for MCMKL method, which yield the best performance for MKL and MCMKL method respectively.

Since we choose one vs. one strategy for our multi-class expression classification, the total number of models we need to train is C_2^n, where n is the total number of expression classes in the dataset. Therefore, we have trained 28 models for 8 categories of expressions, in which each model contains one set of feature weights for the base features, i.e., face, location (loc), motion area (MA), and both hands' Image-HOG (imgHOG) and MHI-

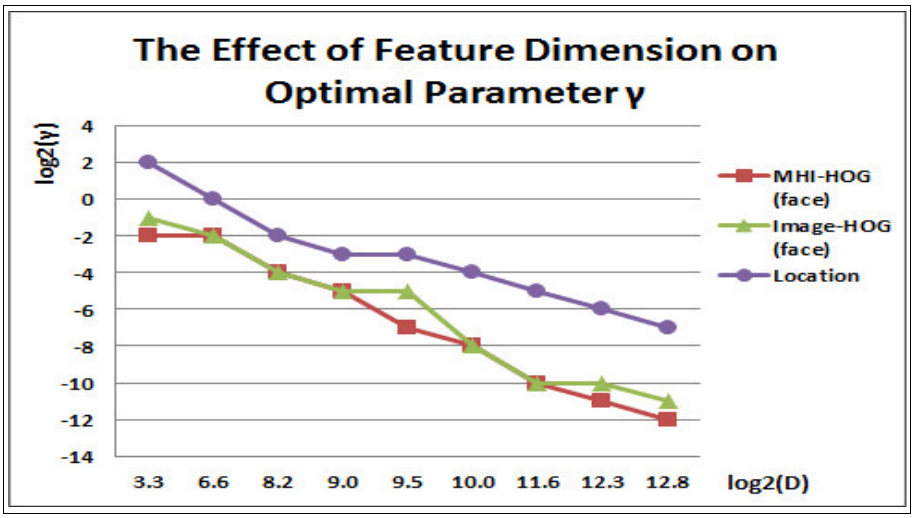


Figure 36: The effect of feature dimension of three base features over the selection of the optimal RBF kernel parameter γ.

HOG (mhiHOG) features. Then we take the mean feature weight of the 28 models over each base feature, followed by the proper normalization.

Figure 35 shows the distribution of average feature weights over the 5 base feature types for MKL and MCMKL method. As expected, MKL selects only the most discriminative base feature, i.e., face feature. More specifically, it assigns more than 98% of the total feature weights to the face feature. The MKL method ignores all the gesture features, i.e., location, and motion area etc., even though these gesture features have been proven to provide complementary information to the face feature [23].

As shown in Figure 35, the proposed MCMKL obtains a more reasonable feature weight distribution. Similar to MKL methods, it recognizes the face feature as the most discriminative base feature by assigning the largest feature weight of 48%. At the same time, it also incorporates other less discriminative gesture features according to their discriminative power.

Figure 35 has verified the effectiveness of the proposed margin constraints and the dimensionally normalized RBF kernel (DNRBF). It is obvious that the additional constraints on the feature weights according to the separation margin of each base feature can enforce the model to assign small weights to the less discriminative base features. However, it may not be intuitive how the DNRBF contribute to a more reasonable feature weight assignment.

Before we provide such intuition, we examine the relationship between the optimal γ value and feature dimension experimentally. We select three base features, i.e., facial point's MHI-HOG, the facial point's Image-HOG and the location feature. Then we manually vary the PCA dimension of the Image-HOG and the MHI-HOG, or the number of normalization frames of the location feature, so that their feature dimensions can be gradually increased. Then we use SVM's 5-fold cross validation to find out the optimal kernel parameter γ for each of the three base features at the selected feature dimension. Figure 36 has suggested the inverse relationship between the optimal γ value and the feature dimension, which has verified our analysis in section 4.2.3.

In the experiments of the last section, the most discriminative feature, i.e., the face feature, has the optimal γ of 2^{-15}. Since other less discriminative gesture feature has much smaller feature dimension as we can see from Table 2. Their optimal γ value is much larger. Therefore, at the γ of 2^{-15}, the other gesture features has almost no discriminative power since their optimal γ values are very far away from 2^{-15}. Therefore, MKL method assigns almost zero feature weights to other gesture features.

After we perform the dimensionally normalization as in Equation (4-13). The optimal γ values become very close for different base features

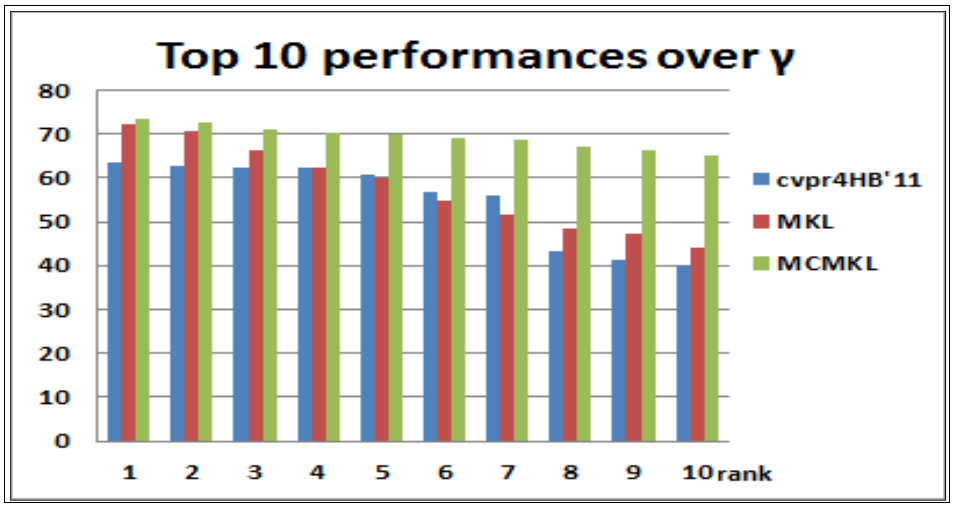


Figure 37: Explore contamination from noisy feature. The average performance of the top 10 ranks by sweeping kernel parameter γ for each of the three methods, i.e., concatenation (cvpr4HB'11), MKL, and MCMKL.

regardless the differences of their feature dimensions. Therefore, MCMKL can utilize the maximum discriminative power of all base features at the same time. That is another reason why MCMKL method can incorporate other less discriminative base features, which provide complementary information.

4.4.4 Contamination from Less Discriminative Features

In this section, we examine the contamination from the less discriminative base features, particularly those with large feature dimensions. From the feature weight distribution in Figure 35, we know that the Image-HOG and MHI-HOG of hands are the least discriminative features. So we intentionally increase their feature dimension to 1200 by including more PCA dimensions. At the same time, we also decrease the dimension of the most discriminative feature, i.e., the face feature, down to 90. Now, we also sweep kernel parameter $\log_2(\gamma)$ and select the top 10

performances for each fusion method, i.e., concatenation, MKL, and MCMKL. Then we rank these 10 performances by the descending order of their accuracy. The experimental results are shown in Figure 37.

We observe that the rank 1 result of the MCMKL method outperforms the concatenation fusion method by almost 10%, which indicate that MCMKL method is more effective to shield the contamination from the less discriminative base features, as compared with the concatenation fusion method.

4.5 Discussion

In this chapter, we have proposed a margin-constrained multiple-kernel learning (MCMKL) method, which extends the multiple-kernel learning (MKL) method by constraining feature weight range according to the separation margin of each base feature. The dimensionally normalized RBF kernel (DNRBF) is also proposed and employed in MCMKL in order to fuse the features from multiple modalities, which is possible to have very different feature dimensions. Our experimental results demonstrate favorable results as compared to the state-of-the-art results on the FABO database. We also demonstrate the significant improvement as compared to the conventional MKL method.

The training time of MCMKL can increase slightly since it needs to find separation margin for each feature type. Nevertheless, by constraining the fusing weight range of each feature type, the MKL training might converge faster, which we need to verify in our future work.

Chapter 5

Discriminative Hierarchical K-Means Tree

5.1 Summary

A key challenge in large-scale image classification is how to achieve efficiency in terms of both computation and memory, but without compromising classification accuracy. The learning-based classifiers achieve state-of-the-art accuracies, but have been criticized for the complexity that grows linearly with the number of classes. The non-parametric nearest neighbor (NN) based classifiers naturally handle large numbers of categories, but incur prohibitively expensive computation and memory costs. In this chapter, we present a novel classification scheme, i.e., Discriminative Hierarchical K-means Tree (D-HKTree), which combines the advantages of both learning-based and NN-based classifiers. The complexity of D-HKTree only grows sub-linearly with the number of categories, which is much better than the recent hierarchical SVM based methods. The memory usage in D-HKTree also benefits from precluding all training features, which is order of magnitude less than the recent NBNN based methods. In the evaluations on several challenging benchmarks, D-HKTree obtains state-of-the-art accuracies, while with significantly lower computation cost and memory requirement.

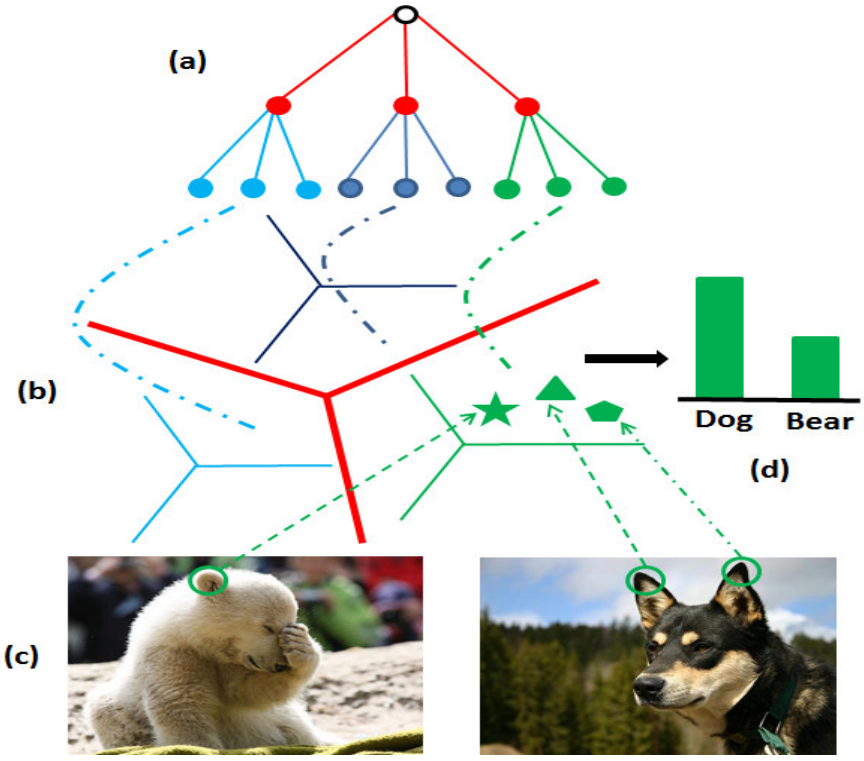


Figure 38: Labeled Hierarchical K-means Tree (L-HKTree). (a) L-HKTree structure. (b) The corresponding feature space partitions projected onto two-dimensional space. (c) Example images from the classes of bear and dog respectively. (d) The label histogram associated with one leaf node, *i.e.*, the frequency of training features falling in the leaf node over all class labels.

5.2 Construction of Discriminative Hierarchical K-means Tree

One attractive property of NN-based classifiers is able to naturally handle large numbers of categories in object and scene classification. However, the aforementioned issues of expensive classification cost and inferior performance have hindered their applications on large-scale image classification, due to high variances in a large-scale dataset but limited training samples. The proposed D-HKTree addresses these issues and

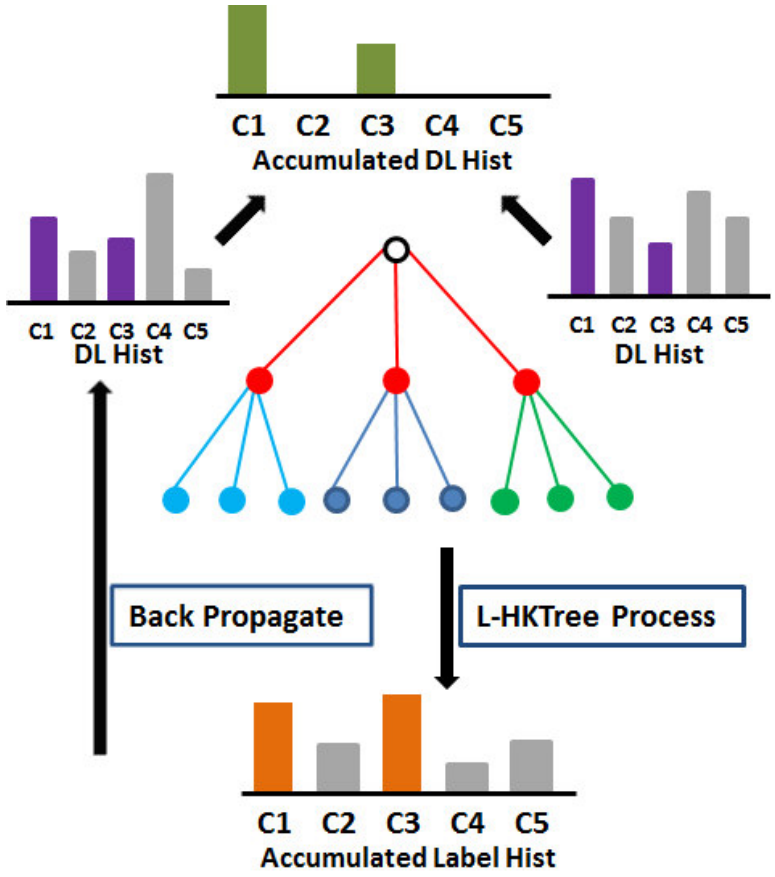


Figure 39: The framework of D-HKTree. In the forward L-HKTree process, query features come down the L-HKTree, and the label histograms of associated leaf node are summed up into the accumulated label histogram at the bottom. Top P performance class labels are selected. Then query features propagate back to their parent non-leaf nodes, and the DL histogram at the non-leaf nodes of the selected class labels are summed up into the accumulated DL histogram. The class label is finally predicted according to the maximum value in the accumulated DL histogram.

extended NN-based classification scheme to large-scale object and scene classification. In this section, we describe the detailed theoretical derivations and computational procedures to build D-HKTree and L-HKTree.

5.2.1 Algorithm Overview

Figure 39 demonstrates the framework of D-HKTree. Query features in a testing image come down L-HKTree and arrive at their nearest neighbor leaf nodes. The label histograms at the associated leaf nodes are summed up into the accumulated label histogram at the bottom, where top p performance class labels are selected. After the forward L-HKTree process, query features propagate back to their parent non-leaf nodes at a certain level, where their discriminatively learned (DL) histograms over the selected p class labels are summed up into the accumulated DL histogram at the top. The predicted class label is finally inferred from the accumulated DL histogram. The DL histograms associated with non-leaf nodes of a particular level can be generated by training a suitable discriminatively learning classifier. In order to maintain a high speed, the accumulations of label histograms associated with leaf nodes and DL histograms associated with non-leaf nodes are implemented by simple operations.

5.2.2 Labeled Hierarchical K-means Tree

The main structure of D-HKTree builds upon L-HKTree. In this subsection, we provide detailed procedures in deriving and building L-HKTree.

(A) Towards L-HKTree

As the classification rules in NBNN, we make the two assumptions, i.e. uniform prior over all class labels and the independence of query features d_i in a testing image Q [10]. The predicted label can be obtained by Equation (5-1).

$$\hat{C} = \underset{C}{\operatorname{argmax}} \quad \sum_{i=1}^{N} \log p(d_i \mid C) \qquad . \qquad (5\text{-}1)$$

We explicitly model unbalanced data over class labels in the Parzen window estimator. The total number of training features in class C is L_C.

$$\log p(d_i \mid C) = \frac{1}{L_C} \sum_{j=1}^{Lc} K(d_i - d_j^C) \qquad . \qquad (5\text{-}2)$$

Instead of using the Gaussian kernel for *K* as in the local NBNN, we use a uniform kernel with the bandwidth of r_i.

$$K(d_i - d_j^C) = B_f U(1 - \frac{\left\| d_i - d_j^C \right\|}{r_i}) \qquad , \qquad (5\text{-}3)$$

where the bandwidth $r_i > 0$, B_f is a positive constant and U is a unit step function, which is 1 if the Euclidean distance between the training feature d_j^C and the query feature d_i is less than the bandwidth r_i; and otherwise is 0. If substituting Equation (5-3) to Equation (5-2), we have

$$\log p(d_i \mid C) = \frac{B_f}{L_C} \sum_{j=1}^{Lc} U(1 - \frac{\left\| d_i - d_j^C \right\|}{r_i}) \qquad . \qquad (5\text{-}4)$$

The summation term in Equation (5-4) denotes the total number of training features in class *C*, which have Euclidean distance smaller than r_i away from the query feature d_i. This can be illustrated using Figure 40(a). The center of the purple circle is at the query feature d_i with the radius equal to the bandwidth of r_i. The summation term in Equation (5-4) for the triangle class is the total number of triangle training features falling within the purple circle, i.e. 2 in Figure 40(a). Similarly, the summation terms for the pentagon class and the star class are 3 and 1, respectively. As illustrated

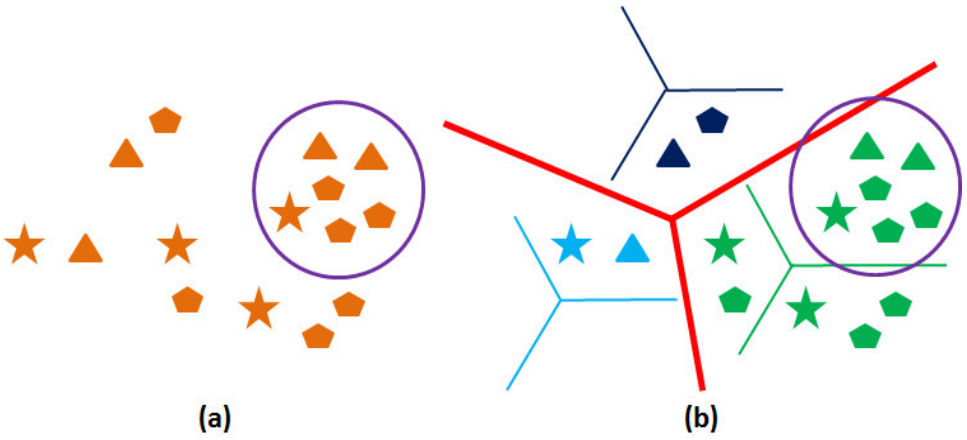


Figure 40: Illustration of unit step function. (a) The purple circle illustrates the unit step function in equation (5-4). The center is at the query feature di with the radius equal to the bandwidth of ri. (b) Approximate the unit step function of the purple circle with the feature space partition by one of the leaf nodes in the HKTree.

in Figure 40(b), we further approximate the unit step function U with the feature space boundary defined by the leaf node f_i, in which the query feature d_i falls within.

$$U(1-\frac{\|d_i - d_j^C\|}{r_i}) \cong \delta(f_i, LEAF(d_j^C)) \qquad (5\text{-}5)$$

where $LEAF(d_j^C)$ is the nearest neighbor leaf node of the training feature d_j^C; δ equals to 1 when feature d_i and d_j^C falls within the same feature space partition defined by the leaf node f_i. If we substitute Equation (5-5) to Equation (5-4), it becomes

$$\log p(d_i \mid C) = \frac{B_f}{L_C} \sum_{j=1}^{L_C} \delta(f_i, LEAF(d_j^C)) \quad . \qquad (5\text{-}6)$$

Note the right hand side of the Equation (5-6) does not depend on the query feature d_i except that f_i is the leaf node of d_i. Therefore, we can pre-

compute the right hand side of Equation (5-6) for each class label C, and store the results as the label histogram associated with the leaf node f_i, denoted as $LH(f_i, C)$.

The summation term in Equation (5-6) is the number of training features from category C, which falls within the feature space partition of the leaf node f_i. L_C corresponds to the total number of training features in category C. B_f is a L1-Norm constant. Substituting Equation (5-6) to Equation (5-1), we have the L-HKTree classification rule.

$$\begin{aligned}\hat{C} &= \underset{C}{\operatorname{argmax}} \quad \sum_{i=1}^{N}\left[\frac{B_f}{L_C}\sum_{j=1}^{L_C}\delta(f_i, LEAF(d_j^C))\right] \\ &= \underset{C}{\operatorname{argmax}} \quad \sum_{i=1}^{N}\left[LH(LEAF(d_i), C)\right]\end{aligned} \quad . \qquad (5\text{-}7)$$

As demonstrated in the derivation of the L-HKTree, we do not need to compute the pair-wise distance of the query feature d_i with each training feature d_j^C online. Instead, we only employ the label histogram associated with the leaf node where d_i falls within. Hence, L-HKTree does not retain any training feature. This results in a significant saving in the memory, which allows us to extend this NN-based classifier to a large-scale classification task.

As suggested in Equation (5-7), in the testing phase, we only need accumulate label histograms of the leaf nodes, at which query features arrive. The computation cost is therefore significantly reduced as compared to the conventional NN-based classifiers [10] and the recently proposed local NBNN [62]. Note the complexity of L-HKTree is independent of the number of classes, which is a very attractive characteristic for image classification on large-scale dataset with huge number of categories.

Algorithm 1: Building L-HKTree

Require:
(1) Hierarchical Kmean Tree HKT
(2) Initialize label histograms and L_C to 0

for all categories C do
  for all training feature d_i in category C do
    $f = LEAF(d_i)$
    $LH(f, C) = LH(f, C) + 1$
    $L_C = L_C + 1$
  end for
end for

for all leaf nodes f do
  for all categories C in f do
    $LH(f, C) = LH(f, C)/L_C$
  end for
  L1 normalize leaf node the label histogram of f
end for

return HKT

(B) Building L-HKTree

L-HKTree is first constructed by the hierarchical K-means tree [69] from training samples with L levels and K branches. Figure 38(a) illustrates a two-level tree with three branches. The corresponding feature space partition of each leaf node projected on the two-dimensional space is shown in Figure 38(b). We modify the original hierarchical K-means tree to automatically adjust the number of branches of a non-leaf node if the average training features arriving at its children nodes is below a threshold N.

Algorithm 1 provides the pseudo code to build L-HKTree. A label histogram associated with each leaf node, as shown in Figure 38(d), accumulates the number of training features arriving at this leaf node over

all class labels. For example, Figure 38(c) shows two images from the classes of bear and dog. One training feature from the bear class and two training features from the dog class arrive at the feature space partition of one green leaf nodes in Figure 38(b). The label histogram of this green leaf node in Figure 38(d) describes the frequency of training features arriving at this leaf node over the class labels of dog and the bear.

In order to handle unbalanced training data in different categories, we normalize label histograms using the total number of training features of corresponding class labels, i.e., L_C in Algorithm 1.

All leaf nodes in L-HKTree have defined their corresponding feature space boundaries as illustrated in Figure 38(b). Any feature arriving at a leaf node can be considered as a nearest neighbor of this leaf node. The label histogram associated with each leaf node counts numbers of training features over class labels, which correspond to nearest neighbors to the leaf node within the predefined feature space boundary. Intuitively, the more nearest neighbor features are from a class label C in the label histogram of the leaf node f, the smaller distance is between the leaf node f and the class label C. Therefore, we can have another interpretation of the label histogram as the inverse distance from leaf node f to different classes.

(C) Pre-Classification with L-HKTree

The classification rule of L-HKTree is shown in Equation (5-7). Query features from a testing image arrive at corresponding leaf nodes. The summation of the label histograms associated with leaf nodes forms the Accumulated Label Histogram(ALH). The P class candidates can be selected by the top P values in ALH. The predicted class label (top 1 class candidate) corresponds to the maximum value in ALH.

In our implementation, instead of using every query feature in a testing image to accumulate label histograms, we employ the non-leaf nodes at a particular level as a filter to remove some noisy query features. This is motivated by the max pooling scheme [56] widely used in computing BOW. If we deem the non-leaf nodes at a particular level as a visual vocabulary, a coding scheme (e.g., local soft assignment) and a pooling scheme (e.g., max pooling) can be used to compute a BOW histogram H over the selected non-leaf nodes. Similarly in L-HKTree, only the query features, which provide the max response at the non-leaf nodes at a certain level, are allowed to continue down. In this way, we can also weight label histograms by the responses of corresponding query features at the selected non-leaf nodes. In the end, the weighted label histograms are accumulated into ALH.

5.2.3 Discriminatively Learned Histogram

We choose the one-versus-all linear SVM as the discriminative learning algorithm to incorporate into NN-based L-HKTree. In this subsection, we show how to transform the learned SVM weights to the discriminatively learned (DL) histograms of non-leaf nodes at a selected level.

As stated above, in building L-HKTree, we can also compute BOW H for training images using the selected non-leaf nodes as a visual vocabulary, e.g., the red non-leaf nodes of L-HKTree in Figure39. We then employ one-versus-all linear SVM to obtain the weights W^C for class C. The score vector over T classes can be computed by Equation (5-8). For convenience, we ignore the bias term.

$$Y = [y^1, y^2, ..., y^C, ..., y^T] \quad , \quad (5\text{-}8)$$

where $y^C = W^C \cdot H = \sum_{k=1}^{M} W_k^C H_K$ and M is the number of non-leaf nodes at the selected level. Substituting y^C into Y, we have

$$Y = \sum_{k=1}^{M} [W_k^1 H_k, W_k^2 H_k, ..., W_k^C H_K, ..., W_k^T H_K] \quad . \quad (5\text{-}9)$$

If we further factor out H_k from the vector Y, we have

$$Y = \sum_{k=1}^{M} H_k W_k \quad , \quad (5\text{-}10)$$

where $W_k = [W_k^1, W_k^2, ..., W_k^C, ..., W_k^T]$.

W_k is the discriminatively learned histogram over T classes at the kth non-leaf node at the selected level. As label histograms are associated with leaf nodes, discriminatively learned histograms are attached to selected non-leaf nodes as well.

5.2.4 Classification with D-HKTree

As shown in Figure 39, a testing image is first pre-classified by L-HKTree according to Equation (5-7). We obtain the accumulated label histogram and select the top P performance class labels from this histogram. Query features in the leaf nodes are then propagated back to their parent non-leaf nodes at the selected level. Their discriminatively learned histograms of the selected P class labels are multiplied by the maximum responses at corresponding non-leaf nodes, which are summed up as the accumulated discriminatively learned histogram. The bias terms of trained SVM models are then subtracted from the accumulated discriminatively learned histogram for the selected class label. Finally, we select the class

label with the highest score from the accumulated discriminatively learned histogram.

The number of top class candidates P is adaptively selected for different testing images. We utilize the distribution of the accumulated label histogram to build a cumulative confidence level, which is the cumulative probability of the class labels in the accumulated label histogram. In our experiments, we set a threshold of 0.2 to the cumulative confidence level to determine the top P class candidates. By adaptively select the top P classes in the forward L-HKTree process, D-HKTree is able to achieve a complexity that grows only sub-linearly with the number of classes. On the other hand, the best performing learning-based classifiers has the computation cost at least increases linearly with the number of categories. Therefore, D-HKTree can handle large-scale image classification with huge numbers of categories more efficiently than learning-based classifiers while maintain state-of-the-art accuracy.

5.3 Experiments

We evaluate our proposed models on object and scene recognition datasets including Caltech 101 [31], Caltech 256 [39], and SUN dataset [99]. D-HKTree significantly outperforms all previous NN-based classifiers in terms of classification accuracy, computation cost, and memory requirement. The relative computational complexity of D-HKTree is also significantly improved compared to the state-of-the-art learning-based classifiers. Experimental results demonstrate that D-HKTree can scale very well to large-scale datasets with large numbers of categories.

Table 3: Comparison to NN-based classifiers on accuracy and speed (sec/image).

Dataset	Caltech 101		Caltech 256		SUN	
Method	accuracy	speed	accuracy	speed	accuracy	speed
1-NN	-	-	7.6±0.7* [10]	-	13* [22]	-
NBNN [2]	70.4*	23.29	37*	450.5	-	-
NBNN [20]	65.5±1.0*	-	-	-	-	-
Local NBNN	71.9 ±0.6*	2.89	40.1±0.1*	112.4	-	-
D-HKTree	**77.6±0.66**	**1.34**	**45.5±0.58**	**3.7**	**35.7±0.18**	**1.89**

5.3.1 Experimental Setup

We employ the dense SIFT features augmented by x and y coordinates throughout our experiments. The two spatial dimensions in the augmented feature vector are weighted by 1.6 in Caltech 101 and 0.75 in Caltech 256 and SUN, as recommended in [62, 63]. L-HKTree has 2 levels with maximum branch factor of 65K in Caltech 101 and 130K in Caltech 256 and SUN. We employ the approximate nearest neighbor library FLANN [67] to find the nearest neighbor branch for a query feature at each level.

In order to facilitate a fair comparison, we follow the evaluation conventions, i.e. we use 30 images per category as training data and 15 images per category as testing data in Caltech 101 and Caltech 256 datasets. We repeat the experiments 10 times with random selection of non-overlapping training and testing data. The average accuracy with the standard deviation is reported in the paper. As for SUN, we use exactly the same training and attesting splitting as in [99], i.e. 50 images for both training and testing sets.

5.3.2 Comparisons to NN-based Classifiers

In this paper, we use the "*" sign after a number to indicate that the number is directly quoted from original papers. The "-" in the table indicate that the data is not available.

Table 3 shows the comparisons of D-HKTree with other NN-based classifiers on both classification accuracy and computation cost. D-HKTree has achieved the highest accuracies, i.e. 77.6% and 45.5%, on Caltech 101 and Caltech 256, respectively. We achieve 35.7% accuracy on the SUN dataset. To the best of our knowledge, this is the highest accuracy on this dataset using a single feature type. Since the D-HKTree takes advantages of both NN-based and learning-based classifiers, the performance is much better than the conventional NN based classifiers. The second step of D-HKTree tree, i.e. the learning based classifier, has boosted up the classification accuracy. We also quote the classical 1-NN classifier results on Caltech 256 and SUN datasets for the comparison in the Table 3. The 1-NN classifier is a correlation classifier in the feature space with pixel intensities of a resized image [39]. The 1-NN results reported in [99] use multiple feature types. If using a single feature type, the result is probably even worse.

The testing speed of D-HKTree is significantly faster than the conventional NN-based classifiers, especially on larger datasets. To evaluate the testing speed, we use the code provided by [62] with the recommended parameters for NBNN and local NBNN. The L-HKTree defines partition boundary in feature space, which pre-stores nearest neighbor feature statistics within each boundary. The testing of a query image does not need to go through all features in training data. Instead, it only requires the query features to find the corresponding boundary and update the nearest neighbor statistics.

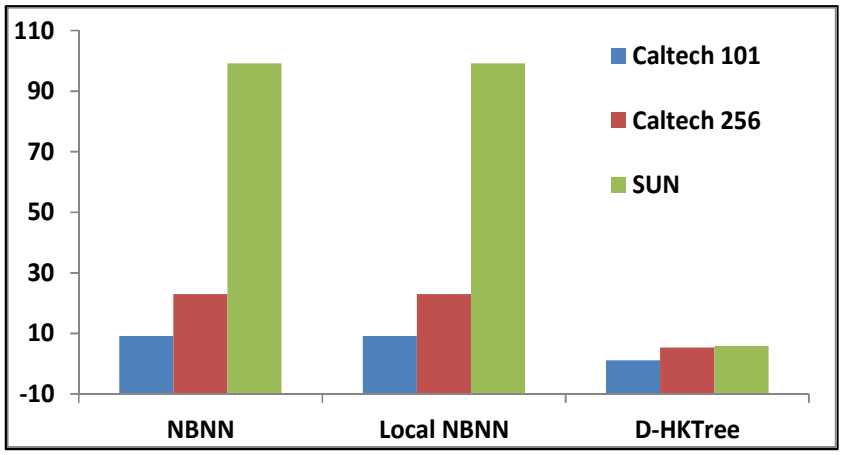


Figure 41: Comparing memory usage over different NN-based classifiers as the scale of the dataset increases.

If we use 30 training images per category on Caltech 256, the testing speed of D-HKTree is 30 times faster than local NBNN, and 120 times faster than NBNN. As shown in Table 3, local NBNN is 10 times faster than NBNN on Caltech 101, but only 4 times faster on Caltech 256, which is different from the results reported in [62].

We think the discrepancy may be due to the number of training images in each category used and the number of descriptors extracted in each training image. If the total number of training features is too large, the testing time might increase exponentially instead of logarithmically depending on the implementation. On the other hand, the time increment is significantly lower for D-HKTree as the dataset changes from Caltech 101 to Caltech 256. It is interesting to observe the computation cost of SUN is even lower than that of Caltech 256, i.e. 1.9 second per image versus 3.7 second per image. This is due to the structure difference of the L-HKTrees built for the two datasets, since the major computation costs of D-HKTree is from the pre-classification of L-HKTree.

Table 4: Comparison of classification accuracy to learning based classifiers on (a) Caltech 101 and Caltech 256 datasets; (b) SUN dataset.

Method	Caltech 101	Caltech 256
Orig SPM	64.6±0.8* [12]	34.1±0.2* [10]
SC SPM [23]	73.2±0.54*	34±0.35*
SLC	**78.39±0.59**	**45.71±0.59**
KC [9]	64.1±1.5*	27.2±0.4*
D-HKTree	**77.6±0.66**	**45.5±0.58**

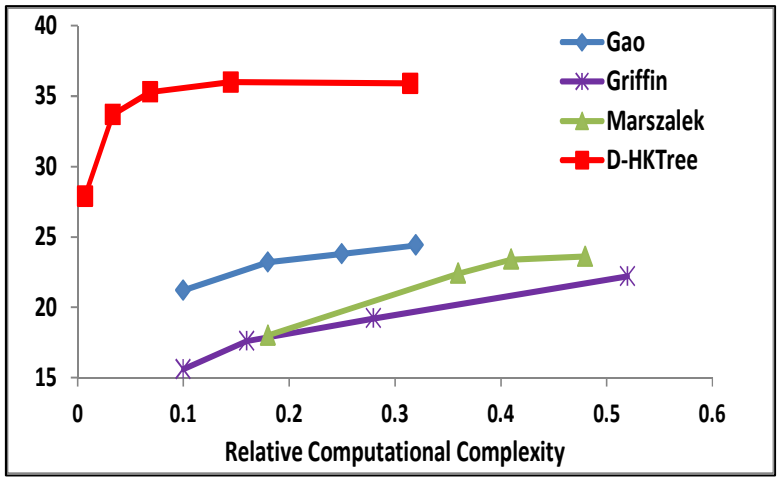
(a)

Method	SUN
Orig SPM [22]	21.5*
Spatial HOG[22]	27.2*
Spatial HOG[7]	27*
D-HKTree	**35.7±0.18**

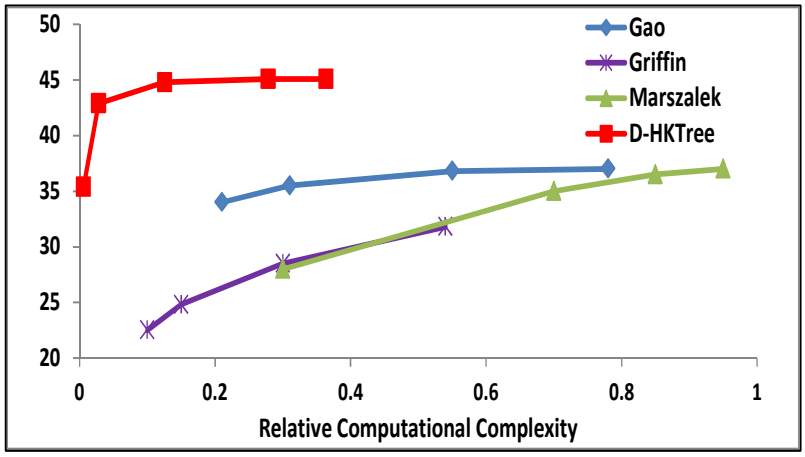
(b)

Figure 41 compares the memory requirements of different NN-based classifiers as the scale of dataset increases. Since the memory usage of conventional NN-based classifiers is directly related to the size of training data size, we can estimate their memory usages according to the training data size. As for D-HKTree, the memory usage is estimated from the size of L-HKTree. As shown in this figure, the memory usage of D-HKTree only increases slightly from Caltech 101 to Caltech 256, then SUN. However, the memory consumptions of NBNN and local NBNN grow significantly as dataset scale increases. For example, the memory requirement is around 100GB for both NBNN and local NBNN under our experimental settings in SUN dataset, while the memory usage of D-HKTree is only 6GB.

5.3.3 Comparisons to Learning-based Classifiers



(a)



(b)

Figure 42: Comparison of the tradeoff between accuracy and relative computational complexity to other hierarchical SVM based methods for large-scale data, i.e. Gao [34], Griffin [38], Marszalek [59], on (a) SUN dataset; (b) Caltech 256 dataset; Note that the results from the other three methods are directly estimated from the plots in the paper [34].

Table 4 demonstrates the comparisons of classification accuracy between D-HKTree and the state-of-the-art learning-based methods. D-HKTree outperforms most learning-based methods and achieves comparable performance to the recently proposed Spatially Local Coding (SLC) [63] on Caltech 101 and Caltech 256. As for the SUN dataset in Table 4(b), D-

HKTree significantly outperforms the current state-of-the-art with a single feature type by more than 8% in classification accuracy.

To further evaluate the scalability to large-scale image classification, we compare D-HKTree with several hierarchical SVM based classifiers [34, 38, 59] in Figure 42. All of these hierarchical SVM based classifiers attempt to improve the efficiency of one-versus-all linear SVM classifier, so that their complexity can grow sub-linearly with the number of categories.

However, these classifiers have to sacrifice classification accuracy for the improvement on speed. We adopt the relative computational complexity metric introduced in [34] for our evaluation. In the case of a linear kernel, the relative complexity is the ratio between the number of categories evaluated and the total number of categories in the dataset.

The relative computational complexity of D-HKTree can be tuned by varying the number of top selected class labels from the accumulated label histogram. Although the computation cost of L-HKTree is not reflected in this metric, this cost is independent of the number of categories. As demonstrated in Figure 42, D-HKTree dominates the classification accuracies on Caltech 256 and SUN datasets, especially when the relative computational complexity is low. For instance, at the relative computational complexity of 0.06 in Figure 42(a), D-HKTree achieves 35% on the SUN dataset, which is more than 10% higher than that of the best method reported in [34]. Similar results are shown on the Caltech 256 dataset in Figure 42(b). It is also interesting to observe that the classification accuracy of D-HKTree tend to saturate around the relative computational complexity of 0.1, which means D-HKTree is more effective to reduce the relative computational complexity and maintain a desirable accuracy.

The top p categories, the L-HKTree selected, are the most confident class labels specifically for the testing image. It has pruned away a large number of unrelated class labels. Hence, the learning based classifier in the second step can focus only on those class labels which are confused by the L-HKTree, and make the final prediction with better accuracy.

On the other hand, leaf node in the hierarchical SVM based classifier also has several class candidates to make the final prediction. The difference is that the class candidates are not the most confident class labels specifically to the testing image. Therefore, D-HKTree yields better performance.

5.3.4 Comparisons to Hybrid Classifiers

There are very few work [85, 107] on combining learning-based and NN-based classifiers to take advantages of both classifier types. Table 5 compares D-HKTree with two other methods that hybrid both classifier types, i.e. SVM-KNN [107] and NBNN Kernel [85]. As shown in this table, D-HKTree significantly outperforms SVM-KNN and NBNN Kernel by 11% and 8% in accuracy respectively. Note that NBNN Multi-Kernel is actually combining NBNN Kernel with other kernels of different feature types instead of a single feature type. Nevertheless, D-HKTree still obtains the highest accuracy as shown in Table 5.

Table 5: Comparison to the hybrid classifiers, i.e. SVM-KNN [24], and NBNN Kernel [20], on the Caltech 101 dataset.

Method	SVM-KNN	NBNN Kernel	NBNN Multi-kernel	D-HKTree
Accuracy	66.2±0.5*	69.6±0.9*	75.2±1.2*	**77.6±0.66**

SVM-KNN [107] selects k-nn neighbor images for a query image and training a local multi-class SVM classifier to predict query image. As the training data closely coupled with the nearest neighbor results, the nn-based and svm based classifier cannot complement each other very well. That may cause the significant performance decrease as compared with the D-HKTree.

5.4 Discussion

In this chapter, we have proposed a novel classification scheme, i.e. D-HKTree, for larg-scale image classification. D-HKTree takes advantages of both learning-based and NN-based methods. It extends the ability of NN-based classifiers to scale to large numbers of image classes due to much lower computation cost and memory requirement, while achieving state-of-the-art classification accuracies. Compared to NN-based methods, D-HKTree significantly outperforms NBNN and local NBNN in classification accuracy, computation cost, and memory usage. Compared with learning-based methods, D-HKTree largely improves the accuracy of hierarchical SVM based methods at much lower relative computational complexity. Compared to previous hybrid methods, D-HKTree also obtains much better performance than SVM-KNN and NBNN Kernel.

Hierarchical K-Mean tree is employed to partition feature space for training data in the D-HTree. However, any approximate nearest neighbor

method, such as hashing algorithms, can be used to generate the partition regions.

Chapter 6

Conclusion and Future Work

6.1 Discussions and Conclusion

Visual classification contains three major components, i.e., feature extraction, feature representation and classification. Each component can have significant impact on both classification accuracy and efficiency. The recent progress on each component of visual classification is impressive and encouraging toward a more robust visual classification system. In this dissertation, we have made contributions for feature representation and classifier design. Especially, we focus on multi-feature fusion to improve classification accuracy, and large-scale learning algorithm, which takes advantages of both learning-based and nearest neighbor based classifiers.

We have proposed a new feature representation, i.e., EigenMap, which extends the BOW model with spatial information. An EigenMap describes the distribution of a visual word in the image's spatial space. By collecting EigenMaps of all visual words as an object representation, we not only understand what object parts are in the image, but also know where these object parts occur relative to the image space. Unlike spatial pyramid matching method, EigenMap does not require manual partition of image space, and has lower dimension.

To effectively combine multiple types features, we have proposed a margin constrained multiple-kernel learning framework (MCMKL), which finds optimal combination of base features' kernels. The MCMKL utilizes the discriminative power of each individual type of features, i.e., the separation margin of features, to guide the learning of optimal weights for kernel fusion.

The dimensionally normalized RBF kernel (DNRBF) is also proposed and employed in MCMKL in order to fuse features with very different feature dimensions. Our experiments demonstrate that MCMKL achieves better performance as compared to the state-of-the-art results on affection recognition by combining facial features and body gestures.

For large-scale visual classification problem, we have proposed a new classifier, Discriminative Hierarchical K-Means Tree (D-HKTree), which combined the advantages of both learning-based and nearest neighbor (NN) based classifiers. D-HKTree significantly outperforms conventional NN-based classifiers in classification accuracy, while with much lower computation cost and memory usage. Compared to learning-based methods, D-HKTree achieves comparable performance with lower relative computational complexity.

6.2 Limitations and Future Work

Similar to other feature representations, which are based on absolute spatial information, EigenMap is not invariant to translation or rotations. Future work on feature representation will capture the relative spatial relationship among different object parts while maintaining high efficiency.

Margin Constrained Multiple-kernel learning (MCMKL) requires iteratively solving SVM problem and projected gradient descent problem many times. The total training time can become prohibitively high, as one iteration already has expensive computational cost. To scale the algorithm to the large-scale data, our future work will focus on improving the efficiency on multi-feature fusion.

Even though the Discriminative Hierarchical K-Means Tree (D-HKTree) achieves the state of the art performances on several large-scale datasets, its complexity, however, greatly depends on the accuracy of the underlying nearest neighbor algorithm, i.e., L-HKTree. We will evaluate its performance impact as the dataset becomes larger.

Another limitation of D-HKTree is the construction time of L-HKTree. As the data size increases, the computational cost to construct the L-HKTree also increases. If we can build a universal L-HKTree, which only need be modified slightly according to a specific dataset, then we can significantly reduce the training complexity. We would like to continue develop a universal L-HKTree in future.

Bibliography

1. S. Arya, D. Mount, N. Netanyahu, R. Silverman, and Y. Wu, "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions", Journal of the ACM, 45:891–923, 1998.
2. F. Bach, G. Lanckriet, M. Jordan, "Multiple kernel learning, conic duality and the SMO algorithm", NIPS, 2004.
3. R. Baeza-Yates and B. Ribeiro-Neto. Modern Information Retrieval. ACM Press, ISBN: 020139829, 1999.
4. H. Bay, A. Ess, T. Tuytelaars, L. Gool, "Speeded-Up Robust Features (SURF)", Computer Vision and Image Understanding Vol. 110, Issues 3, PP. 346-359, June 2008.
5. R. Behmo, P. Marcombes, A. Dalalyan, V. Prinet, "Towards Optimal Naïve Bayes Nearest Neighbor", European Conference of Computer Vision (ECCV), 2010.
6. S. Belongie, J. Malik, and J. Puzicha, "Shape Matching and Object Recognition Using Shape Contexts," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 2, no. 4, pp. 509-522, Apr. 2002.
7. S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multiclass task. In NIPS, 2010.
8. S. Bernard, "Density Estimation for Statistics and Data Analysis", 1st Edition, Chapman and Hall, 1986.
9. D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation", Journal of Machine Learning Research, 3:993-1022, 2003.

10. A. Bobick and J. Davis, “The recognition of human movement using temporal templates”. PAMI, 2001.
11. O. Boiman, E. Shechtman, M. Irani, “In Defense of Nearest Neighbor Based Image Classification”, Computer Vision and Pattern Rrecognition (CVPR), 2008.
12. A. Bosch, A. Zisserman, X. Munoz, “Scene classification via pLSA”, Proc. 9th European Conference on Computer Vision (ECCV'06) Springer Lecture Notes in Computer Science 3954: 517~530.
13. A. Bosch, A. Zisserman, X. Munoz, “Image classification using random forests and ferns”, Proc. 11th International Conference on Computer Vision (ICCV'07) (Rio de Janeiro, Brazil): 1-8.
14. Y. Boureau, F. Bach, Y. LeCun, J. Ponce, “Learning Mid-Level Features For Recognition”, Computer Vision and Pattern Recognition (CVPR), 2010.
15. Y. Boureau, N. Roux, F. Bach, J. Ponce, Y. LeCun, “Ask the locals: multi-way local pooling for image recognition”, ICCV 2011.
16. C. Burges, “A tutorial on support vector machines for pattern recognition”, Data Mining and Knowledge Discovery, 2(2), 121–167, 1998.
17. G. Burghouts, J. Geusebroek, “Performance evaluation of local colour invariants”, *Computer Vision and Image Understanding* 113: 48-62, 2009.
18. Y. Cao, C. Wang, Z. Li, L. Zhang, L. Zhang, “Spatial Bag of Features”, CVPR, 2010.
19. C. Chang and C. Lin, “LIBSVM : a library for support vector machines”, ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011.

20. O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, “Choosing Multiple Parameters for Support Vector Machines”, Machine Learning, 2002.
21. Y. Chen, M. Crawford, and J. Ghosh. Integrating support vector machines in a hierarchical output space decomposition framework. In IGARSS, 2004
22. S. Chen, Y. Tian, Q. Liu, and D. Metaxas, Recognizing Expressions from Face and Body Gesture by Temporal Normalized Motion and Appearance Features, Image and Vision Computing, Volume 31 Issue 2, February, Pages 175-185, 2013, DOI: 10.1016/j.imavis.2012.06.014
23. S. Chen, Y. Tian, Q. Liu, D. Metaxas, “Recognizing Expressions from Face and Body Gesture by Temporal Normalized Motion and Appearance Features”, IEEE Int'l Conf. Computer Vision and Pattern Recognition workshop for Human Communicative Behavior Analysis (CVPR4HB). 2011.
24. T. Cootes, C. Taylor, D. Cooper and J. Graham, “Active Shape Models – Their Training and Application”, Computer Vision and Image Understanding, 1995.
25. C. Cortes, V. Vapnik, "Support-Vector Networks", Machine Learning, 20, 1995
26. G. Csurka, C. Dance, L. Fan, J. Willamowski, C. Bray, “Visual Categorization with Bag of Keypoints”, ECCV, 2004.
27. N. Dalal, B. Triggs, “Histograms of Oriented Gradients for Human Detection”, CVPR 2005.
28. J. Deng, A. Berg, K. Li, and L. Fei-Fei, “What does classifying more than 10,000 image categories tell us”, ECCV, 2010.

29. C. Evans, “Notes on the OpenSURF Library”, University of Bristol, 2009.
30. L. Fei-Fei and P. Perona, “A Bayesian hierarchy model for learning natural scene categories”, CVPR 2005.
31. L. Fei-Fei, R. Fergus and P. Perona, “One-Shot learning of object categories”, IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), 2006.
32. P. Felzenszwalb and D. Huttenlocher, “Efficient Graph-Based Image Segmentation”, IJCV, 2004.
33. W. Freeman and E. Adelson, “The Design and Use of Steerable Filters,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 13, no. 9, pp. 891-906, Sept. 1991.
34. T. Gao, D. Koller, “Discriminative Learning of Relaxed Hierarchy for Large Scale Visual Recognition”, International Conference on Computer Vision (ICCV),2011.
35. P. Gehler and S. Nowozin, “On feature combination for multiclass object detection”, International Conference on Computer Vision (ICCV), 2009.
36. J. Gemert, J. Geusebroek, C. Veenman, A. Smeulders, “Kernel codebooks for scene categorization”, In ECCV 2008, pages 696–709, 2008.
37. J. Gemert, C. Veenman, A. Smeulders, J. Geusebroek, “Visual Word Ambiguity”, IEEE TPAMI, 99, 2009
38. G. Griffin and P. Perona, “Learning and using taxonomies for fast visual categorization”, Computer Vision and Pattern Rrecognition (CVPR), 2008.

39. G. Griffin, A. Holub, and P. Perona, “Caltech-256 object category dataset”, Technical Report 7694, California Institute of Technology, 2007.
40. H. Gunes, M. Piccardi, “Automatic temporal segment detection and affect recognition from face and body display”, IEEE Trans. Syst. Man Cybern. B Cybern. 39 (1) (2009).
41. H. Gunes and M. Piccardi, “A bimodal face and body gesture database for automatic analysis of human nonverbal affective behavior”, International Conferenece Pattern Recognition, 2006.
42. C. Harris and M. Stephens, “A combined corner and edge detector”, In Proc. of Alvey Vision Conf., pages 147-151, 1988.
43. J. Huang, S. Kumar, M. Mitra, W. Zhu, R. Zabih, “Image Indexing Using Color Correlograms”, CVPR, 1997.
44. A. Joshi, F. Porikli, N. Papanikolopoulos, “Scalable active learning for multi-class image classification”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012.
45. Y. Ke and R. Sukthankar, “PCA-SIFT: A More Distinctive Representation for Local Image Descriptors,” Proc. Conf. Computer Vision and Pattern Recognition, pp. 511-517, 2004.
46. J. Koenderink and A. van Doorn, “Representation of Local Geometry in the Visual System,” Biological Cybernetics, vol. 55, pp. 367-375, 1987.
47. J. Kovac, P. Peer, F. Solina, “Human Skin Colour Clustering for Face Detection”, EUROCON – Computer as a Tool, 2003.
48. A. Krizhevsky, I. Sutskever, G. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks”, NIPS 2012.

49. S. Lazebnik, C. Schmid, J. Ponce, “Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories”, Computer Vision and Pattern Rrecognition (CVPR), 2006.
50. S. Lazebnik, C. Schmid, and J. Ponce, “Sparse Texture Representation Using Affine-Invariant Neighborhoods,” Proc. Conf. Computer Vision and Pattern Recognition, pp. 319-324, 2003.
51. L. Li, L. Fei-Fei, “What, where and who? Classifying events by scene and object recognition”, ICCV 2007.
52. L. Li, R. Socher, and L. Fei-Fei, “Towards Total Scene Understanding: Classification, Annotation and Segmentation in an Automatic Framework”, CVPR, 2009.
53. Z. Li, J. Imai, M. Kaneko, “Robust Face Recognition Using Block-based Bag of Words”, International Conference on Pattern Recognition, 2010.
54. T. Lindeberg, “Scale-space theory: A basic tool for analyzing structures at different scales”, Journal of Applied Statistics, 21(2):224-270, 1994.
55. T. Lindeberg, “Feature detection with automatic scale selection”, International Journal of Computer Vision, 1998.
56. L. Liu, L. Wang, X. Liu, “In Defense of Soft-assignment Coding”, International Conference on Computer Vision (ICCV), 2011.
57. S. Lloyd, "Least squares quantization in PCM". IEEE Transactions on Information Theory 28 (2): 129–137, 1982.
58. D. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints”, IJCV, 2004.

59. M. Marszalek and C. Schmid, “Constructing category hierarchies for visual recognition”, European Conference of Computer Vision (ECCV), 2008.
60. M. Marszalek, C. Schmid, “Spatial Weighting for Bag-of-Features”, CVPR 2006.
61. J. Matas, O. Chum, U. Martin, T. Pajdla, “Robust wide baseline stereo from maximally stable extremal regions”, In Proc. British Machine Vision Conf., volume I, pages 384{393, 2002.
62. S. McCann, D. Lowe, “Local Naïve Bayes Nearest Neighbor for Image Classification”, Computer Vision and Pattern Rrecognition (CVPR), 2012.
63. S. McCann, D. Lowe, “Spatially Local Coding for Object Recognition”, Asian Conference on Computer Vision (ACCV), 2012.
64. K. Mikolajczyk, C. Schmid, A performance evaluation of local descriptors, IEEE Trans. Pattern Anal. Mach. Intell. 27 (10) (2005) 1615–1630.
65. K. Mikolajczyk and C. Schmid, “An affine invariant interest point detector”, In Proc. European Conf. on Computer Vision, volume I, pages 128-142, 2002.
66. K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. IEEE Trans. on Pattern Analysis and Machine Intelligence, 27(10):1615{1630, 2005.
67. M. Muja, D. Lowe, “Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration”, International Conference on Computer vision Theory and Applications (VISAPP), 2009.

68. H. Murase and S. Nayar, “Visual learning and recognition of 3D objects from appearance”, International Journal on Computer Vision, vol. 14, no. 1, pp. 5–24, 1995.
69. D. Nister, H. Stewenius, “Scalable Recognition with a Vocabulary Tree”, Computer Vision and Pattern Rrecognition (CVPR), 2006.
70. E. Nowak, F. Jurie, B. Triggs, “Sampling Strategies for Bag-of-Features Image Classification”, ECCV, 2006.
71. A. Oliva and A. Torralba, “Modeling the shape of the scene: A Holistic Representation of the Spatial Envelope”, IJCV, 2001.
72. E. Parzen, "On Estimation of a Probability Density Function and Mode". The Annals of Mathematical Statistics 33 (3): 1065, 1962.
73. A. Rakotomamonjy, F. Bach, S. Canu, Y. Grandvalet, “More Efficiency in Multiple Kernel Learning”, ICML, 2007
74. M. Rosenblatt, "Remarks on Some Nonparametric Estimates of a Density Function". The Annals of Mathematical Statistics 27 (3): 832. 1956.
75. S. Savarese, J. Winn, A. Criminisi, “Discriminative Object Class Models of Appearance and Shape by Correlatons”, CVPR, 2006.
76. F. Schaffalitzky and A. Zisserman, “Multi-View Matching for Unordered Image Sets,” Proc. Seventh European Conf. Computer Vision, pp. 414-431, 2002.
77. K. Schmidt and J. Cohn, “Human Facial Expressions as Adaptations: Evolutionary Questions in Facial Expression Research”, Yearbook of Physical Anthropology, 2001.
78. C. Shan, S. Gong and P. McOwan, “Beyond facial expressions: learning human emotion from body gestures”, British Machine Vision Conference, 2007.

79. J. Shi and C. Tomasi, "Good Features to Track", CVPR 1994.
80. J. Sivic and A. Zisserman, “Video google: a text retrieval approach to object matching in videos”, ICCV, 2003.
81. J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman, “Discovering objects and their location in images”, In ICCV, volume 1, pages 370–377, 2005.
82. M. Swain and D. Ballard, “Color indexing” International Journal in Computer Vision, vol. 7, no. 1, pp. 11–32, 1991.
83. Y. Tian, L. Cao, Z. Liu, and Z. Zhang, “Hierarchical Filtered Motion for Action Recognition in Crowded Videos”, IEEE Transactions on Systems, Man, and Cybernetics--Part C: Applications and Reviews, 2011.
84. M. A. Turk and A. P. Pentland, “Eigenfaces for face recognition,” in Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 586–591, 1991.
85. T. Tuytelaars, M. Fritz, K. Saenko, T. Darrell, “The NBNN Kernel”, International Conference on Computer Vision (ICCV), 2011.
86. J. Uijlings, A. Smeulders, R. Scha, “Real-time Bag of Words, Approximately”, CIVR, 2009.
87. M. Varma and A. Zisserman, “Classifying Images of Materials: Achieving Viewpoint and Illumination Independence”, ECCV, 2002.
88. M. Varma, D. Ray, “Learning The Discriminative Power-Invariance Trade-Off”, ICCV, 2007.
89. A. Vedaldi, V. Gulshan, M. Varma, A. Zisserman, “Multiple Kernels for Object Detection”, ICCV, 2009.
90. P. Viola, M. Jones, “Rapid Object detection using a boosted cascade of simple features”, CVPR 2001.

91. V. Vural and J. G. Dy. A hierarchical method for multi-class support vector machines. In ICML, 2004
92. J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, Y. Gong, Locality-constrained linear coding for image classification, CVPR 2010.
93. C. Wang, D. Blei and L. Fei-Fei, “Simultaneous Image Classification and Annotation”, CVPR, 2009.
94. Y. Wei, “Research on Facial Expression Recognition and Synthesis”, Master Thesis, 2009, software available at: http://code.google.com/p/asmlibrary
95. J. Weijer, C. Schmid, “Coloring local feature extraction”, Proc. 9th European Conference on Computer Vision (ECCV'06) Springer Lecture Notes in Computer Science 3952: 334-348.
96. J. Willamowski, D. Arregui, G. Csurka, C. R. Dance, and L. Fan, “Categorizing nine visual classes using local appearance descriptors”, In IWLAVS, 2004.
97. J. Winn, A. Criminisi, T. Minka, “Object Categorization by Learned Universal Visual Dictionary”, ICCV 2005.
98. S. Xia, J. Li, L. Xia, and C. Ju. Tree-structured support vector machines for multi-class classification. In ISNN, 2007.
99. J. Xiao, J. Hays, K. Ehinger, A. Oliva, A. Torralba, “SUN Database: Large-Scale Scene Recognition from Abbey to Zoo”, Computer Vision and Pattern Rrecognition (CVPR), 2010.
100. J. Yang, K. Yu, Y. Gong, T. Huang, “Linear Spatial Pyramid Matching Using Sparse Coding for Image Classification”, Computer Vision and Pattern Rrecognition (CVPR), 2009.
101. Y. Yang and S. Newsam, "Bag-Of-Visual-Words and Spatial Extensions for Land-Use Classification," ACM SIGSPATIAL

International Conference on Advances in Geographic Information Systems (ACM GIS), 2010.

102. J. Yang, Y. Li, Y. Tian, L. Duan, W. Gao, “Group-Sensitive Multiple Kernel Learning for Object Categorization”, ICCV 2009.
103. K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. In NIPS, 2009.
104. X. Yuan, W. Lai, T. Mei, X. Hua, X. Wu, and S. Li. Automatic video genre categorization using hierarchical svm. In ICIP, 2006.
105. L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. Eighteenth Annual Conference on Neural Information Processing Systems, (NIPS), 2004.
106. Y. Zhang, Z. Jia, T. Chen, “Image Retrieval with Geometry-Preserving Visual Phrases”, CVPR, 2011.
107. H. Zhang, A. Berg, M. Maire, J. Malik, “SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition”, Computer Vision and Pattern Rrecognition (CVPR), 2006.
108. X. Zhou, B. Bhanu, “Feature fusion of side face and gait for video-based human identification”, Pattern Recognition 41, 2008.
109. A. Zweig and D. Weinshall. Exploiting object hierarchy: Combining models from different category levels. In ICCV, 2007.
110. YouTube, 23 Apr, 2013 <http://www.youtube.com/yt/press/statistics.html>.
111. Money Bag. Digital Image. Rambling of an English Teacher. Web. 1 May 2013. <http://andrewleggett.wordpress.com/2011/11/24/the-word-bag-strategy/>

112. Van. Digital Image. 2013 Toyota Sienna LE for Sale. Web. 1 May 2013. <http://www.usedcarsgroup.com/wappingersfalls-ny/2013-toyota-sienna-5tdkk3dc1ds360034.html>.

113. Dolphin. Digital Image. Animal Rights Group Rescues Dolphin Off Cape Cod. Web. 1 May 2013. <http://www.opposingviews.com/i/animal-rights-group-rescues-dolphins-off-cape-cod>

114. Dolphin. Digital Image. Mysterious Dolphin Deaths Continue in Gulf of Mexico. Web. 1 May 2013. <http://www.banderasnews.com/1301/nb-mysteriousdolphindeaths.htm>

115. Tiger. Digital Image. Wallpapers for resolution of 1600x1200. Web. 1 May 2013. <http://im05.coolwallpapers.org/resolution/1600x1200/162>

116. Watermelon. Digital Image. Geometry Shapes with an example. Web. 1 May 2013. <http://quizlet.com/18009319/geometry-shapes-with-an-example-flash-cards/>

117. Orange. Digital Image. Strategy. Web. 1 May 2013. <http://stratkomuncut.com/category/strategy/>.

118. Strawberry. Digital Image. Marietta College. Web. 1 May 2013. <https://secure.www.alumniconnections.com/olc/pub/MRO/event/showEventForm.jsp?form_id=144710>.

119. Pink lily flower. Digital Image. Full of Life. Web. 1 May 2013. <http://banishloneliness.org/2012/09/27/non-attachment-v-loneliness/>.

120. Rose. Digital Image. Flickr. Web. 1 May 2013. <http://www.flickr.com/photos/niceflowers48/5021857417/in/photostream/>.

121. Lion. Digital Image. African Lion King. Web. 1 May 2013. <http://www.wallpapersshop.net/wallpaper/african-lion-king/>.

122. Blue whale. Digital Image. Reflection at Point Lookout. Web. 1 May 2013. <http://www.reflectionsatpointlookout.com/?_escaped_fragment_=gallery>.

123. Office partition. Digital Image. Singapore Office Partitions. Web. 1 May 2013. <http://detail.en.china.cn/provide/1001809044.html>.

124. Office. Digital Image. Portillo Cleaning Services. Web. 1 May 2013. <http://portillocleaninglck.com/services.html>.

125. Office. Digital Image. Prestigious Cambridge Location. Web. 1 May 2013. <http://geekoffices.com/properties/>.

126. Office furniture. Digital Image. World Class Wooden Furniture. Web. 1 May 2013. <http://wooden-furnitures.co.in/>.

127. G. Kim, C. Faloutsos, M. Hebert, "Unsupervised Modeling of Object Categories Using Link Analysis Techniques", CVPR, 2008.

128. T. Dean, M. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, J. Yagnik, "Fast, Accurate Detection of 100,000 Object Classes on a Single Machine", CVPR, 2013.

My Publication List

Journal and Book Chapter

1. H. Gunes, C. Shan, S. Chen, and Y. Tian, "Bodily Expression for Automatic Affect Recognition", In Advances in Emotion Recognition, A. Konar, A. Chakraborty (Eds.), Wiley-Blackwell, 2011.
2. Y. Tian and S. Chen, “Understanding Effects of Image Resolution for Facial Expression Analysis”, Journal of Computer Vision and Image Processing, Vol. 2, No. 1, March 2012.
3. S. Chen and Y. Tian, “Describing Visual Scene through EigenMaps”, Journal of Computer Vision and Image Processing, Vol. 2, No. 1, March 2012.
4. S. Chen, Y. Tian, Q. Liu, and D. Metaxas, Recognizing Expressions from Face and Body Gesture by Temporal Normalized Motion and Appearance Features, Image and Vision Computing, Volume 31 Issue 2, February, Pages 175-185, 2013, DOI: 10.1016/j.imavis.2012.06.014

Conference

5. S. Chen, Y. Tian. Evaluating Effectiveness of Latent Dirichlet Allocation Model for Scene Classification. IEEE Wireless and Optical Communications Conference (WOCC). 2011.

6. S. Chen, Y. Tian, Q. Liu and D. Metaxas. Segment and Recognize Expression Phase by Fusion of Motion Area and Neutral Divergence Features. IEEE Int'l Conf. on Automatic Face and Gesture Recognition (AFGR). 2011.
7. S. Chen, Y. Tian, Q. Liu, D. Metaxas. Recognizing Expressions from Face and Body Gesture by Temporal Normalized Motion and Appearance Features. IEEE Int'l Conf. Computer Vision and Pattern Recognition workshop for Human Communicative Behavior Analysis (CVPR4HB). 2011.
8. S. Chen, Y. Tian, and W. S. Weiss, "Detecting Human Activities in the Arctic Ocean by Constructing and Analyzing Super-Resolution Images from MODIS Data", Imaging and Geospatial Technologies - Into the Future, ASPRS 2012 Annual Conference, Sacramento, California USA, March 19-23, 2012.
9. S. Chen, D. M. Quintian and Y. Tian, "Towards a Visual Speech Learning System for the Deaf by Matching Dynamic Lip Shapes", International Conference on Computers Helping People with Special Needs (ICCHP), 2012.
10. S. Chen and Y. Tian, Margin-Constrained Multiple Kernel Learning Based Multi-Modal Fusion for Affect Recognition, The 2nd Int'l Workshop on Emotion Representation, Analysis and Synthesis in Continuous Time and Space (EmoSPACE) 2013.