ABSTRACT

Title of dissertation: NON-LINEAR AND SPARSE REPRESENTATIONS FOR MULTI-MODAL RECOGNITION

Hien Van Nguyen, Doctor of Philosophy, 2013

Dissertation directed by: Professor Rama Chellappa
Department of Electrical and Computer Engineering

In the first part of this dissertation, we address the problem of representing 2D and 3D shapes. In particular, we introduce a novel implicit shape representation based on Support Vector Machine (SVM) theory. Each shape is represented by an analytic decision function obtained by training an SVM, with a Radial Basis Function (RBF) kernel, so that the interior shape points are given higher values. This empowers support vector shape (SVS) with multifold advantages. First, the representation uses a sparse subset of feature points determined by the support vectors, which significantly improves the discriminative power against noise, fragmentation and other artifacts that often come with the data. Second, the use of the RBF kernel provides scale, rotation, and translation invariant features, and allows a shape to be represented accurately regardless of its complexity. Finally, the decision function can be used to select reliable feature points. These features are described using gradients computed from highly consistent decision functions instead of conventional edges. Our experiments on 2D and 3D shapes demonstrate promising results.

The availability of inexpensive 3D sensors like Kinect necessitates the design of

new representation for this type of data. We present a 3D feature descriptor that represents local topologies within a set of folded concentric rings by distances from local points to a projection plane. This feature, called as Concentric Ring Signature (CORS), possesses similar computational advantages to point signatures yet provides more accurate matches. CORS produces compact and discriminative descriptors, which makes it more robust to noise and occlusions.

It is also well-known to computer vision researchers that there is no universal representation that is optimal for all types of data or tasks. Sparsity has proved to be a good criterion for working with natural images. This motivates us to develop efficient sparse and non-linear learning techniques for automatically extracting useful information from visual data. Specifically, we present dictionary learning methods for sparse and redundant representations in a high-dimensional feature space. Using the kernel method, we describe how the well-known dictionary learning approaches such as the method of optimal directions and KSVD can be made non-linear. We analyse their kernel constructions and demonstrate their effectiveness through several experiments on classification problems. It is shown that non-linear dictionary learning approaches can provide significantly better discrimination compared to their linear counterparts and kernel PCA, especially when the data is corrupted by different types of degradations.

Visual descriptors are often high dimensional. This results in high computational complexity for sparse learning algorithms. Motivated by this observation, we introduce a novel framework, called *sparse embedding* (SE), for simultaneous dimensionality reduction and dictionary learning. We formulate an optimization problem for learning a transformation from the original signal domain to a lower-dimensional one in a way that

preserves the sparse structure of data. We propose an efficient optimization algorithm and present its non-linear extension based on the kernel methods. One of the key features of our method is that it is computationally efficient as the learning is done in the lower-dimensional space and it discards the irrelevant part of the signal that derails the dictionary learning process. Various experiments show that our method is able to capture the meaningful structure of data and can perform significantly better than many competitive algorithms on signal recovery and object classification tasks.

In many practical applications, we are often confronted with the situation where the data that we use to train our models are different from that presented during the testing. In the final part of this dissertation, we present a novel framework for *domain adaptation using a sparse and hierarchical network* (DASH-N), which makes use of the old data to improve the performance of a system operating on a new domain. Our network jointly learns a hierarchy of features together with transformations that rectify the mismatch between different domains. The building block of DASH-N is the latent sparse representation. It employs a dimensionality reduction step that can prevent the data dimension from increasing too fast as traversing deeper into the hierarchy. Experimental results show that our method consistently outperforms the current state-of-the-art by a significant margin. Moreover, we found that a multi-layer DASH-N has an edge over the single-layer DASH-N.

NON-LINEAR AND SPARSE REPRESENTATIONS FOR OBJECT RECOGNITION

by

Hien Van Nguyen

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2013

Advisory Committee:
Professor Rama Chellappa (Chair/Advisor)
Professor Larry Davis
Professor Min Wu
Doctor Jonathon Phillips
Associate Professor Wojciech Czaja (Dean's Representative)

DEDICATION

To my parents, Ut Hien and Cam Ha.

Acknowledgments

First and foremost I would like to express my deepest gratitude to my advisor, Professor Rama Chellappa for his extraordinary supervision. He has always been there whenever I needed advice and help and there hasn't been an occasion when I've come to his office and he didn't give me time. His intellectual charisma is a tremendous source of inspiration and his witty sense of humor teaches me to look on the bright side of life. Professor Rama's approach to mentoring is more like being a beacon in which he encouraged me to find my own way but provided wise counsel whenever I was at lost. He inspired me and gave enough freedom to independently explore my own research interests. There are many more good deeds that he has done for me and my family, for which I want to thank him in heart and not in word.

I am also extremely grateful to Dr. Fatih Porikli and Dr. Oncel Tuzel for their hospitality when I was doing my internships at the Mitsubishi Electric Research Laboratories. Dr. Fatih's whole-hearted support have a tremendous positive impact on my current research and future career. I am very thankful to Dr. Nasser Nasrabadi for teaching me to learn from the basics. Dr. Vishal Patel deserves my special appreciation for his help in editing papers, discussing new ideas and sharing with me his research experience.

I am grateful to my dissertation committee members, Professor Larry Davis, Professor Min Wu, Professor Wojciech Czaja and Dr. Jonathon Phillips for spending their valuable time on reading my thesis and attending my dissertation defence. I would also want to acknowledge the help from administrative staffs including Ms. Janice Perrone, Dr. Tracy Chung, Ms. Melanie Prange, Ms. Maria Hoo, Ms. Meg Richmond for helping

me to smoothly get through all the complicated paper work.

The bright fellow graduate students at the Center for Automation Research have enriched my graduate life in many ways and I am truly grateful to all of them, Jaishanker Pillai, Nazare Batool, Tho Ho, Ashish Srivastava , Sumit Shekhar, Yi-Chen Chen, Raviteja Vemulapalli, Garrett Warnell, Wu Tao, Sima Taheri, Qiang Qiu, Jie Ni, Aswin Sankaranarayanan, Pavan Turaga, Kaushik Mitra, Dikpal Reddy, Ming-Yu Liu, Ruonan Li, Raghuram Gopalan, Mahesh Ramachandran, and other colleagues. Without these companions, the journey would not be as fulfilling and enjoyable.

I have been extremely fortunate to know many wonderful Vietnamese friends in Maryland who have been a significant part of my life in the last five years and made me feel so home. Special gratitude is due to Dr. Ha Minh Nguyen who frequently drove me to grocery stores for the first two years. I am immensely grateful to Mr. Ba and his wife, my grandfather, my ants, Mrs. Hang, Mrs. Diep for helping my parents when I am away from home. I can't be more grateful to Dr. Amador and Ms. Ling for their contributions in improving my parents' well-being.

I owe my deepest gratitude to my family (father, mother, Ut Hien, Cam Ha) who have endured a difficult time for me to accomplish my graduate program. Words just can't do enough justice for what they have done for me.

It is impossible to remember all, and I apologize to those I've inadvertently left out. Finally, Thank you all!

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

We live in a complex visual world. A typical visual scene is composed of a large number of image elements that vary in luminance, color, shape, and motion. Unfortunately, the information of interests is often highly non-linear. High-level concepts such as car or human form very convoluted regions within the image space. Several theorists have proposed that natural images lie along a continuous curved 'manifold' embedding in the high-dimensional state space of images. The manifold represents the smooth changes that follow from the transformations, such as translations and rotations that are likely to occur in natural scenes. For this reason, finding a good way to represent images or videos is not a trivial task.

Good representations of data are crucial to the successes of many computer vision systems. However, there is no universal representation that is optimal for all types of data. As a result, significant research efforts have been spent on designing good representations that are appropriate for specific tasks such as detection and recognition. The availability of new sensor modalities, such as 3D and hyperspectral sensors capable of collecting high-dimensional data in real-time, necessitates the design of new representations and processing techniques. In this dissertation, we address the problems of designing representations for 2D and 3D shapes. In addition, we propose novel learning techniques capable of learning non-linear and sparse representations directly from a set of images.

2D shapes are often represented by a set of features that are computed based on boundaries or edges points. The most popular features include shape context [1] and inner distance shape context [2]. While these approaches are widely used to design features for 2D shape matching, the have some disadvantages. For example, the constructions of these features are sensitive to various effects such as missing pixels, internal discontinuities, and branching offshoots. As opposed to the conventional approach, in chapter 2 we propose to represent a shape as an analytic function similar to the decision function of the support vector machine [3]. This representation enables the extraction of a novel set of invariant features that is robust against the above-referenced degradations. These features are extracted from highly consistent decision function's gradients instead of edges as in conventional approaches. Our experiments demonstrate promising results of shape matching and object recognition on several datasets including MPEG7.

Range images are more and more popular due to the availability of inexpensive sensors like Kinect. However, it is not straight-forward to extend the traditional 2D features into this sensor modality. The reasons include low resolution, irregular sampling density, and differences in underlying physics. Common 3D feature extractors such as spin image [4], 3D shape context [5], and spherical harmonic [6] are often computationally expensive. These drawbacks motivate the design of new features for this type of data. In this dissertation, we propose a novel feature for 2.5D range images called the Concentric Ring Signature (CORS) [7]. Our feature is patch-based and thus is a natural choice for range images. Unlike volume-based features (e.g. spin image and spherical harmonics) that require the normalization of the sampling density at every 3D voxel, CORS is more compact and robust against irregular sampling effects. CORS achieves state-of-the-art

recognition results on several difficult datasets including TOSCA and MIAN which consist of multiple objects and scenes that come with various non-rigid 3D transformations.

The process of designing representations for visual data is tedious and time consuming. It requires a deep understanding and a careful examination of the underlying physics that governs the generation of data. Recently, there is an explosion of interest in modelling data using sparse representation. It has been shown that sparse methods can automatically extract useful information such as edges, corners as well as more complex structures from natural images. The set of techniques learned sparse representations directly from the data is called *dictionary learning*. The dictionaries designed by MOD [8] and KSVD [9] are based on a linear representation of the data. However, linear representations are almost always inadequate for representing non-linear structures of the data which arise in many practical applications. For example, images of a human face are subjected to various sources of non-linear transformations such as rotation and deformation.

In addition, many types of descriptors in computer vision have intrinsic non-linear similarity measure functions. The most popular ones include the spatial pyramid descriptor [10] which uses a pyramid match kernel, and the region covariance descriptor [11] which uses a Riemannian metric as the similarity measure between two descriptors. Both of these distance measures are highly non-linear. The linear model used by the traditional dictionary learning methods, e.g. MOD and KSVD, inevitably leads to poor performances for many datasets, e.g., object classification of Caltech-101 [12] dataset, even when discriminant power is taken into account during the training [13].

Motivated by the drawbacks of the current methods and the needs of many practical applications, in chapter 3 we propose kernel dictionaries which are basically dictionaries

in high dimensional feature spaces. Our dictionary learning methods yield representations that are more compact than kernel principal component analysis (KPCA) [14] and are able to handle the non-linearity better their linear counterparts.

To enhance the efficacy and adaptability of kernel dictionary, we introduce a novel framework, called *sparse embedding* (SE), for simultaneous dimensionality reduction and dictionary learning in chapter 4. We formulate an optimization problem for learning a transformation from the original signal domain to a lower-dimensional one in a way that preserves the sparse structure of data. We propose an efficient optimization algorithm and present its non-linear extension based on the kernel methods. One of the key features of our method is that it is computationally efficient as the learning is done in the lower-dimensional space and it discards the irrelevant part of the signal that derails the dictionary learning process. Various experiments show that our method is able to capture the meaningful structure of data and can perform significantly better than many competitive algorithms on signal recovery and object classification tasks.

In many practical computer vision applications, we are often confronted with the situation where the data that we use to train our algorithm have different distribution or representation from that presented during the testing. The ubiquity of this problem is well-known to computer vision researchers. For instance, indoor images are quite different from outdoor images, just as videos captured with a high definition camera are from those collected with a webcam. This detrimental effect is often a dominant factor accounting for the poor performances of many computer vision algorithms. As an example of the effect of distribution mismatch, Ben-David *et al.* [15] show that, under certain assumption, the bound on the test error linearly increases with the ℓ_1 divergence between the training

and testing distributions. Even worse, data from the test domain are often scarce and expensive to obtain. This makes it impractical to re-train an algorithm from the scratch since a learning algorithm would generalize poorly when insufficient amount of data is presented [16].

In order to address this issue, chapter 5 proposes a novel approach for domain adaptation, based on the sparse embedding framework, that possesses the following advantages. First, the adaptation is performed on multiple levels of the feature hierarchy in order to maximize the knowledge transfer. The hierarchical structure allows the transfer of useful information that might not be well captured by existing domain adaptation techniques. Second, domain adaptation is done jointly with feature learning. Our method learns a hierarchy of sparse codes and uses them to describe a visual object instead of relying on any low-level feature. Finally, unlike existing hierarchical networks, our network is more computationally efficient with a mechanism to prevent the data dimension from increasing too fast as the number of layer increases. We provide extensive experiments to show that our approach outperforms the current state-of-the-art by a large margin. This is interesting since our training is entirely generative followed by a linear SVM while several other methods employ discriminative training together with non-linear kernels.

Chapter 2

Shape Representations

Shape of an object represents the geometrical information that is independent of the transformational (scaling, rotation, articulation, etc) effects. Understanding shape is essential in many computer vision applications from recognition of people and their actions in video surveillance to design and inspection in industrial manufacturing [17, 18].

Recent psychophysical findings [19] suggest that the perceptual representation of a shape is primarily based on qualitative properties whose topological structures remain relatively stable over transformational conditions. Other empirical studies [20, 21] have shown that the neural processing of shape in the brain is broadly distributed throughout the ventral (*what*) pathway that is involved in object recognition, and the dorsal (*where*) pathway that is involved in spatial localization. In other words, an adequate mathematical representation of shape needs to be invariant to viewpoint changes and articulated object motion, and discriminative enough to enable detection and classification.

2.1 Related Work On 2D and 3D Shape Representations

Two main approaches dominate previous work on shape representation: global approaches model an object as a whole segment, while part-based approaches advocate segmentation of shape into constituent regions. The drawback of a purely global approach is the exclusion of articulation and the sensitivity to occlusion. The drawback of a purely

part-based approach is that a consistent partitioning is generally not possible in the face of numerous combinations of possibilities and object shape variations. Besides, segmentation itself is ill-posed, except under controlled environments or in restricted application domains.

2.1.1 Global Approaches

Prominent global shape representations include variational [22] and level set approaches [23, 24]. These approaches have been applied for scene segmentation [25] and tracking [26, 27]. Brookstein initiated the use of thin-plate splines [28] to analyze deformable shape, which were then improved by [29, 30]. These methods are landmark-based and suffer from inconsistency in landmark selection. [31] fits a parametric model to a shape using mixture of Gaussian densities. This method requires a clustering process to estimate cluster centers and therefore has the same drawback with other landmark-based approaches. [32] assigns every internal point of the silhouette a value proportional to mean time of random walk from the point to the boundary. This can be achieved by solving a Poisson equation. Other popular methods are statistical moments [33], eigenshapes [34], curvature scale space [35], elastic matching [36], parametric curves (polylines), image signatures, etc. Zernike moments are a class of orthogonal moments that are invariant to rotation and translation. Eigenshapes decompose a distance matrix of boundary points into an ordered set of eigenvectors and finds the modes of these eigenvectors. Elastic matching evaluates the similarity as a sum of local deformations needed to change one shape into another. Scale space representation successively smooths contour while

decreasing the number of curvature zero crossings. In general, global models need additional mechanisms to compensate for articulated motion and non-rigid deformation.

2.1.2 Part-Based Approaches

In comparison, part-based approaches describe shapes in terms of their part structure. Parts are defined to be nearly convex shapes separated from the rest of the object at concavity extrema [37, 38, 39, 40]. It is possible to build a discriminative classifier from a collection of parts [41] or a bag-of-feature to solve correspondence [42]. These methods often require a multitude of training samples, prior knowledge on the number of parts, and a precise formulation of articulation. Other part-based methods try to learn the part structure by considering the shape interior. For instance, shock graphs [43] are defined as the cyclic tree of singularities of a curve evolution. The inner distance [44], geodesic distance [45] and random walk [46] also consider the interior of the shape to build descriptors. Given a pair of points, the inner distance is determined by finding their closest points on the shape skeleton, then measuring the distance along the skeleton. The geodesic distance is the length of the shortest path on the surface. While shock graphs benefit from the skeletons robustness to articulation they suffer from boundary noise. The inner and geodesic distances are robust to disturbances along boundaries, yet they are highly sensitive to occlusions and fragmentations. Recently, [47] proposed Gibbs Random Fields to model shapes as spatial compositions of simple parts.

Pioneering work on the spin image [48] describes the relative spatial distribution of shape points around a set of feature points. It considers a cylindrical support region

and accumulates a histogram of points. The shape context [49, 50] is similar to the spin image except that the support region is a sphere. Since both generate sparse matrices, the distance computation is sensitive to the shape structure. In [51] each shape is indexed based on a variety of features such as inner distance, Euclidean distance, contour distance, etc. that characterize pairwise geometric relationships between interest points on the shape. Shapes in the database are ordered according to their similarity with the query shape and similar shapes are retrieved using a scheme which does not involve shape-wise alignment.

2.2 Support Vector Shape Representation

The above referenced methods provide satisfactory results under ideal conditions with strong priors and clean segmentation masks. Their representation capacity substantially degrades when the shape boundary is noisy (part-based methods, shock graphs), shape has internal crevices, branching offshoots and excessive discontinuities (inner-distance, spin images, shape context), and non-conforming articulations (global models). Besides, they would not necessarily extend to higher dimensions or generalize over most shape classes.

Quite different from existing approaches, we propose a novel implicit shape representation based on SVM theory. Each shape is represented by a classification decision function obtained by training an SVM with interior and exterior shape points providing positive and negative training samples, respectively. The RBF kernel is used with SVM to make our representation rotation and translation invariant. The decision boundary is

a hypersurface on the high-dimensional feature space that separates the positive labelled points, *shape*, from the negative labelled points, *its surroundings*. Our shape representation is not just the shape boundary or the decision function boundary but the function itself. Instead of using the edge or surface gradients on a discrete grid, we use the *gradient* of the classification decision function, which is an analytic function that is defined everywhere in the data space. Furthermore, the use of the RBF kernel enables SVM to model any complicated shape due to the infinite dimensional nature of the associated Hilbert space. Several other advantages are explained in the following section.

To summarize the main contributions, this chapter

1. proposes a novel method to represent $2D$ and $3D$ shapes using support vector classifiers,

2. provides an in depth theoretical analysis for a better understanding of this representation, and

3. presents experimental results on several datasets to evaluate its performance on challenging datasets.

2.2.1 Formulation of Support Vector Shape

We define the SVS representation to be the decision function of a classifier. The name SVS comes from the fact that the decision function is parametrized by a set of support vectors, learned from an SVM. Through out the chapter, the terms "SVS" and "decision function" are used interchangeably. This representation facilitates the extraction of feature points that correspond to salient components of the shape, which are then

described using local statistics of the decision function around each point.

To the best of our knowledge, this is the first effort that considers the shape representation as a classification problem. This classifier-based representation offers several advantages. First, it is general enough to be applied to 2D shapes and 3D volumes. Second, the classification function depends only on a sparse subset of key points, which makes the representation robust against noise, missing data, and other artifacts including interior fragmentations. Finally, the descriptors are also more discriminative and stable against transformation and disturbances than edge-based descriptors [48, 52, 44] since they are extracted from the dense gradient field of the decision function but not from the original data.

Basically, SVS involves the following tasks:

1. Learn a decision function from a given shape,
2. Select feature points using the gradient, i.e. the first derivative, of the decision function. (sec. 2.2.2),
3. Compute local descriptors (sec. 2.2.3).

which are explained in the following sections.

As it will be clear, SVS enables selecting a small set of salient features for shape matching and retrieval. These features are described using local statistics of the decision function around each point. For instance, a 2D variant of SVS features picks the high gradient points of the decision function as feature points and uses the local histogram of oriented gradients computed on the decision function as the descriptors.

2.2.1.1 Learning a Decision Function:

Let $S = \{\mathbf{x_n}\}_{\mathbf{n=1}}^{\mathbf{N}}$ be a set of points representing a shape[1] and $\bar{S}$ be the set of points not in S, i.e outside. We wish to learn a classifier:

$$f(\mathbf{x}) = \begin{cases} \geq 0, & \mathbf{x} \in \mathbf{S} \\ \\ < 0, & \mathbf{x} \in \bar{\mathbf{S}} \end{cases} \tag{2.1}$$

Two classifiers, $f(\mathbf{x})$ and $g(\mathbf{x})$, are said to represent the same shape if

$$\text{sign}[f(\mathbf{x})] = \text{sign}[\mathbf{g}(\mathbf{x})] = \begin{cases} \geq 0, & \mathbf{x} \in \mathbf{S} \\ \\ < 0, & \mathbf{x} \in \bar{\mathbf{S}} \end{cases} \tag{2.2}$$

The following theorem states that if the binary shapes generated by taking the zero-level crossings of these classifiers are equivalent (as in 2.2) to each other then the decision functions are equivalent with a constant factor:

Theorem 2.1 (Curtis '87 [53])**.** *Let $f(\mathbf{x})$ and $g(\mathbf{x})$ be real, 2D, band-limited and irreducible functions. If $f(\mathbf{x})$ and $g(\mathbf{x})$ take on both positive and negative values in a closed bounded region $D \subset \mathbb{R}^2$, and sign$[f(\mathbf{x})] =$ sign$[\mathbf{g}(\mathbf{x})]$ for all $\mathbf{x}$ in D, then $f(\mathbf{x}) = \kappa\mathbf{g}(\mathbf{x})$, where κ is a real positive constant.*

Here, *irreducible function* means that its Fourier transform is not factorable. The band-limited condition implies the Fourier transform will exist and it will have a compact region of support for finite-energy signals (interested readers are referred to [54] for further discussions on band-limitedness condition).

[1] We denote vectors in bold letters

The above theorem suggests that the gradient orientation is equivalent for such two functions satisfying the above constraints. In other words, two decision functions representing the same shape will be consistent and exhibit invariance properties in terms of their gradient orientations if these functions are real, 2D, band-limited, irreducible, and have almost identical responses.

One such example is the radial basic function (RBF) an SVM [55] operates on. It is real and band-limited: it has the form of the sum of finite number of weighted exponential, thus, its Fourier transform is a sum of weighted exponentials, which has finite-energy and a compact region of support especially when insignificant coefficients of the radial kernels are disregarded. A Theorem for the irreducibility of the general class of functions including the RBF kernel is given below with a proof in Sec. 2.2.5.

Theorem 2.2. *A function of the following form:*

$$f(\mathbf{x}) = \sum_{\mathbf{i=1}}^{\mathbf{m}} \alpha_{\mathbf{i}} \mathbf{g}(\mathbf{x} - \mathbf{x}_{\mathbf{i}}^{*}), \quad \mathbf{m} \geq \mathbf{5} \tag{2.3}$$

is irreducible if the Fourier transform of $g(\mathbf{x})$ does not have any zero (either real or complex).

For an ideal shape classifier as defined in (2.1), the decision function is positive for the shape regions, zero on the shape boundary $\mathcal{B}$, and negative otherwise. In other words, the gradient of the decision function along the tangent space of a point on the boundary $\mathcal{B}$ is zero (see Figure 2.1). This means that the gradient of the decision function must be perpendicular to the tangent plane, thus, the gradient itself coincides with the normal vector of the shape.

Proposition 2.1. *Gradient $\nabla f(x)$ at a shape boundary point is a close approximation for the normal vector at that point.*

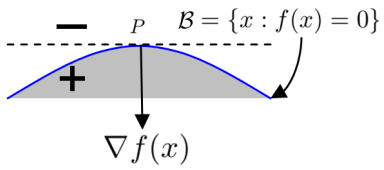


Figure 2.1: Illustration of SVS decision boundary. The dotted line is the tangent space.

This property is very desirable for SVS since the normal direction is an essential input for the construction of many descriptors [49, 56, 48]. It is especially useful for computing descriptors from 3D point clouds where the knowledge about points orientations is missing.

As moving from the boundary $\mathcal{B}$ to the interior of shape, the gradient $\nabla f(x)$ are an indication of the combined effects of local edge segments. This effectively fuses local topologies and enhances the discriminative power of local descriptors (Section 2.2.3).

We are interested in representing a shape by a continuous function that has the mathematical form of (2.3) and at the same time satisfies (2.1) as much as possible. Notice that the class of functions in (2.3) contains the decision function of SVM with the RBF kernel [57, 58]. Therefore, we employ SVM to learn our parametric shape representation. To our advantage, the SVM decision function is analytic, i.e., it is in the form of weighted sum of kernel responses, thus its gradient can be efficiently computed at a point in the space. Furthermore, the RBF kernel functions can effectively map data $\mathbf{x_n}$ to an infinite-dimensional space where S and $\bar{S}$ would be linearly separable, thus even for intricate boundaries a high classification performance, and accurate shape representation,

is guaranteed.

SVMs construct a hyperplane in a high (or infinite) dimensional feature space between a set of labelled input vectors $\mathbf{x}$ that can be either $+1$ for shape pixels, or -1 for non-shape pixels by definition for binary SVMs. The decision boundary is defined in terms of a typically small subset of training examples, called as support vectors, that result in a maximum margin separation between these classes. The decision function of SVM is given as

$$f(\mathbf{x}) = \sum_{\mathbf{i=1}}^{\mathbf{m}} \alpha_{\mathbf{i}} [\mathbf{\Phi}(\mathbf{x}).\mathbf{\Phi}(\mathbf{x_i^*})] \tag{2.4}$$

where $\mathbf{x_i^*}$ are support vectors, α_i are the corresponding weights of the support vectors, m is the number of non-zero support vectors, and Φ is a mapping function in some dot product space $\mathcal{H}$. By defining a similarity measure k in $\mathcal{H}$ as

$$k(\mathbf{x}, \mathbf{x_i^*}) = \mathbf{\Phi}(\mathbf{x}).\mathbf{\Phi}(\mathbf{x_i^*}), \tag{2.5}$$

every dot product in the decision function is replaced by a kernel function. This allows efficient computations without having to venture into the high dimensional feature space $\mathcal{H}$. The transformation may be non-linear; thus though the classifier is a hyperplane in $\mathcal{H}$, it may be non-linear in the original input space. If the kernel used is a Gaussian, the corresponding feature space is a Hilbert space of infinite dimension

$$\Phi(\mathbf{x}).\mathbf{\Phi}(\mathbf{x_i^*}) = \exp(-\gamma ||\mathbf{x} - \mathbf{x_i^*}||^{\mathbf{2}}) \tag{2.6}$$

where γ stands for the Gaussian kernel width. By using the RBF, it is always possible to find a decision function that perfectly represents a shape. Such a decision function has the form

$$f(\mathbf{x}) = \sum_{\mathbf{i=1}}^{\mathbf{m}} \alpha_{\mathbf{i}} \exp(-\gamma ||\mathbf{x} - \mathbf{x_i^*}||^{\mathbf{2}}) \tag{2.7}$$

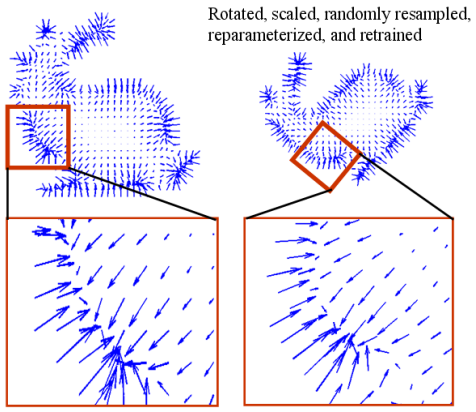


Figure 2.2: Orientation of $\nabla f(\mathbf{x})$ remains stable even if the shape transforms and different set of parameters and points are used to train the SVS on the right.

and the final classification is made by $l(\mathbf{x}) = \text{sign}[\mathbf{f}(\mathbf{x})]$. It is worth noting that the set of support vectors is usually small in comparison with the entire training set.

Since the decision function in (2.7) only depends on the distance between points, the SVS representation is robust to translation and rotation[2]. This can be observed from the Figure 2.2 which shows that the gradient directions, computed from SVS decision functions of transformed versions of the same shape, are almost same.

Furthermore, the γ multiplier in the kernel function is inversely proportional to the squared scale change. For a given shape with unknown scale change from its original, the mean of pairwise distances between all points can be used to normalize the shape. This scale normalization technique has been used in shape context [50] and has been proven to be effective. In addition, our analysis demonstrates that small variations of γ would not perturb the best possible classification accuracy.

For training, we select a random subset of internal points to be positive training

[2]Note that, for linear and polynomial kernels such an invariance does not apply as they impose inner products of point coordinates

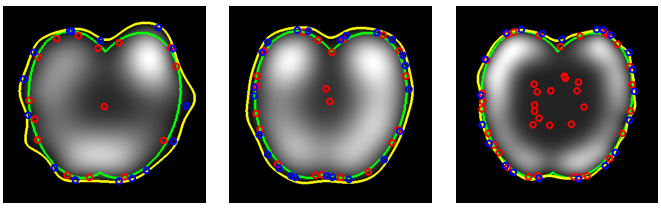


Figure 2.3: Support vectors (red circles $\alpha_i > 0$, blue circles $\alpha_i \leq 0$), the decision function boundaries $f(\mathbf{x}) = \mathbf{0}$ (yellow), and the original shape (green) for support vector numbers 25, 44, and 61. Classifier accuracies are 93%, 97.2%, and 99.8%, respectively. Original shape contains $\sim$54K points ($\sim$1K points on the boundary), yet only a small fraction (e.g. 61) of points are needed to encode the shape.

samples from a given shape. Another random subset of points surrounding the shape is chosen to be negative samples. Random selection is preferred just for computational efficiency. The input to the classifier is the coordinates and the corresponding inside/outside labels of the training points.

Figure 2.3 shows support vectors and decision boundaries for a sample shape. It can be noticed that support vectors are not required to lie on shape edges. This is because the kernel mapping data to the high dimensional space, where non-edge points might happen to lie on the decision boundary of the learning algorithms. The number of support vectors typically varies from 0.1% to 3% of the total number of points.

2.2.1.2 $\nu-$SVM and One-Class SVM:

We employ $\nu-$SVM [57] and one-class SVM [58] for learning the decision function as its parameters have a natural interpretation for shapes. Given a set of labelled samples

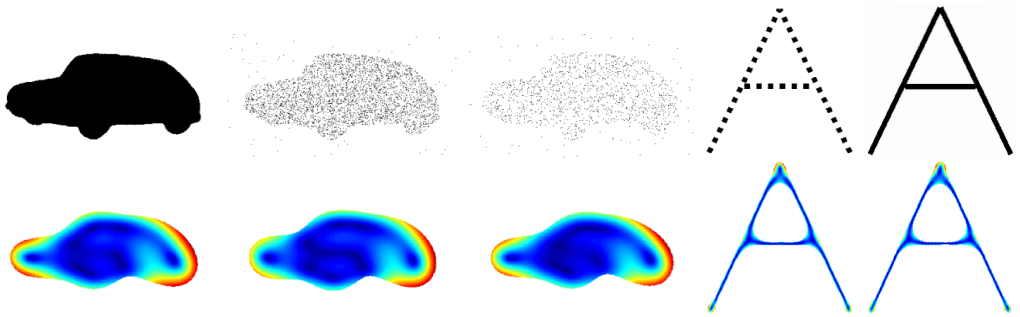


Figure 2.4: Comparison of the decision function responses for noise-free and noisy shapes. In the first experiment, a car shape is distorted by randomly removing 20% and 50% of points and adding speckle noise to the background. In the second experiment, the letter *A* is interleaved with blank spaces. The differences of decision function for both cases are hardly noticeable. They are both trained using one-class SVM.

(x_i, y_i), the learning problem of $\nu-$SVM is formulated as the minimization of

$$\underset{\mathbf{w},\boldsymbol{\xi},\rho}{\arg\min} \ \frac{1}{2}||\mathbf{w}||^2 - \nu\rho + \frac{1}{l}\sum_{i=1}^{l}\xi_i \tag{2.8}$$

subject to:

$$y_i.(\mathbf{w}.\Phi(\mathbf{x_i}) + \mathbf{b}) \geq \rho - \xi_\mathbf{i}, \tag{2.9}$$

$$\xi_i \geq 0, \quad \rho \geq 0 \tag{2.10}$$

where Φ is a function that maps the input data to some Hilbert space. The above optimization tries to correctly classify as many data as possible by penalizing the misclassified samples through variable ξ_i. At the same time, the minimization of $||\mathbf{w}||$ keeps the model as simple as possible, and the margin is made as large as possible through maximization of variable ρ.

The trade-off between the model complexity and the training error is controlled by parameter $\nu \in [0, 1]$. It is also the lower and upper bound on the number of examples

that are support vectors and lie on the wrong side of the hyperplane, respectively. The larger (smaller) we make ν the more (less) points are allowed to lie inside the margin, which gives coarser (finer) shape representations. It is possible to use a small value which results in a larger number of support vectors to allow accurate representation of complex shapes, while smaller numbers of support vectors enhance robustness against corrupted training data.

The formulation and parameters of one-class SVM is similar with $\nu-$SVM. The only difference is that one-class SVM allows learning from data with positive samples only. It separates data from the origin in the feature space instead of separating positive samples from negative samples. This becomes extremely useful to deal with missing data due to occlusion and camera sampling error.

The selection of parameters for SVM algorithms (e.g. kernel width γ of (2.6), error margin ν of (2.8)) is done automatically by imposing a constraint on the cross-validation accuracy. Specifically, we divide the data points into two subsets, one for training and another one for testing. We then perform cross-validation and select the set of parameters that produce a classification accuracy higher than 99%. Note that if heavy occlusions are present, we allow the classification accuracy to be lower.

Figure 2.4 illustrates the robustness of SVS representation under different noise effects. In particular, we compare the color-coded response of the decision functions for a car shape and an A-letter shape before and after being distorted. As for the car, we randomly remove pixels from the shape and also add noise to the background. The A-letter shape is interwoven with white spaces to create distortion of the shape boundary. For both cases, it can be seen from Figure 2.4 that the color-coded responses, thus, the

associated decision functions remain quite stable.

From the computational perspective, the SVS complexity is essentially the same with that of SVM algorithms. In general, it is polynomial in the number of input points. In our experiments, it takes about 0.15 seconds to compute SVS for a shape in the MPEG7 dataset [59] using a 2.4 GHz Quad Core machine. The learning process can be significantly speed up using approximate variants of SVM [60, 61]. The query of the decision function is very efficient. It is linear in the number of support vectors, which is a small fraction of the total number of points (e.g. 1%). In addition, it can be accelerated by two orders of magnitude [62] using statistical approximations.

2.2.2 SVS Feature Points

Shape matching algorithms using SVS representation comprise two constituents: feature (interest) points and their descriptors. This section discusses possible ways of selecting the feature points. All these methods are based on the previously explained decision function $f(x)$.

The feature points are desired to be stable under local and global shape perturbations (including affine transformations and articulated motion) as well as noise and other artifacts. Such feature points should be reliably computed with high degree of reproducibility. In addition, the local descriptors computed at these feature points have to be discriminative enough for reliable shape matching and registration, therefore structure around the feature points should be rich in terms of local information content. In what follows we will elaborate on different possibilities of selecting good feature points for

SVS representation.

2.2.2.1 Gradient-based Feature Points

A corollary of Theorem 2.1 is that the gradient orientation is stable while gradient magnitude differs only by a constant factor. The gradient orientation is given by

$$\frac{\nabla f(\mathbf{x})}{||\nabla f(\mathbf{x})||}, \quad \text{where} \tag{2.11}$$

$$\nabla f(\mathbf{x}) = \mathbf{2}\gamma \sum_{\mathbf{i=1}}^{\mathbf{m}} \alpha_{\mathbf{i}} \exp(-\gamma ||\mathbf{x} - \mathbf{x_i^*}||^{\mathbf{2}})(\mathbf{x_i^*} - \mathbf{x}). \tag{2.12}$$

To evaluate the stability of the gradient orientation, we randomly choose a set of 500 points on each of 70 SVSs created from different shapes in the MPEG7 database [59] and examine their gradients as the training parameters (γ, ν) vary (γ by $6\times$ and ν by $10\times$ with respect to the smallest value of each parameter). Note that each different parametrization may generate a different decision function in magnitude, however we are interested in how the *direction* of the gradient of the decision function changes. Therefore, to account for the multiplication factor of the decision function, we normalize the decision function values by their mean value yielding relative magnitudes.

Figure 2.5 shows how the standard deviation of gradient direction changes with respect to the relative gradient magnitudes for 500 points from one of the 70 SVSs. One can easily notice that the gradients vary with respect to the training parameters. This is because large variation of the training parameters (γ, ν) results in different classification functions that do not satisfy strictly the condition $\text{sign}[f(\mathbf{x})] = \text{sign}[\mathbf{g}(\mathbf{x})]$ as in Theorem 2.1.

However, the variation exhibits a strong dependency on gradient magnitude. The

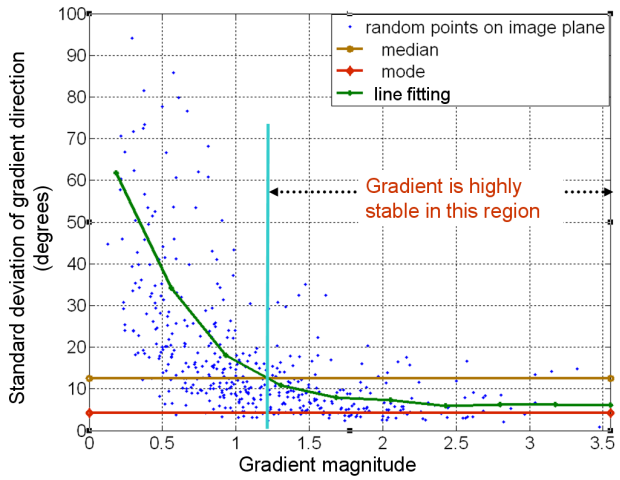


Figure 2.5: The relative gradient magnitude at a point is computed by dividing its gradient magnitude by mean of gradient magnitude across the entire shape.

higher the gradient magnitude gets, the smaller the standard deviation is. For points with gradient magnitude of more than two, the standard deviation, which directly corresponds to direction changes, is as small as 4°. Note that in practice the variation should be smaller than what we see in Figure 2.5 since the constraint of high classification accuracy implicitly requires a consistent set of parameters (γ, ν).

Since the gradient orientation is stable especially for higher gradient magnitude points we choose a small subset of such points for matching. We apply iterative search method that finds the maximum gradient magnitude point on $\nabla f(\mathbf{x})$ until it selects 100 points and build a list by ordering them according to their angles from the center of the shape in a circular fashion. The starting (0°) orientation of circular sweep is set with respect to a dominant gradient direction of $\nabla f(\mathbf{x})$.

Severe occlusions or distortions might lead to change of the dominant gradient directions. In such a scenario, we allow the generation of multiple sets of descriptors corre-

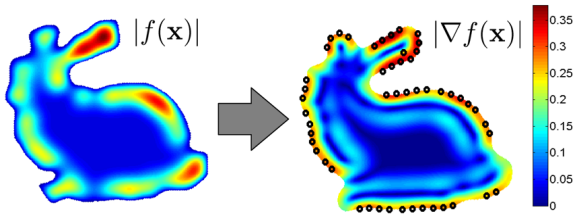


Figure 2.6: Decision function responses (left) and the gradient magnitude of the decision function (right) for the SVS representing Stanford Bunny. Points with higher values in $\nabla(f\mathbf{x})$ seem to be good candidates to be selected as discriminative feature points. Black circles are the 51 feature points (102 support vectors).

sponding to different tentative orientations. During the shape matching phase, the distance between two shapes is the smallest matching cost among all the orientations. Figure 2.6 shows the selected feature points for a sample shape.

2.2.2.2 Support Vector-based Feature Points

The support vectors are sufficient to construct the decision function and its gradient, thus they are good candidates for feature points.

One can ask whether the set of support vectors enable reliable shape matching. The answer to this question largely depends on the problem at hand. For non-articulated transformations, the support vectors remain stable. In Figure 2.7 we show that support vector locations are quite stable when the kernel width varies $2\times$. Constraining the classification accuracy to be sufficiently high (e.g. 99%) and preventing the kernel width from changing too much would produce similar support vectors. Yet, if the kernel width changes too much, e.g. $20\times$, support vectors change significantly as in Figure 2.3. Besides, the support vectors are sensitive to shape articulations due to the topology changes as illustrated

in Figure 2.7 bottom-pair.

2.2.2.3 Curvature-based Feature Points

It is possible to select points with high-curvature on SVS by looking at the Hessian matrix of a decision function. More specifically, for the 2D case, we first solve the eigenvalues of the Hessian matrix, which is proportional to curvatures.

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} = \mathbf{Q} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \mathbf{Q^{-1}} \tag{2.13}$$

Points with high curvatures associate with large eigenvalues of both dimensions (λ_1, λ_2). We randomly sample 200 points to compute curvatures. Feature points are chosen where the first eigenvalue λ_1 and the second eigenvalue λ_2 are larger than the median of λ_1 and λ_2, respectively. To mitigate noise effects, the decision function can be smoothed at different scales before computing curvatures. Gaussian smoothing of the decision function (2.7) is a mixture of Gaussian functions, which can be computed efficiently with a closed-form expression.

An advantage of this selection scheme is that local descriptors are highly discriminative. For instance, SIFT can be computed to find point-wise correspondences for aligning two similar shapes. However, this point selection method is not appropriate for shape matching. For example, choosing only those points around corners makes it impossible to differentiate a rectangle from a square. This disadvantage arises for shapes characterized mainly by their dimensions and shapes characterized by the arrangement of similar local structures. Another disadvantage of this method is the difficulty of computing curvatures when generalizing to higher dimensions.

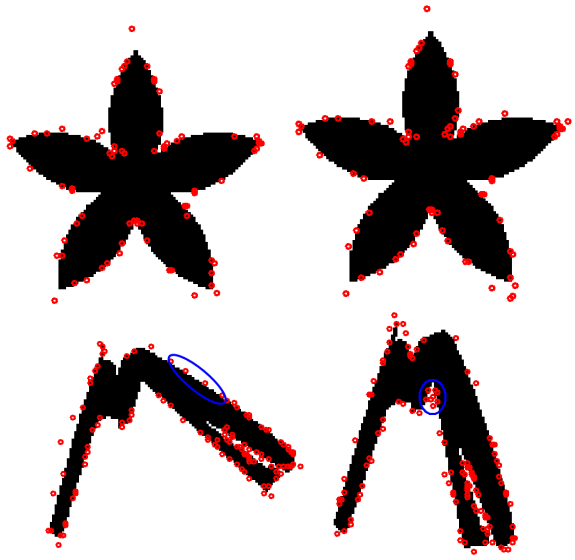


Figure 2.7: Stability of support vectors with respect to variation of RBF kernel width (γ) and articulation. (**top**) Support vectors when γ varies. $\gamma = 0.0027$ in the first shape, and $\gamma = 0.0054$ in the second shape. 62 out of 78 support vectors in the first shape appear in the second shape. The overall positioning of the support vectors is very similar. (**bottom**) Support vectors with shape articulation. Blue circles indicate SVs that do not have correspondences on the other shape. 83 out of 114 SVs in the first shape appear in the second shape.

2.2.2.4 Entropy-based Feature Points

Another possibility is selecting a small subsets of points whose local gradient orientation have high entropy. To compute this entropy, we first create a histogram of gradients over the local window of size 0.25×0.25 (with respect to mean pairwise distances). The entropy is then computed as follows:

$$-\sum_{i=1}^{n} p_i log_2(p_i) \tag{2.14}$$

where p_i is the weight of the i^{th} bin and n is the number of bins in the histogram. High entropy is equivalent to high variation of gradient orientations. Therefore, it is a good

indication of complex local topologies of SVS, thus, more discriminative local descriptors. This strategy is similar to the high-curvature selection method. However, it does not involve the computation of principal curvatures which can be difficult for higher dimensional cases.

2.2.3 SVS Descriptors

SVS facilitates the computation of a set of descriptors extracted from the decision function. Below, we give only an examples of possible descriptors.

We compute a local histogram of oriented gradients (HOG) descriptor around each point *on the decision function gradient* ∇f but not on the conventional edge gradient, thus our HOG$_f$ is significantly different from existing descriptors.

For a given feature point, a 4×4 array of 8-bin orientation histograms is constructed within a local window. The size of the window is set to be 0.25×0.25 (relative with respect to mean pairwise distance). Our experiments indicated that this size provides satisfactory results for both very coarse and fine shapes. A histogram is populated for each sub-block by aggregating the gradient magnitude of the decision function into the corresponding orientation bin of the gradient direction of the decision function.

Since gradients with larger magnitudes are more stable, the contribution of each gradient to the histogram is set to be proportional to its magnitude. We impose a Gaussian kernel to weight gradients based on their relative distances with respect to the feature point. This spatial weighting puts more emphasis on gradients that are closer to the center and helps improve the discriminative power of the local descriptors.

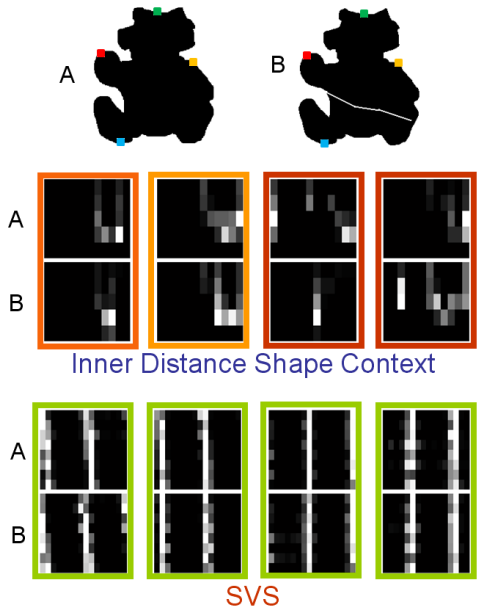


Figure 2.8: Comparison of IDSC with the SVS computed at four locations indicated by square dots on the shapes. Descriptors are displayed from left to right corresponding to the following order of dots' colors: red, yellow, green, cyan. The second image has a crack in the shape. IDSC changes drastically (it highly depends on the boundary) while the SVS remains robust generating almost the identical descriptors. Box color indicates the severity of the mismatch: Green ($< 20\%$) ; Orange ($20 \rightarrow 50\%$); Red ($> 50\%$).

To prevent problems due to the coarse binning issues, the value of each gradient point is interpolated into adjacent histogram bins. Finally, the mean of gradients for the local window is taken to be the orientation of the descriptor and the histogram bins are reoriented accordingly with respect to this mean orientation to achieve the rotation invariance of the descriptor. The histograms are then concatenated into a 128-dimensional vector.

Figure 2.8 shows a comparison of the Inner Distance Shape Context descriptors (IDSC) [44] with the SVS descriptors for a pair of images where one contains irregularity

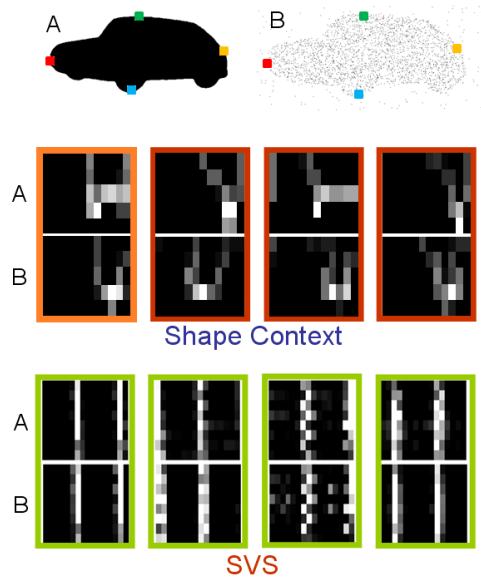


Figure 2.9: Comparison of Shape Context (SC) with the SVS computed at four locations indicated by square dots on the shapes. Descriptors are displayed from left to right corresponding to the following order of dots' colors: red, yellow, green, cyan. Box color indicates the severity of the mismatch: Green ($< 20\%$) ; Orange ($20 \rightarrow 50\%$); Red ($> 50\%$).

in the shape (assuming even after some morphology such artifacts remain). The inner distance responses change drastically while the descriptors computed from SVS stays very robust. Figure 2.9 demonstrates the strength of the SVS descriptors even for very noisy data (note that, fitting an outer shell, morphological filtering, etc. would not help for this shape as the noise is dispersed into the background). In this case, we compare with Shape Context (SC) since IDSC will not work due to the presence of disconnected interior. To the best of our knowledge, there is no other representation that gives such a robustness that SVS provides for heavily noisy and disconnected data. For all of the experiments presented in Figure 2.8, and 2.9, we use one-class SVM to learn the decision functions.

2.2.4 Experiments

For our experiments, we need to establish the distance between two shapes. Given two SVS decision functions $f_A(\mathbf{x})$ and $f_B(\mathbf{x})$, for shapes A and B respectively, we compute a distance score between their descriptors.

As we explained before, since the feature points are already ordered with respect to the dominant gradient direction, initial alignment for the shapes with similar overall structures is almost accurate. Next, we use the local descriptors for comparison of two shapes. The advantage of using local descriptors is that they are robust against occlusion and shape articulation.

Let two sets of the descriptors for two shapes A and B be $\Lambda^A : \{\lambda_1^A, \lambda_2^A, ..\lambda_t^A\}$ and $\Lambda^B : \{\lambda_1^B, \lambda_2^B, ..\lambda_s^B\}$ where t does not have to be equal to s assuming $t \geq s$. The correspondence is established through a mapping function h such that $h : \Lambda^B \rightarrow \Lambda^A$. If a descriptor λ_i^B is matched to another λ_j^A then $h(i) = j$. We define a cost function as

$$\boldsymbol{E}(h) = \sum_{1 \leq i \leq s} \epsilon(h(i), i) \tag{2.15}$$

where the descriptor distance is computed using χ^2 statistic

$$\epsilon(h(i), i) = \sum_{1 \leq k \leq 128} \frac{[\lambda_{h(i)}^A(k) - \lambda_i^B(k)]^2}{\lambda_{h(i)}^A(k) + \lambda_i^B(k)}. \tag{2.16}$$

This cost represents the overall distance for the corresponding pairing of the descriptors. Note that, the mapping h is neither one-to-one nor overlapping, but keeps the ordering of the descriptors.

To minimize $\boldsymbol{E}$, we use dynamic programming to find the solution to (2.16). It is worth noting that the start points and the end points of two sequences are already roughly

aligned for the dynamic programming algorithms to converge to the correct solution. Under certain conditions, the initial alignment provided by the ordered lists may not be valid. To overcome this, we find and compensate for the angle that maximizes the correlation between two 'global' histogram of oriented gradients, which are defined as HOG_f yet computed for the entire shapes, of two given shapes before minimizing the above cost function.

We run the first set of experiments on the entire MPEG7 shape benchmark dataset [59], which has 70 classes and 20 shapes for each class, a total of 1400 images. The performance is measured by the standard Bullseye test. For each shape, the retrieval accuracy is measured by counting how many of twenty correct shapes are in the top forty matches.

For the gradient-based method, we apply an iterative search method that finds the maximum gradient magnitude point on $\nabla f(\mathbf{x})$ until it selects 100 features. In the entropy-based feature selection method, 200 points are randomly selected in each shape where HOG_f descriptors and their entropy are computed. Points associated with entropy larger than 1.5 times the median of overall entropy value are selected for matching. In the curvature-based method, we also randomly sample 200 points to compute curvatures.

Table 2.1: Comparison of MPEG7 classification results using four different feature point selection methods in Section 2.2.2 based on gradients, support vectors, curvatures, and entropies. The last column shows the baseline classification result when gradients are computed from edges instead of SVS.

Method	Gradient	SV	Curvature	Entropy	Edges
Accuracy (%)	91.07	70.13	72.52	62.25	30.25

Feature points are chosen where the first eigenvalue λ_1 and the second eigenvalue λ_2 are larger than the median of λ_1 and λ_2, respectively. For the support vector-based method, we randomly select half of the support vectors set as feature points for shape matching (we observed using only the most significant support vectors is in fact deteriorates the performance).

Algorithm	**Accuracy** (%)	**Algorithm**	**Accuracy** (%)
Gopalan [63]	93.67	Shape L'Âne Rouge [69]	85.25
IDSC + LCDP [64]	93.32	Generative Models [70]	80.03
Mixture of Gaussian + tSL [65]	89.1	Curve Edit [71]	78.14
Shape-tree [66]	87.7	SC+TPS [49]	76.51
IDSC + DP + EMD [67]	86.56	Visual Parts [59]	76.45
Biswas [51]	86.48	CSS [72]	75.44
Hierarchical Proscrustes [68]	86.35		
IDSC + DP [44]	85.4	**SVS + DP**	**91.07**

Table 2.2: Comparison for the MPEG7 dataset.

Table 2.1 gives the correct retrieval percentages for four feature point selection methods (gradient-based, support vector-based, curvature-based, and entropy-based) presented in Section 2.2.2. Results indicate that the gradient-based method produces more consistent feature points and better matching performances. We also compare our performances with HOG computed on the edges gradients (30.25%) to verify that SVSs improve the discriminative power of our descriptors. Edge gradients are commonly used to extract feature points for 2D shape representation.

As in Table 2.1, the best overall accuracy on MPEG7 dataset using the Bulleyes

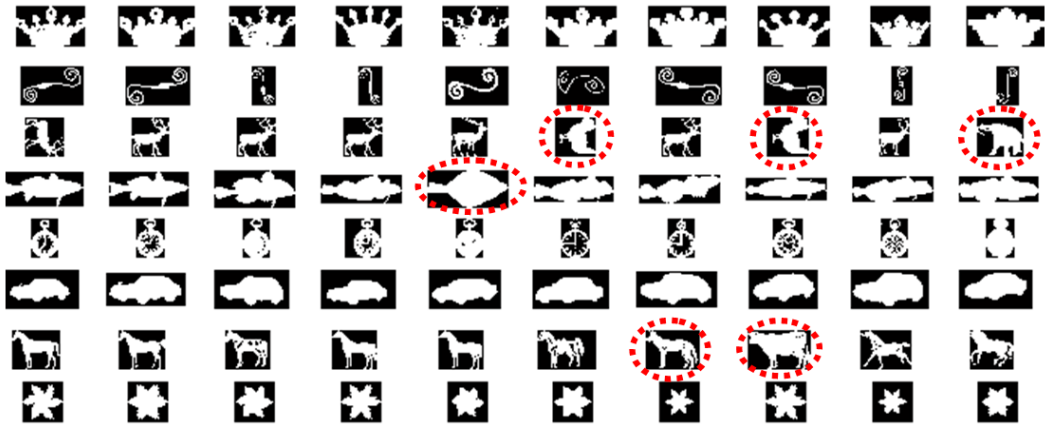


Figure 2.10: SVS results: red circles show the incorrect matches. Note that none is in top 4 rankings.

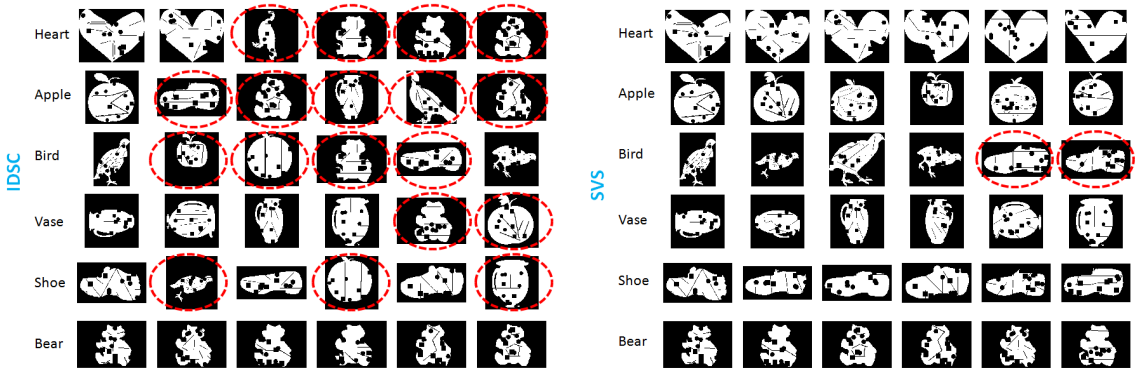


Figure 2.11: Comparison of the retrieval results for noisy dataset. Accuracy of IDSC is 57.0%, SVS is 93.7%. Red circles show incorrect matches.

test is 91.07%, which is based on the gradient-based feature selection method. This is better than the performances of SC [49] (76.51%), IDSC [44] (85.40%), the post-refined version of [51] (86.48%), and shape tree [66] (87.70%) as summarized in Table 2.2. The best performance 93.67% is reported in [63], however their part-based algorithm is highly sensitive to segmentation errors especially for noisy data. A matching scheme [64], which takes into account the influence of the other shapes while computing the similarity of a

Table 2.3: MPEG7 classification results versus the number of features selected using the gradient-based selection strategy.

♯ Features	10	50	100	200	300
Accuracy (%)	15.93	68.83	91.07	90.38	89.92

Table 2.4: Random distortion results on partial MPEG7 dataset

Method	IDSC	Denoised IDSC	SVS-BoW	SVS-DP
Accuracy	57.0%	85.9%	81.4%	93.7%

pair of shapes, has reported an accuracy of 93.32%. SVS can be effectively used as the shape representation in [64]. Fig. 2.10 shows sample retrieval results in the descending order of matching scores using the gradient-based features.

We also investigate the trade-off between the model complexity and the classification performance by varying the number of feature points from $10 \rightarrow 300$. The results are summarized in the Table 2.3. In general, more feature points would yield better accuracies. However, it can be noticed that the classification accuracy slightly suffers when the number of features exceeds 100. It is because a large number of features inevitably leads to less-discriminative interest points to be selected, therefore, negatively interfering with the shape matching algorithm.

In the second experiment, we pick a subset of 6 shape classes (8 samples per class) from the MPEG7 database and add random distortions into them. The performance is measured by counting how many of 8 correct shapes appear in the first 16 matches for each shape. Instead of using $\nu-$SVM algorithm like in the first two experiments, we use

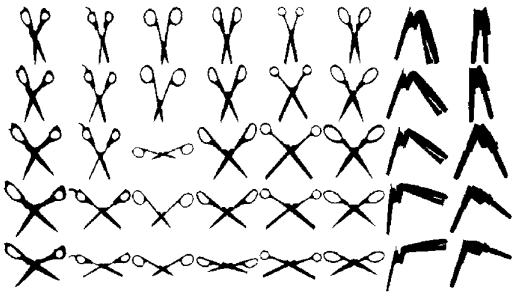


Figure 2.12: Samples from the articulation database. Note that, each **column** corresponds to a different object.

one-class SVM for this experiment since it produces better retrieval results. In addition to classification using SVS and IDSC, we zero-threshold our SVS representations to reconstruct the original binary shapes. These binary shapes are then used as the input for computing IDSC. We call this scheme *denoised IDSC*.

The overall performance of IDSC is 57.0%, while the SVS method, using gradient-based feature points, gives 93.7% as given in Table 2.4. Besides, Table 2.4 shows that denoised IDSC (85.9%) performs significantly better than IDSC. The result implies that our shape representation provides some denoising effects over random distortions. We also compare the performance of SVS when using the bag-of-word approach (81.4%) instead of the dynamic programming technique. Figure 2.11 shows retrieval results for both IDSC and SVS methods in descending order of the matching scores. IDSC's performance dramatically degrades on this database because artifacts within a shape severely changes the inner distances. Shape context and other edge-based methods will also have the same issue. In contrast, the SVS method invariably gives accurate retrieval results thanks to classifier-based representation.

In the third set of experiments, we test our matching algorithm, with the gradient-

Table 2.5: Matching results on the articulation dataset

Descriptor type	1^{st} Match	2^{nd} Match	3^{rd} Match	4^{th} Match
ℓ_2 (baseline)	25/40	15/40	12/40	10/40
SC [49]	20/40	10/40	11/40	5/40
IDSC [44]	40/40	34/40	35/40	27/40
Biswas [51]	40/40	38/40	33/40	20/40
Lin [73]	40/40	38/40	36/40	33/40
SVS HOG$_f$	**40/40**	**38/40**	**35/40**	**31/40**

based feature selection strategy, on the articulation database reported in [44]. This database includes 40 images from 8 objects with different articulations as shown in Figure 2.12. This consists of highly similar shapes like types of scissors with only minor differences but significant articulations. The recognition result is evaluated for each shape by choosing four most similar matches and then sort them according to their matching scores. Table 2.5 summarizes the matching results. We compare against the ℓ_2 distance on a bag-of-features, the shape context (SC), the inner distance with shape context (IDSC), the multiple feature indexing [51], and the layered graph matching [73]. In this experiment, the SVS uses the gradient-based feature point selection. It is apparent that our method handles articulation as well as (even better for later matches) IDSC, and much more accurately than the SC thanks to the robust nature of the SVS feature point selection and the locality of its descriptor.

2.2.5 Proof of SVM-RBF Irreducibility

The decision function of SVM with RBF kernel has the following form:

$$f(\mathbf{x}) = \sum_{\mathbf{i=1}}^{\mathbf{m}} \alpha_{\mathbf{i}} \exp(-\gamma ||\mathbf{x} - \mathbf{x}_{\mathbf{i}}^*||^{\mathbf{2}}) \tag{2.17}$$

Fourier transform of the function is as follows:

$$\begin{aligned} \mathcal{F}(\omega) &= \sum_{i=1}^{m} \alpha_i \mathcal{F}\{\exp(-\gamma ||\mathbf{x} - \mathbf{x}_{\mathbf{i}}^*||^{\mathbf{2}})\} = \sum_{\mathbf{i=1}}^{\mathbf{m}} \alpha_{\mathbf{i}} \mathcal{F}\{\exp(-\gamma ||\mathbf{x}||^{\mathbf{2}})\} \exp(-\mathbf{j}\omega \mathbf{x}_{\mathbf{i}}^*) \\ &= \sum_{i=1}^{m} \alpha_i \left[\sqrt{\frac{\pi}{\gamma}}\right]^n \exp\left(-\frac{\pi^2}{\gamma}||\omega||^2\right) \exp(-j\omega \mathbf{x}_{\mathbf{i}}^*) = \mathbf{\Phi}(\omega)\mathbf{\Psi}(\omega) \end{aligned} \tag{2.18}$$

where n is the dimension of $\mathbf{x}$ and,

$$\Phi(\omega) = \left[\sqrt{\frac{\pi}{\gamma}}\right]^n \exp\left(-\frac{\pi^2}{\gamma}||\omega||^2\right) \tag{2.19}$$

$$\Psi(\omega) = \sum_{i=1}^{m} \alpha_i \exp(-j\omega \mathbf{x}_{\mathbf{i}}^*) \tag{2.20}$$

where n is the dimension of $\mathbf{x}$. The Fourier transform of RBF kernel obviously does not have any zero.

From now on, consider $\Phi(\omega)$ as the Fourier transform of an arbitrary kernel function. By assumption, the entire function $\Phi(\omega)$ never vanishes, i.e. it does not have any zero just like the case of the RBF kernel. Therefore, it remains to show that the function $\Psi(\omega)$ is not factorable to conclude that $f(x)$ is irreducible.

Assume that $\Psi(\omega)$ is reducible. It means that it can be factored into a product of two entire functions. These functions are required to have non-empty zeros set (either real or complex). Then they must have the following form:

$$\sum_{k=1}^{K} a_k \exp(-j\omega y_k) \sum_{l=1}^{L} b_l \exp(-j\omega z_l) = \sum_{k=1}^{K} \sum_{l=1}^{L} a_k b_l \exp\{-j\omega(y_k + z_l)\} \tag{2.21}$$

Note that an exponential function $e^{j\omega\tau}$ never vanishes. This property together with non-empty zeros set constrain implicitly require:

$$K \geq 2, \quad L \geq 2 \tag{2.22}$$

The number of constrains when equating (2.21) to (2.20) is $(KL + m)$. The breakdown of constrains is as follows:

- KL constrains to equate the set of exponents $\{y_k + z_l\}$ with the set $\{x_i^*\}$.
- m constrains to equate $\{a_k b_l\}$ with $\{\alpha_i\}$.

The total number of variables is $2(K + L)$, and we require this to be at least equal to the total number of constrains, which gives:

$$2(K + L) \geq KL + m \tag{2.23}$$

It is easy to verify that this condition does not hold because $K \geq 2$, $L \geq 2$, $m \geq 5$ due to (2.22) and our initial our assumption. Therefore $f(x)$ in (2.3) is irreducible.

2.3 Concentric Ring Signature

In the second part of this chapter, we present a 3D feature descriptor that represents local topologies within a set of folded concentric rings by distances from local points to a projection plane. This feature, called as Concentric Ring Signature (CORS), possesses similar computational advantages to point signatures yet provides more accurate matches. CORS produces compact and discriminative descriptors, which makes it more robust to noise and occlusions.

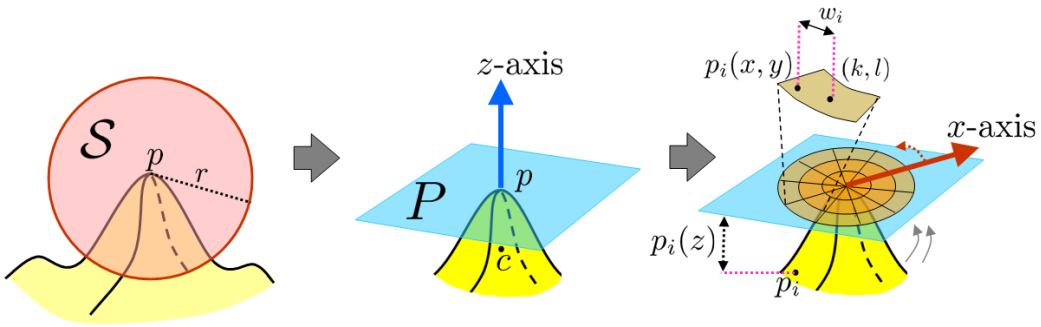


Figure 2.13: a) CORS is constructed at p by finding a spherical support region $\mathcal{S}$. b) A plane is fitted to local neighborhood and translated to the point p. The normal direction is taken to be z-axis c) Selecting a reference orientation for x-axis and projecting the distances from the surface to plane into the corresponding patches.

2.3.1 CORS Construction

To construct CORS we first find the 3D data points within the local support region, then determine a plane of projection, decide the reference orientation in that plane, and finally compute the patch responses that are arranged into a matrix form as illustrated in Fig. 2.13.

Local Support

Let p a point in a 3D cloud data and r the radius of a spherical volume centered on p. We assign the set of points falling within the spherical volume as the local support $\mathcal{S} = \{p_i : ||p_i - p|| \leq r\}$. The choice of the radius r is data-dependent. For example, larger radius is preferred for smooth and rigid shapes while smaller radius is preferred for shapes with articulations or structural variations. As r increases, CORS is more discriminative but more vulnerable against occlusions. A good choice of r would balance well these two

factors. We perform cross-validation on a subset of databases to choose the value r that gives best results.

Plane of Projection and Reference Axes

A plane P is fitted to the local support $\mathcal{S}$. There are two possible choices for fitting planes. One can use the all data points within the local support, fit a plane by least-squares as the system is almost always over-determined, and parallel move the origin of P at p. Alternatively, it is possible to select a subset of points along the perimeter of the local support, e.g. intersecting the sphere support with the object surface.

Fitting to the perimeter would be more appropriate particularly for the points along ridges as illustrated in Fig. 2.14. It is more meaningful for the plane of projection to be *tangent* to the surface rather than *slice* into it as our descriptor is based on elevations of points. In case the projection plane slices into the surface, the resulting descriptor would be zero.

We define a local reference coordinates so that the local descriptor is invariant to camera view. Let c be the (Karcher) mean, that is the coordinate having the minimal overall distance to the other points in the local support $c = \arg\min \sum_i \|p_i - c\|$. We set the z-axis to be orthogonal to P and pointing in a direction such as the dot product of the unit vector $\vec{z}$ with vector $\vec{cp}$ is positive. We define a local reference axis (x-axis) so that the local descriptor is invariant to camera view. x-axis points away from p to the projection of the 3D point that has the maximum distance from the fitted plane P within the local support $\mathcal{S}$. y-axis is defined by the cross product $\vec{z} \times \vec{x}$. With such assignments,

P corresponds to the xy plane going through point p. These two conditions define z-axis without any ambiguity.

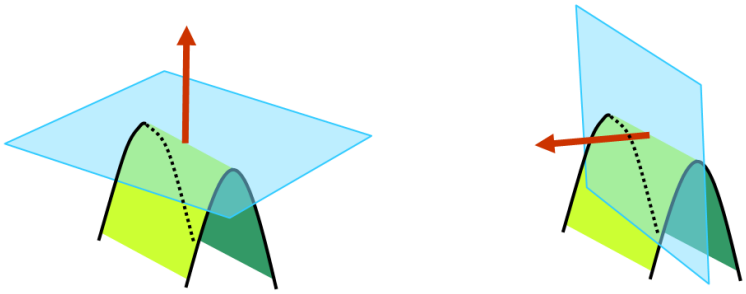


Figure 2.14: A special case where fitting a plane to local support is different from fitting a plane to its perimeter. a) Fitting to the perimeter results in a tangent plane, while b) fitting to the entire local neighborhood results in a slicing plane

In case the projection distances from P to the xy plane have more than one peak, multiple reference axes can be generated. When this situation occurs during training phase, multiple descriptors, each corresponds to one peak, are computed and stored in the database. During the matching phase, only one descriptor corresponding to the largest peak is needed for each query point, even when its projection distances have multiple similarly large peaks. The reason is because the query descriptor can always find its correct match in the model database where multiple descriptors have been generated to take into account ambiguity of peaks. We observe that this situation occurs at only around 1% of points and the inclusion of multiple peaks improves matching of descriptors.

Populating Patches

After fitting the plane and determining the reference axes, each 3D point p_i in the local neighborhood $\mathcal{S}$ is now represented by a tensor $p_i(x, y, z)$ independent of the camera viewing angle. The z coordinates $p_i(z)$ correspond to the distance from the plane in this tensor, and the xy-plane coordinates $p_i(x, y)$ correspond to the projection on the plane P. We estimate a representative elevation value of the given data points within the patches of this grid as follows:

1. We apply a polar grid along azimuth and radial directions on the xy plane centered on the original point p. The patches of this grid can be considered as the 2D histogram bins. Let $\{(k, l)\}$ be the set of sampled grid locations with $k = 1 \dots K$ and $l = 1 \dots L$, where K and L are the numbers of sampling intervals along the radial and the azimuthal directions, respectively. In other words, we will extract a 2D matrix $F_{K \times L}$ on this grid where each coefficient corresponds to a patch of the grid.

2. At each grid location (k, l), we estimate a representative elevation value $F(k, l)$. Elevation at a location is estimated by interpolating the elevation values of the given 3D points within the immediate patches. This significantly improves sparsity related issues, e.g. sudden jumps of the estimated elevation when there are insufficient number of 3D points and boundary issues e.g. being very close to boundary but falling into another bin, etc.

   The representative elevation score $F(k, l)$ is estimated as follows:

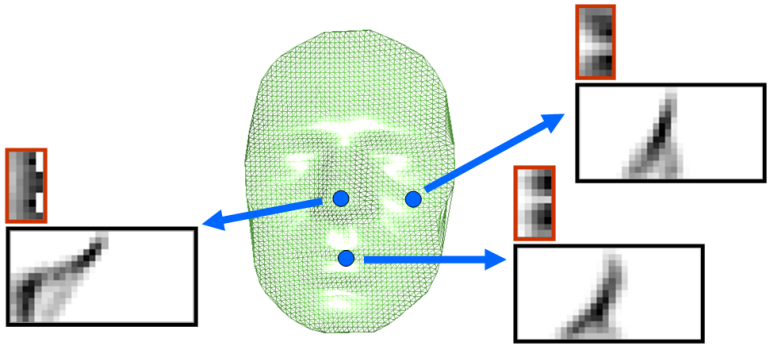


Figure 2.15: Illustration of CORS (red border) and spin image (black border) at different points on a 3D human face. The radius of support region is set to 15 for both descriptors. The number of azimuthal and radial quantizations of CORS are 10 and 5, respectively. The bin size of spin image is set equal to the scanner resolution as suggested by its author. For this setting, the dimension of CORS is around $6.5\times$ more compact than that of spin image.

$$F(k,l) = \frac{\sum_i w_i.p_i(z)}{\sum_i w_i} \tag{2.24}$$

where p_i are 3D points within the immediate neighboring bins of the bin of (k,l) and the weight is computed as:

$$w_i = \begin{cases} 1/\alpha, & d \leq \alpha \\ 1/d, & \alpha \leq d \leq 2\alpha \\ 0, & \text{otherwise} \end{cases} \tag{2.25}$$

and $d = \|(k,l) - p_i(x,y)\|$.

Basically, $F(k,l)$ is the weighted average of elevation of points surrounding (k,l). The contribution of each surrounding point's elevation to the estimation of representative

elevation is controlled by a weight w_i negatively proportional to distance to (k, l). the parameter α controls the smoothness of a descriptor. Higher α values yield smoother descriptors while a smaller α makes the descriptor sensitive to positional shifts. The choice of α depends on the sampling interval along azimuth and radial directions. We observed that the average Euclidean distance between bin centers and their adjacent bins is a satisfactory value. Using a fixed value of α makes bins close to the origin in a polar coordinate system look more similar than those further away. α could be set in an adaptive manner to overcome this issue. Also, imposing a minimum distance constraint enables improving the robustness against small differences in shape very close to the center.

In addition to the mean orthogonal distance from S to the P, the standard deviation of the projection distances and the density of points falling into each bin also possess complementing discriminant power and can be incorporated into similar matrices. An advantage of the mean distance is that it does not require point density estimation and normalization. Figure 2.15 provides a visual illustration of CORS computed at different locations on a 3D data cloud of human face. Note that the dimension of CORS is 6.5 times smaller than that of spin image. Such dimensional reduction increases the descriptors matching efficiency, yet does not compromise the discriminative power as shown in the experimental analysis section.

2.3.2 Fast Approximation CORS

In practice, the computation of CORS can be significantly speed up by using the normal vectors whenever available as z-axis of the local reference frame. This eliminates

Fast CORS	Point Signature	CORS	Spin Image
1.02s	1.27s	1.34s	2.76s

Table 2.6: Amount of time taken to compute 500 descriptors.

the need for fitting a plane to the neighborhood at every location. Table 2.6 compares the computation of CORS, speed-up CORS, and other descriptors. As visible, CORS is as fast or faster than the conventional methods. In this experiments, we use models from TOSCA dataset, each of which has around $50K$-60K points. The computation times are for 500 descriptors. No normals are available beforehand for CORS.

A wide spectrum of applications involve 2.5D image that is a view-point specific simplified representation of 3D data in which every pixel on the camera's image plane contains only one depth value measuring its distance to the scene. This data structure is effectively a 2D hash table allowing efficient constant-time retrieval of local neighborhood $\mathcal{S}$ around a point p.

To speed up the matching of a CORS descriptor to large databases (hundreds of thousands of signatures), a coarse-to-fine approach can be adapted. For each CORS descriptor, we row-sum up all elements of the $K \times L$ matrix, i.e. those lying on the same ring to create a subordinate signature. This can be used for quickly pruning unlikely candidates.

2.3.3 Similarity Measure and Matching

Euclidean distance is used as the dissimilarity measure between two CORS matrices:

$$dist(F_1, F_2) = \sqrt{\sum_{k,l}(F_1(k,l) - F_2(k,l))^2} \tag{2.26}$$

Matching of CORS descriptors is not limited to Euclidean distance. Since the representation of CORS is in a matrix form, it can be considered to possess a manifold structure where the matching score is defined as the geodesic distance connecting two CORS descriptors on the manifold. In addition, a manifold can be flattened using ISOMAP, etc. In this work, we concentrate on the Euclidean distance.

The dissimilarity measure is mainly dictated by the applications at hand and the contributions of different bins can be modified accordingly. For example, if the goal is to highlight symmetric local structures, CORS with similar bin values along the azimuthal dimension should be weighted significantly higher than the others.

The best match of a query descriptor can be efficiently extracted using approximate nearest neighbor techniques such as kd-trees and box decomposition tree based search.

2.3.4 Experimental Results

We conducted several detection, recognition, registration, and retrieval experiments to analyse the performance of CORS descriptors in comparison to existing descriptors including spin images and point signatures.

2.3.4.1 Saliency

In the first experiment, we use five synthetic models, each of around 150K points, to compare the saliency, i.e. discriminative power, of point signature, spin image, and CORS. A reference database of 10K signatures is computed at random points on each model. Another 10K set of signatures at randomly sampled points on the same model is used as the query database. We make sure that no point from the reference set is selected for the query set. A query signature at location q_i is said to have a good match to its model at location m_i if their distance $d_i = |m_i - q_i| \leq \epsilon$. We chose ϵ to be 5 times of the scanner resolution. Figure 2.16 shows the percentage of correct matches within the first k nearest neighbors. CORS out performs both point signature and spin image. The correct matching rate of CORS is approximately 2.5 times higher than that of point signatures. The error rate reduces from 18% for spin image to 12% for CORS, which is more than 33% of improvement.

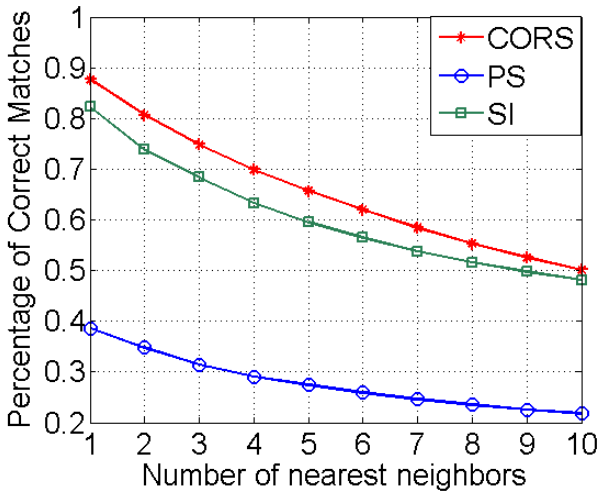


Figure 2.16: Percentage of correct matches within k-nearest neighbors.

Discriminant Ratio

Finding descriptor correspondence is essential to many recognition and retrieval problems. It is desirable to have a robust mechanism of filtering out potentially erroneous matches and keeping only the useful ones for further processing. When searching descriptors into a large database or finding correspondences within noisy observations, the nearest neighbor matching would result in a large number of incorrect pairs. For instance, Fig. 2.17 shows that the rate of finding correct correspondences (dashed lines) decrease with the noise and the database size.

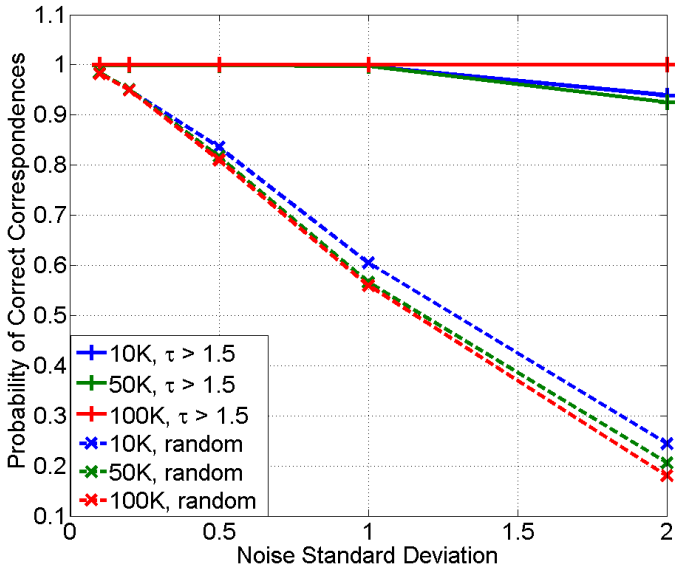


Figure 2.17: Comparison of correct correspondences for two methods: random selection and selecting with the discriminant ratio $\tau \geq 1.5$. Gaussian noise is added to all descriptors before matching. The noise standard deviation varies from 0.1 to 2, and database size varies from 10K to 100K.

To deal with this issue, it is possible to impose a global threshold on the Euclidean

distances of the descriptors to their closest matches. However, applying a threshold does not work as most of the less discriminative descriptors tend to have multiple matches with only small distances.

Taking a similar approach to [74], we compare the distance of the closest neighbor to that of the second nearest neighbor. We define the discriminant ratio as a measure for this comparison:

$$\tau = \frac{dist_2}{dist_1} \tag{2.27}$$

where $dist_1$ and $dist_2$ are the Euclidean distances between a query descriptor and its first and second best matches in the database, respectively. Higher discriminant ratios require correct matches to have the closest neighbor significantly closer than the closest incorrect matches. As a result, matches with high discriminant ratios tend to be much more reliable. For false matches, there will likely be a number of other false matches within similar distances due to the high dimensionality of the feature space.

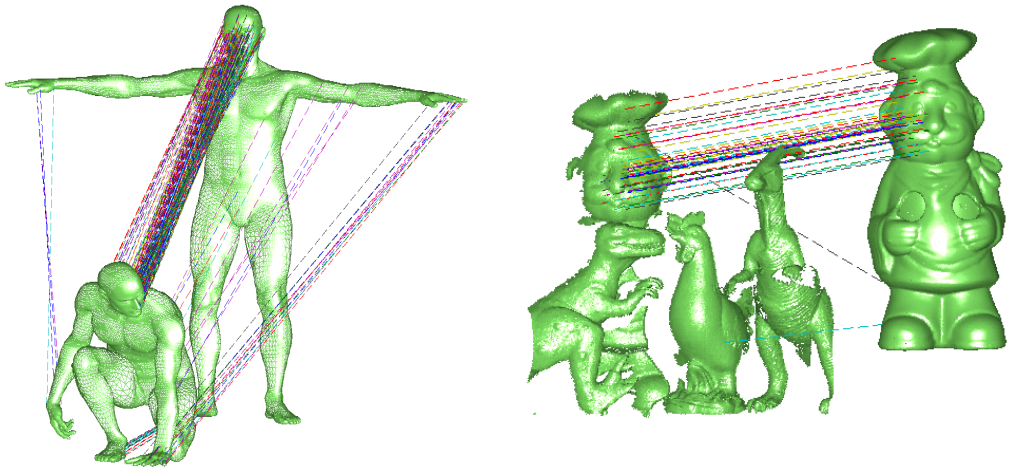
Imposing a limit on the discriminant ratio dramatically increases the correct correspondence rate. Figure 2.17 shows the probability of correct correspondences within selected points under different settings. Shapes from TOSCA and Mian datasets are used to construct the descriptor databases. Dashed lines represent the conventional correspondences by the nearest neighbor matching and random point selection. Solid lines represent the refined correspondences by the nearest neighbors with the constraint that their discriminant ratios are at least $\tau \geq 1.5$. While the performance of the conventional method degrades almost linearly with the Gaussian noise standard deviation, the performance of the second method remains high and stable as the noise level increases. Impressively, the

correct correspondence rate of the refined method reaches up to 100% for the the database of the largest size ($100K$), while the conventional method has the lowest performance on this setting (see red lines). The reason is that as more false matches are available, the comparison based on the distance to the closest neighbor and to the second closest neighbor becomes more robust. In other words , a point that can stand out from a crowded database, in terms of discriminant ratio, must be more reliable.

Although the reliability of selected points is proportional to the discriminant ratio, the number of detected points reduces as the discriminant ratio increases. We observed that $\tau \geq 1.5$ is a good trade-off between the correct correspondences rate and the number of total correspondences. We use both Mian and TOSCA datasets to create the descriptors databases. CORS parameters set are as follows: the local support radius $r = 12, K = 10, L = 20$, where the value is relative to scanner resolution. Figure 2.18 gives sample sets of correspondences found using the proposed discriminant ratio. Feature correspondences are demonstrated to be very accurate even under nonrigid transformations and occlusions.

2.3.4.2 Shape Detection and Registration

Given a 2.5D range scan query scene the task is to make a reliable decision whether the scene contains the objects of interests. If an instance of the target is detected, either complete or partially occluded, the algorithm will estimate the transformation that registers the complete model to the target in the scene. This problem is challenging for several reasons. First, range scan images usually contain hundreds of thousands of points, posing a question of how to process them in an efficient yet reliable manner. Second,



(a) Tosca Non-rigid Shapes (b) Mian Occluded Shapes

Figure 2.18: Correspondences of CORS with the discriminant ratio $\tau \geq 1.5$ **a**) Two shapes taken from TOSCA dataset with nonrigid transformations (Note that all correspondences for this pair are correct while the plotted lines are sometimes hidden behind the surface giving an impression of matching the face with the back of the head). **b**) A model and an occluded scene from Mian *et al.* dataset. (All correspondences, except two on the leg of the reference model, are correct.)

target is only partially visible due to the self-occlusion and the cluttering effects (see Figure 2.18b), rendering many global shape descriptors useless. Three main steps of our detection method using CORS, then, can be listed as:

- Computate CORS for a subset of randomly distributed points in the scene,
- Find correspondence between the query signatures of the randomly distributed points and the model signatures computed off-line, and
- Iteratively estimate motion parameters with geometric constraints within the RANSAC

framework to locate and find the pose of the object.

We evaluate our method on the datasets provided by Mian [75], which consists of 5 models and 50 scenes taken with a Konica-Minolta range scanner. The scenes in this dataset are highly occluded and cluttered by putting objects very close to each other. We are interested in evaluating the recognition rate that is defined as the number of correct detections over the total number of the scenes. An object is said to be correctly detected if the resulting errors of the translation and pose estimations, compared to the ground truth, are smaller than one-tenth of the object's diameter and $12\,^{\circ}$, respectively. These criteria are the same with that of Drost *et al.* [56], therefore allowing the comparison to their methods.

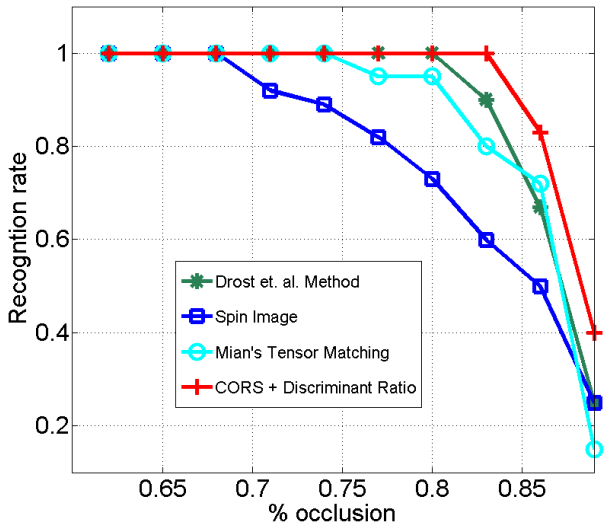


Figure 2.19: Comparison of the recognition rates vs. the percentage of occlusions: for spin image, tensor matching, Drost method, and CORS.

Our algorithm converges after, on average, only 3 RANSAC iterations. It produces satisfactory estimates of R and t even without any further processing. Figure 2.20 shows

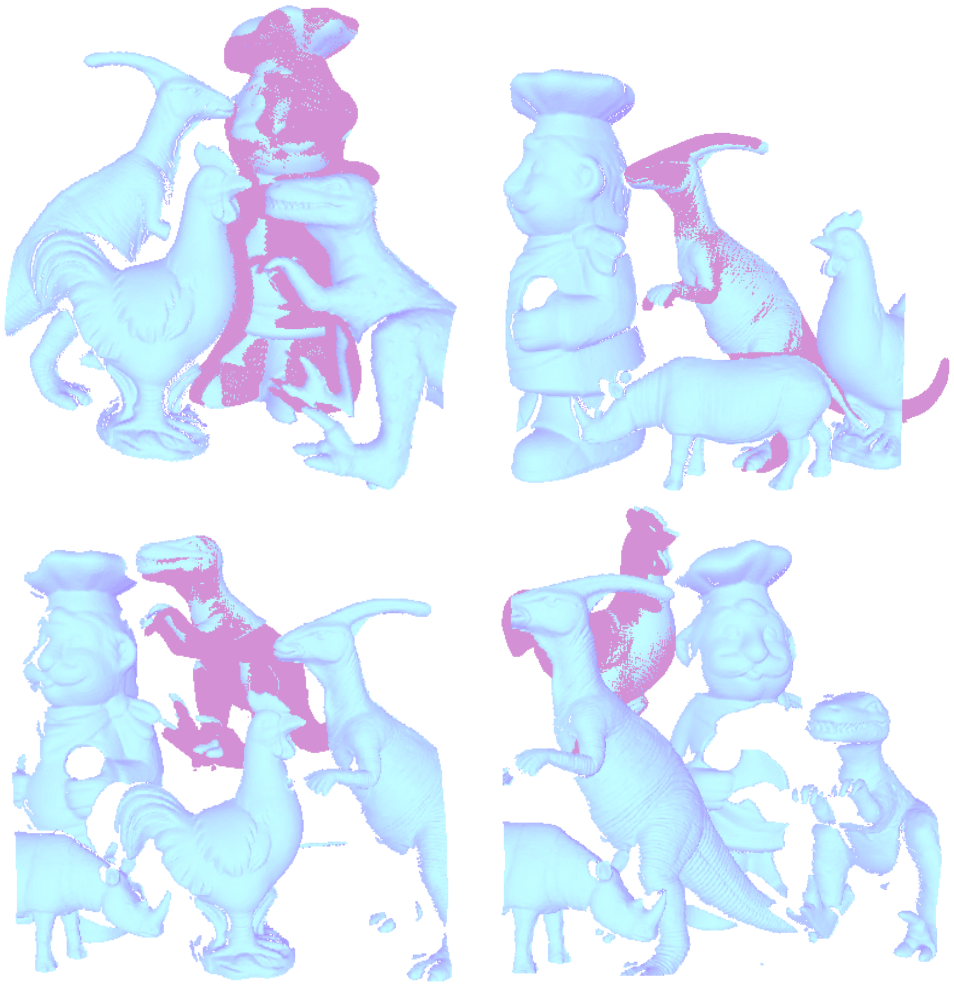


Figure 2.20: Recognition and registration of 3D model point clouds into the occluded scenes. The cyan and pink colors are used to render the scenes and models, respectively. The transformed models closely overlap with the targets within all the scenes. 50 different scenes are used for testing our approach.

the registration results of four different models without any refinement. Figure 2.19 shows the overall recognition result of our method. As given, it outperforms all other methods in terms of the recognition rate with respect to occlusions. The key to the higher robustness against occlusions lies in the facts that the detected correspondences are highly reliable and accurate, and the use of the discriminant ratio constraint successfully filters

out potentially invalid matches.

2.3.4.3 Shape Recognition and Retrieval

Figure 2.21: Six different articulations from a model in the TOSCA dataset.

Given a query shape, the goal is to extract all instances of this shape, which might come with 3D non-rigid articulations, from the database . It is necessary for an algorithm to be 3D articulation-independent in order to perform well on this test.

A bag-of-feature approach is used to retrieve 3D shapes. We compare CORS at a subset of salient points to evaluate the similarity between a model and a query. Similar to the approach in the previous section, we perform salient points selection and shapes retrieval jointly. Given a query point clouds, we first compute CORS at a subset of randomly selected points. Two best matches for each query CORS are extracted from the model database from which discriminant ratios are computed. We define a dissimilarity score δ between a query and a model using only those correspondences with high discriminant ratios. More specifically, it is computed as:

$$\delta = \frac{1}{|\mathcal{S}_\tau|} \sum_{q_i \in \mathcal{S}_\tau} dist_1(q_i), \quad \mathcal{S}_\tau = \{q_i : \tau_{q_i} \geq \tau_0\} \tag{2.28}$$

where $\mathcal{S}_\tau$ is the set of feature points having discriminant ratio higher than a threshold τ_0.

In contrast to the previous section where the threshold τ_0 is kept constant, here we keep the size of $|\mathcal{S}_\tau|$ fixed (50 in our experiments) and let τ_0 to vary. This effectively forces two dissimilar shapes to accept some poor matches leading to higher dissimilarity score. Shapes with dissimilarity score lower than a predefined threshold are considered to be from the same object.

We evaluate our algorithm on the TOSCA dataset which consists of 9 models and 80 shapes. This is a challenging database since all models come with various nonrigid shape transformations. Figure 2.21 shows a model in the dataset with six different 3D articulations. We perform a leave-one-out matching experiment. In other words, each shape is excluded from and matched against the rest of the database. The recognition rate is computed by counting how many best matches that come from the same models of query shapes. Recognition and retrieval results of our method are presented in Fig. 2.22. Our method achieves 100% correct recognition and retrieval rate, i.e. all shapes from correct models are retrieved before those from incorrect models. The recognition rate of spin image is only 55%, and the retrieval result is presented in Fig. 2.23. It is easy to observe that our method produces consistent scores. This makes it possible to impose a threshold on the score and reject shapes from incorrect models. On the contrary, dissimilarity scores of spin image are quite erratic inhibiting the determination of the cut-off point between correct and incorrect models.

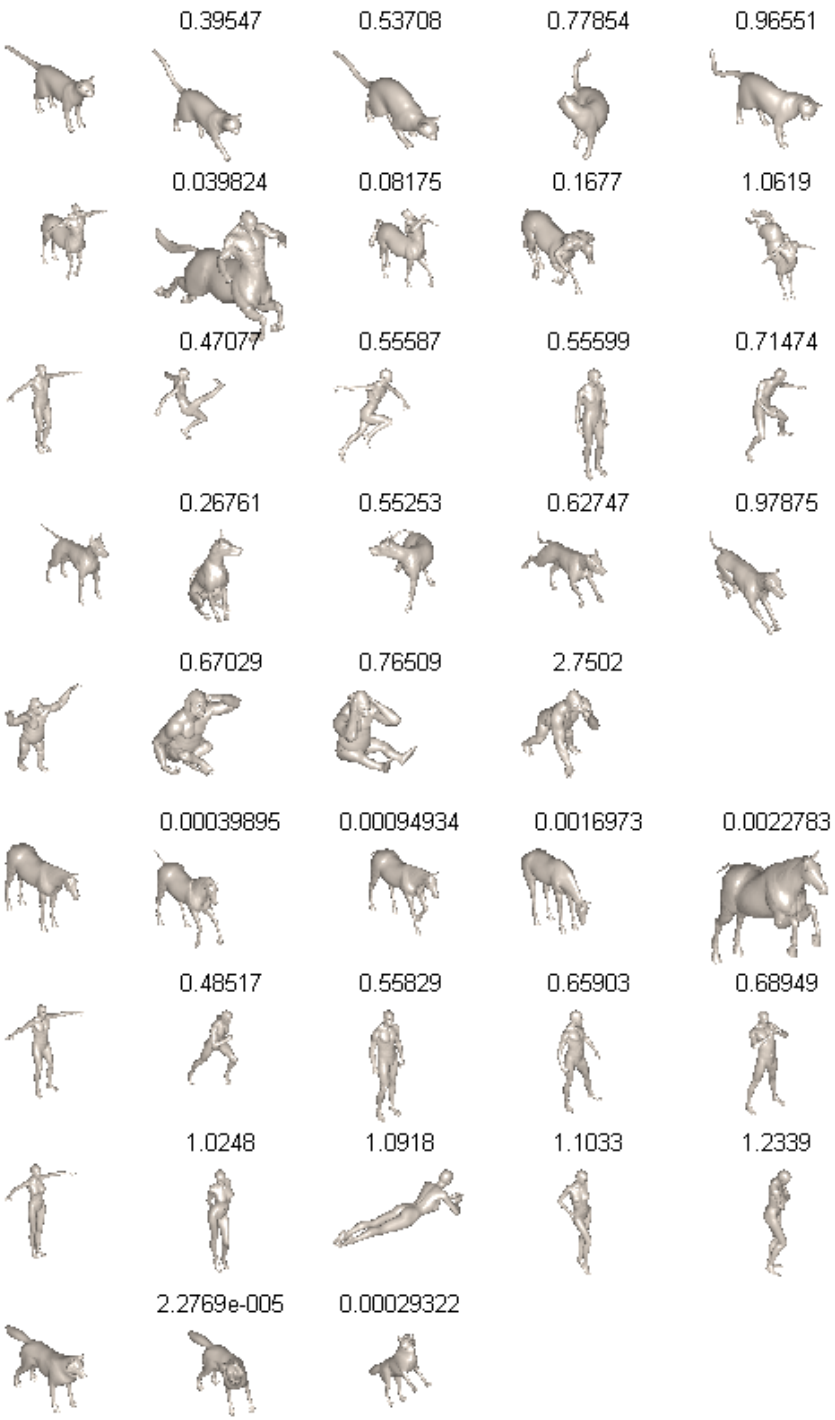


Figure 2.22: Top retrieval shapes of our CORS method. All shapes are correctly retrieved. (The 5^{th} model *gorilla* has three and the 9^{th} model *wolf* has two instances in the dataset.)

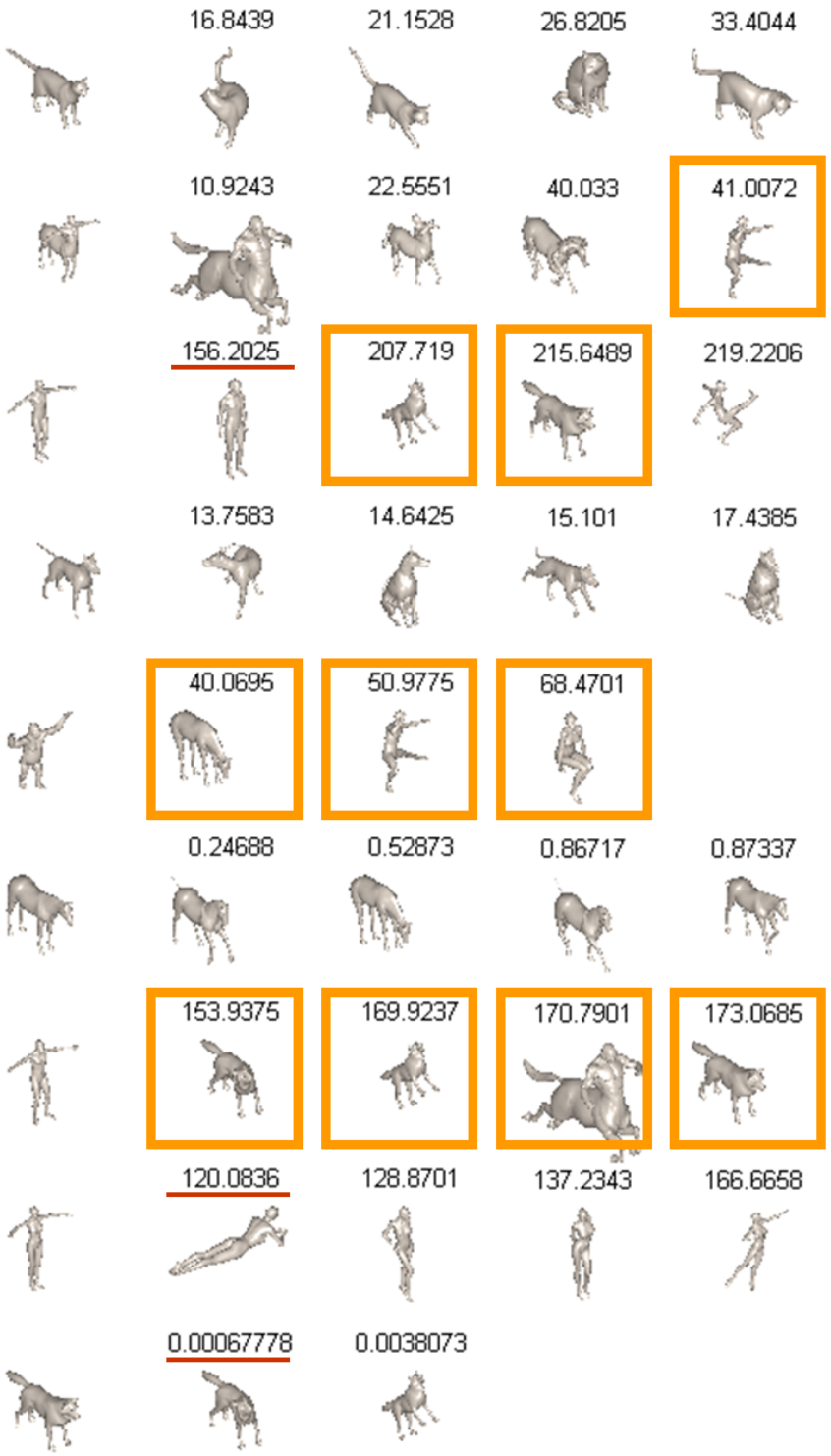


Figure 2.23: Top four retrieval shapes of spin image. Wrong retrievals are highlighted with orange boxes. Distance scores are erratic (see red underlines) inhibiting the determination of a cut-off value between the correct and incorrect shapes.

Chapter 3

Sparse Representations

3.1 Background on Sparse Coding and Dictionary Learning

Sparse and redundant signal representations have recently drawn much interest in vision, signal and image processing [76], [77], [78], [79]. This is due in part to the fact that signals and images of interest can be sparsely represented or compressible given an appropriate dictionary. In particular, we say a signal $\mathbf{y} \in \mathbb{R}^n$ is sparsely represented by a dictionary $\mathbf{D} \in \mathbb{R}^{n \times K}$ when it can be well approximated by a linear combination of a few columns of $\mathbf{D}$ as $\mathbf{y} \approx \mathbf{D}\mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^K$ is the sparse representation vector and $\mathbf{D}$ is a dictionary that contains a set of basics (atoms) as its columns. Finding a sparse representation vector entails solving the following optimization problem

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_0 \;\; s.t. \;\; \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2 \leq \varepsilon, \tag{3.1}$$

where ε is an error tolerance, $\|\mathbf{x}\|_0$ is the ℓ_0 sparsity measure that counts the number of nonzero elements in the vector $\mathbf{x}$, and $\|\mathbf{y} - \mathbf{D}\mathbf{x}\|_\mathbf{2}$ is the mean squared error resulted from the sparse approximation. Solving (3.1) is NP-hard and can be approximated by various methods [80], [81], [82].

Instead of using a pre-determined dictionary, one can directly learn a dictionary from the data. Indeed, it has been observed that learning a dictionary directly from the training data rather than using a predetermined dictionary (e.g. wavelet) usually leads

to a more compact representation and hence can provide better results in many image processing applications such as restoration and classification [76], [77], [78] [83], [84], [85].

Several algorithms have been developed for the task of learning a dictionary. Two of the most well-known algorithms are the method of optimal directions (MOD) [8] and the KSVD algorithm [9]. Given a set of N signals $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_N]$, the goal of KSVD and MOD algorithms is to find a dictionary $\mathbf{D}$ and a sparse matrix $\mathbf{X}$ that minimize the following representation error

$$\begin{aligned} (\hat{\mathbf{D}}, \hat{\mathbf{X}}) = \arg\min_{\mathbf{D},\mathbf{X}} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 \;\; s.t. \\ \|\mathbf{x}_i\|_0 \leq T_0, \; \forall i = 1 \ldots N \end{aligned} \tag{3.2}$$

where $\mathbf{x}_i$ represents the i-th column of $\mathbf{X}$, $\|\mathbf{A}\|_F$ denotes the Frobenius norm of $\mathbf{A}$, and T_0 denotes the sparsity level. Both MOD and KSVD are iterative methods that alternate between sparse-coding and dictionary update steps. First, a dictionary $\mathbf{D}$ with ℓ_2 normalized columns is initialized. Then, the main iteration is composed of the following two stages:

- *Sparse coding*: In this step, $\mathbf{D}$ is fixed and the following optimization problem is solved to compute the representation vector $\mathbf{x}_i$ for each example $\mathbf{y}_i$

$$\min_{\mathbf{x}_i} \|\mathbf{y}_i - \mathbf{D}\mathbf{x}_i\|_2^2 \;\; s.t. \;\; \|\mathbf{x}_i\|_0 \leq T_0, \quad \forall i = 1 \ldots N \tag{3.3}$$

- *Dictionary update*: This is where both MOD and KSVD algorithms differ. The MOD algorithm updates all the atoms simultaneously by solving an optimization problem whose solution is given by $\mathbf{D} = \mathbf{Y}\mathbf{X}^\dagger$, where $\mathbf{X}^\dagger$ denotes the Moore-Penrose pseudo-inverse. Even though the MOD algorithm is very effective and

usually converges in a few iterations, it suffers from the high complexity of the matrix inversion as discussed in [9]. In the case of KSVD, the dictionary update is performed atom-by-atom in an efficient way rather than using a matrix inversion. It has been observed that the KSVD algorithm requires fewer iterations to converge than the MOD method.

3.2 Non-Linear Kernel Dictionary Learning

The dictionaries designed by MOD and KSVD are based on a linear representation of the data. However, linear representations are almost always inadequate for representing non-linear structures of the data which arise in many practical applications. For example, many types of descriptors in computer vision have intrinsic non-linear similarity measure functions. The most popular ones include the spatial pyramid descriptor [10] which uses a pyramid match kernel, and the region covariance descriptor [11] which uses a Riemannian metric as the similarity measure between two descriptors. Both of these distance measures are highly non-linear. Unfortunately, the traditional dictionary learning methods, e.g. MOD and KSVD, are based on a linear model. This inevitably leads to poor performances for many datasets, e.g., object classification of Caltech-101 [12] dataset, even when discriminant power is taken into account during the training [13]. Motivated by the drawbacks of the current methods and the needs of many practical applications, we propose kernel dictionaries which are basically dictionaries in high dimensional feature spaces. Our dictionary learning methods yield representations that are more compact than kernel principal component analysis (KPCA) [14] and are able to handle the non-linearity

better their linear counterparts.

3.2.1 Problem Formulation

Let $\Phi : \mathbb{R}^n \rightarrow \mathcal{F} \subset \mathbb{R}^{\tilde{n}}$ be a non-linear mapping from $\mathbb{R}^n$ into a dot product space $\mathcal{F}$. We restrict our formulation to Hilbert spaces. The reason is because a Hilbert space associates each pair of vectors with a dot product. This allows the use of *Mercer* kernels to carry out computations implicitly without venturing into the high-dimensional feature space. It is worth noting that in practice $\tilde{n}$ is often much larger than n, and possibly infinite. More discussion on the use of Mercer kernels is delayed until later in this section.

Our goal is to learn a non-linear dictionary $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_K]$ in the feature space $\mathcal{F}$ by solving the following optimization problem:

$$\begin{aligned} &\arg\min_{\mathbf{D},\mathbf{X}} \ \|\Phi(\mathbf{Y}) - \mathbf{D}\mathbf{X}\|_F^2 \ \ s.t \\ &\|\mathbf{x}_i\|_0 \leq T_0, \ \ \|\mathbf{d}_j\|_2 = 1, \ \ \forall i, j. \end{aligned} \tag{3.4}$$

where $\Phi(\mathbf{Y}) = [\Phi(\mathbf{y}_1), \ldots, \Phi(\mathbf{y}_N)]$, $\mathbf{D} \in \mathbb{R}^{\tilde{n} \times K}$ is the sought dictionary, $\mathbf{X} \in \mathbb{R}^{K \times N}$ is a matrix whose i-th column is the sparse vector $\mathbf{x}_i$ corresponding to the sample $\Phi(\mathbf{y}_i)$, with maximum of T_0 non-zero entries.

The following proposition facilitates the re-formulation of (3.4). To the best of our knowledge, none of the previous work on dictionary learning has proven this property.

Proposition 3.1. *There exists an optimal solution* $\mathbf{D}^*$ *to (3.4) that has the following form:*

$$\mathbf{D}^* = \Phi(\mathbf{Y})\mathbf{A}, \tag{3.5}$$

for some $\mathbf{A} \in \mathbb{R}^{N \times K}$.

Table 3.1: Parameters' dimensions.

$\mathbf{Y}$	$\Phi(\mathbf{Y})$	$\mathbf{D}$	$\mathbf{A}$	$\mathbf{X}$	$\mathbf{d}_i$	$\mathbf{x}_i$
$n \times N$	$\tilde{n} \times N$	$\tilde{n} \times K$	$N \times K$	$K \times N$	$\tilde{n} \times 1$	$K \times 1$

Proof. See Appendix 3.5. □

At first sight, the solution in (5.9) might look similar to the double-sparsity model $\mathbf{D} = \mathbf{BA}$ proposed in [86]. However, there are significant differences between the two models. First, we prove in the proposition 3.1 that $\mathbf{B} = \Phi(\mathbf{Y})$ is a natural choice for our case instead of relying on any manual selection of $\mathbf{B}$. In addition, we do not require columns of $\mathbf{D}$ to be sparse with respect to the base $\mathbf{B}$ as in [86]. Although $\mathbf{B}$ is fixed in both models, the dictionary can be tuned to the training data via modifying the coefficient matrix $\mathbf{A}$. For the reader's convenience, table 3.1 summarizes the dimensions of all important parameters. Parameters that involve $\tilde{n}$ are those associated with the feature space $\mathcal{F}$. As a result, we can seek an optimal dictionary through optimizing $\mathbf{A}$ instead of $\mathbf{D}$. By substituting (5.9) into (3.4), the problem can be re-written as follows:

$$\begin{aligned} \underset{\mathbf{A},\mathbf{X}}{\arg\min} \ \ & \|\Phi(\mathbf{Y}) - \Phi(\mathbf{Y})\mathbf{AX}\|_F^2 \ \ s.t. \\ & \|\mathbf{x}_i\|_0 \leq T_0, \forall i = 1 \dots N. \end{aligned} \tag{3.6}$$

An important property of this formulation is that the objective function in (3.6) can be minimized through the use of Mercer kernels as we shall see shortly in the next section.

A Mercer kernel is a function $\kappa(\mathbf{x}, \mathbf{y})$ that satisfies Mercer's condition [87] [88]: For all data $\{\mathbf{y}_i\}_{i=1}^N$, the function gives rise to a positive semi-definite matrix $[\boldsymbol{\mathcal{K}}_{ij}] =$

$[\kappa(\mathbf{y}_i, \mathbf{y}_j)]$. If κ satisfies the Mercer' condition, it can be shown that κ corresponds to some mapping Φ in the Hilbert feature space $\mathcal{F}$. That is, $\kappa(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle$. Mercer kernels are often used to implicitly specify the mapping Φ. It helps avoid the computationally expensive mapping of data into the high-dimensional feature space. Some commonly used kernels include polynomial kernels $\kappa(\mathbf{x}, \mathbf{y}) = \langle (\mathbf{x}, \mathbf{y} \rangle + c)^d$ and Gaussian kernels $\kappa(\mathbf{x}, \mathbf{y}) = \exp(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{c})$, where c and d are the parameters.

In order to see the advantage of the formulation in (3.6) over the original one, we will examine the objective function. Through some algebraic manipulations, the cost function can be re-written as:

$$\begin{aligned} &\|\Phi(\mathbf{Y}) - \Phi(\mathbf{Y})\mathbf{A}\mathbf{X}\|_F^2 = \|\Phi(\mathbf{Y})(\mathbf{I} - \mathbf{A}\mathbf{X})\|_F^2 = \\ &\mathbf{tr}((\mathbf{I} - \mathbf{A}\mathbf{X})^T \Phi(\mathbf{Y})^T \Phi(\mathbf{Y})(\mathbf{I} - \mathbf{A}\mathbf{X})) \\ &\mathbf{tr}((\mathbf{I} - \mathbf{A}\mathbf{X})^T \boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y})(\mathbf{I} - \mathbf{A}\mathbf{X})), \end{aligned} \tag{3.7}$$

where $\boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y})$ is a kernel matrix whose elements are computed from $\kappa(\mathbf{y}_i, \mathbf{y}_j) = \Phi(\mathbf{y}_i)^T \Phi(\mathbf{y}_j)$. It is more efficient to optimize this objective function since it only involves a kernel matrix of finite dimension $\boldsymbol{\mathcal{K}} \in \mathbb{R}^{N \times N}$, instead of dealing with a possibly infinite dimensional dictionary in the original formulation (3.4).

In order to illustrate this point, we take an example where training data are images of size 16×16 pixels. That is, $n = 256$. Assuming that a polynomial kernel of degree 4 is used, then the dimension of the feature space is in order of $\tilde{n} \sim \binom{256}{4} \sim 10^8$. This is often much larger than the number of training samples. For instance, Caltech-256 dataset only contains $n = 30607$ images for both training and testing. If a Gaussian kernel is used, the corresponding feature space is the Hilbert space of infinite dimension. Therefore, it is computationally prohibited to carry out computations directly on feature

spaces. Instead, we can work with the kernel matrix as in (3.7) and achieve a better computational efficiency.

3.2.2 Optimization Procedure

In this section, we present our approach for learning dictionaries in the feature space. Just as in the case of KSVD and MOD, our method of learning dictionaries involve two stages: sparse coding and dictionary update. In what follows, we describe them in detail.

3.2.2.1 Kernel Orthogonal Matching Pursuit (KOMP)

In this stage, the matrix $\mathbf{A}$ is assumed to be fixed and then we seek for the sparse codes contained in the matrix $\mathbf{X}$. Note that, the penalty term in (3.6) can be re-written as

$$\|\Phi(\mathbf{Y}) - \Phi(\mathbf{Y})\mathbf{A}\mathbf{X}\|_F^2 = \sum_{j=1}^{N} \|\Phi(\mathbf{y}_i) - \Phi(\mathbf{Y})\mathbf{A}\mathbf{x}_i\|_2^2. \tag{3.8}$$

Hence, (3.6) can be reformulated as solving N different problems of the following form

$$\min_{\mathbf{x}_i} \ \|\Phi(\mathbf{y}_i) - \Phi(\mathbf{Y})\mathbf{A}\mathbf{x}_i\|_2^2 \ \ s.t \ \ \|\mathbf{x}_i\|_0 \leq T_0, \tag{3.9}$$

for $i = 1, \cdots, N$. The above problem can be solved by any pursuit algorithms [80, 81], with the modification that signals are now in the feature space with a very high dimensionality. In what follows, we show how the well-known orthogonal matching pursuit algorithm (OMP) [81, 82] can be generalized using kernels to solve (3.11).

In this stage, the matrix $\mathbf{A}$ is assumed to be fixed and then we seek for the sparse

codes contained in the matrix $\mathbf{X}$. Note that, the penalty term in (3.6) can be re-written as

$$\|\Phi(\mathbf{Y}) - \Phi(\mathbf{Y})\mathbf{A}\mathbf{X}\|_F^2 = \sum_{j=1}^{N} \|\Phi(\mathbf{y}_i) - \Phi(\mathbf{Y})\mathbf{A}\mathbf{x}_i\|_2^2. \tag{3.10}$$

Hence, (3.6) can be reformulated as solving N different problems of the following form

$$\min_{\mathbf{x}_i} \ \|\Phi(\mathbf{y}_i) - \Phi(\mathbf{Y})\mathbf{A}\mathbf{x}_i\|_2^2 \ \ s.t \ \ \|\mathbf{x}_i\|_0 \leq T_0, \tag{3.11}$$

for $i = 1, \cdots, N$. The above problem can be solved by any pursuit algorithms [80, 81], with the modification that signals are now in the feature space with a very high dimensionality. In what follows, we show how the well-known orthogonal matching pursuit algorithm (OMP) [81, 82] can be generalized using kernels to solve (3.11).

Given a signal $\mathbf{z} \in \mathbb{R}^n$ and the kernel dictionary represented via $\mathbf{A}$, we seek a sparse combinations of dictionary atoms that approximate $\mathbf{z}$ in the feature space:

$$\Phi(\mathbf{z}) = \Phi(\mathbf{Y})\mathbf{A}\mathbf{x}_s + \mathbf{r}_s = \Phi(\mathbf{Y})\boldsymbol{\nu}_s + \mathbf{r}_s. \tag{3.12}$$

Here, $\boldsymbol{\nu}_s = \mathbf{A}\mathbf{x}_s \in \mathbb{R}^N$ is an intermediate notation. We can also think of it as the current loading of the signal $\Phi(\mathbf{z})$ over a base $\Phi(\mathbf{Y})$. Finally, $\mathbf{r}_s$ is the current residual.

The pseudo-code for KOMP is given in the Fig. 3.1. Let I_s denote the set of indices of the atoms that have already been selected. In the first step, the residual is projected onto the remaining dictionary atoms $\mathbf{d}_i = \Phi(\mathbf{Y})\mathbf{a}_i$, where $i \notin I_s$ and $\mathbf{a}_i$ is the i-th column of $\mathbf{A}$. Let τ_i denote the projection of the residual onto $\mathbf{d}_i$, which is computed simply by:

$$\begin{aligned} \tau_i = \mathbf{r}_s^T(\Phi(\mathbf{Y})\mathbf{a}_i) &= \big(\Phi(\mathbf{z}) - \Phi(\mathbf{Y})\boldsymbol{\nu}_s\big)^T\big(\Phi(\mathbf{Y})\mathbf{a}_i\big) \\ &= \big(\boldsymbol{\mathcal{K}}(\mathbf{z}, \mathbf{Y}) - \boldsymbol{\nu}_s^T\boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y})\big)\mathbf{a}_i, \quad i \notin I_s \end{aligned} \tag{3.13}$$

Input: A signal $\mathbf{z}$, a kernel function κ, a coefficient matrix $\mathbf{A}$, and a sparsity level T_0.

Task: Find a coefficient vector $\mathbf{x} \in \mathbb{R}^K$ with at most T_0 non-zero coefficients such that $\|\Phi(\mathbf{Y})\mathbf{A}\mathbf{x} - \Phi(\mathbf{z})\|^2$ is minimized.

Initialize: $s = 0$, $I_0 = \emptyset$, $\mathbf{x}_0 = \mathbf{0}$, $\boldsymbol{\nu}_0 = \mathbf{0}$

Procedure:

1. $\tau_i = \left(\boldsymbol{\mathcal{K}}(\mathbf{z}, \mathbf{Y}) - \boldsymbol{\nu}_s^T \boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y})\right) \mathbf{a}_i, \quad \forall i \notin I_{s-1}$

2. $i_{max} = \arg\max_i |\tau_i|, \ \forall i \notin I_{s-1}$

3. Update the index set $I_s = I_{s-1} \bigcup i_{max}$

4. $\mathbf{x}_s = \left(\mathbf{A}_{I_s}^T \boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y}) \mathbf{A}_{I_s}\right)^{-1} \left(\boldsymbol{\mathcal{K}}(\mathbf{z}, \mathbf{Y}) \mathbf{A}_{I_s}\right)^T$

5. $\boldsymbol{\nu}_s = \mathbf{A}_{I_s} \mathbf{x}_s$

6. $s \leftarrow s + 1$; Repeat steps 1–6 T_0 times

Output: Sparse vector $\mathbf{x} \in \mathbb{R}^K$ satisfying $\mathbf{x}(I_s(j)) = \mathbf{x}_s(j), \forall j \in I_s$ and zero elsewhere.

Figure 3.1: The KOMP algorithm.

where, with a slight abuse of notation, we denote

$$\boldsymbol{\mathcal{K}}(\mathbf{z}, \mathbf{Y}) = [\kappa(\mathbf{z}, \mathbf{y_1}), \kappa(\mathbf{z}, \mathbf{y_2}), \ldots, \kappa(\mathbf{z}, \mathbf{y_N})]. \tag{3.14}$$

The algorithm then selects a new dictionary atom in the remaining set that gives the largest projection coefficient in (3.13). This selection guarantees the biggest reduction of the approximation error.

In the fourth step of Fig. 3.1, we want to update the coefficients in $\mathbf{x}$ corresponding

to the selected indices in I_s. Let $\mathbf{x}_s$ denote a vector formed by removing from $\mathbf{x}$ all coefficients whose indices do not belong to I_s. Similarly, let $\mathbf{A}_{I_s}$ indicate the set of dictionary atoms whose indices are from the set I_s. Then $\mathbf{x}_s$ is obtained by projecting the signal $\Phi(\mathbf{z})$ onto the subspace spanned by the selected dictionary atoms $\Phi(\mathbf{Y})\mathbf{A}_{I_s}$. Note that the matrix

$$(\Phi(\mathbf{Y})\mathbf{A}_{I_s})^T(\Phi(\mathbf{Y})\mathbf{A}_{I_s}) = \mathbf{A}_{I_s}^T\boldsymbol{\mathcal{K}}(\mathbf{Y},\mathbf{Y})\mathbf{A}_{I_s} \tag{3.15}$$

is positive semi-definite since $\boldsymbol{\mathcal{K}}(\mathbf{Y},\mathbf{Y})$ is positive semi-definite. The projection coefficients are obtained as follows:

$$\begin{aligned}\mathbf{x}_s &= \big((\Phi(\mathbf{Y})\mathbf{A}_{I_s})^T(\Phi(\mathbf{Y})\mathbf{A}_{I_s})\big)^{-1}\big(\Phi(\mathbf{Y})\mathbf{A}_{I_s}\big)^T\Phi(\mathbf{z}) \\ &= \big(\mathbf{A}_{I_s}^T\boldsymbol{\mathcal{K}}(\mathbf{Y},\mathbf{Y})\mathbf{A}_{I_s}\big)^{-1}\big(\boldsymbol{\mathcal{K}}(\mathbf{z},\mathbf{Y})\mathbf{A}_{I_s}\big)^T\end{aligned} \tag{3.16}$$

Note also that the computation in (3.16) can be efficiently implemented in a recursive manner as in [81]. Once the coefficients $\mathbf{x_s}$ are found, the representation $\boldsymbol{\nu}_s$ are updated as in step 5 of Fig. 3.1. We can think of $\boldsymbol{\nu}_s$ as the current loading vector of the signal $\Phi(\mathbf{z})$ over the base $\Phi(\mathbf{Y})$. The procedure is repeated until T_0 atoms are selected.

3.2.3 Dictionary Update with Kernel MOD

Once the sparse codes for each training data are calculated, the dictionary is updated such that the error,

$$\|\Phi(\mathbf{Y}) - \Phi(\mathbf{Y})\mathbf{A}\mathbf{X}\|_F^2 \tag{3.17}$$

is minimized. Note that unlike the traditional dictionary learning framework, a kernel dictionary is learned by optimizing the coefficient matrix $\mathbf{A}$ instead of the dictionary $\mathbf{D}$. Taking the derivative of this error with respect to $\mathbf{A}$ and after some manipulations, we

have:

$$\begin{aligned} &\frac{\partial \|\Phi(\mathbf{Y}) - \Phi(\mathbf{Y})\mathbf{A}\mathbf{X}\|_F^2}{\partial \mathbf{A}} = \\ &\frac{\partial \mathbf{tr}((\mathbf{I} - \mathbf{A}\mathbf{X})^T \boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y})(\mathbf{I} - \mathbf{A}\mathbf{X}))}{\partial \mathbf{A}} = \\ &-2\boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y})(\mathbf{I} - \mathbf{A}\mathbf{X})\mathbf{X}^T \end{aligned} \tag{3.18}$$

By setting (3.18) to zero, we obtain the relation $\mathbf{A} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}$ which leads to the following iterative update:

$$\mathbf{A}_{k+1} = \mathbf{X}_k^T(\mathbf{X}_k\mathbf{X}_k^T)^{-1} = \mathbf{X}_k^{\dagger}. \tag{3.19}$$

where the subscript k denotes iteration index. This procedure of updating the dictionary is essentially the idea behind the MOD method [8]. We call this kernel MOD, since this is done in the feature space. As discussed in [9], one of the major drawbacks of MOD is that it suffers from the high complexity of matrix inversion during the dictionary update stage. Specifically, the inversion of matrix $\mathbf{X}_k\mathbf{X}_k^T$ has the complexity of $\mathcal{O}(K^3)$, where K is the number of dictionary atoms. This becomes very expensive as the dictionary size is usually large. Several other methods have also been proposed that focus on reducing this complexity. One such algorithm is KSVD [9]. This algorithm eliminates the need of a large-matrix inversion and converges faster than MOD [9]. Following the procedure of KSVD, in what follows, we describe a more sophisticated way of updating the dictionary.

3.2.4 Dictionary Update with Kernel KSVD

Let $\mathbf{a}_k$ and $\mathbf{x}_T^j$ denote the k-th column of $\mathbf{A}$ and the j-th row of $\mathbf{X}$, respectively. The error

$$\|\Phi(\mathbf{Y}) - \Phi(\mathbf{Y})\mathbf{A}\mathbf{X}\|_F^2 \tag{3.20}$$

Input: A set of signals $\mathbf{Y}$, a kernel function κ.

Task: Find a dictionary via $\mathbf{A}$ to represent these signals as sparse decompositions in the feature space by solving the optimization problem in (3.6).

Initialize: Set a random element of each column in $\mathbf{A}^{(0)}$ to be 1. Normalize each column in the initial dictionary to a unit norm. Set iteration $J = 1$.

Stage 1: *Sparse coding*

Use the KOMP algorithm described in Fig. 3.1 to obtain the sparse coefficient matrix $\mathbf{X}^{(J)}$ given a fixed dictionary $\mathbf{A}^{(J-1)}$.

Stage 2: *Dictionary update*

For each column $\mathbf{a}_k^{(J-1)}$ in $\mathbf{A}^{(J-1)}$, where $k = 1, \ldots, K$, update it by

- Define the group of examples that use this atom: $\omega_k = \{i | 1 \leq i \leq N, \mathbf{x}_T^k(i) \neq 0\}$

- Define the representation error matrix, $\mathbf{E}_k$, by (3.24).

- Restrict $\mathbf{E}_k$ by choosing only the columns corresponding to ω_k by: $\mathbf{E}_k^R = \mathbf{E}_k \Omega_k$

- Apply SVD decomposition to get

$$(\mathbf{E}_k^R)^T \boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y})(\mathbf{E}_k^R) = \mathbf{V} \triangle \mathbf{V}^T.$$

Choose updated $\mathbf{a}_k^{(J)} = \sigma_1^{-1} \mathbf{E}_k^R \mathbf{v}_1$, where $\mathbf{v}_1$ is the first vector of $\mathbf{V}$ corresponding to the largest singular value σ_1^2 in $\triangle$.

- Set $J = J + 1$. Repeat from stage 1 until a stopping criterion is met.

Output: $\mathbf{A}$ and $\mathbf{X}$.

Figure 3.2: The kernel KSVD algorithm.

can be re-written as:

$$\left\| \Phi(\mathbf{Y}) - \Phi(\mathbf{Y}) \sum_{j=1}^{K} \mathbf{a}_j \mathbf{x}_T^j \right\|_F^2 \tag{3.21}$$

$$= \left\| \Phi(\mathbf{Y}) \left(\mathbf{I} - \sum_{j \neq k} \mathbf{a}_j \mathbf{x}_T^j \right) - \Phi(\mathbf{Y})(\mathbf{a}_k \mathbf{x}_T^k) \right\|_F^2 \tag{3.22}$$

$$= \|\Phi(\mathbf{Y})\mathbf{E}_k - \Phi(\mathbf{Y})\mathbf{M}_k\|_F^2 \tag{3.23}$$

where,

$$\mathbf{E}_k = \left(\mathbf{I} - \sum_{j \neq k} \mathbf{a}_j \mathbf{x}_T^j \right), \quad \mathbf{M}_k = (\mathbf{a}_k \mathbf{x}_T^k). \tag{3.24}$$

$\Phi(\mathbf{Y})\mathbf{E}_k$ indicates the error between the approximated signals and the true signals when removing the k-th dictionary atom. $\Phi(\mathbf{Y})\mathbf{M}_k$ indicates the contribution of the k-th dictionary atom to the estimated signals.

At this stage, we assume that only ($\mathbf{a}_k$,$\mathbf{x}_T^k$) are variables and the rest are fixed, hence $\mathbf{E}_k$ is also constant for each k. The minimization of the above problem is equivalent to finding $(\mathbf{a}_k, \mathbf{x}_T^k)$ such that the rank-1 matrix $\Phi(\mathbf{Y})\mathbf{M}_k$ best approximates $\Phi(\mathbf{Y})\mathbf{E}_k$ in terms of mean squared error. The optimal solution can be obtained via a singular value decomposition (SVD). However, there are two reasons that make the direct SVD inappropriate. Firstly, it would yield a dense vector $\mathbf{x}_T^k$, which increases the number of non-zeros in the representation $\mathbf{X}$. Secondly, the matrix might have infinitely large row dimension, which is computationally prohibitive.

In order to minimize the objective function while keeping the sparsity of all the representations fixed, we work only with a subset of columns. Note that the columns of $\mathbf{M}_k$ associated with zero-value elements of $\mathbf{x}_T^k$ are all zero. These columns do not affect the minimization of the objective function. Therefore, we can shrink the matrices $\mathbf{E}_k$

and $\mathbf{M}_k$ by discarding these zero columns. An advantage of working with the reduced matrices is that only the non-zero coefficients in $\mathbf{x}_T^k$ are allowed to vary and therefore the sparsity are preserved [9].

Define ω_k as the group of indices pointing to examples $\{\Phi(\mathbf{y}_i)\}$ that use the atom $(\Phi(\mathbf{Y})\mathbf{A})_k$:

$$\omega_k = \{i | 1 \leq i \leq N,\ \mathbf{x}_T^k(i) \neq 0\}. \tag{3.25}$$

Let Ω_k be a matrix of size $N \times |\omega_k|$, with ones on the $(\omega_k(i), i)$-th entries and zeros elsewhere. When multiplying with Ω_k, all zeros within the row vector $\mathbf{x_T^k}$ will be discarded resulting in the row vector $\mathbf{x_R^k}$ of the length $|\omega_k|$. The column-reduced matrices are obtained as

$$\mathbf{E}_k^R = \mathbf{E}_k \Omega_k; \quad \mathbf{M}_k^R = \mathbf{M}_k \Omega_k. \tag{3.26}$$

We can now modify the cost function in (3.21) so that its solution has the same support with the original $\mathbf{x_T^k}$:

$$\begin{aligned} &\left\| \Phi(\mathbf{Y})\mathbf{E}_k^R - \Phi(\mathbf{Y})\mathbf{M}_k^R \right\|_F^2 = \\ &\left\| \Phi(\mathbf{Y})\mathbf{E}_k^R - \Phi(\mathbf{Y})\mathbf{a}_k\mathbf{x_R^k} \right\|_F^2. \end{aligned} \tag{3.27}$$

Recall the fact that $\Phi(\mathbf{Y})\mathbf{a}_k\mathbf{x_R^k}$ is a rank-1 matrix. Therefore, the optimal solution of (3.27) can be obtained by first decomposing $\Phi(\mathbf{Y})\mathbf{E}_k^R$ into rank-1 matrices using SVD, and then equating $\Phi(\mathbf{Y})\mathbf{a}_k\mathbf{x_R^k}$ to the rank-1 matrix corresponding to the largest singular value. That is,

$$\Phi(\mathbf{Y})\mathbf{E}_k^R = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{3.28}$$

$$\Phi(\mathbf{Y})\mathbf{a}_k\mathbf{x_R^k} = \sigma_\mathbf{1}\mathbf{u_1}\mathbf{v_1^T}, \tag{3.29}$$

where $\mathbf{u}_1$ and $\mathbf{v}_1$ are the first columns of $\mathbf{U}$ and $\mathbf{V}$ corresponding to the largest singular value $\sigma_1 = \mathbf{\Sigma}(1,1)$, respectively. A valid breakdown for the assignment (3.29) is given. The reason for putting the multiplier σ_1 in (3.30) instead of in (3.31) will become clearer when solving for $\mathbf{a}_k$. Basically, such an assignment guarantees that the resulting dictionary atom on the feature space is normalized to unit-norm

$$\mathbf{x}_{\mathbf{R}}^{\mathbf{k}} = \sigma_1 \mathbf{v}_1^T \tag{3.30}$$

$$\Phi(\mathbf{Y})\mathbf{a}_k = \mathbf{u}_1. \tag{3.31}$$

However, as mentioned before, we can not perform direct SVD decomposition on $\Phi(\mathbf{Y})\mathbf{E}_k^R$ as in (3.28) since this matrix can have infinitely large row dimension. This is an important difference between the linear KSVD and kernel KSVD algorithm. A remedy for this issue comes from the fact that SVD decomposition is closely related to eigen-decomposition of the Gram matrix, which is independent of the row dimension.

$$\begin{aligned} \left(\Phi(\mathbf{Y})\mathbf{E}_k^R\right)^T \left(\Phi(\mathbf{Y})\mathbf{E}_k^R\right) = \\ \left(\mathbf{E}_k^R\right)^T \boldsymbol{\mathcal{K}}(\mathbf{Y},\mathbf{Y})(\mathbf{E}_k^R) = \mathbf{V} \triangle \mathbf{V}^T, \end{aligned} \tag{3.32}$$

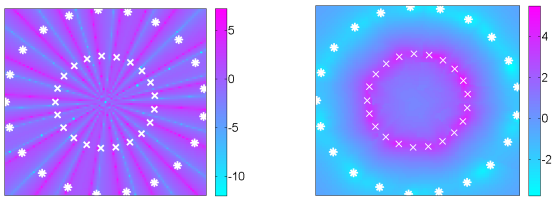
where $\triangle = \mathbf{\Sigma}^T\mathbf{\Sigma} \in \mathbb{R}^{N\times N}$. This gives us $\mathbf{v}_1$ as the first column of $\mathbf{V}$, and $\sigma_1 = \sqrt{\triangle(\mathbf{1},\mathbf{1})}$. Hence, $\mathbf{x}_{\mathbf{R}}^{\mathbf{k}}$ can be found using the relation in (3.30).

To solve for $\mathbf{a}_k$, we first observe that by right-multiplying both sides of (3.28) by $\mathbf{V}$ and considering only the first column, we get

$$\Phi(\mathbf{Y})\mathbf{E}_k^R\mathbf{v}_1 = \sigma_1\mathbf{u}_1. \tag{3.33}$$

The solution for $\mathbf{a}_k$ is obtained by substituting (3.31) into (3.33), yielding:

$$\Phi(\mathbf{Y})\mathbf{E}_k^R\mathbf{v}_1 = \sigma_1\Phi(\mathbf{Y})\mathbf{a}_k. \tag{3.34}$$



(a) (b)

Figure 3.3: Comparison of error ratios for (a) KSVD and (b) Kernel KSVD on a common logarithm scale.

Hence,

$$\mathbf{a}_k = \sigma_1^{-1}\mathbf{E}_k^R\mathbf{v}_1. \tag{3.35}$$

One can easily verify that this updating procedure of $\mathbf{a}_k$ results in a dictionary atom of unit-norm in the feature space. The pseudo-code for the kernel KSVD algorithm is given in Fig. 3.2.

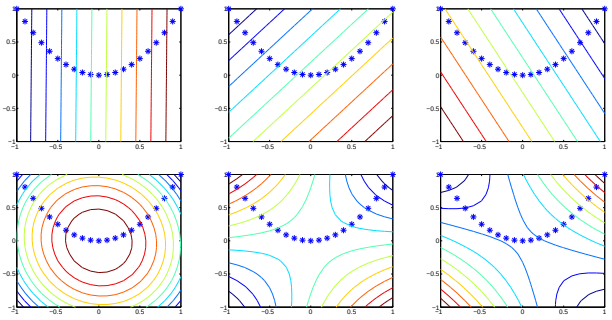


Figure 3.4: Comparison between the level curves of the projection coefficients for three different dictionary atoms corresponding to linear KSVD and kernel KSVD. In this figure, the first row corresponds to KSVD and the second row corresponds to kernel KSVD.

3.3 Experimental Results

In this section, we present several experimental results demonstrating the effectiveness of the proposed dictionary learning method for classification tasks. In particular, we present classification results on the USPS digit dataset and the Caltech-101 object dataset. We first examine the effectiveness of a learned dictionary in the feature space using synthetic data.

3.3.1 Synthetic Data

In the first set of experiments, we generate 1500 samples for a two-class problem by randomly sampling a two dimensional circle $\{\mathbf{y} = [y_1, y_2] \in \mathbb{R}^2 \mid y_1^2 + y_2^2 = r^2\}$. The radius r of the first circle (class 1) is a half that of the second circle (class 2). The radius of the second circle is set to 1 in this experiment. Separate dictionaries are learned for different classes and different methods. This means that we learn two different dictionaries for two classes using KSVD, and two kernel dictionaries using kernel KSVD. The following parameters are used to learn dictionaries using both KSVD and kernel KSVD: dictionaries are learned with 30 atoms, $T_0 = 3$, and the maximum number of training iterations is set to 80. For kernel KSVD, we use a polynomial kernel of degree 2.

Fig. 3.3 shows the color-coded map of the error ratios obtained by dividing reconstruction errors of the second class to that of the first class for all points on the $\mathbb{R}^2$ plane. The reconstruction errors are computed when the sparse coding only use one dictionary atom (i.e. $T_0 = 1$). Since data samples from the two classes lie roughly on the same linear subspace, which is the entire plane in $\mathbb{R}^2$, dictionaries learned using KSVD are indistin-

guishable for the two classes. This is clearly seen from Fig. 3.3 (a), where the error ratios are quite random even for points lying on the circles.

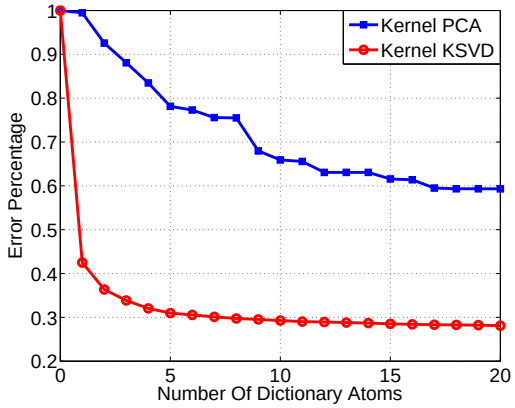


Figure 3.5: Comparison of error percentage using kernel KSVD and kernel PCA.

On the other hand, as can be seen from Fig. 3.3 (b), error ratios corresponding to a dictionary learned in the feature space exhibit strong differences between the two classes. In particular, error ratios are very high for points lying close to the first class, and very small for points lying close to the second class. Moreover, points on the same circle have similar error ratios. This observation implies that kernel KSVD correctly learns the non-linear structure of the data and embeds this information into kernel dictionary atoms.

In the second set of synthetic experiments, we learn dictionaries from 1500 data samples that are generated from a 2-dimensional parabola $\{\mathbf{y} = [y_1, y_2] \in \mathbb{R}^2 \mid y_2 = y_1^2\}$. All parameters are set to the same values as before. Fig. 3.4 shows the level curves of the projection coefficients for three different dictionary atoms. The curves are obtained as follows. First, we project every point $\mathbf{y} \in \mathbb{R}^2$ onto the selected dictionary atom $\mathbf{d}_i$ to get the projection coefficients. Then, the points with the same projection coefficients are shown with the same color map. In other words, every contour in Fig. 3.4 is a collection

of points with the same projection coefficients onto a dictionary atom. For kernel KSVD, the projection coefficients for a specific $\mathbf{y}$ is computed on the feature space:

$$\Phi(\mathbf{y})^T \mathbf{d}_i = \Phi(\mathbf{y})^T \Phi(\mathbf{Y}) \mathbf{a}_i = \boldsymbol{\mathcal{K}}(\mathbf{y}, \mathbf{Y}) \mathbf{a}_i \tag{3.36}$$

Here, $\mathbf{Y}$ denotes the set of points sampled along the parabola and $\mathbf{y}$ is any point in $\mathbb{R}^2$. The direction in which the projection coefficient varies (e.g. the direction in which the contour's color changes) indicates the direction of the dictionary atom. For instance, in the top left image of Fig. 3.4, the KSVD dictionary atom points in the horizontal direction, from left to right. Similarly, for the second and third images in the first row of Fig 3.4, the KSVD dictionary atoms point diagonally.

In contrast to KSVD coefficients, the coefficients of kernel KSVD (second row) in Fig. 3.4 change in non-linear manners. Each kernel dictionary atom captures variations along both branches of the parabola. This observation implies that our dictionary learning method can capture nonlinearities within the data and provide more compact representations.

3.3.2 Kernel sparse representation

In this section, we examine the representation obtained by kernel KSVD and kernel PCA using the USPS dataset. Fig. 3.5 compares the mean-squared-error (MSE) of an image from the USPS dataset when approximated using the first m dominant kernel PCA components and $m = [1, \ldots, 20]$ kernel dictionary atoms (i.e. $T_0 = m$). It is clearly seen that the MSE decays much faster for kernel KSVD than kernel PCA with respect to the number of selected bases. This observation implies that the image is *nonlinearly*

sparse and learning a dictionary in the high dimensional feature space can provide a better representation of data.

3.3.3 Digit Recognition

We apply our dictionary learning approach on the real-world handwritten digits classification problem. We use the USPS database [89] which contains ten classes of 256-dimensional handwritten digits. In other words, input training samples are the vectorizations of USPS digit images with the dimension of $n = 256$. For each class, we randomly select $N = 500$ samples for training and $N_{test} = 200$ samples for testing. The selection of parameters is done through a 5-fold cross-validation. Specifically, we choose the following parameters for learning the dictionaries for all classes: each class dictionary is learned with $K = 300$ atoms, $T_0 = 5$, maximum number of training iterations is set to 80. A polynomial kernel of degree 4 is used for both kernel KSVD and kernel PCA unless otherwise stated.

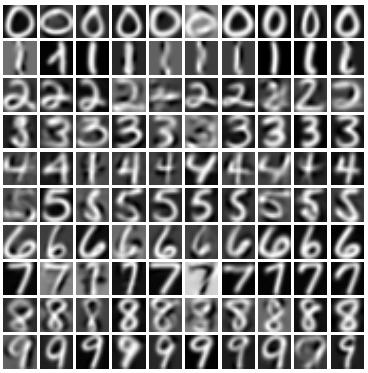


Figure 3.6: The pre-images of the kernel KSVD dictionary atoms.

We first use a sparse subspace approach for classification. Let $\mathbf{Y}_i = [\mathbf{y}_{i,1} \dots \mathbf{y}_{i,N}] \in$

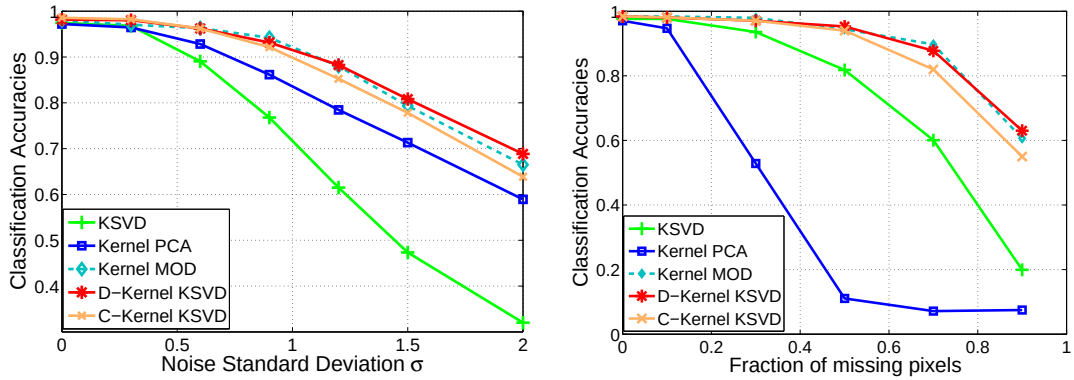
$\mathbb{R}^{256\times500}$ represents the set of training samples of the i-th class, where $i \in \{1 \dots 10\}$. Let $\mathbf{D}_i = \Phi(\mathbf{Y}_i)\mathbf{A}_i$ denote the learned kernel dictionary for each class, where $\mathbf{A}_i \in \mathbb{R}^{500\times300}$. Given a query image $\mathbf{z} \in \mathbb{R}^{256}$, we first perform KOMP separately for each $\mathbf{D}_i$, as in Fig. 3.1, to get the sparse code $\mathbf{x_i}$. Since sparse coding is done separately for each class, we call this a *distributive* approach. The sparse setting is the same as the training phase, i.e., $T_0 = 5$. The reconstruction error is then computed as:

$$r_i = ||\Phi(\mathbf{z}) - \Phi(\mathbf{Y}_i)\mathbf{A}_i\mathbf{x_i}||_2^2 = \boldsymbol{\mathcal{K}}(\mathbf{z},\mathbf{z}) - 2\boldsymbol{\mathcal{K}}(\mathbf{z},\mathbf{Y}_i)\mathbf{A}_i\mathbf{x_i}$$
$$\dots + \mathbf{x_i}^T\mathbf{A}_i^T\boldsymbol{\mathcal{K}}(\mathbf{Y}_i,\mathbf{Y}_i)\mathbf{A}_i\mathbf{x_i}\ ,\ \forall i = [1 \dots 10] \tag{3.37}$$

The test sample is simply classified to the class that gives the smallest reconstruction error.

We also evaluate the classification performance using a *collective* approach where dictionaries from all classes are concatenated to form a common dictionary $\mathbf{D} = [\mathbf{D}_1 \dots \mathbf{D}_{10}]$. This approach is different from the first approach only in the sparse coding step. For a given test sample $\mathbf{z}$, the sparse coding is done jointly using KOMP on $\mathbf{D}$ to obtain a sparse coefficient $\mathbf{x} = [\mathbf{x_1}, \dots, \mathbf{x_{10}}] \in \mathbb{R}^{\mathbf{3000}}$, where $\mathbf{x}_i$ contains the sparse coefficients associated with the dictionary $\mathbf{D}_i$. The sparsity level T_0 is chosen to be 10, which is the setting that yields the best performance on a 5-fold cross validation. The residual of each class is computed as in (3.37). The test sample is classified in a similar fashion with that of the distributive approach. Our method can be considered as the kernel version of [90].

For kernel PCA, we follow the same approach as in [14] to perform classification. First, we aggregate training samples from all classes. This results in a training set of 5000 samples $\mathbf{Y} = [\mathbf{Y}_1, \dots, \mathbf{Y}_{10}] \in \mathbb{R}^{256\times5000}$. Training samples are projected onto the first



(a) Gaussian noise

(b) Missing pixels

Figure 3.7: Comparison of digit recognition accuracies for different methods in the presence of Gaussian noise and missing-pixel effects. Red color and orange color represent the distributive and collective classification approaches for kernel KSVD, respectively.

3000 principal components of largest eigenvalues, which are found by applying the kernel PCA method on $\mathbf{Y}$. Classification is then done by learning a linear SVM classifier trained on the 3000-dimensional projection coefficients.

3.3.3.1 Pre-images of learned atoms

In order to visualize the learned kernel dictionary atoms, we find images that lie on the input space and best approximate these atoms in terms of the mean square errors. Recall that the k-th kernel dictionary atom of the i-th class is represented by $\Phi(\mathbf{Y}_i)\mathbf{a}_{i,k}$, where $\mathbf{a}_{i,k} \in \mathbb{R}^N$ is the representation of the kernel dictionary atom with respect to the base $\Phi(\mathbf{Y}_i)$ in the feature space $\mathcal{F}$. The pre-image of $\Phi(\mathbf{Y}_i)\mathbf{a}_{i,k}$ is obtained by seeking a vector in the input space $\mathbf{d}_{i,k} \in \mathbb{R}^n$ that minimizes the cost function $||\Phi(\mathbf{d}_{i,k}) - \Phi(\mathbf{Y}_i)\mathbf{a}_{i,k}||_2^2$. Due to various noise effects and the generally non-invertible

mapping Φ, the exact pre-image does not always exist. However, the approximated pre-image can be reconstructed without venturing into the feature space using the techniques described in [91, 92]. Fig. 3.6 shows the pre-images of the kernel KSVD dictionary atoms for 10 digits.

3.3.3.2 Classification results

In the first set of experiments, we present the results for the situation where the test samples are corrupted by random Gaussian noise with different standard deviations as shown in Fig. 3.7a. Likewise, Fig. 3.7b shows the results obtained when pixels are randomly removed from the test images. In both experiments, kernel KSVD (both distributive and collective) and kernel MOD (distributive) consistently outperform linear KSVD and kernel PCA. As the distortion level increases the performance difference between kernel dictionaries and linear dictionaries become more drastic. Another interesting observation is that the distributive classification approach outperforms the collective one for the cases with high noise levels. For this reason, we choose the former classification scheme for all other experiments on the USPS dataset.

The second set of experiments examines the effects of parameters choices on the overall recognition performances. Fig. 3.8 shows the classification accuracy of kernel KSVD as we vary the degree of polynomial kernel. The best error rate of 1.6% is achieved with the polynomial degree of 4. This error rate is lower than the error rates from other methods in the literature: nearest neighbor (5.6%) [93], neural net (4.2%) [94], neural net + boosting (2.6%) [94], invariant support vectors (3%) [95], supervised dictionary

learning (3.54%) [96].

We also compare the performance of our method in the case of missing pixels. For the case when 50% of the pixels in the test samples are randomly removed, the kernel KSVD method shows similar trend of performance. For instance, the accuracy peaks when the polynomial degree 4 is used and as the polynomial degree increases further, the accuracy goes down slightly and becomes saturated. For this reason, we fix the polynomial degree to be 4 for all other experiments on the USPS dataset.

Fig. 3.9 compares the performances of KSVD, kernel PCA, and kernel KSVD when varying the sparsity T_0 in the range from $5 \rightarrow 50$. For kernel PCA, sparsity means the number of principal components or the dimension of the PCA subspace. It can be easily seen that the performance of KSVD is very sensitive to the sparsity parameter, while kernel KSVD and kernel PCA remain rather stable against this variation. Kernel KSVD also outperforms other methods considered here for all Gaussian noise levels and

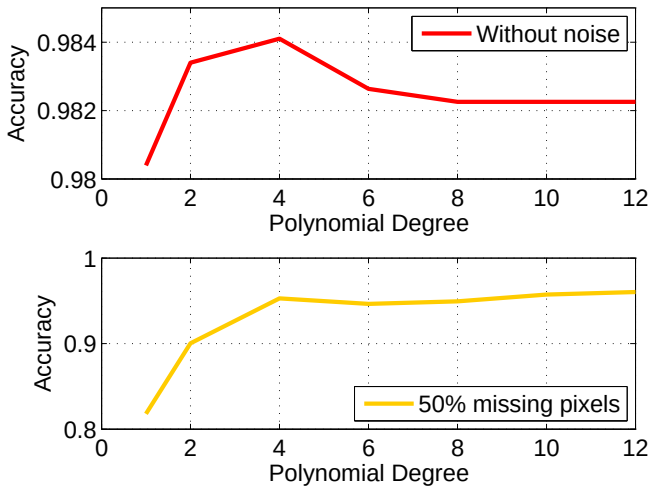


Figure 3.8: Kernel KSVD classification accuracy versus the polynomial degree of the USPS dataset.

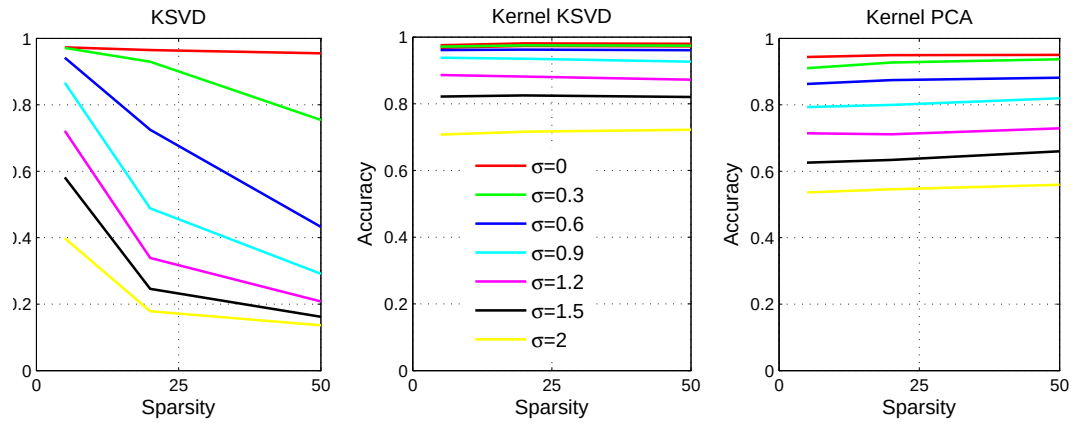


Figure 3.9: Comparison of digit recognition accuracies for different sparsity levels in the presence of Gaussian noise with standard deviation σ. All algorithms train on clean images and test on noisy images. The dictionary size is set to 50 for this experiments.

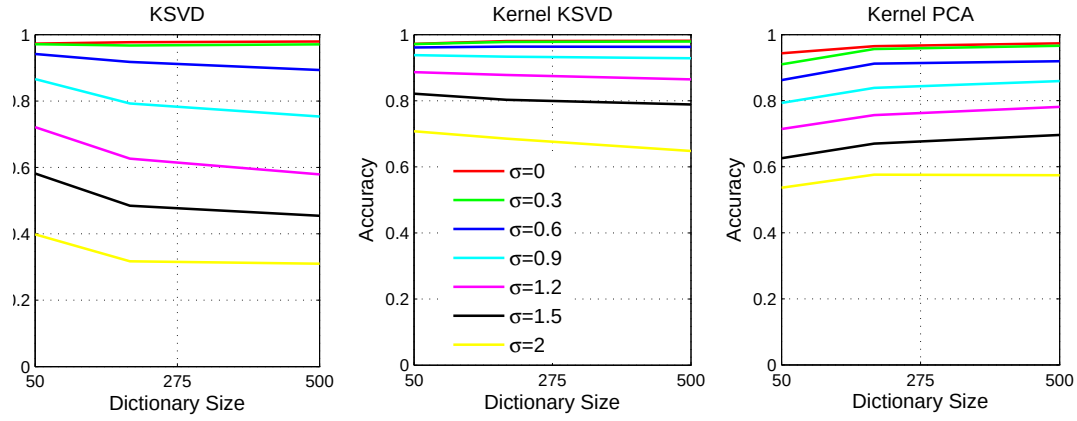


Figure 3.10: Comparison of digit recognition accuracies for different dictionary sizes in the presence of Gaussian noise with standard deviation σ. All algorithms train on clean images and test on noisy images. The sparsity is set to $T_0 = 5$ in this experiment.

all sparsity settings. This can be seen by examining the curves of the same colors in the three plots in Fig. 3.9. Interestingly, while KSVD (with a proper sparsity setting) beats kernel PCA for cases with moderate noise levels, its performance rapidly deteriorates as noise level increases and becomes even worse than kernel PCA for the case of heavy noise, e.g. $\sigma \geq 1.5$.

Fig. 3.10 compares the recognition accuracies on USPS dataset for different number of dictionary atoms and principal components. Both KSVD and kernel KSVD exhibit a similar trend. In particular, in the presence of noise, their performances decrease as the number of dictionary atoms increases. The performance of KSVD decreases at a faster rate than kernel KSVD. In addition, both KSVD and kernel KSVD are rather indifferent to the variation of number of dictionary atoms when there is no noise. These observations are quite understandable since larger dictionaries do not generalize well, making them more brittle in dealing with noise. Interestingly, kernel PCA performs slightly better when allowing more principal components. This is the case because principal components are orthogonal to each others, thus, they are less compact and need more components to capture enough useful information. The benefit of additional information outweighs the inclusion of noise and leads to a slightly better performances for kernel PCA. Overall, we observe that the setting with dictionary size $K = 200$ and sparsity setting $T_0 = 5$ provides a good trade-off between the performance, the robustness against noise, and the computational complexity for this experiment.

3.3.4 Caltech-101 Object Recognition

We report the results of object recognition experiments using the Caltech-101 database [97]. This database comprises 101 object classes, and 1 background class collected randomly from Internet. Each category contains from 31 to 800 images. Most objects are centered in the foreground and same-class objects are aligned to similar stereo-typical pose. The average size of images is about 300×300 pixels. The database is very diverse and challenging as it includes objects like buildings, musical instruments, animals and natural scenes.

We combine 39 different features using the boosting approach as in [98]. These features include the spatial pyramid [10], the PHOG shape [99], the region of covariance [11], the local binary patterns (LBP) [100], the V1S+ [101], etc. All features have non-linear distance metrics. For instance, the χ^2 distance is used for the PHOG shape and the LBP descriptors, the geodesic distance is used for region of covariance descriptor, whereas the Gaussian kernel is used for all other types of features. Kernel functions are computed as $\kappa = exp(-d(\mathbf{y}_i, \mathbf{y}_j)/\delta)$, with d being the distance and δ set to the mean of pairwise distances. Any learning method should be able to take these non-linear similarity measures into account to be effective. Our kernel dictionary learning algorithm fits naturally into the framework of the multiple features combination. In contrast, the current dictionary learning techniques [12, 13], which use the Euclidean distance as the similarity measure between feature vectors, are unable to adequately capture the non-linear structures of the feature space.

We follow the suggested protocol in [10, 12, 102], namely, we train on N images,

where $N \in \{5, 10, 15, 20, 25, 30\}$, and test on the rest. Note that because some categories are very small, we might end up with just one single image for testing. To compensate for the variations in class sizes, we normalize the recognition results by the number of test images to get per-class accuracies. The final recognition accuracy is then obtained by averaging per-class accuracies across 102 categories.

We use similar approaches as in Sec.3.3.3 for classification. The algorithm starts with learning a dictionary for each class. In particular, we aggregate descriptors of all training images for each category into $\mathbf{Y}_i = [\mathbf{y}_{i,1} \dots \mathbf{y}_{i,N}]$, where $i \in \{1 \dots 102\}$. The kernel KSVD algorithm is then run on $\mathbf{Y}_i$ with the sparsity level $T_0 = 3$ and the maximum number of iterations set to 80. The dictionary size is equal to $\{4, 8, 12, 15, 20, 25\}$, respectively. The parameter selection is done with a 5-fold cross validation.

Let $\mathbf{D}_i = \Phi(\mathbf{Y}_i)\mathbf{A}_i$ denotes the learned kernel dictionary, where $\mathbf{A}_i \in \mathbb{R}^{N \times K}$. In order to perform classification on a newly given test feature $\mathbf{y}_t$ using the collective approach, we first concatenate dictionaries from all classes:

$$\mathbf{D} = [\mathbf{D}_1 \dots \mathbf{D}_{102}] = [\Phi(\mathbf{Y}_1)\mathbf{A}_1 \dots \Phi(\mathbf{Y}_{102})\mathbf{A}_{102}] \tag{3.38}$$

The final dictionary $\mathbf{D}$ is then used in kernel OMP to solve the sparse decompositions of $\mathbf{y}_t$, as in Fig. 3.1, yielding the sparse coefficients $\mathbf{x}_t = [\mathbf{x}_{1,t} \dots \mathbf{x}_{102,t}]$, where $\mathbf{x}_{i,t}$ contains the sparse coefficients associated with the dictionary $\mathbf{D}_i$ of the i-th class. During the testing phase, we set the sparsity level to be $T_0 = 10$. The reconstruction error is

computed by:

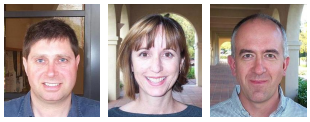
$$
\begin{aligned}
r_i &= \|\Phi(\mathbf{y}_t) - \Phi(\mathbf{Y}_i)\mathbf{A}_i\mathbf{x}_{i,t}\|_2^2 \\
&= \left((\Phi(\mathbf{y}_t) - \Phi(\mathbf{Y}_i)\mathbf{A}_i\mathbf{x}_{i,t})^T(\Phi(\mathbf{y}_t) - \Phi(\mathbf{Y}_i)\mathbf{A}_i\mathbf{x}_{i,t})\right) \\
&= \boldsymbol{\mathcal{K}}(\mathbf{y}_t, \mathbf{y}_t) - 2\boldsymbol{\mathcal{K}}(\mathbf{y}_t, \mathbf{Y}_i)\mathbf{A}_i\mathbf{x}_{i,t} + \mathbf{x}_i^T\mathbf{A}_i^T\boldsymbol{\mathcal{K}}(\mathbf{Y}_i, \mathbf{Y}_i)\mathbf{A}_i^T\mathbf{x}_i
\end{aligned}
\tag{3.39}
$$

The classification is done simply by assigning the test feature to the class with smallest reconstruction error. The distributive classification scheme is similar except that the sparse coding is done separately for each dictionary.

(a) Minaret (100%)

(b) Panda(100%)

(c) Okapi (100%)



(d) Face easy (99.75%)

(e) Wild cat (25%)

(f) Background (34%)

(g) Beaver (37.5%)

(h) Crocodile (40%)

Figure 3.11: The first two rows and the last two rows show the classes that our method performed the best and worst, respectively.

Table 3.2 and table 3.3 show the average recognition accuracy of our algorithm in comparison with other methods on the Caltech-101 and Caltech-256 datasets, respec-

Table 3.2: Comparison of recognition results on Caltech-101 dataset

Number of train samp.	5	10	15	20	25	30
Malik [103]	46.6	55.8	59.1	62.0	-	66.2
Lazebnik [10]	-	-	56.4	-	-	64.6
Griffin [104]	44.2	54.5	59.0	63.3	65.8	67.6
Irani [105]	-	-	65.0	-	-	70.4
Grauman [102]	-	-	61.0	-	-	69.1
Venkatesh [106]	-	-	42.0	-	-	-
Gemert [107]	-	-	-	-	-	64.16
Yang [108]	-	-	67.0	-	-	73.2
Wang [109]	51.15	59.77	65.43	67.74	70.16	73.44
SRC [90]	48.8	60.1	64.9	67.7	69.2	70.7
KSVD [9]	49.8	59.8	65.2	68.7	71.0	73.2
D-KSVD [13]	49.6	59.5	65.1	68.6	71.1	73.0
LC-KSVD [12]	54	63.1	67.7	70.5	72.3	73.6
LP-β [98]	54.2	65.0	70.4	73.6	75.7	77.8
D-Kernel KSVD	**52.1**	**63.5**	**68.3**	**72.4**	**74.6**	**76.8**
C-Kernel KSVD	**56.5**	**67.2**	**72.5**	**75.8**	**77.6**	**80.1**

Table 3.3: Comparison of recognition results on Caltech-256 dataset

# train samples	15	30
Griffin [104]	28.3	34.1
Gemert [107]	-	27.2
Yang [110]	34.4	41.2
D-Kernel KSVD	**34.5**	**41.4**
C-Kernel KSVD	**34.8**	**42.5**

tively. Kernel KSVD with a collective dictionary outperforms all the methods for any number of training images n. In contrast to the previous experiments on the USPS dataset, we notice that the collective classification approach significantly outperforms the distributive approach on the Caltech-101 dataset. An explanation for this observation is that the set of training data is insufficient (less than 30 samples per class). Consequently, the learned dictionaries do not generalize well enough. This leads to a deterioration of the distributive classification scheme which highly depends on the representation power of the learned dictionaries.

It is worth noting that the best recognition performance of [98] using multiple features with boosting techniques on Caltech-101 is only 77.8% in comparison with 80.1% of our method. Since the classifier in [98] is learned directly from the feature vectors, it is quite brittle and may not generalize well, especially in the case where the number of training samples is small (e.g., less than 30 per class). A more robust approach would

be to extract sparse features that are common among images of the same category and learn a classifier on top of these features. LC-KSVD [12] follows such an approach, which uses the spatial pyramid descriptors in addition to the discriminative dictionary learning algorithm to improve the classification result. One drawback of this approach is that the pyramid match kernel or any other non-linear kernel cannot be incorporated within this algorithm. As a result, the learning algorithm operates in the Euclidean space instead of the manifold that the combined features lie on. This inhibits the learning of common non-linear structures that can be very useful for classification. For these reasons, the recognition accuracies of both methods ([98] and LC-KSVD [12]) are lower than our results. This observation justifies the effectiveness of kernel dictionary learning and recognizes its important role in improving the classification accuracy.

Fig. 3.12 plots the confusion table between all categories for our best result (i.e. 80.1%). Fig. 3.13 displays per-class accuracy where classes are sorted in the ascending order of recognition accuracies. There are 15 categories that achieve perfect recognition. Fig. 4.6 displays a few of the "easiest" and the "hardest" object categories. The successful classes include "Okapi" and "Minaret", and the difficult classes include "Beaver" and "Crocodile". This is consistent with the result in [10]. Interestingly, several classes like "Ant" have the rather high performances while they are among the hardest classes in [10]. This shows the benefit of combining multiple features with dictionary learning.

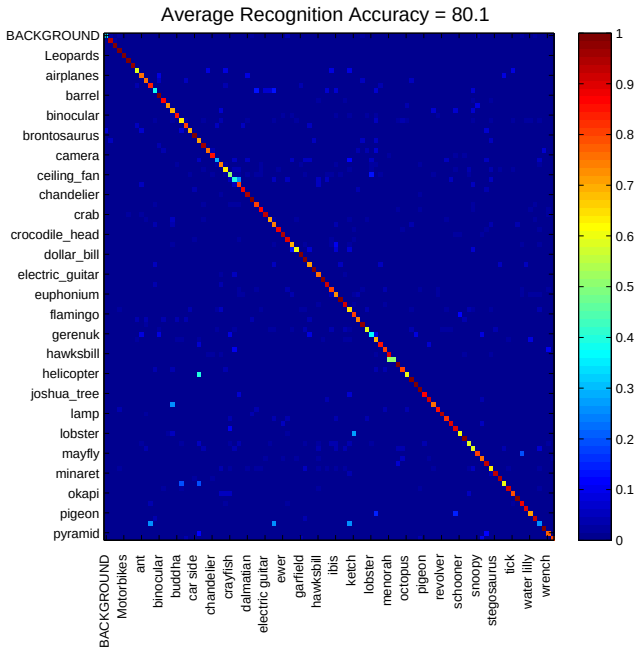


Figure 3.12: Confusion matrix of our recognition performances on Caltech 101 dataset using kernel KSVD. The rows and columns correspond to true labels and predicted labels, respectively. The dictionary is learned from 3030 images where each class contributes 30 images. The sparsity is set to be 30 for both training and testing. Although the confusion matrix contains all classes, only a subset of class labels is displayed for better legibility.

3.4 Discussion and Conclusion

We have presented two non-linear dictionary learning algorithms that exploit sparsity of data in high dimensional feature space by extending MOD and KSVD dictionary learning algorithms through an appropriate choice of kernel. Experimental results indicate that exploiting non-linear sparsity via learning dictionaries in a non-linear feature space can provide better discrimination than the traditional linear approaches and kernel PCA. Significant improvements over the current state of the art on two standard datasets

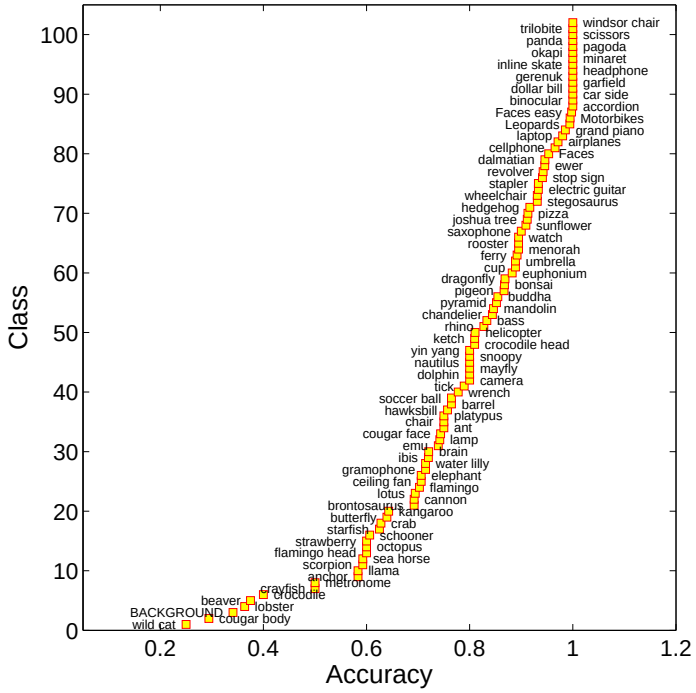


Figure 3.13: Average accuracy of each class in Caltech-101 dataset. Classes are sorted in the ascending order of their recognition accuracies.

show that the kernel dictionary learning technique is superior over its linear counterpart in terms of classification accuracy. Especially, kernel dictionary demonstrates robustness with respect to noise allowing robust recognition and better tolerance against different types of degradations.

3.5 Proof of Proposition 3.1

Proof. Using the standard orthogonal decomposition, we can write $\mathbf{D}^*$ as follows:

$$\mathbf{D}^* = \tilde{\mathbf{D}} + \mathbf{D}_\perp \tag{3.40}$$

where $\tilde{\mathbf{D}}$ and $\mathbf{D}_\perp$ are matrices whose columns lie within and orthogonal to column subspace of $\Phi(\mathbf{Y})$, respectively. In other words, $\tilde{\mathbf{D}} = \Phi(\mathbf{Y})\mathbf{A}$ and $\Phi(\mathbf{Y})^T\mathbf{D}_\perp = \mathbf{0}$. The optimal cost is equivalent to:

$$
\begin{aligned}
&||\Phi(\mathbf{Y}) - \mathbf{D}^*\mathbf{X}^*||_F^2 = ||\Phi(\mathbf{Y}) - (\tilde{\mathbf{D}} + \mathbf{D}_\perp)\mathbf{X}^*||_F^2 \\
=&||\Phi(\mathbf{Y})(\mathbf{I} - \mathbf{A}\mathbf{X}^*) - \mathbf{D}_\perp\mathbf{X}^*||_F^2 \\
=&\mathbf{tr}\left([\Phi(\mathbf{Y})(\mathbf{I} - \mathbf{A}\mathbf{X}^*) - \mathbf{D}_\perp\mathbf{X}^*]^T[\Phi(\mathbf{Y})(\mathbf{I} - \mathbf{A}\mathbf{X}^*) - \mathbf{D}_\perp\mathbf{X}^*]\right) \\
=&\mathbf{tr}[(\mathbf{I} - \mathbf{A}\mathbf{X}^*)^T\Phi(\mathbf{Y})^T\Phi(\mathbf{Y})(\mathbf{I} - \mathbf{A}\mathbf{X}^*)] \\
&+ \mathbf{tr}[(\mathbf{X}^*)^T\mathbf{D}_\perp^T\mathbf{D}_\perp\mathbf{X}^*].
\end{aligned} \tag{3.41}
$$

The second terms can be expressed as:

$$
\mathbf{tr}[(\mathbf{X}^*)^T\mathbf{D}_\perp^T\mathbf{D}_\perp\mathbf{X}^*] = \sum_{i=1}^{n} \mathbf{x}_i^T\mathbf{D}_\perp^T\mathbf{D}_\perp\mathbf{x}_i \geq 0
$$

The inequality comes from the fact that $\mathbf{D}_\perp^T\mathbf{D}_\perp$ is a positive semidefinite matrix. As a result, the second term of (3.41) can only increase the cost function. For the cost to be optimal, the second term must vanish. In short, $\mathbf{D}_\perp = \mathbf{0}$ and $\mathbf{D}^* = \tilde{\mathbf{D}} = \Phi(\mathbf{Y})\mathbf{A}$ is an optimal solution. □

Chapter 4

Sparse Representations On Low-Dimensional Feature Space

In this chapter, we propose a novel framework, called *sparse embedding* (SE), that brings the strength of both dimensionality reduction and sparse learning together. In this framework, the dimension of signals is reduced in a way such that the sparse structures of signals are promoted. The algorithm simultaneously learns a dictionary in the reduced space, yet, allows the recovery of the dictionary in the original domain. This empowers the algorithm with two important advantages: 1) Ability to remove the distracting part of the signal that negatively interferes with the learning process, and 2) Learning in the reduced space with smaller computational complexity. In addition, our framework is able to handle sparsity in non-linear model through the use of Mercer kernels.

4.1 Motivation and Related Work

Signals are usually assumed to lie on a low-dimensional manifold embedded in a high dimensional space. Dealing with the high-dimension is not practical for both learning and inference tasks. As an example of the effect of dimension on learning, Stone [111] showed that, under certain regularity assumption including that samples are identically independent distributed, the optimal rate of convergence for non-parametric regression decreases exponentially with the dimension of the data. As the dimension increases, the Euclidean distances between feature vectors become closer to each other making the infer-

ence task harder. This is known as the concentration phenomenon [112], To address these issues, various linear and non-linear dimensionality reduction (DR) techniques have been developed [113]. Some examples include PCA [114], ISOMAP [115], LLE [116], Laplacian Eigenmaps [117], etc. In general, these techniques map data to a lower-dimensional space such that non-informative or irrelevant information in the data are discarded.

Recently, there has been an explosion of activities in modelling a signal using appropriate sparse representations (see [118] and references therein). This approach is motivated by the observation that most of signals encountered in practical applications are compressible. In other words, their sorted coefficient magnitudes in some basis obey power law decay. For this reason, signals can be well-approximated by linear combinations of a few columns of some appropriate basis or dictionary $\mathbf{D}$. Although predefined basis such as wavelets or Fourier basis give rather good performances in signal compression, but it has been shown that dictionaries learned from the data can be more compact leading to better performance in many important tasks such as reconstruction and classification [119, 90].

However, the current algorithms for finding a good dictionary have some drawbacks. The learning of $\mathbf{D}$ is challenging due to the high dimensional nature of the training data, as well as the lack of training samples. Therefore, DR seems to be a natural solution. Unfortunately, the current DR techniques are not designed to respect and promote underlying sparse structures of data. Therefore, they cannot help the process of learning the dictionary $\mathbf{D}$. Note that the recently developed DR technique [120] based on the sparse linear model is also not suitable for the purpose of sparse learning since it assumes that the dictionary is given. Ideally, we want an algorithm that can discard non-informative

part of the signal and yet does not destroy the useful sparse structures within the data.

The second disadvantage of the current sparse learning framework is its inability to handle sparse signals within non-linear models. Linear model used for learning $\mathbf{D}$ is often inadequate to capture the non-linear relationships within the data that naturally arise in many important applications. For example, in [121, 122, 123] it has been shown that by taking into account non-linearity, one can do better in reconstruction and classification. In addition, spatial pyramid [10], a popular descriptor for object and scene classification, and region of covariance [11], a popular descriptor for object detection and tracking, both have non-linear distance measures thus making the current sparse representation inappropriate.

4.2 Sparse Embedding Framework

The classical approach to learn sparse representations [9] is by minimizing the empirical lost over a finite set of signals subject to some sparsity constraint. Let $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N] \in \mathbb{R}^{n \times N}$ denotes the matrix of N input signals, where $\mathbf{y}_i \in \mathbb{R}^n$. A popular formulation is:

$$\{\mathbf{D}^*, \mathbf{X}^*\} = \underset{\mathbf{D},\mathbf{X}}{\operatorname{argmin}} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2, \tag{4.1}$$

$$\text{subject to: } \|\mathbf{x}_i\|_0 \leq T_0, \forall i$$

where $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_K] \in \mathbb{R}^{n \times K}$ is called the dictionary that we seek to learn, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}^{K \times N}$ is the sparse representation of $\mathbf{Y}$ over $\mathbf{D}$, and T_0 is the sparsity level. The cost function in (4.1) promotes a dictionary $\mathbf{D}$ that can best represents $\mathbf{Y}$ by linearly combining only a few of its columns. This type of optimization can be done efficiently using the current methods [123, 9].

Different from the classical approaches, we develop an algorithm that embeds input signals into a low-dimensional space, and simultaneously learns an optimized dictionary. Let $\mathcal{M}$ denote the mapping that transforms input signals into the output space. In general, $\mathcal{M}$ can be non-linear. However, for the simplicity of notations, we temporarily restrict our discussion to linear transformations. The extension to the non-linear case will be presented in section 4.4. As a result, the mapping $\mathcal{M}$ is characterized using a matrix $\mathbf{P} \in \mathbb{R}^{d \times n}$. We can learn the mapping together with the dictionary through minimizing some appropriate cost function $\mathcal{C}_{\mathbf{Y}}$:

$$\{\mathbf{P}^*, \mathbf{D}^*, \mathbf{X}^*\} = \underset{\mathbf{P},\mathbf{D},\mathbf{X}}{\operatorname{argmin}} \mathcal{C}_{\mathbf{Y}}(\mathbf{P}, \mathbf{D}, \mathbf{X}) \tag{4.2}$$

This cost function $\mathcal{C}_{\mathbf{Y}}$ needs to have several desirable properties. First, it has to promote sparsity within the reduced space. At the same time, the transformation $\mathbf{P}$ resulting from optimizing $\mathcal{C}_{\mathbf{Y}}$ has to retain the useful information within the original signals. The second criterion is needed in order to prevent the pathological case when everything is mapped into the origin, which obtains the sparsest solution but is obviously of no interest. Towards this end, we propose the following optimization:

$$\{\mathbf{P}^*, \mathbf{D}^*, \mathbf{X}^*\} = \underset{\mathbf{P},\mathbf{D},\mathbf{X}}{\operatorname{argmin}} \left(\|\mathbf{PY} - \mathbf{DX}\|_F^2 + \lambda \|\mathbf{Y} - \mathbf{P}^T\mathbf{PY}\|_F^2\right) \tag{4.3}$$

$$\text{subject to: } \mathbf{P}\mathbf{P}^T = \mathbf{I}, \text{ and } \|\mathbf{x}_i\|_0 \leq T_0, \ \forall i$$

where $\mathbf{I} \in \mathbb{R}^{d \times d}$ is the identity matrix, λ is a non-negative constant, and the dictionary is now in the reduced space, i.e., $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_K] \in \mathbb{R}^{\mathbf{d} \times K}$. The first term of the cost function promotes sparsity of signals in the reduced space. The second term is the amount of energy discarded by the transformation $\mathbf{P}$, or the difference between low-dimensional

approximations and the original signals. In fact, the second term is closely related to PCA as by removing the first term in (4.3), it can be shown that the solution of $\mathbf{P}$ coincides with the principal components of the largest eigenvalues, when the data are centered.

In addition, we also require rows of $\mathbf{P}$ to be orthogonal and normalized to unit norm. $\mathbf{P}$ plays the role of selecting the right subspace, or equivalently the right features, in which the useful sparse structures within data are revealed. There are several compelling reasons to keep the orthogonality constraint. First, this constraint leads to a computationally efficient scheme for optimization and classification. Second, it allows the extension of SE to the non-linear case. Note that the columns of the dictionary $\mathbf{D}$ still form a non-orthogonal basis in the output space despite the orthogonality constraint of $\mathbf{P}$.

4.3 Optimization Procedure

Proposition 4.1. *There exists an optimal solution* $\mathbf{P}^*$ *and* $\mathbf{D}^*$ *to the problem (4.3) that has the following form:*

$$\mathbf{P}^* = (\mathbf{YA})^T; \quad \mathbf{D}^* = \mathbf{A}^T\mathbf{Y}^T\mathbf{YB} \tag{4.4}$$

for some $\mathbf{A} \in \mathbb{R}^{N\times d}$*, and some* $\mathbf{B} \in \mathbb{R}^{N\times K}$.

Proof. See Appendix 4.7 □

As a corollary of the proposition 4.1, it is sufficient to seek an optimal solution for the optimization in Eq. (4.3) through $\mathbf{A}$ and $\mathbf{B}$. By substituting Eq. (4.4) into Eq. (4.3), we have:

$$\mathcal{C}_{\mathbf{Y}}(\mathbf{P}, \mathbf{D}, \mathbf{X}) = \|\mathbf{A}^T\boldsymbol{\mathcal{K}}(\mathbf{I} - \mathbf{BX})\|_F^2 + \lambda\|\mathbf{Y}(\mathbf{I} - \mathbf{AA}^T\boldsymbol{\mathcal{K}})\|_F^2 \tag{4.5}$$

where $\mathcal{K} = \mathbf{Y}^T\mathbf{Y}$ and λ is a regularization parameter. The equality constraint becomes

$$\mathbf{P}\mathbf{P}^T = \mathbf{A}^T\mathcal{K}\mathbf{A} = \mathbf{I} \tag{4.6}$$

The solution can be derived as

$$\{\mathbf{A}^*, \mathbf{B}^*, \mathbf{X}^*\} = \underset{\mathbf{A},\mathbf{B},\mathbf{X}}{\operatorname{argmin}} \left(\|\mathbf{A}^T\mathcal{K}(\mathbf{I} - \mathbf{B}\mathbf{X})\|_F^2 + \lambda\|\mathbf{Y}(\mathbf{I} - \mathbf{A}\mathbf{A}^T\mathcal{K})\|_F^2\right) \tag{4.7}$$

$$\text{subject to: } \mathbf{A}^T\mathcal{K}\mathbf{A} = \mathbf{I}, \text{ and } \|\mathbf{x}_i\|_0 \leq T_0$$

The advantage of this formulation will become clear later. Basically, this formulation allows a joint update of $\mathbf{P}$ and $\mathbf{D}$ via $\mathbf{A}$. As we shall see later in section 4.4, because the objective function is not explicitly represented in terms of $\mathbf{Y}$, it is then possible to use the kernel method to make the algorithm non-linear. Despite (4.7) being non-convex, our experiments show that effective solutions can be found through iterative minimization.

Solving for $\mathbf{A}$

In this stage, we assume that $(\mathbf{B}, \mathbf{X})$ are fixed. As a result, we can remove the sparsity constraint of (4.7). The following proposition shows that $\mathbf{A}$ can be solved efficiently after some algebraic manipulation:

Proposition 4.2. *The optimal solution of (4.7) when* $\mathbf{B}$ *and* $\mathbf{X}$ *are fixed is:*

$$\mathbf{A}^* = \mathbf{V}\mathbf{S}^{-\frac{1}{2}}\mathbf{G}^* \tag{4.8}$$

where $\mathbf{V}$ *and* $\mathbf{S}$ *come from the eigendecomposition of* $\mathcal{K} = \mathbf{V}\mathbf{S}\mathbf{V}^T$*, and* $\mathbf{G}^* \in \mathbb{R}^{N\times d}$ *is*

the optimal solution of the following minimum eigenvalues problem:

$$\{\mathbf{G}^*\} = \underset{\mathbf{G}}{argmin} \quad \mathbf{tr}\left[\mathbf{G}^T\mathbf{H}\mathbf{G}\right] \tag{4.9}$$

$$\textit{subject to: } \mathbf{G}^T\mathbf{G} = \mathbf{I}$$

where $\mathbf{H} = \mathbf{S}^{\frac{1}{2}}\mathbf{V}^T((\mathbf{I}-\mathbf{B}\mathbf{X})(\mathbf{I}-\mathbf{B}\mathbf{X})^T - \lambda\mathbf{I})\mathbf{V}\mathbf{S}^{\frac{1}{2}} \in \mathbb{R}^{N\times N}$.

Proof. The cost function can be expanded as follows:

$$\mathcal{C}_{\mathbf{Y}}(\mathbf{P},\mathbf{D},\mathbf{X}) = \|\mathbf{A}^T\boldsymbol{\mathcal{K}}(\mathbf{I}-\mathbf{B}\mathbf{X})\|_F^2 + \lambda\|\mathbf{Y}(\mathbf{I}-\mathbf{A}\mathbf{A}^T\boldsymbol{\mathcal{K}})\|_F^2 \tag{4.10}$$

$$= \mathbf{tr}\left[(\mathbf{I}-\mathbf{B}\mathbf{X})(\mathbf{I}-\mathbf{B}\mathbf{X})^T\boldsymbol{\mathcal{K}}^T\mathbf{Q}^T\boldsymbol{\mathcal{K}} + \lambda(\boldsymbol{\mathcal{K}} - 2\boldsymbol{\mathcal{K}}^T\mathbf{Q}^T\boldsymbol{\mathcal{K}} + \boldsymbol{\mathcal{K}}^T\mathbf{Q}^T\boldsymbol{\mathcal{K}}\mathbf{Q}\boldsymbol{\mathcal{K}})\right] \tag{4.11}$$

where $\mathbf{Q} = \mathbf{A}\mathbf{A}^T \in \mathbb{R}^{N\times N}$. The constraint $\mathbf{A}^T\boldsymbol{\mathcal{K}}\mathbf{A} = \mathbf{I}$ leads to the new constraint $\mathbf{A}\mathbf{A}^T\boldsymbol{\mathcal{K}}\mathbf{A}\mathbf{A}^T = \mathbf{A}\mathbf{A}^T$ or $\mathbf{Q}\boldsymbol{\mathcal{K}}\mathbf{Q}^T = \mathbf{Q}$. Using this equality constraint, and also notice that $\mathbf{tr}(\boldsymbol{\mathcal{K}})$ is a constant, the objective function in (4.11) can be simplified to a more elegant form:

$$\tilde{\mathcal{C}}_{\mathbf{Y}}(\mathbf{P},\mathbf{D},\mathbf{X}) = \mathbf{tr}\left[((\mathbf{I}-\mathbf{B}\mathbf{X})(\mathbf{I}-\mathbf{B}\mathbf{X})^T - \lambda I)\boldsymbol{\mathcal{K}}^T\mathbf{Q}^T\boldsymbol{\mathcal{K}}\right] \tag{4.12}$$

With a simple change of variable $\mathbf{G} = \mathbf{S}^{\frac{1}{2}}\mathbf{V}^T\mathbf{A}$, and noting that $\mathbf{Q} = \mathbf{A}\mathbf{A}^T$, the cost function can be further simplified as:

$$\tilde{\mathcal{C}}_{\mathbf{Y}}(\mathbf{P},\mathbf{D},\mathbf{X}) = \mathbf{tr}\left[((\mathbf{I}-\mathbf{B}\mathbf{X})(\mathbf{I}-\mathbf{B}\mathbf{X})^T - \lambda I)\mathbf{V}\mathbf{S}^{\frac{1}{2}}(\mathbf{S}^{\frac{1}{2}}\mathbf{V}^T\mathbf{A})(\mathbf{A}^T\mathbf{V}\mathbf{S}^{\frac{1}{2}})\mathbf{S}^{\frac{1}{2}}\mathbf{V}^T\right]$$

$$= \mathbf{tr}\left[\mathbf{G}^T\mathbf{S}^{\frac{1}{2}}\mathbf{V}^T((\mathbf{I}-\mathbf{B}\mathbf{X})(\mathbf{I}-\mathbf{B}\mathbf{X})^T - \lambda I)\mathbf{V}\mathbf{S}^{\frac{1}{2}}\mathbf{G}\right] = \mathbf{tr}\left[\mathbf{G}^T\mathbf{H}\mathbf{G}\right] \tag{4.13}$$

On the other hand, the equality constraint can also be simplified as:

$$\mathbf{A}^T\boldsymbol{\mathcal{K}}\mathbf{A} = \mathbf{A}^T\mathbf{V}\mathbf{S}^{\frac{1}{2}}\mathbf{S}^{\frac{1}{2}}\mathbf{V}^T\mathbf{A} = \mathbf{G}^T\mathbf{G} = \mathbf{I} \tag{4.14}$$

Eqs. (4.13) and (4.14) show that the original optimization in (4.7) is equivalent to (4.9), and the optimal solution $\mathbf{A}^*$ can be recovered as in (4.8), i.e., $\mathbf{A}^* = \mathbf{V}\mathbf{S}^{-\frac{1}{2}}\mathbf{G}^*$. Note that since $\boldsymbol{\mathcal{K}}$ is a positive semidefinite matrix, the diagonal matrix $\mathbf{S}$ has non-negative entries. $\mathbf{S}^{-\frac{1}{2}}$ is obtained by setting non-zero entries along the diagonal of $\mathbf{S}$ to the inverse of their square root and keeping zero elements the same. This completes the proof.

□

Solving for $\mathbf{B}$ and $\mathbf{X}$

In order to solve for $\mathbf{B}$, we keep $\mathbf{A}$ and $\mathbf{X}$ fixed. The second term in (4.7) disappears, and the objective function reduces to:

$$\|\mathbf{A}^T\boldsymbol{\mathcal{K}}(\mathbf{I}-\mathbf{B}\mathbf{X})\|_F^2 = \mathbf{tr}\ \left(\boldsymbol{\mathcal{K}}^T\mathbf{Q}\boldsymbol{\mathcal{K}} - 2\mathbf{X}^T\mathbf{B}^T\boldsymbol{\mathcal{K}}^T\mathbf{Q}\boldsymbol{\mathcal{K}} + \mathbf{X}^T\mathbf{B}^T\boldsymbol{\mathcal{K}}^T\mathbf{Q}\boldsymbol{\mathcal{K}}\mathbf{B}\mathbf{X}\right) \tag{4.15}$$

A possible way of solving for $\mathbf{B}$ is by taking the derivative of the objective function with respect to $\mathbf{B}$ and setting it to zero, which yields:

$$-2(\boldsymbol{\mathcal{K}}^T\mathbf{Q}\boldsymbol{\mathcal{K}})\mathbf{X}^T + 2(\boldsymbol{\mathcal{K}}^T\mathbf{Q}\boldsymbol{\mathcal{K}})\mathbf{B}(\mathbf{X}\mathbf{X}^T) = 0 \tag{4.16}$$

$$\mathbf{B} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^\dagger \tag{4.17}$$

This is similar to the method of optimal direction (MOD) [124] updating step except that $\mathbf{B}$ is not the dictionary but its representation coefficients over the training data $\mathbf{Y}$.

It is also possible to update $\mathbf{B}$ using the KSVD algorithm [9]. First, let us rewrite the objective function (4.15) to a more KSVD-friendly form:

$$\|\mathbf{A}^T\boldsymbol{\mathcal{K}}(\mathbf{I}-\mathbf{B}\mathbf{X})\|_F^2 = \|\mathbf{Z}-\mathbf{D}\mathbf{X}\|_F^2 \tag{4.18}$$

where $\mathbf{Z} = \mathbf{A}^T\boldsymbol{\mathcal{K}} \in \mathbb{R}^{d\times N}$ is the set of output signals. We can solve for $\mathbf{B}$ in two steps. First, we apply the KSVD algorithm to learn a dictionary $\mathbf{D}_{ksvd}$ from $\mathbf{Z}$. Second, we try to recover $\mathbf{B}$ from $\mathbf{D}_{ksvd}$. From proposition 1, it follows that the optimal dictionary has to be in the columns subspace of the input signals $\mathbf{Z}$. We can observe from the KSVD algorithm that its output dictionary also obeys this property. As a result, we can recover $\mathbf{B}$ exactly by simply taking the pseudo-inverse:

$$\mathbf{B} = (\mathbf{Z})^{\dagger}\mathbf{D}_{ksvd} \tag{4.19}$$

We choose a KSVD-like updating strategy because it is more computationally efficient. Our experiments show that both updating strategies produce similar performances for applications like object classification.

Sparse coding that solves for $\mathbf{X}$ can be done by any off-the-shelves pursuit algorithms. We use the orthogonal matching pursuit (OMP) [81] due to its high efficiency and effectiveness. Note that sparse coding is the most expensive step in many dictionary learning algorithms. Kernel KSVD [122] is an extreme example where sparse coding has to be done in the high-dimensional feature space. Our algorithm performs sparse coding in the reduced space leading to a huge computational advantage, yet, is capable of taking into account the non-linearity as we shall demonstrate in the next section.

4.4 Non-linear Extension of Sparse Embedding

There are many important applications of computer vision that deal with non-linear data [10, 11]. Non-linear structures in data can be exploited by transforming the data into a high-dimensional feature space where they may exist as a simple Euclidean geometry.

Input: A kernel matrix $\boldsymbol{\mathcal{K}}$, sparse setting T_0, dictionary size K, dimension d, and λ.

Task: Find $\mathbf{A}^*$ and $\mathbf{B}^*$ by solving Eq. (4.4).

Initialize:

- Set iteration $J = 1$. Perform eigendecomposition $\boldsymbol{\mathcal{K}} = \mathbf{V}\mathbf{S}\mathbf{V}^T$

- Set $\mathbf{A} = \mathbf{V}(:, \mathcal{I}_0)$, where $\mathcal{I}_0$ is the index set of d largest eigenvalues of $\boldsymbol{\mathcal{K}}$

Stage 1: *Dictionary Update*

- Learn a dictionary $\mathbf{D}$ and $\mathbf{X}$ from the reduced signals $\mathbf{Z} = \mathbf{A}^T\boldsymbol{\mathcal{K}}$ using KSVD

- Update $\mathbf{B} = (\mathbf{Z})^\dagger\mathbf{D}$

Stage 2: *Embedding update*

- Compute $\mathbf{H} = \mathbf{S}^{\frac{1}{2}}\mathbf{V}^T((\mathbf{I} - \mathbf{B}\mathbf{X})(\mathbf{I} - \mathbf{B}\mathbf{X})^T - \lambda\mathbf{I})\mathbf{V}\mathbf{S}^{\frac{1}{2}}$

- Perform eigendecomposition of $\mathbf{H} = \mathbf{U}\Lambda\mathbf{U}^T$

- Set $\mathbf{G} = \mathbf{U}(:, \mathcal{I}_J)$, where $\mathcal{I}_J$ is the index set of d smallest eigenvalues of $\mathbf{H}$

- Update $\mathbf{A} = \mathbf{V}\mathbf{S}^{-\frac{1}{2}}\mathbf{G}$

- Increment $J = J + 1$. Repeat from stage 1 until stopping conditions reached.

Output: $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{X}$.

Figure 4.1: The SE algorithm for both linear and non-linear cases.

In order to avoid the computational issues related to high-dimensional mapping, Mercer kernels are often used to carry out the mapping implicitly. We adopt the use of Mercer kernels to extend our analysis to the non-linear case.

Let $\Phi : \mathbb{R}^n \rightarrow \mathcal{H}$ be a mapping from the input space to the reproducing kernel Hilbert space $\mathcal{H}$. Let $k : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ be the kernel function associated with Φ. The

mapping $\mathcal{M}$ from the input space to the reduced space is no longer linear. It, however, can be characterized by a compact, linear operator $\boldsymbol{\mathcal{P}} : \mathcal{H} \rightarrow \mathbb{R}^d$ that maps every input signal $\mathbf{s} \in \mathbb{R}^n$ to $\boldsymbol{\mathcal{P}}\Phi(\mathbf{s}) \in \mathbb{R}^d$. Similar to the proposition 1, by letting $\boldsymbol{\mathcal{K}} = \langle \Phi(\mathbf{Y}), \Phi(\mathbf{Y}) \rangle_{\mathcal{H}} = [k(\mathbf{y}_i, \mathbf{y}_j)]_{i,j=1}^N$, we can show that:

$$\boldsymbol{\mathcal{P}}^* = \mathbf{A}^T \Phi(\mathbf{Y})^T; \ \mathbf{D}^* = \mathbf{A}^T \boldsymbol{\mathcal{K}} \mathbf{B}, \tag{4.20}$$

Using Eq. (4.20), we can write the mapping $\mathcal{M}$ in an explicit form:

$$\mathcal{M} : \mathbf{s} \in \mathbb{R}^n \rightarrow \boldsymbol{\mathcal{P}}\Phi(\mathbf{s}) = \mathbf{A}^T \langle \Phi(\mathbf{Y}), \Phi(\mathbf{s}) \rangle_{\mathcal{H}} = \mathbf{A}^T [k(\mathbf{y}_1, \mathbf{s}), \ldots, k(\mathbf{y}_N, \mathbf{s})]^T \tag{4.21}$$

Similar to the linear case, the non-linear SE gives rise to the following cost function:

$$\|\boldsymbol{\mathcal{P}}\Phi(\mathbf{Y}) - \mathbf{D}\mathbf{X}\|_F^2 + \lambda \|\Phi(\mathbf{Y}) - \boldsymbol{\mathcal{P}}^T \boldsymbol{\mathcal{P}} \Phi(\mathbf{Y})\|_{\mathcal{H}}^2 \tag{4.22}$$

which can be expressed in terms of $\mathbf{A}$ and $\mathbf{B}$ using Eq. (4.20), yielding an equivalent optimization:

$$\{\mathbf{A}^*, \mathbf{B}^*, \mathbf{X}^*\} = \underset{\mathbf{A},\mathbf{B},\mathbf{X}}{\operatorname{argmin}} \|\mathbf{A}^T \boldsymbol{\mathcal{K}} (\mathbf{I} - \mathbf{B}\mathbf{X})\|_F^2 + \lambda \mathbf{tr}\left[(\mathbf{I} - \mathbf{A}\mathbf{A}^T \boldsymbol{\mathcal{K}})^T \boldsymbol{\mathcal{K}} (\mathbf{I} - \mathbf{A}\mathbf{A}^T \boldsymbol{\mathcal{K}}) \right]$$
$$\text{subject to: } \mathbf{A}^T \boldsymbol{\mathcal{K}} \mathbf{A} = \mathbf{I}, \text{ and } \|\mathbf{x}_i\|_0 \leq T_0, \forall i \tag{4.23}$$

The resulting optimization can be solved in the same way as in the linear case. Fig. 4.1 summarizes the SE algorithm. Note that in the non-linear case, the dimension of the output space can be higher than the dimension of the input space, and is only upper bounded by the number of training samples.

4.5 Experiments

In this section, we evaluate our algorithm on both synthetic and real datasets. In addition, we propose a novel classification scheme that leads to competitive performances

on 3 different datasets: USPS, Caltech-101, and Caltech-256. We also analyse and compare our method with the state of the art. For all the experiments in this section, we set the maximum number of iteration J for our SE algorithm shown in Fig. 4.1 and that of the KSVD algorithm to 5 and 80, respectively.

4.5.1 Recovery of Dictionary Atoms

Similar to the previous works [9, 81], we first run our algorithm on a set of synthetic signals. The goal is to verify if our algorithm is able to learn the sparse patterns from a set of training data that comes with distortions.

Generation of Training Signals: Let $\mathbf{D} \in \mathbb{R}^{80\times 50}$ be a *generating dictionary*. The first 30 elements in each column of $\mathbf{D}$ are generated randomly, and the last 50 elements are set to zero. Each column of $\mathbf{D}$, which we will call a *dictionary atom*, is normalized to unit norm. 2000 training signals are generated by linearly combining 3 random atoms from this dictionary and superimposed with distortion signals:

$$\mathbf{Y} = \mathbf{D}\mathbf{X} + \alpha\mathbf{E} \tag{4.24}$$

where α is the distortion level; $\mathbf{X} \in \mathbb{R}^{50\times 2000}$ is a matrix where each of its column has at most 3 non-zero elements at independent locations with random values; $\mathbf{E} \in \mathbb{R}^{80\times 2000}$ is a matrix where each of its column is called a *distortion signal* and generated as follows: The first 30 elements in each column of $\mathbf{E}$ are set to zero, and the last 50 elements are generated randomly and independently under Gaussian distribution. Each column of $\mathbf{E}$ is also normalized to unit norm.

Our task is to recover $\mathbf{D}$ from $\mathbf{Y}$. We will first use SE to simultaneously reduce the

dimension via $\mathbf{A}$ and learn a dictionary via $\mathbf{B}$. The original dictionary can be estimated as $\hat{\mathbf{D}} = \mathbf{YB}$. We compare our results with KSVD. To demonstrate the benefit of our proposed joint dimensionality reduction and dictionary learning, we also compare our results with the approach when the dimensionality reduction is done using PCA before learning the dictionary using KSVD.

Let $\mathbf{P}_{pca}$ represent the PCA transformation. We learn a dictionary, denoted $\mathbf{D}_{pca}$, using KSVD on the set of reduced signals $\mathbf{P}_{pca}\mathbf{Y}$. The original dictionary is recovered by first computing the coefficient matrix $\mathbf{B}_{pca} = (\mathbf{P}_{pca}\mathbf{Y})^{\dagger}\mathbf{D}_{pca}$, and then $\hat{\mathbf{D}} = \mathbf{YB}_{pca}$. Note that since the columns of $\mathbf{D}_{pca}$ are in the column subspace of $\mathbf{P}_{pca}\mathbf{Y}$, the computation of the coefficient matrix $\mathbf{B}_{pca}$ is exact.

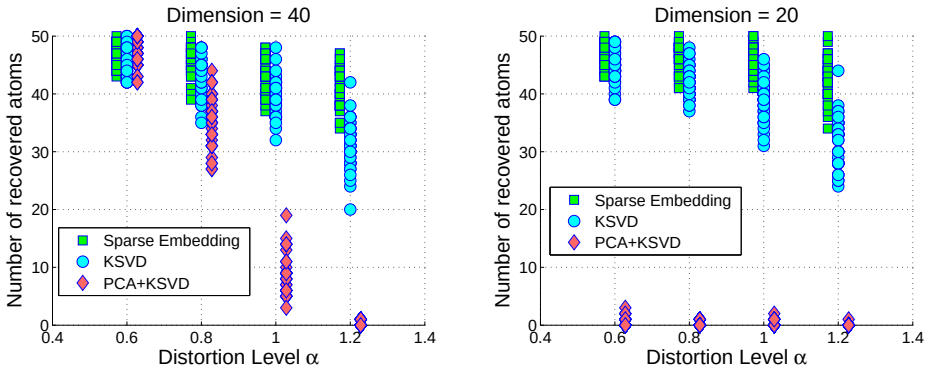


Figure 4.2: Comparison of number of recovered dictionary atoms over 40 trials.

Verification of Recovered Dictionaries: For all methods, the computed dictionary was compared against the generating dictionary. This comparison is done by sweeping through the columns of the generating dictionary to find the closest column (in ℓ_2 distance). The distance is measured by $\left(1 - |\mathbf{d}_i^T \hat{\mathbf{d}}_i|\right)$, where $\hat{\mathbf{d}}_i$ is the i-th estimated dictionary atom, and $\mathbf{d}_i$ is its closest atom in the original dictionary. A distance of less than 0.01 is considered a success. We learn dictionaries with sparsity level $T_0 = 3$, the dictionary

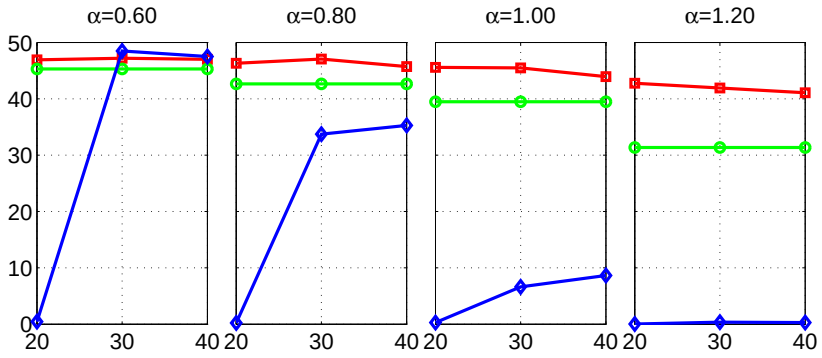


Figure 4.3: Average number of successfully recovered dictionary atoms versus the dimension of the reduced space for different distortion level α. Blue color line corresponds to results for PCA+KSVD scheme, green color line for KSVD, and red color line for SE.

size $K = 50$, and $\lambda = 1.1$.

Fig. 4.2 compares the number of successes over 40 trials. Fig. 4.3 plots the average number of success versus the dimension of the output space for different distortion levels. SE consistently outperforms both KSVD and PCA+KSVD with larger and larger performance gaps as the distortion level increases. Fig. 4.3 shows that the performance of PCA+KSVD decreases drastically not only when the level of distortion level increases, but also when the dimension gets smaller than 30, which is the true dimension of our

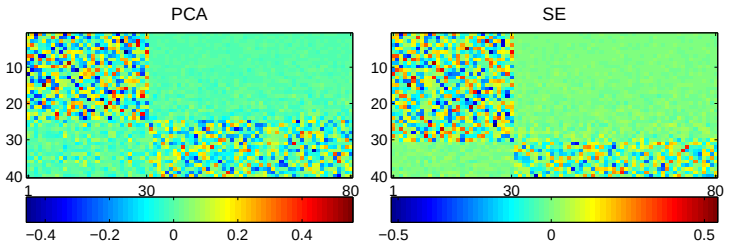


Figure 4.4: Comparison of PCA mapping (left) and the transformation $\mathbf{P}$ learned from SE (right). Distortion level $\alpha = 1$, dimension of the reduced space is 40.

sparse signals. In contrast, SE outperforms KSVD even when the dimension goes below 30. Interestingly, for the high distortion level like $\alpha = 1.2$, it is beneficial to reduce the dimension to even below the true dimension of sparse signals (see the right most chart of Fig. 4.3).

In order to understand the reason behind the good results of SE, we visually inspect the transformation $\mathbf{P}$. Fig. 4.4 shows the images of $\mathbf{P}$ and the PCA mapping. The first 30 rows of the $\mathbf{P}$ weight heavily on the first 30 dimensions of $\mathbf{Y}$. In other words, SE efficiently preserves sparse structures of signals and discards non-sparse distortions. In contrast, PCA does not preserve the sparse patterns since it is attempting to capture more of the signal variation. Only around 24 rows of PCA focus on the first 30 dimensions and the rest put more emphasis on non-sparse structures. Being able to get rid of distortions while preserving the sparse structures enables SE to achieve a higher recovery rate.

4.5.2 Latent Sparse Embedding Residual Classifier (LASERC)

Classification is an important component in many computer vision applications. In this section, we propose a novel classification scheme motivated by the SE framework. For generality, we will consider the non-linear setting. Let there be C different classes of signals $\{\mathbf{Y}_i = [\mathbf{y}_j^i]_{j=1}^{N_i} \in \mathbb{R}^{n \times N_i}\}_{i=1}^{C}$. We use the SE algorithm in Fig. 4.1 to learn $\{\mathbf{A}_i, \mathbf{B}_i\}_{i=1}^{C}$, which implicitly provides $\{\boldsymbol{\mathcal{P}}_i, \mathbf{D}_i\}_{i=1}^{C}$. Given a new test sample $\mathbf{s}_t$, the classification is done in three steps:

I) We compute output signals $\mathbf{z}_i$ by mapping $\mathbf{s}_t$ via $\mathcal{M}_i$,

$$\mathcal{M}_i : \ \mathbf{s}_t \rightarrow \mathbf{z}_i = \boldsymbol{\mathcal{P}}_i \Phi(\mathbf{s}_t) = \mathbf{A}_i^T \langle \Phi(\mathbf{Y}_i), \Phi(\mathbf{s}_t) \rangle_{\mathcal{H}} = \mathbf{A}_i^T \mathbf{k}_{i,t} \tag{4.25}$$

$$\text{where: } \mathbf{k}_{i,t} = [k(\mathbf{y}_1^i, \mathbf{s}_t), \ldots, k(\mathbf{y}_{N_i}^i, \mathbf{s}_t)]^T \in \mathbb{R}^{N_i} \tag{4.26}$$

II) We obtain the sparse code $\mathbf{x}_i$ for $\mathbf{z}_i$ over the dictionary $\mathbf{D}_i = \mathbf{A}_i^T \boldsymbol{\mathcal{K}}_i \mathbf{B}_i$ using the OMP algorithm, where

$$\boldsymbol{\mathcal{K}}_i = \langle \Phi(\mathbf{Y}_i), \Phi(\mathbf{Y}_i) \rangle_{\mathcal{H}} = [k(\mathbf{y}_j^i, \mathbf{y}_k^i)]_{j,k=1}^{N_i} \in \mathbb{R}^{N_i \times N_i} \tag{4.27}$$

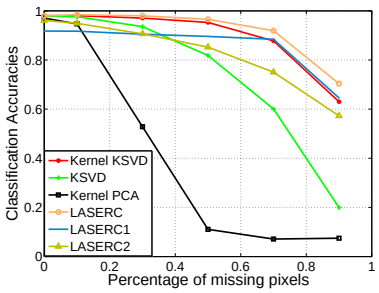
III) We compute the residual for each class as follows:

$$r_i = \|\Phi(\mathbf{s}_t) - \boldsymbol{\mathcal{P}}_i^T \mathbf{D}_i \mathbf{x}_i\|_{\mathcal{H}}^2 = k(\mathbf{s}_t, \mathbf{s}_t) - 2\mathbf{k}_{i,t}^T \mathbf{A}_i \mathbf{D}_i \mathbf{x}_i + \mathbf{x}_i^T \mathbf{D}_i^T \mathbf{A}_i^T \boldsymbol{\mathcal{K}}_i \mathbf{A}_i \mathbf{D}_i \mathbf{x}_i \tag{4.28}$$

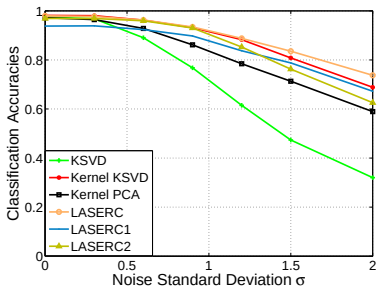
Here, $\mathbf{D}_i \mathbf{x}_i$ is the estimated signal in the output space, and $\boldsymbol{\mathcal{P}}_i^T \mathbf{D}_i \mathbf{x}_i$ is the estimated signal in the feature space. The sparse coding step makes our algorithm more resilient to noise. Finally, each signal is assigned to the class that yields the smallest reconstruction residual. We call this *latent sparse embedding residual classifier* (LASERC). The term "latent" comes from the fact that the residual errors are computed in the feature space instead of the input space which does not take into account the non-linearities of the signal. The output space is also not suitable for classification because it does not retain sufficient discriminatory power due to its low-dimensional nature.

4.5.2.1 USPS Digit Recognition

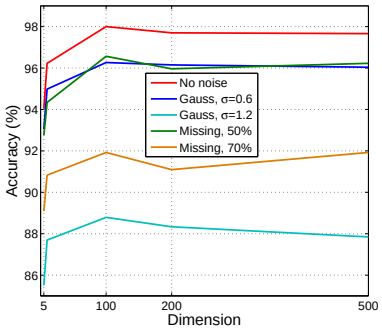
We apply our classifier on the USPS database [89] which contains ten classes of 256-dimensional handwritten digits. A dictionary is learned for each class using samples



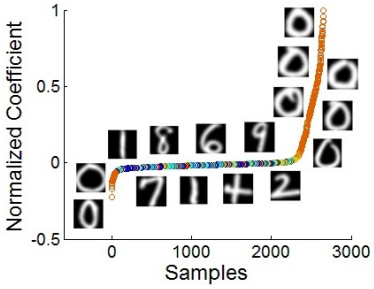
(a) Missing pixels



(b) Gaussian noise



(c) Accuracy versus dimension



(d) Projection Coefficient

Figure 4.5: (a,b) Comparison of classification accuracy against noise level. (c) Accuracy versus dimension. (d) Projection coefficient of all samples onto a dictionary atom.

from the training set with the following parameters setting: 500 dictionary atoms, $T_0 = 5$, $d = 100$, $\lambda = 1.1$, and the maximum number of iterations is set to 80. A polynomial kernel of degree 4 with the constant of value 1 is used for SE, kernel KSVD, and kernel PCA.

Our first experiment presents the results for the case when the pixels are randomly removed from the test images shown in Fig. 4.5(a), and when the test samples are corrupted by Gaussian noise with different standard deviations shown in Fig. 4.5(b). In both scenarios, LASERC consistently outperforms kernel KSVD, linear KSVD, and kernel PCA. As the distortion level increases the performance differences between sparse em-

bedding and linear KSVD become more drastic.

It is also worthwhile to investigate cases when the objective function in (4.7) has only the first term ($\lambda = 0$), denoted by LASERC1, and when there is only the second term ($\lambda \to \infty$), denoted by LASERC2. Fig 4.5(a) and 4.5(b) show that LASERC2 performs better for the low-noise cases and worse for the high-noise cases in comparison with LASERC1.

In order to see the effect of dimension on the classification performance of LASERC, we compare the results for different values of dimension $d = \{5, 10, 100, 200, 500\}$. The corresponding sparsity level is $T_0 = \{2, 3, 5, 5, 10\}$. Fig. 4.5(c) shows that the classification result improves as the dimension increases from $5 \to 100$. Beyond 100, the accuracy decreases slightly for the noiseless case, but faster for the very noisy cases like Gaussian noise with $\sigma = 1.2$.

We project test samples onto a random dictionary atom of the first class (digit 0). Fig. 4.5(d) plots the sorted projection coefficients of all the samples, color-coded by their class labels. We can observe from the plot 4.5(d) that, in the feature space, the dictionary atom is almost perpendicular to all samples except those from the first class (orange color at the two ends). This implies that the learned dictionary has taken into account the non-linearities of signals.

4.5.2.2 Caltech-101 and Caltech-256 Object Detection

We perform the second set of object recognition experiments on the Caltech-101 database [97]. This database comprises of 101 object classes, and 1 background class

(a) Okapi (100%) (b) Leopard (100%) (c) Face (100%)

(d) Mayfly (20%) (e) Cougar (35%) (f) Seahorse (42%)

Figure 4.6: Sample images from the classes corresponding to the highest accuracy (top row) and the lowest accuracy (bottom row) of LASERC.

# train samples	5	10	15	20	25	30
Malik [103]	46.6	55.8	59.1	62.0	-	66.2
Lazebnik [10]	-	-	56.4	-	-	64.6
Griffin [104]	44.2	54.5	59.0	63.3	65.8	67.6
Irani [105]	-	-	65.0	-	-	70.4
Yang [108]	-	-	67.0	-	-	73.2
Wang [109]	51.15	59.77	65.43	67.74	70.16	73.44
SRC [90]	48.8	60.1	64.9	67.7	69.2	70.7
KSVD [9]	49.8	59.8	65.2	68.7	71.0	73.2
D-KSVD [13]	49.6	59.5	65.1	68.6	71.1	73.0
LC-KSVD [12]	54.0	63.1	67.7	70.5	72.3	73.6
LASERC	**55.2**	**65.6**	**69.5**	**73.1**	**75.8**	**77.3**

# train samples	15	30
Griffin [104]	28.3	34.1
Gemert [107]	-	27.2
Yang [110]	34.4	41.2
LASERC	**35.2**	**43.6**

Time	Train (s)	Test (ms)
SVM	2.1	8.1
Ker. KSVD	2598	3578
SRC	N/A	520
D-KSVD	450	12.8
LASERC	**7.2**	**9.4**

Table 4.1: Comparison of recognition results on Caltech-101 dataset (left), recognition results on Caltech-256 dataset (upper right), and the computing time (lower right).

collected from Internet. The database contains a diverse and challenging set of images from buildings, musical instruments, animals and natural scenes, etc. Spatial pyramid

descriptors [10] are used with the following settings: 16×16 patch size, 8 pixels grid spacing, 3 pyramid levels, 200 quantized clusters of feature vectors. This results in a 4200-dimensional descriptor. The similarity is measured using the pyramid match kernel [10], which is also a Mercer kernel.

We follow the suggested protocol in [10, 102], namely, we train on m images, where $m \in \{5, 10, 15, 20, 25, 30\}$, and test on the rest. The corresponding parameters settings of SE are: $T_0 = \{3, 4, 5, 7, 8, 9\}$, $d = \{5, 10, 15, 20, 25, 30\}$, and $\lambda = 1.1$. To compensate for the variation of the class size, we normalize the recognition results by the number of test images to get per-class accuracies. The final recognition accuracy is then obtained by averaging per-class accuracies across 102 categories. We also repeat the same experiment on Caltech-256 dataset.

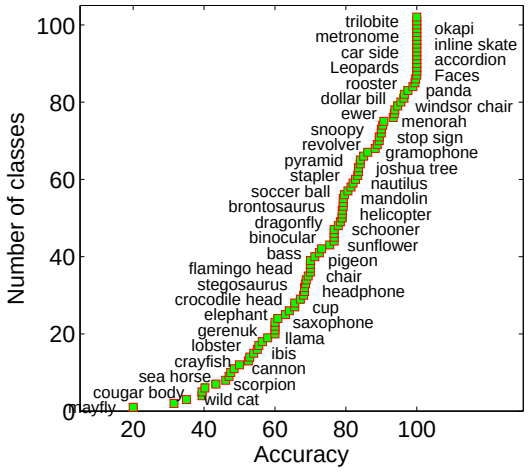


Figure 4.7: Caltech-101 Per Class Accuracy

Table 4.1 shows the comparison of our classification accuracy with the state of the art. It is interesting that our method significantly outperforms the other discriminative approaches like LC-KSVD [12] and D-KSVD [13]. Thanks to the help of DR, our method

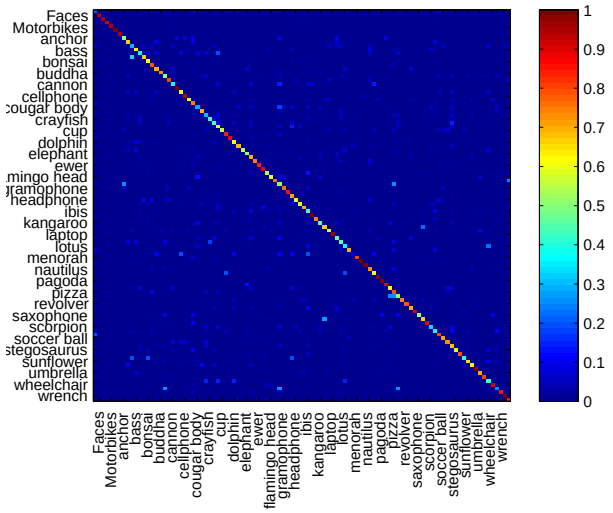


Figure 4.8: Caltech-101 Confusion Matrix

achieves a significant speed-up in the training process over the other sparse learning methods as shown in table 4.1. Fig. 4.6 shows sample images from the easiest classes as well as the most difficult classes. Fig. 4.7 shows the recognition accuracy per class, and Fig. 4.8 shows their confusion matrix.

4.6 Conclusion

This chapter presented a novel framework for a joint dimensionality reduction and sparse learning. It proposes an efficient algorithm for solving the resulting optimization problem. It designs a novel classification scheme leading to a state-of-the-art performance and robustness on several popular datasets. Through extensive experimental results on real and synthetic data, we showed that sparse learning techniques can benefit significantly from dimensionality reduction in terms of both computation and accuracy.

4.7 Proof of Proposition 4.1

The proposed optimization:

$$\{\mathbf{P}^*, \mathbf{D}^*, \mathbf{X}^*\} = \underset{\mathbf{P},\mathbf{D},\mathbf{X}}{\operatorname{argmin}} \left(\|\mathbf{PY} - \mathbf{DX}\|_F^2 + \lambda\|\mathbf{Y} - \mathbf{P}^T\mathbf{PY}\|_F^2\right) \tag{4.29}$$

$$\text{subject to: } \mathbf{PP}^T = \mathbf{I}, \text{ and } \|\mathbf{x}_i\|_0 \leq T_0, \ \forall i$$

Recall the proposition that we want to prove:

Proposition 4.1. *There exists an optimal solution* $\mathbf{P}^*$ *and* $\mathbf{D}^*$ *to (4.29), for sufficiently large* λ*, that has the following form:*

$$\mathbf{D}^* = \mathbf{A}^T\mathbf{Y}^T\mathbf{YB} \quad \mathbf{P}^* = (\mathbf{YA})^T; \tag{4.30}$$

for some $\mathbf{A} \in \mathbb{R}^{N\times d}$*, and some* $\mathbf{B} \in \mathbb{R}^{N\times K}$.

Proof. First, we will show that for a specific $\mathbf{P}$, there exist an optimal solution of $\mathbf{D}$ that has the form $\mathbf{PYB}$, for some $\mathbf{B} \in \mathbb{R}^{N\times K}$.

Let $\mathbf{D}_\mathbf{p}^*$ be an optimal solution corresponding to $\mathbf{P}$. For simplicity of notation, we will write $\mathbf{D}^*$ instead of $\mathbf{D}_\mathbf{p}^*$. Since the columns of $\mathbf{D}^*$ have the same dimension with the columns of $\mathbf{PY}$, we can express $\mathbf{D}^*$ using the orthogonal decomposition as follows:

$$\mathbf{D}^* = \mathbf{D}_\parallel + \mathbf{D}_\perp \tag{4.31}$$

$$\text{where:} \quad \mathbf{D}_\parallel = (\mathbf{PY})\mathbf{B} \quad \text{and} \quad \mathbf{D}_\perp^T(\mathbf{PY}) = \mathbf{0} \tag{4.32}$$

for some appropriate $\mathbf{B} \in \mathbb{R}^{N\times K}$. Basically, columns of $\mathbf{D}_\parallel$ and $\mathbf{D}_\perp$ are in and orthogonal to the column subspace of $\mathbf{PY}$, respectively. Since $\mathbf{P}$ is fixed, the cost function in (4.3)

only depends on the first term:

$$\|\mathbf{PY} - \mathbf{D}^*\mathbf{X}\|_F^2 = \|\mathbf{PY} - (\mathbf{D}_\parallel + \mathbf{D}_\perp)\mathbf{X}\|_F^2 \tag{4.33}$$

$$= \mathbf{tr}\left(\mathbf{Y}^T\mathbf{P}^T\mathbf{PY} + \mathbf{Y}^T\mathbf{P}^T(\mathbf{D}_\parallel + \mathbf{D}_\perp)\mathbf{X} + \mathbf{X}^T(\mathbf{D}_\parallel + \mathbf{D}_\perp)^T(\mathbf{D}_\parallel + \mathbf{D}_\perp)\mathbf{X}\right) \tag{4.34}$$

$$= \mathbf{tr}\left(\mathbf{Y}^T\mathbf{P}^T\mathbf{PY} + \mathbf{Y}^T\mathbf{P}^T\mathbf{D}_\parallel\mathbf{X} + \mathbf{X}^T\mathbf{D}_\parallel^T\mathbf{D}_\parallel\mathbf{X} + \mathbf{X}^T\mathbf{D}_\perp^T\mathbf{D}_\perp\mathbf{X}\right) \tag{4.35}$$

$$\geq \mathbf{tr}\left(\mathbf{Y}^T\mathbf{P}^T\mathbf{PY} + \mathbf{Y}^T\mathbf{P}^T\mathbf{D}_\parallel\mathbf{X} + \mathbf{X}^T\mathbf{D}_\parallel^T\mathbf{D}_\parallel\mathbf{X}\right) \tag{4.36}$$

The last equality is true because $\mathbf{X}^T\mathbf{D}_\perp^T\mathbf{D}_\perp\mathbf{X}$ is a positive semi-definite matrix, thus, its trace is non-negative. In order for $\mathbf{D}^*$ to be an optimal solution, the cost function must achieve the smallest value, i.e., the term in (4.35) has to be equal to the term in (4.36). A sufficient condition is $\mathbf{D}_\perp = \mathbf{0}$. In other words, $\mathbf{D}_\parallel$ is also an optimal solution. As a result, there exists an optimal solution of $\mathbf{D}$ for each $\mathbf{P}$ of the form:

$$\mathbf{D}_\parallel = \mathbf{PYB} \tag{4.37}$$

Moreover, the optimal solution of this form has smallest Frobenius norm among all optimal solutions of $\mathbf{D}$ because $\|\mathbf{D}^*\|_F^2 = \|\mathbf{D}_\parallel\|_F^2 + \|\mathbf{D}_\perp\|_F^2 \geq \|\mathbf{D}_\parallel\|_F^2$ or $\|\mathbf{D}^*\|_F \geq \|\mathbf{D}_\parallel\|_F$.

Second, we will show that the optimal solution of $\mathbf{P}$, denoted by $\mathbf{P}^*$, must have the form $\mathbf{P}^* = (\mathbf{YA})^T$, for some $\mathbf{A} \in \mathbb{R}^{N\times d}$.

Using the orthogonal decomposition of $\mathbf{P}^*$, we have:

$$\mathbf{P}^* = \mathbf{P}_\parallel + \mathbf{P}_\perp \tag{4.38}$$

$$\text{where:} \quad \mathbf{P}_\parallel = (\mathbf{YA})^T \quad \text{and} \quad \mathbf{P}_\perp\mathbf{Y} = \mathbf{0} \tag{4.39}$$

for some appropriate $\mathbf{A} \in \mathbb{R}^{N\times d}$. Rows of $\mathbf{P}_\parallel$ and $\mathbf{P}_\perp$ are in and orthogonal to the column subspace of $\mathbf{Y}$, respectively. Using Eqs. (4.37) and (4.38), we can rewrite the first term

of the objective function in (4.3),

$$\|\mathbf{P}^*\mathbf{Y}(\mathbf{I}-\mathbf{B}\mathbf{X})\|_F^2 = \|(\mathbf{P}_\parallel+\mathbf{P}_\perp)\mathbf{Y}(\mathbf{I}-\mathbf{B}\mathbf{X})\|_F^2 \tag{4.40}$$

$$= \|\mathbf{P}_\parallel\mathbf{Y}(\mathbf{I}-\mathbf{B}\mathbf{X})\|_F^2 = \mathbf{tr}\left(\mathbf{P}_\parallel\mathbf{Y}(\mathbf{I}-\mathbf{B}\mathbf{X})(\mathbf{I}-\mathbf{B}\mathbf{X})^T\mathbf{Y}^T\mathbf{P}_\parallel^T\right) \tag{4.41}$$

and the second term,

$$\|\mathbf{Y}-\mathbf{P}^{*T}\mathbf{P}^*\mathbf{Y}\|_F^2 = \mathbf{tr}\left(\mathbf{Y}^T\mathbf{Y}-2\mathbf{Y}^T\mathbf{P}^{*T}\mathbf{P}^*\mathbf{Y}+\mathbf{Y}^T\mathbf{P}^{*T}\mathbf{P}^*\mathbf{P}^{*T}\mathbf{P}^*\mathbf{Y}\right) \tag{4.42}$$

$$= \mathbf{tr}\left(\mathbf{Y}^T\mathbf{Y}-\mathbf{Y}^T\mathbf{P}^{*T}\mathbf{P}^*\mathbf{Y}\right) = \mathbf{tr}\left(\mathbf{Y}^T\mathbf{Y}-\mathbf{Y}^T(\mathbf{P}_\parallel+\mathbf{P}_\perp)^T(\mathbf{P}_\parallel+\mathbf{P}_\perp)\mathbf{Y}\right) \tag{4.43}$$

$$= \mathbf{tr}\left(\mathbf{Y}^T\mathbf{Y}-\mathbf{Y}^T{\mathbf{P}_\parallel}^T\mathbf{P}_\parallel\mathbf{Y}\right) = \mathbf{tr}\left(\mathbf{Y}^T\mathbf{Y}-\mathbf{P}_\parallel\mathbf{Y}\mathbf{Y}^T{\mathbf{P}_\parallel}^T\right) \tag{4.44}$$

From Eq. (4.42) to Eq. (4.43), we use the constraint $\mathbf{P}^*\mathbf{P}^{*T} = \mathbf{I}$. Putting Eqs. (4.39) (4.41) (4.44) together, and let $\boldsymbol{\mathcal{K}} = \mathbf{Y}^T\mathbf{Y}$, the objective function becomes

$$\mathbf{tr}(\boldsymbol{\mathcal{K}}) - \mathbf{tr}\left(\mathbf{P}_\parallel\mathbf{Y}\left(\lambda\mathbf{I}-(\mathbf{I}-\mathbf{B}\mathbf{X})(\mathbf{I}-\mathbf{B}\mathbf{X})^T\right)\mathbf{Y}^T\mathbf{P}_\parallel^T\right) \tag{4.45}$$

$$= \mathbf{tr}(\boldsymbol{\mathcal{K}}) - \mathbf{tr}\left(\mathbf{A}^T\boldsymbol{\mathcal{K}}\left(\lambda\mathbf{I}-(\mathbf{I}-\mathbf{B}\mathbf{X})(\mathbf{I}-\mathbf{B}\mathbf{X})^T\right)\boldsymbol{\mathcal{K}}\mathbf{A}\right) \tag{4.46}$$

Let $\boldsymbol{\mathcal{K}} = \mathbf{V}\mathbf{S}\mathbf{V}^T$, $\tilde{\mathbf{H}} = \mathbf{S}^{\frac{1}{2}}\mathbf{V}^T\left(\lambda\mathbf{I}-(\mathbf{I}-\mathbf{B}\mathbf{X})(\mathbf{I}-\mathbf{B}\mathbf{X})^T\right)\mathbf{V}\mathbf{S}^{\frac{1}{2}}$, and $\mathbf{G} = \mathbf{S}^{\frac{1}{2}}\mathbf{V}^T\mathbf{A}$. Note that $\tilde{\mathbf{H}}$ is positive semi-definite for sufficiently large value of λ. We can derive the lower bound of the cost function

$$\mathbf{tr}(\boldsymbol{\mathcal{K}}) - \mathbf{tr}\left(\mathbf{A}^T\mathbf{V}\mathbf{S}^{\frac{1}{2}}\tilde{\mathbf{H}}\mathbf{S}^{\frac{1}{2}}\mathbf{V}^T\mathbf{A}\right) \tag{4.47}$$

$$= \mathbf{tr}(\boldsymbol{\mathcal{K}}) - \mathbf{tr}\left(\mathbf{G}^T\tilde{\mathbf{H}}\mathbf{G}\right) \geq \mathbf{tr}(\boldsymbol{\mathcal{K}}) - \sum_{i=1}^{d}\alpha_i \tag{4.48}$$

where α_i is the i-th largest eigenvalue of $\tilde{\mathbf{H}}$. In order for the objective function to achieve its minimum, columns of $\mathbf{G}$ have to be the same with eigenvectors corresponding to largest eigenvalues of $\tilde{\mathbf{H}}$. Therefore, $\mathbf{G}^T\mathbf{G} = \mathbf{I}$. Equivalently,

$$\mathbf{G}^T\mathbf{G} = \mathbf{A}^T\boldsymbol{\mathcal{K}}\mathbf{A} = \mathbf{P}_\parallel\mathbf{P}_\parallel^T = \mathbf{I}-\mathbf{P}_\perp\mathbf{P}_\perp^T = \mathbf{I} \tag{4.49}$$

which means that $\mathbf{P}_\perp = \mathbf{0}$. In short, the optimal solution of $\mathbf{P}$ has the form

$$\mathbf{P}^* = \mathbf{P}_\parallel = (\mathbf{YA})^T \tag{4.50}$$

This also solution has the smallest Frobenius norm since all optimal solutions of $\mathbf{P}$ have Frobenius norm equal to $\sqrt{d}$:

$$\|\mathbf{P}^*\|_F = \sqrt{\mathbf{tr}(\mathbf{P}^{*T}\mathbf{P}^*)} = \sqrt{\mathbf{tr}(\mathbf{P}^*\mathbf{P}^{*T})} = \sqrt{\mathbf{tr}(\mathbf{I})} = \sqrt{d} \tag{4.51}$$

Using the form of $\mathbf{P}^*$ and Eq. (4.37), we conclude that there exist an optimal solution of $\mathbf{D}$ that has the form $\mathbf{D}^* = \mathbf{P}^*\mathbf{YB} = \mathbf{A}^T\mathbf{Y}^T\mathbf{YB}$. This completes the proof.

□

Chapter 5

Domain Adaptation Using a Sparse and Hierarchical Network

5.1 Motivation

In many practical computer vision applications, we are often confronted with the situation where the data that we use to train our algorithm have different distribution or representation with that presented during the testing. The ubiquity of this problem is well-known to computer vision researchers. For instance, indoor images are quite different from outdoor images, just as videos captured with a high definition camera are from that collected with a webcam. This detrimental effect is often a dominant factor accounting for the poor performances of many computer vision algorithms. As an example of the effect of distribution mismatch, Ben-David *et al.* [15] show that, under certain assumption, the bound on the test error linearly increases with the ℓ_1 divergence between the training and testing distributions. Even worse, data from the test domain are often scarce and expensive to obtain. This makes it impractical to re-train an algorithm from the scratch since a learning algorithm would generalize poorly when insufficient amount of data is presented [16].

Understanding the problem caused by domain change has received substantial attention in recent years (see [125] and references therein). The problem can be informally stated as follows. Given a source domain whose representation or distribution can be different from that of the target domain, how to effectively utilize the model trained on the

source to achieve a good performance on the target data. It is also often assumed that the source domain has plentiful labelled training samples while there are only a few (both labelled and unlabelled) samples available from the target domain. It has been shown in [126, 127, 128, 129, 130] that domain adaptation techniques can significantly improve the performance of computer vision tasks such as visual object detection and recognition.

Although the formulation differs, most of the algorithms for adapting a recognition system to a new visual domain share a common architecture containing two main stages. First, features are extracted *separately* for source and target using *hand-crafted* feature descriptors, followed by the second stage where transformations are learned in order to rectify the discrepancy between the two domains. This architecture has several drawbacks. Without any knowledge about the target domain, the feature extraction performed on the source data can ignore information important to the target data. For instance, let us consider a picture of a small bird in a forest background. The forest features apparently dominate the picture description. While these features might be sufficient for classifying the picture into land animals or sea animals, they do not benefit the target domain which has the same task on images of animals without background.

Another issue is that discriminative information can be embedded in multiple levels of the features hierarchy. High-level features are sometimes more useful than low-level ones. In fact, this is one of the main motivations behind the development of hierarchical networks (e.g. [131, 132]) so that more complex abstraction from a visual object can be captured. The traditional framework of domain adaptation employs a shallow architecture containing a single layer. This ignores the possibility of transferring at multiple levels of the feature hierarchy.

Finally, the process of designing features, such as SIFT [133] or SURF [134], is tedious and time-consuming. It requires a deep understanding and a careful examination of the underlying physics that governs the generation of data. Such requirements might be impractical given that the data from the target domain are often very scarce.

Contributions: In order to address the limitations of existing approaches, we propose a novel approach for domain adaptation that possesses the following advantages:

- Adaptation is performed on multiple levels of the feature hierarchy in order to maximize the knowledge transfer. The hierarchical structure allows the transfer of useful information that might not be well captured by existing domain adaptation techniques.

- Adaptation is done jointly with feature learning. Our method learns a hierarchy of sparse codes and uses them to describe a visual object instead of relying on any low-level feature.

- Unlike existing hierarchical networks, our network is more computationally efficient with a mechanism to prevent the data dimension from increasing too fast as the number of layer increases.

We provide extensive experiments to show that our approach outperforms the current state-of-the-art by a large margin. This is interesting since our training is entirely generative followed by a linear support vector machine while several other methods employ discriminative training together with non-linear kernels. Furthermore, we introduce a new set of data for benchmarking the performance of our algorithm. The new dataset has two domains containing half-toning and edge images, respectively.

5.2 Related Works

While domain adaptation was first investigated in speech and natural language processing [135, 136], it has been studied extensively in other areas such as machine learning [125, 15] and computer vision, especially in the context of visual object recognition [126, 127, 128, 129, 130]. For instance, the semi-supervised approach proposed by Saenko *et al.* [126] employed metric learning to learn the domain shift using partially labeled data from the target domain. This work was extended by Kulis *et al.* [127] to handle asymmetric domain transformations. Gopalan *et al.* [128] addressed the problem of unsupervised domain adaptation, where samples from the target domain are unlabeld, using an incremental approach based on Grassmann manifolds. By formulating a geodesic flow kernel, [129, 137] extended this approach to integrate an infinite number of subspaces on the geodesic flow from the source domain to the target domain.

Sparse methods have also been used to address the domain shift problem [138, 139]. In particular, [138] modelled dictionaries across different domains with a parametric mapping function, while [139] enforced different domains to have a common sparse representation on some latent domain. Another class of techniques performed domain adaptation by directly learning a target classifier from classifiers trained on the source domain(s) [140, 141]. A drawback of the existing approaches is that the domain shifting transformation is considered only at a single layer and may not capture adequately the shift between the source and target domain. It is worth noting that although [142] also named their method hierarchical domain adaptation, it is not quite related to ours. They made use of hierarchical Bayesian prior while we employ a multi-layer network of sparse representa-

tion.

The design of multi-layer networks has been an active research topic in computer vision. Some of the early work includes [143], which used a multistage system to extract salient features in the image at different spatial scales. By learning higher-level feature representations from unlabelled data, deep belief networks (DBN) [131] and its variants, such as convolutional DBNs [132] and deep autoencoders [144], have been shown to be effective when applied to classification problems. Motivated by recent works on deep learning, multi-layer sparse coding networks [145, 146, 147] proposed to build feature hierarchies layer by layer using sparse codes and spatial pooling. A common problem with these hierarchical approaches is that the dimension of the learned feature vectors keeps increasing after each layer and thus, may make the computations become costly.

5.3 Proposed Approach

In this section, we briefly summarize some related work in sparse coding and dictionary learning. Recent advances in feature learning using hierarchical sparse coding are also discussed. We then proceed to introduce the main formulation of DASH-N.

5.3.1 Latent Sparse Representation

Given a set of training samples $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n] \in \mathbb{R}^{d \times n}$, the problem of learning a dictionary together with the sparse codes is typically posed as the minimization of the following cost function over ($\mathbf{D}$, $\mathbf{X}$):

$$\|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \beta\|\mathbf{X}\|_1 \;\; \text{s.t.} \;\; \|\mathbf{d}_i\|_2 = 1, \forall i \in [1, K] \tag{5.1}$$

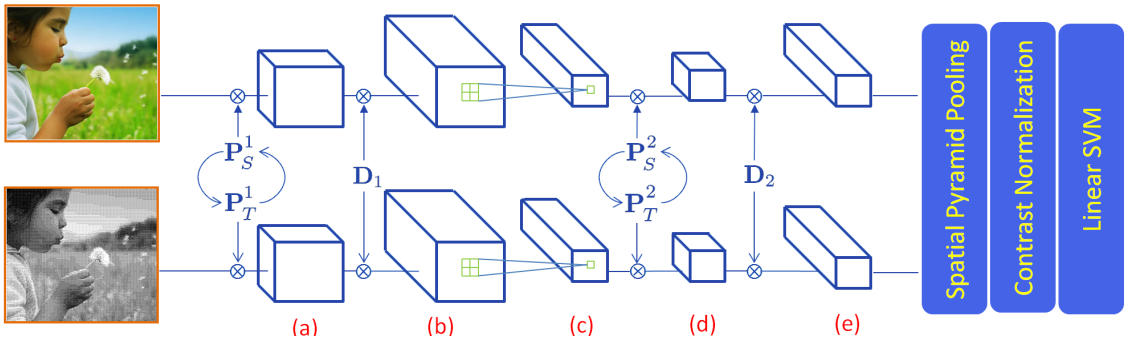


Figure 5.1: An illustration of DASH-N algorithm. The source domain is RGB images and the target domain is halftone images. First, images are divided into small overlapping patches. These patches are vectorized while maintaining their spatial arrangements. (a) Performing contrast-normalization and dimensionality reduction using $\mathbf{P}_S$ for source images and $\mathbf{P}_T$ for target images. The circular feedbacks between $\mathbf{P}_S$ and $\mathbf{P}_T$ indicate that these two transformations are learned jointly. (b) Obtaining sparse codes using the common dictionary $\mathbf{D}_1$. (c) Performing max pooling. The process then repeats for layer 2 (d & e), except that the input is the sparse codes from layer 1 instead of pixel intensities. At the final stage, spatial pyramid with max pooling are used to create image descriptors. Classification is done using linear support vector machine.

where $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_K] \in \mathbb{R}^{d \times K}$ is the sought dictionary, $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{K \times n}$ is the horizontal concatenation of the sparse codes, and β is a non-negative constant. The ℓ_1 constraint is well-known for promoting sparsity of the coefficients in $\mathbf{X}$. ℓ_0 pseudo-norm can also be used for sparse regularization. However, we choose to use ℓ_1 norm because the optimization algorithms possesses a better theoretical guarantee for successfully recovering the true sparse signals due to its convex nature [148].

From the observation that signals often lie on a low-dimensional manifold, several authors have proposed to perform dictionary learning and sparse coding on a latent space [149, 150]. We call it *latent sparse representation* to distinguish from the formulation in (5.1). This is done by minimizing the following cost function over ($\mathbf{P}$, $\mathbf{D}$, $\mathbf{X}$):

$$\begin{aligned} &\mathcal{L}(\mathbf{Y}, \mathbf{P}, \mathbf{D}, \mathbf{X}, \alpha, \beta) = \\ &\|\mathbf{PY} - \mathbf{DX}\|_F^2 + \alpha\|\mathbf{Y} - \mathbf{P}^T\mathbf{PY}\|_F^2 + \beta\|\mathbf{X}\|_1 \\ &\text{s.t.} \quad \mathbf{PP}^T = \mathbf{I} \text{ and } \|\mathbf{d}_i\|_2 = 1, \ \forall i \in [1, K] \end{aligned} \tag{5.2}$$

where $\mathbf{P} \in \mathbb{R}^{p \times d}$ is a linear transformation that brings the data to a low-dimensional feature space ($p < d$). Note that the dictionary is now in the low-dimensional space $\mathbf{D} \in \mathbb{R}^{p \times K}$. Besides the computational advantage, [150] shows that this optimization can recover the underlying sparse representation better than the traditional dictionary learning methods. This formulation is attractive since it allows the transformation of the data into another domain to better handle different sources of variation such as illumination and geometric articulation.

5.3.2 Multi-layer Feature Learning

Designing features for visual object is a time-consuming and challenging task that requires a deep understanding of the domain knowledge. It is also non-trivial to adapt these manually designed features to new types of data such as hyperspectral or range-scan images. For this reason, learning features from the raw data has become increasingly popular and demonstrated competitive performances on practical computer vision tasks [146]. In order to capture the richness of data, a hierarchical network is employed to learn a spectrum of features, layer by layer.

Very recently, multi-level sparse coding networks including hierarchical sparse coding [145, 146, 147] have been proposed for feature learning. These networks contain a coding layer and a pooling layer. A dictionary is learned at each coding layer, using (5.1), which then serves as a codebook for obtaining sparse codes from image patches or pooled features. Spatial pooling schemes, most notably max-pooling, group the sparse codes from adjacent blocks into common entities. This operation makes the resulting features more invariant to certain changes caused by translation and rotation. The pooled sparse codes from one layer serve as the input to the next layer. While feature learning has been successfully applied to object recognition on RGB images and 3-D point clouds [151], to the best of our knowledge, our method is one of the first attempts for integrating the feature learning step into domain adaptation.

5.3.3 Hierarchical Domain Adaptation

Motivated by the work of [146], we propose a method to perform hierarchical domain adaptation jointly with feature learning. Figure 5.1 shows an overview of the proposed method. The network contains multiple layers, each of which contains 3 sub-layers as illustrated in Figure 5.1. The first sub-layer performs contrast-normalization and dimensionality reduction on the input data. Sparse coding is carried out in the second sub-layer. In the final sub-layer, adjacent features are max-pooled together to produce a new features. Output from one layer becomes the input to the next layer. For the simplicity of notation, we consider a single source domain. The extension of DASH-N to multiple source domains is straight forward (see the supplementary).

In each layer of DASH-N, we learn a *joint* latent sparse representation. This can be formulated as the minimization of the following cost function w.r.t. $(\mathbf{P}_S, \mathbf{P}_T, \mathbf{D}, \mathbf{X}_S, \mathbf{X}_T)$:

$$\mathcal{L}(\mathbf{Y}_S, \mathbf{P}_S, \mathbf{D}, \mathbf{X}_S, \alpha, \beta) + \lambda \mathcal{L}(\mathbf{Y}_T, \mathbf{P}_T, \mathbf{D}, \mathbf{X}_T, \alpha, \beta) \tag{5.3}$$

$$\text{s.t.} \quad \mathbf{P}_S \mathbf{P}_S^T = \mathbf{P}_T \mathbf{P}_T^T = \mathbf{I}, \ \|\mathbf{d}_i\|_2 = 1, \ \forall i \in [1, K] \tag{5.4}$$

where (α, β, λ) are the non-negative constants and $\mathbf{D} \in \mathbb{R}^{p \times K}$ is the common dictionary. $\mathbf{Y}_s \in \mathbb{R}^{d_S \times n_S}$ and $\mathbf{Y}_T \in \mathbb{R}^{d_T \times n_T}$ are the input data at each layer, $\mathbf{P}_S \in \mathbb{R}^{p \times d_S}$ and $\mathbf{P}_T \in \mathbb{R}^{p \times d_T}$ are the transformations to the latent domain, $\mathbf{X}_S \in \mathbb{R}^{K \times n_S}$ and $\mathbf{X}_S \in \mathbb{R}^{K \times n_S}$ are the sparse codes of the source and the target, respectively. In this formulation, two domains are forced to share a common dictionary in the latent domain. Together with sparsity constraint, the common $\mathbf{D}$ provides a coupling effect that promotes the discovery of common structure between the two domains.

In our experiments, we learn a network of two layers as we found empirically that

adding a third layer did not provide significant improvements over the first two layers.

Layer 1: We perform dense sampling on each training image to get a set of overlapping patches of size 5×5 pixels. These patches are then contrast-normalized as explained in layer 2. In order to make the computation more efficient, only a random subset of patches from each image is used for learning the latent sparse representation. We found that setting this number to 150 for images of maximum size of 150×150 provides a good trade-off between accuracy and computational efficiency. After learning the dictionary $\mathbf{D}_1$ and the transformations $(\mathbf{P}_S^1, \mathbf{P}_T^1)$, the sparse codes $(\mathbf{X}_S^1, \mathbf{X}_T^1)$ are computed for all sampled patches by solving:

$$\min_{\mathbf{X}_*^1} \|\mathbf{P}_*^1 \mathbf{Y}_*^1 - \mathbf{D}_1 \mathbf{X}_*^1\|_2^2 + \beta_1 \|\mathbf{X}_*^1\|_1, \tag{5.5}$$

where $*$ indicates that it can be either source or target. Each column of $\mathbf{Y}_*^1$ is the vectorized pixel values inside a patch. A fast implementation of the LARS algorithm is used for solving this optimization problem [152].

Spatial max pooling is used to aggregate the sparse codes over each 4×4 neighborhood as this pooling method is particularly well-suited for the separation of sparse features [153].

Layer 2: In this layer, we perform similar computations except that the input is the sparse codes from layer 1 instead of image pixels. The features obtained from the previous layer are aggregated by concatenation over each 4×4 neighborhood and contrast-normalized. This results in a new representation that is more robust to occlusion and illumination. If $\mathbf{f}$ is an aggregated feature vector, the contrast-normalization can be per-

formed as in [147]

$$\hat{\mathbf{f}} = \frac{\mathbf{f}}{\sqrt{\|\mathbf{f}\|^2 + \epsilon}}, \tag{5.6}$$

where $\epsilon = 0.1$ is found to work well in our experiments. Similar to layer 1, we also randomly sample 150 normalized feature vectors $\hat{\mathbf{f}}$ from each image for training. ℓ_1 optimization is again employed to compute the sparse codes of the normalized features $\hat{\mathbf{f}}$.

At the end of layer 2, the sparse codes are then aggregated using max pooling in a multi-level patch decomposition (spatial pyramid max pooling). At level 0 of the spatial pyramid, a single feature vector is obtained by performing max pooling over the whole image. At level 1, the image is divided into four quadrants and max pooling is applied to each quadrant, yielding 4 feature vectors. Similarly for level 2, we obtain 9 feature vectors, and so on. In this chapter, max pooling using a three level spatial pyramid is used. As a result, the final feature vector returned by the second layer for each image is a result of concatenating 14 feature vectors from the spatial pyramid.

5.4 Optimization Procedure

In this section, we elaborate on the details of how the cost function in (5.3) is minimized. First, let us define

$$\boldsymbol{\mathcal{K}}_S = \mathbf{Y}_S^T \mathbf{Y}_S,\ \boldsymbol{\mathcal{K}}_T = \mathbf{Y}_T^T \mathbf{Y}_T,\ \boldsymbol{\mathcal{K}} = \begin{pmatrix} \boldsymbol{\mathcal{K}}_S & \mathbf{0} \\ \mathbf{0} & \sqrt{\lambda}\boldsymbol{\mathcal{K}}_T \end{pmatrix} \tag{5.7}$$

to be the Gram matrix of source, target, and their block diagonal concatenation, respectively. It can be shown that (see the supplementary) the optimal solution of (5.3) takes the

form:

$$\mathbf{P}_S = (\mathbf{Y}_S\mathbf{A}_S)^T, \;\; \mathbf{P}_T = (\mathbf{Y}_T\mathbf{A}_T)^T \tag{5.8}$$

$$\mathbf{D} = [\mathbf{A}_S^T\boldsymbol{\mathcal{K}}_S, \; \sqrt{\lambda}\mathbf{A}_T^T\boldsymbol{\mathcal{K}}_T]\mathbf{B} \tag{5.9}$$

for some $\mathbf{A}_S \in \mathbb{R}^{n_S \times p}$, $\mathbf{A}_T \in \mathbb{R}^{n_T \times p}$ and $\mathbf{B} \in \mathbb{R}^{(n_S+n_T)\times K}$. Notice that rows of each transformation live in the column subspace of the data from its own domain. In contrast, columns of the dictionary are jointly created by the data of both source and target.

Solving for $(\mathbf{A}_S, \mathbf{A}_T)$: The orthogonal constraint in (5.4) can be re-written using (5.8) as:

$$\mathbf{A}_S^T\boldsymbol{\mathcal{K}}_S\mathbf{A}_S = \mathbf{I}, \;\; \mathbf{A}_T^T\boldsymbol{\mathcal{K}}_T\mathbf{A}_T = \mathbf{I}. \tag{5.10}$$

By substituting (5.8), (5.9) into (5.3) and making use of the orthogonal constraint in (5.10), the formulation can be simplified to (see derivation in the supplementary):

$$\min_{\mathbf{G}} \mathbf{tr}(\mathbf{G}^T\mathbf{H}\mathbf{G}) \quad \text{s.t.} \quad \mathbf{G}_S^T\mathbf{G}_S = \mathbf{G}_T^T\mathbf{G}_T^T = \mathbf{I}. \tag{5.11}$$

where $\mathbf{H}$ is defined as:

$$\mathbf{H} = \boldsymbol{\Lambda}^{\frac{1}{2}}\mathbf{V}^T\boldsymbol{\mathcal{K}}((\mathbf{I}-\mathbf{B}\mathbf{X})(\mathbf{I}-\mathbf{B}\mathbf{X})^T - \alpha\mathbf{I})\boldsymbol{\mathcal{K}}\mathbf{V}\boldsymbol{\Lambda}^{\frac{1}{2}} \tag{5.12}$$

$$\mathbf{V} = \begin{pmatrix} \mathbf{V}_S & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_T \end{pmatrix}, \; \boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_S & \mathbf{0} \\ \mathbf{0} & \sqrt{\lambda}\boldsymbol{\Lambda}_T \end{pmatrix}, \tag{5.13}$$

$$\boldsymbol{\mathcal{K}}_S = \mathbf{V}_S\boldsymbol{\Lambda}_S\mathbf{V}_S^T, \; \boldsymbol{\mathcal{K}}_T = \mathbf{V}_T\boldsymbol{\Lambda}_T\mathbf{V}_T^T, \tag{5.14}$$

where (5.14) is given by the eigen-decompositions of Gram matrices. Finally, $\mathbf{G}$ is de-

fined as:

$$\mathbf{G} = [\mathbf{G}_S,\ \sqrt{\lambda}\mathbf{G}_T], \tag{5.15}$$

$$\mathbf{G}_S = \mathbf{\Lambda}_S^{\frac{1}{2}}\mathbf{V}_S^T\mathbf{A}_S,\ \mathbf{G}_T = \mathbf{\Lambda}_T^{\frac{1}{2}}\mathbf{V}_T^T\mathbf{A}_T. \tag{5.16}$$

The optimization in (5.11) is non-convex due to the orthogonality constraints. However, $\mathbf{G}$ can be learned efficiently using the algorithm proposed by [154]. Given $\mathbf{G}$, the solution of $(\mathbf{A}_S, \mathbf{A}_T)$ is simply given by

$$\mathbf{A}_S = \mathbf{V}_S\mathbf{\Lambda}_S^{-\frac{1}{2}}\mathbf{G}_S,\ \mathbf{A}_T = \mathbf{V}_T\mathbf{\Lambda}_T^{-\frac{1}{2}}\mathbf{G}_T. \tag{5.17}$$

We note that the optimization step involves the eigen-decompositions of large Gram matrices whose dimensions equal to the number of training samples ($\approx 10^5$ in our experiments). This is computationally infeasible. We propose a remedy for this. The source is taken for the illustration purpose and the computation for the target is similar. First, we compute the eigen-decomposition of the matrix:

$$\mathbf{C}_S = \mathbf{Y}_S\mathbf{Y}_S^T = \mathbf{U}_S\mathbf{\Lambda}'_S\mathbf{U}_S^T \in \mathbb{R}^{d_S \times d_S} \tag{5.18}$$

then the d_S dominant eigenvectors of $\boldsymbol{\mathcal{K}}_S$ can be recovered as in (5.19). The relationship in (5.19) between $\mathbf{V}_S$ and $\mathbf{U}_S$ can be easily verified using an SVD-decomposition of $\mathbf{Y}_S$.

$$\mathbf{V}_S = \mathbf{Y}_S^T\mathbf{U}_S{\mathbf{\Lambda}'_S}^{-\frac{1}{2}}. \tag{5.19}$$

The signal dimension d_S is much smaller than the number of training samples n_S in our experiments (e.g. 10^3 versus 10^5). The eigen-decomposition of $\mathbf{C}_S$ is therefore much more efficient than that of $\boldsymbol{\mathcal{K}}_S$. Finally, non-zero eigenvalues in $\boldsymbol{\Lambda_S}$ are given by the diagonal coefficients of $\mathbf{\Lambda}'_S$.

Solving for $(\mathbf{B}, \mathbf{X})$: If we fix $(\mathbf{A}_S, \mathbf{A}_T)$, then learning $(\mathbf{B}, \mathbf{X}_S, \mathbf{X}_T)$ can be done using any dictionary learning algorithms. In order to see this, let us define:

$$\mathbf{Z} = [\mathbf{A}_S \boldsymbol{\mathcal{K}}_S,\ \sqrt{\lambda}\mathbf{A}_T \boldsymbol{\mathcal{K}}_T], \tag{5.20}$$

$$\mathbf{X} = [\mathbf{X}_S,\ \sqrt{\lambda}\mathbf{X}_T]. \tag{5.21}$$

The cost function can be re-written in a familiar form

$$\|\mathbf{Z} - \mathbf{D}\mathbf{X}\|_F^2 + \beta(\|\mathbf{X}_S\|_1 + \lambda\|\mathbf{X}_T\|_1). \tag{5.22}$$

We use the Lasso to solve for the sparse codes $\mathbf{X}$ and the efficient online dictionary learning algorithm [152] to solve for $\mathbf{D}$. The solution of $\mathbf{B}$ can be recovered, using the relationship in (5.9), simply by $\mathbf{B} = \mathbf{Z}^{\dagger}\mathbf{D}$, where $\dagger$ denotes the Moore-Penrose pseudo-inverse.

5.5 Experiments

The proposed algorithm is evaluated in the context of object recognition using a recent domain adaptation dataset [126], containing 31 classes, with the addition of images from the Caltech-256 dataset [155]. There are 10 common classes between the two datasets (BACKPACK, TOURING-BIKE, CALCULATOR, COMPUTER-KEYBOARD, HEADPHONES, LAPTOP-101, COMPUTER-MONITOR, COMPUTER-MOUSE, COFFEE-MUG, and VIDEO-PROJECTOR) which contain a total of 2533 images. Domain shifts are caused by variations in factors such as pose, lighting, resolution, etc., between images in different domains. Figure 5.2 shows example images from the LAPTOP-101 class with respect to different domains. We compare our method with state-of-the-art adaptation al-

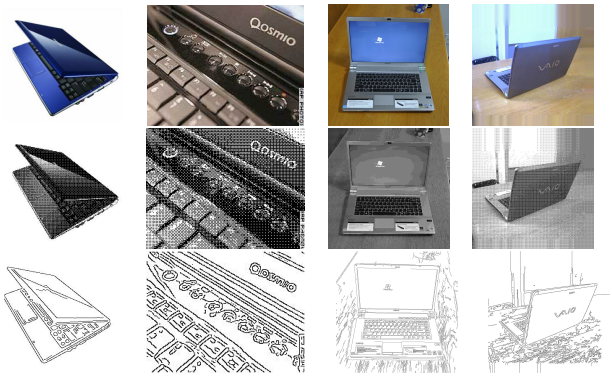


Figure 5.2: Example images from the LAPTOP-101 class in different domains: (a) Amazon, (b) Caltech, (c) DSLR, (d) Webcam. First row: original images, second row: halftone images, third row: edge images.

gorithms such as [126, 128, 129, 139]. In order to better assess the ability to adapt to a wide range of domains, experimental results are also reported on new images obtained by performing halftoning [156] and edge detection [157] algorithms on images from the datasets in [126, 155].

5.5.1 Experiment Setup

We follow the experimental set-ups of [129]. The results using 10 as well as 31 common classes are reported. In both cases, experiments are performed in 20 random trials for each pair of source and target domains. If the source domain is Amazon or Caltech, 20 samples are used in the training. Otherwise, only 8 training samples are used for DLSR and Webcam. The number of target training samples is always set to 3. The remaining images from the target domain in each split are used for testing.

Table 5.1: Recognition rates of different approaches on four domains (C: Caltech, A: Amazon, D: DSLR, W: Webcam). 10 common classes are used. Red color denotes the best recognition rates. Blue color denotes the second best recognition rates.

Method	C → A	C → D	A → C	A → W	W → C	W → A	D → A	D → W
Metric [126]	33.7 ± 0.8	35.0 ± 1.1	27.3 ± 0.7	36.0 ± 1.0	21.7 ± 0.5	32.3 ± 0.8	30.3 ± 0.8	55.6 ± 0.7
SGF [128]	40.2 ± 0.7	36.6 ± 0.8	37.7 ± 0.5	37.9 ± 0.7	29.2 ± 0.7	38.2 ± 0.6	39.2 ± 0.7	69.5 ± 0.9
GFK (PLS+PCA) [129]	46.1 ± 0.6	55.0 ± 0.9	39.6 ± 0.4	56.9 ± 1.0	32.8 ± 0.1	46.2 ± 0.6	46.2 ± 0.6	$\mathbf{80.2 \pm 0.4}$
SDDL [139]	49.5 ± 2.6	76.7 ± 3.9	27.4 ± 2.4	72.0 ± 4.8	29.7 ± 1.9	49.4 ± 2.1	48.9 ± 3.8	72.6 ± 2.1
DASH-N (1st layer)	60.3 ± 2.7	79.6 ± 3.1	52.2 ± 2.1	74.1 ± 4.6	45.31 ± 3.7	68.7 ± 2.9	65.9 ± 2.1	76.3 ± 2.3
DASH-N (1st+2nd layers)	$\mathbf{71.6 \pm 2.2}$	$\mathbf{81.4 \pm 3.5}$	$\mathbf{54.9 \pm 1.8}$	$\mathbf{75.5 \pm 4.2}$	$\mathbf{50.2 \pm 3.3}$	$\mathbf{70.4 \pm 3.2}$	$\mathbf{68.9 \pm 2.9}$	77.1 ± 2.8

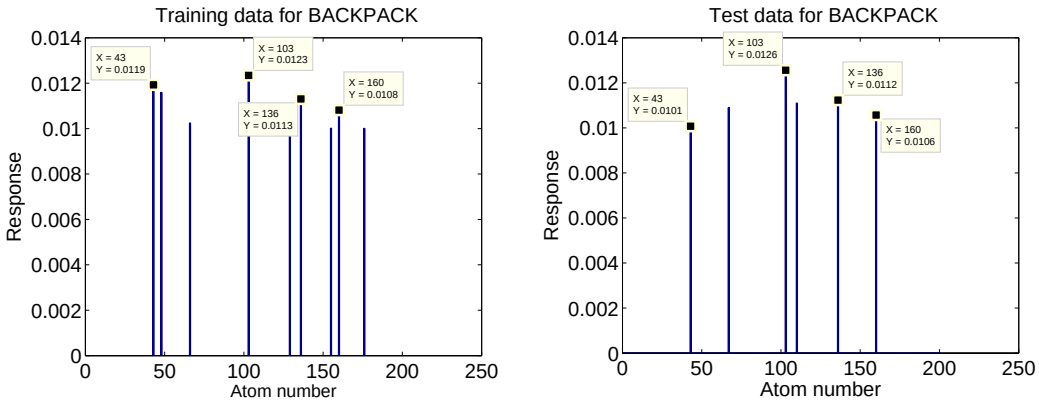


Figure 5.3: Dictionary responses of training (left) and testing (right) data for the BACKPACK class for the pair DSLR-Webcam domains in the first layer.

Parameter Settings: In our experiments, all images are resized to be no larger than 150×150 with preserved ratio and converted to grayscale. The patch size is set to 5×5. Parameter λ is set to 4 in order to account for less training samples from the target than that from the source, and α is set to 1.5 for all experiments. We also found that using

Table 5.2: Single-source recognition rates on all 31 classes.

Method	A $\rightarrow$ W	D $\rightarrow$ W	W $\rightarrow$ D
Metric [126]	44	31	27
RDALR [130]	50.7 ± 0.8	36.9 ± 19.9	32.9 ± 1.2
SGF [128]	57 ± 3.5	36 ± 1.1	37 ± 2.3
GFK (PLS+PCA) [129]	46.4 ± 0.5	61.3 ± 0.4	66.3 ± 0.4
SDDL [139]	50.1 ± 2.5	51.2 ± 2.1	50.6 ± 2.6
DASH-N (1st layer)	59.9 ± 2.7	65.8 ± 1.3	69.6 ± 2.1
DASH-N (1st+2nd layers)	$\mathbf{60.6 \pm 3.5}$	$\mathbf{67.9 \pm 1.1}$	$\mathbf{71.1 \pm 1.7}$

$\beta_{train} = 0.3$ for training and $\beta_{test} = 0.15$ for testing yields best performance. Smaller sparsity constant often makes the decoding more stable, thus, leads to more consistent sparse codes. This is similar to the finding in [146]. The number of dictionary atoms is set to 200 and 1500 in the first and second layer, respectively. The dimension of the latent domain is set to 20 and 750 in the first and second layer, respectively. It is worth noting that the input feature to layer 2 has the dimension of 3200. This results from aggregating sparse codes obtained from the first layer over a 4×4 spatial cell ($4 \times 4 \times 200$). By projecting them into a latent domain of dimension of 750, the computations become more tractable. A three level spatial pyramid, partitioned into 1×1, 2×2, and 3×3, is used to perform the max pooling in the final layer. Linear SVM [158] with the regularization parameter of 10 is employed for classification.

Computation Time: It takes an average of 35 minutes to perform the dictionary learning and feature extraction of all training samples using our Matlab implementation

Table 5.3: Multiple-source recognition rates on all 31 classes

Method	{D, A} → W	{A, W} → D	{W, D} → A
A-SVM [159]	30.4 ± 0.6	25.3 ± 1.1	17.3 ± 0.9
RDALR [130]	36.9 ± 1.1	31.2 ± 1.3	20.9 ± 0.9
SGF [128]	52 ± 2.5	39 ± 1.1	28 ± 0.8
FDDL [160]	41.0 ± 2.4	38.4 ± 3.4	19.30 ± 1.2
SDDL [139]	57.8 ± 2.4	56.7 ± 2.3	24.1 ± 1.6
DASH-N (1st layer)	61.7 ± 2.5	64.1 ± 3.5	39.6 ± 1.3
DASH-N (1st+2nd layers)	$\mathbf{64.5 \pm 2.3}$	$\mathbf{68.6 \pm 3.7}$	$\mathbf{41.8 \pm 1.1}$

on a computer with a 3.8 GHz Intel i7 processor. It takes less than 2 seconds to compute the feature for a test image of size 150×150 using both layers of the hierarchy.

5.5.2 Object Recognition

10 Common Classes: The recognition results of different algorithms on 8 pairs of source-target domains are shown in Table 5.1. It can be seen that DASH-N outperforms all compared methods in 7 out of 8 pairs of source-target domains. For pairs such as Caltech-Amazon, Webcam-Amazon, or DSLR-Amazon, we achieve more than 20% improvements over the next best algorithm used in the comparison (from 49.5% to 71.6%, 49.4% to 70.4%, and 48.9% to 68.9%, respectively). It is worth noting that while we employ a generative approach for learning the feature, our method consistently achieves better performance than [139], even that this method uses discriminative training together with non-linear kernels. It is also clear from the table that the multi-layer DASH-N out-

performs the single-layer DASH-N. In the case of adapting from Caltech to Amazon, the performance gain by using a combination of features obtained from both layers rather than just features from the first layer is more than 10% (from 60.3% to 71.6%) .

In order to illustrate the encoding of features using the learned dictionary in the first layer, Figure 5.3 shows the responses of the training and testing data for the BACKPACK class with respect to each atom of the dictionary in the first layer for the pair DSLR-Webcam domains. The sparse codes for all the patches of the training and testing images belong to the class are computed. The absolutes of these sparse vectors are summed together and normalized to unit length. Small components of the normalized sparse codes are thresholded to better show the correspondences between the training and testing data. It can be seen from the figure that the sparse codes for the training and testing data for the BACKPACK class both have high responses in four different dictionary atoms (43, 103, 136 and 160).

31 Classes and Multiple Sources: We also compare the recognition results for all 31 classes between our approach and other methods in both cases of single (Table 5.2) and multiple source domains (Table 5.3). It can be seen from Tables 5.2 and 5.3 that our results, even using only features extracted from the first layer, are consistently better than that of other algorithms in all the domain settings. This proves the effectiveness of the feature learning process. The performance of our algorithm further increases when combining features learned from both layers of the hierarchy. Especially in the case of adapting from Webcam and DSLR to Amazon, we achieve an improvement of more than 15% compared to the result of SDDL [139] (from 24.1% to 41.8%).

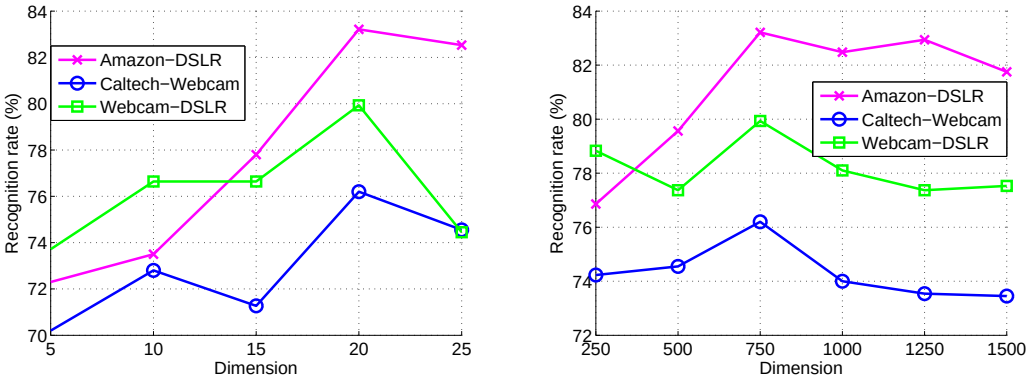
Dimensions of Latent Domains: These are important parameters affecting the per-

formance of DASH-N. Figure 5.4(a) shows the recognition rates with respect to different dimensions of the latent domain in the first layer for three pairs of source-target domains (Amazon-DSLR, Caltech-Amazon and Webcam-DSLR), while keeping the dimension of latent domain in the second layer to 750. As the patch size is set at 5×5, we vary the dimension of the first layer dictionary from 5 to 25. It can be seen from the figure that if the latent domain dimension is too low, the accuracy decreases. The optimal dimension is achieved at 20.

Similarly, the recognition rates with respect to different dimensions of the second layer latent domain are shown in Figure 5.4(b) while the first layer latent dimension is kept at 20. It can be seen from Figure 5.4(b) that the curves for all three pairs of source-target domains peak at the dimension 750. Once again, we observe that the performance decreases if the dimension of the latent domain is too low. More interestingly, as we can observe for the pair Caltech-Webcam and Webcam-DSLR, setting the dimension of the latent domain too high is as detrimental as setting it too low. In all of our experiments, we set the dimension of the latent domain using the cross validation technique.

5.5.3 Half-toning and Edge

In order to evaluate the ability of DASH-N in adapting to a wide range of domains, we also perform experiments on object recognition from the original image domain to two new domains generated by applying half-toning and edge extraction algorithms to the original images. Half-toning images, which imitate the effect of jet-printing technology in the past, are generated using the dithering algorithm in [156]. Edge images are obtained



(a) First layer

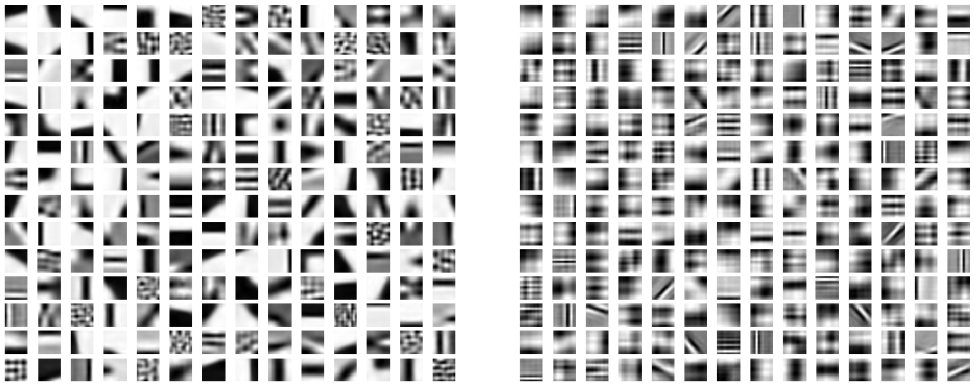
(b) Second layer

Figure 5.4: Recognition rates with respect to different dimensions of the latent domain in the first and second layer.

by applying the Canny edge detector [157] with the threshold set to 0.07.

Figure 5.5 is the visualization of the reconstructed dictionaries atoms at layer 1 when adapting the original images (source) to edge images (target). Reconstructed dictionaries are obtained by $\hat{\mathbf{D}}_*^1 = (\mathbf{P}_*^1)^\dagger \mathbf{D}_1$, where $\dagger$ denotes the Moore-Penrose pseudo-inverse. We observe that the dictionary atoms of original images contain rather fat and smooth regions. In contrast, dictionary atoms of edge images have many thin and highly varying patterns that are more suitable for capturing edges.

Table 5.4 shows the performance of different algorithms when adapting to these new domains. It can be seen from the table that DASH-N outperforms other methods used in the comparison in both cases of half-toning and edge images. This proves the ability of our approach to adapt well to new domains.



(a) Source: Amazon original images (b) Target: Amazon edge images

Figure 5.5: The reconstructed dictionaries at layer 1.

5.6 Conclusion

We have presented a hierarchical method for performing domain adaptation using multi-layer representations of images. In the proposed approach, the features and domain shifts are learned jointly in each layer of the hierarchy in order to obtain a better representation of data from different domains. Unlike the other hierarchical approaches, our method prevents the dimension of feature vectors from increasing too fast as the number of layers increase. Experimental results show that the proposed approach significantly outperforms the other domain adaptation algorithms used in the comparison. In the future, we plan to incorporate non-linear learning frameworks to DASH-N.

Table 5.4: Recognition rates of different approaches on the half-toning and edge datasets. 10 common classes are used.

(a) Half-toning

Method	C → A	C → D	A → C	A → W	W → C	W → A	D → A	D → W
kNN	50.1 ± 5.1	41.9 ± 5.5	29.8 ± 4.3	42.9 ± 2.8	28.9 ± 2.5	48.3 ± 2.1	48.6 ± 1.1	41.4 ± 4.1
Metric [126]	41.1 ± 6.4	38.8 ± 5.6	31.9 ± 5.4	49.4 ± 3.4	32.8 ± 3.2	49.9 ± 3.3	43.8 ± 2.6	49.3 ± 2.6
SDDL [139]	52.2 ± 3.9	66.7 ± 5.5	34.1 ± 3.5	69.2 ± 4.2	34.6 ± 2.8	51.2 ± 3.4	54.1 ± 2.7	71.6 ± 5.3
DASH-N (1st layer)	68.1 ± 3.1	76.6 ± 3.5	49.7 ± 2.8	85.4 ± 3.1	42.1 ± 3.7	62.7 ± 3.2	66.4 ± 2.1	79.3 ± 2.5
DASH-N (1st+2nd layer)	$\mathbf{70.2 \pm 2.7}$	$\mathbf{79.6 \pm 4.3}$	$\mathbf{52.4 \pm 2.3}$	$\mathbf{86.2 \pm 4.1}$	$\mathbf{43.3 \pm 3.9}$	$\mathbf{66.1 \pm 3.7}$	$\mathbf{67.2 \pm 3.5}$	$\mathbf{80.7 \pm 2.1}$

(b) Edge

Method	C → A	C → D	A → C	A → W	W → C	W → A	D → A	D → W
kNN	50.8 ± 1.8	50.4 ± 1.4	32.8 ± 2.9	47.5 ± 4.2	30.4 ± 3.3	51.9 ± 3.1	48.9 ± 1.8	50.2 ± 2.1
Metric [126]	42.8 ± 2.7	43.8 ± 2.5	35.2 ± 2.1	53.6 ± 1.4	36.8 ± 1.8	53.2 ± 3.1	40.8 ± 3.9	54.5 ± 2.7
SDDL [139]	52.9 ± 5.2	63.8 ± 6.3	32.4 ± 3.2	62.5 ± 5.7	33.5 ± 2.9	55.2 ± 2.8	55.4 ± 3.3	65.3 ± 4.7
DASH-N (1st layer)	68.3 ± 4.2	72.9 ± 3.3	33.1 ± 2.1	66.2 ± 4.2	42.6 ± 3.4	59.9 ± 2.7	62.7 ± 2.3	61.5 ± 2.6
DASH-N (1st+2nd layer)	$\mathbf{74.2 \pm 3.9}$	$\mathbf{75.7 \pm 2.6}$	$\mathbf{44.3 \pm 2.5}$	$\mathbf{74.2 \pm 4.5}$	$\mathbf{46.7 \pm 3.1}$	$\mathbf{68.9 \pm 2.2}$	$\mathbf{67.7 \pm 3.4}$	$\mathbf{74.5 \pm 3.2}$

Chapter 6

Future Works

This chapter will discuss several potential directions for the future research. They includes learning invariant representation, learning transformation models to relate images, and learning useful representation of data from compressed observation.

6.1 Invariant Representations

There are various sources that cause the variations of a visual scene such as luminance, shape, pose, etc. While these variations are important to some applications like shape from shading, they are detrimental to many other important applications such as object recognition and detections. Good representations should be sensitive to desirable variations while being invariant to other types of variations.

Recall that we have proposed novel approaches such as kernel KSVD and sparse embedding for learning sparse and non-linear representations directly from the data. Effectively, these methods approximate the manifold of possible scenes by using a set of subspaces in a non-linear feature domain. Careful analyses and designs of sparse learning algorithms might make it possible to capture desirable variations while being insensitive to other effects. Future research work will focus on extending our algorithms to learn invariant representations directly from the data.

6.2 Learning Transformation Models

The human visual system is excellent not only in recognizing objects but also in predicting their possible movements and deformations. Human can even make accurate predictions on an entirely novel object that are not presented to them beforehand. This suggests that human brains might learn a set of universal transformations which can be applied to predict motions and deformations of a data. For example, generic transforamtions such as translation, rotation and scaling are likely to be embedded somewhere in our visual system. This observation motivates the development of new methods for learning transformation models to relate images.

An example in this research direction is the work by Memisevic *et al.* [161] who proposed the relational autoencoder model for learning transformations from pairs of images. The work of Wang *et al.* [162] follows a similar spirit, however, requires the transformation to have a Lie group structure. An exciting research direction would be investigating what role sparsity plays in learning these transformation model.

6.3 Representations On Compressed Domain

Random projection has been proven to be efficient for data collection and dimensionality reduction. It has many desirable properties. First, if the dimension of the mapping is significantly large, which is in the order of $Klog(N)$ where K is the sparsity level and N is the original signal dimension, then we can recover the original signals from the compressed one with an overwhelming probability. In addition the well-known Johnson-Lindenstrauss lemma implies that geometric structures of data are well-preserved under

this transformation.

While much works have been focused on signal reconstruction with random projection, there is little effort in analysing the performances of and designing algorithms for computer vision tasks such as object detection and recognition in a compressed domain associated with the random projection mapping. Lying at the heart of this problem is the issue of finding representations of data, such as videos and images, in the compressed domain. Our future works will address this question. This will enable many computer vision tasks such as object recognition and detection to be performed directly on the compressed domain.

Our first step toward solving this problem is to examine the effects of different random and deterministic measurement matrices on the performances of object recognition for USPS and Caltech-101 datasets. In this experiment, we will use the compressed measurements of entire images as the primary features. Based on the experimental insights, we would develop algorithms for extracting more compact and discriminative features from the compressed measurements for recognition and detection tasks.

Bibliography

[1] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:509–522, 2001.

[2] Haibin Ling and D.W. Jacobs. Shape classification using the inner-distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(2):286 –299, feb. 2007.

[3] H.V. Nguyen and F. Porikli. Support vector shape: A classifier-based shape representation. *IEEE PAMI (Under Review)*.

[4] A. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE PAMI*, 21, 1999.

[5] Marcel Kortgen, Marcin Novotni, and Reinhard Klein. 3d shape matching with 3d shape contexts. In *In The 7th Central European Seminar on Computer Graphics*, 2003.

[6] D. Saupe and D. Vranic. 3D model retrieval with spherical harmonics and moments. In *DAGM*, 2001.

[7] Hien Van Nguyen and F. Porikli. Concentric ring signature descriptor for 3d objects. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 2893 –2896, sept. 2011.

[8] K. Engan, S. O. Aase, and J. H. Husoy. Method of optimal directions for frame design. *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 5:2443–2446, 1999.

[9] M. Aharon, M. Elad, and A. M. Bruckstein. The k-svd: an algorithm for designing of overcomplete dictionaries for sparse representation. *IEEE Trans. Signal Process.*, 54(11):4311–4322, 2006.

[10] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE CVPR*, volume 2, pages 2169 – 2178, 2006.

[11] O. Tuzel, F. M. Porikli, and P. Meer. Region covariance: A fast descriptor for detection and classification. In *ECCV*, pages II: 589–600, 2006.

[12] Zhuolin Jiang, Zhe Lin, and L.S. Davis. Learning a discriminative dictionary for sparse coding via label consistent K-SVD. In *CVPR*, pages 1697 –1704, june 2011.

[13] Qiang Zhang and Baoxin Li. Discriminative K-SVD for dictionary learning in face recognition. In *IEEE CVPR*, pages 2691 –2698, june 2010.

[14] Bernhard Schlkopf, Alexander J. Smola, and Klaus R. Müller. Kernel principal component analysis. *Advances in kernel methods: support vector learning*, pages 327–352, 1999.

[15] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, and F. Pereira. A Theory of Learning from Different Domains. *Machine Learning*, 79:151–175, 2010.

[16] Charles J Stone. Optimal Global Rates of Convergence for Nonparametric Regression. *The Annals of Statistics*, pages 1040–1053, 1982.

[17] I. Biederman. Human image understanding: Recent research and a theory. *Computer Vision, Graphics, and Image Processing*, 32:29–73, 1985.

[18] B. B. Kimia and K. Siddiqi. Parts of visual form: Computational aspects. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(5):544–544, May 1995.

[19] J. Todd. Visual perception of 3D shape. *TRENDS in Cognitive Sciences*, 2004.

[20] Z. Kourtzi and N. Kanwisher. Cortical regions involved in perceiving object shape. *Journal of Neuroscience*, 20:33103318, 2000.

[21] S. Murray, B. Olshausen, and D. Woods. Processing shape, motion and threedimensional shape-from-motion in the human cortex. *Cerebral Cortex*, 13", 2003.

[22] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. *ACM Transactions on Graphics*, 23(3):905–914, August 2004.

[23] S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[24] R. Malladi, J. A. Sethian, and B. C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(2):158–175, 1995.

[25] D. Cremers, N. A. Sochen, and C. Schnorr. A multiphase dynamic labeling model for variational recognition-driven image segmentation. *International Journal of Computer Vision*, 66(1):67–81, January 2006.

[26] N. Paragios. A level set approach for shape-driven segmentation and tracking of the left ventricle. *Medical Imaging, IEEE Transactions on*, 22(6):773 –776, june 2003.

[27] Victor Adrian Prisacariu and Ian Reid. Nonlinear shape manifolds as shape priors in level set segmentation and tracking. IEEE, 2011.

[28] F. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 11(6):567–585, 1989.

[29] H. Chui and A. Rangarajan. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2-3):114–141, February 2003.

[30] Haili Chui, Anand Rangarajan, Jie Zhang, and Christiana Morison Leonard. Unsupervised learning of an atlas from unlabeled point-sets. *IEEE Trans. Pattern Anal. Mach. Intell*, 26(2):160–172, 2004.

[31] F. Wang, B. C. Vemuri, A. Rangarajan, and S. J. Eisenschenk. Simultaneous nonrigid registration of multiple point sets and atlas construction. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(11):2011–2022, November 2008.

[32] L. Gorelick, M. Galun, E. Sharon, R. Basri, and A. Brandt. Shape representation and classification using the poisson equation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(12):1991–2005, December 2006.

[33] Y. Mei and D. Androutsos. Robust affine invariant shape image retrieval using the ICA Zernike moment shape descriptor. *ICIP*, 2009".

[34] B. Gunsel and M. Tekalp. Shape similarity matching for query by example. *Elsevier Journal of Pattern Recognition*, 31(7), 1998.

[35] S. Abbasi, F. Mokhtarian, and J. Kittler. Curvature scale space image in shape similarity retrieval. *ACM Multimedia System*, 7, 1999.

[36] R. Basri, L. Costa, D. Geiger, and D. Jacobs. Determining the similarity of deformable shapes. *Elsevier Journal on Vision Research*, 38, 1998.

[37] M. D. Levine and K. N. Wu. 3D part segmentation using simulated electrical charge distributions. In *ICPR*, pages I: 14–18, 1996.

[38] Lien and Amato. Approximate convex decomposition of polygons. *CGTA: Computational Geometry: Theory and Applications*, 35, 2006.

[39] Jyh-Ming Lien and Nancy M. Amato. Approximate convex decomposition of polyhedra and its applications. *Computer Aided Geometric Design*, 25(7), 2008.

[40] Hairong Liu, Wenyu Liu, and L.J. Latecki. Convex shape decomposition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 97 –104, june 2010.

[41] G. Heitz, G. Elidan, B. Packer, and D. Koller. Shape-based object localization for classification. *IJCV*, 84(1), 2009.

[42] A. Berg, T. Berg, and J. Malik. Multi-scale object detection by clustering lines. *CVPR*, 2005.

[43] T. Sebastian, P. Klein, and B. Kimia. Recognition of shapes by editing their shock graphs. *PAMI*, 26(5), 2004.

[44] H. Ling and D. Jacobs. Shape classification using the inner-distance. *PAMI*, 29, 2007.

[45] J. Hua, Z. Lai, M. Dong, X. Gu, and H. Qin. Geodesic distance-weighted shape vector image diffusion. *IEEE Transactions on Visualization and Computer Graphics*, 14, 2008.

[46] L. Gorelick, M. Galun, E. Sharon, R. Basri, and A. Brandt. Shape representation and classification using the poisson equation. *CVPR*, 2004.

[47] B. Flach and D. Schlesinger. Modelling composite shapes by gibbs random fields. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2177 –2182, june 2011.

[48] A. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *PAMI*, 21(5), 1999.

[49] S. Belongie, J. Malik, and J. Puzicha. Shape matching object recognition using shape context. *PAMI*, 24(24), 2002.

[50] A. Frome, D. Huber, R. Kolluri, T. Bulow, and J. Malik. Recognizing objects in range data using regional point descriptors. 2004.

[51] S. Biswas, G. Aggarwal, and R. Chellappa. An efficient and robust algorithm for shape indexing and retrieval. *IEEE Transactions on Multimedia*, 12:372385, 2010.

[52] H. Zhang and J. Malik. Learning a discriminative classifier using shape context distances. *CVPR*, 2003.

[53] Susan R. Curtis and Alan V. Oppenheim. Reconstruction of multidimensional signals from zero crossings. *J. Opt. Soc. Am. A*, 4(1):221–231, 1987.

[54] D. Slepian. On bandwidth. *Proceedings of the IEEE*, 64:292–300, 1976.

[55] V. Vapnik. The nature of statistical learning theory. *Springer Verlag, New York*, 1995.

[56] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: efficient and robust 3D object recognition. In *CVPR*, pages 998–1005. IEEE, 2010.

[57] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

[58] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

[59] L. J. Latecki, R. Lakamper, and U. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *CVPR*, pages I: 424–429, 2000.

[60] T. Joachims. Making large–scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184, Cambridge, MA, 1999. MIT Press.

[61] Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung. Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

[62] Michele Cossalter, Rong Yan, and Lu Zheng. Adaptive kernel approximation for large-scale Non-Linear SVM prediction. *International Conference on Machine Learning*, 2011.

[63] R. Gopalan, P. Turaga, and R. Chellappa. Articulation-invariant representation of non-planar shapes. *ECCV*, 2010.

[64] X. Yang, S. Koknar-Tezel, and L. Latecki. Locally constrained diffusion process on locally densified distance spaces with applications to shape retrieval. *CVPR*, 2009.

[65] M. Liu, B. C. Vemuri, S.-I. Amari, and F. Nielsen. Total Bregman divergence and its applications to shape retrieval. In *CVPR*, pages 3463 –3468, 2010.

[66] P. Felzenszwalb and J. Schwartz. Hierarchical matching of deformable shapes. *CVPR*, 2007.

[67] H. Ling and K. Okada. An efficient earth mover's distance algorithm for robust histogram comparison. *PAMI*, 29(5):840 –853, may 2007.

[68] G. McNeill and S. Vijayakumar. Hierarchical procrustes matching for shape retrieval. In *CVPR*, pages 885 – 894, 2006.

[69] A. M. Peter, A. Rangarajan, and J. Ho. Shape L'ane rouge: Sliding wavelets for indexing and retrieval. In *CVPR*, pages 1–8, 2008.

[70] Z. W. Tu and A. L. Yuille. Shape matching and recognition: Using generative models and informative features. In *ECCV*, pages Vol III: 195–209, 2004.

[71] T.B. Sebastian, P.N. Klein, and B.B. Kimia. On aligning curves. *PAMI*, 25(1):116 –125, 2003.

[72] F. Mokhtarian, S. Abbasi, and J. Kittler. Efficient and robust retrieval by shape content through curvature scale space, 1996.

[73] L. Lin, X. Liu, and S.-C. Zhu. Layered graph matching with composite cluster sampling. *PAMI*, 32(8):1426 –1442, 2010.

[74] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[75] A. Mian, M. Bennamoun, and R. Owens. On the repeatability and quality of keypoints for local feature-based 3D object retrieval from cluttered scenes. *International Journal of Computer Vision*, 89(2-3):348–361, 2010.

[76] R. Rubinstein, A. M. Bruckstein, and M. Elad. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, submitted 2009.

[77] J. Wright, Yi Ma, J. Mairal, G. Sapiro, T.S. Huang, and Shuicheng Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031 –1044, june 2010.

[78] M. Elad, M.A.T. Figueiredo, and Yi Ma. On the role of sparse and redundant representations in image processing. *Proceedings of the IEEE*, 98(6):972 –982, june 2010.

[79] Alfred M. Bruckstein, David L. Donoho, and Michael Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1):34 –81, 2010.

[80] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Sci. Comp.*, 20(1):33–61, 1998.

[81] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. *1993 Conference Record of the 27th Asilomar Conference on Signals, Systems and Computers*, pages 40–44, 1993.

[82] J. A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. Info. Theory*, 50(10):2231–2242, Oct. 2004.

[83] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, June 1996.

[84] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 689–696, New York, NY, USA, 2009. ACM.

[85] J. Mairal, F. Bach, and J. Ponce. Task-driven dictionary learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):791 –804, april 2012.

[86] R. Rubinstein, M. Zibulevsky, and M. Elad. Double sparsity: Learning sparse dictionaries for sparse signal approximation. *Signal Processing, IEEE Transactions on*, 58(3):1553 –1564, march 2010.

[87] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[88] R. Courant and D. Hilbert. *Methods of Mathematical Physics*. Interscience, 1953.

[89] Usps handwritten digit database. In *http://www-i6.informatik.rwth-aachen.de/ keysers/usps.html*.

[90] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 31(2):210–227, Feb. 2009.

[91] B. Scholkopf and A. J. Smola. *Learning With Kernels, Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

[92] James Tin-Yau Kwok and Ivor Wai-Hung Tsang. The pre-image problem in kernel methods. *IEEE Transactions on Neural Networks*, 15(6):1517–1525, 2004.

[93] D. Keysers, J. Dahmen, T. Theiner, and H. Ney. Experiments with an extended tangent distance. In *ICPR*, pages Vol II: 38–42, 2000.

[94] Patrice Simard, Yann LeCun, John S. Denker, and Bernard Victorri. Transformation invariance in pattern recognition-tangent distance and tangent propagation. In *Neural Networks: Tricks of the Trade, this book is an outgrowth of a 1996 NIPS workshop*, pages 239–27, London, UK, 1998. Springer-Verlag.

[95] Bernhard Schölkopf, Patrice Simard, Alex J. Smola, and Vladimir Vapnik. Prior knowledge in support vector kernels. The MIT Press, 1997.

[96] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Supervised dictionary learning. *CoRR*, abs/0809.3083, 2008. informal publication.

[97] P. Perona, R. Fergus, and F. F. Li. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *Workshop on Generative Model Based Vision*, page 178, 2004.

[98] Peter Gehler and Sebastian Nowozin. On feature combination for multiclass object classification. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 221 –228, 29 2009-oct. 2 2009.

[99] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Representing shape with a spatial pyramid kernel. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, CIVR '07, pages 401–408, New York, NY, USA, 2007. ACM.

[100] Timo Ojala, Matti Pietikinen, and Topi Menp. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 24(7):971–987, 2002.

[101] Nicolas Pinto, David D. Cox, and James J. Dicarlo. Why is real-world visual object recognition hard. *PLoS Computational Biology*, 2008.

[102] P. Jain, B. Kulis, and K. Grauman. Fast image search for learned metrics. In *IEEE CVPR 2008.*, pages 1 –8, june 2008.

[103] Hao Zhang, A.C. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *IEEE CVPR, 2006.*

[104] G. Grifn, A. Holub, and P. Perona. Caltech-256 object category dataset. *Technical Report*, 2007.

[105] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *IEEE CVPR*, pages 1 –8, june 2008.

[106] Duc-Son Pham and S. Venkatesh. Joint learning and dictionary construction for pattern recognition. In *IEEE CVPR 2008.*, pages 1 –8, june 2008.

[107] J. C. van Gemert, J. M. Geusebroek, C. J. Veenman, and A. W. M. Smeulders. Kernel codebooks for scene categorization. In *ECCV*, pages III: 696–709, 2008.

[108] Jianchao Yang, Kai Yu, Yihong Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *IEEE CVPR*, pages 1794 –1801, june 2009.

[109] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, T. Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *IEEE CVPR*, pages 3360 –3367, june 2010.

[110] N. Kulkarni and Baoxin Li. Discriminative affine sparse codes for image classification. In *IEEE CVPR*, pages 1609 –1616, june 2011.

[111] C. J. Stone. Optimal global rates of convergence for nonparametric regression. *The Annals of Statistics*, 10:1040–1053, 1982.

[112] Beyer, Goldstein, Ramakrishnan, and Shaft. When is "nearest neighbor" meaningful? In *ICDT: 7th International Conference on Database Theory*, 1999.

[113] J. A. Lee and M. Verleysen. Nonlinear dimensionality reduction. In *Information Science and Statistics, Springer, 2006*.

[114] K. Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh and Dublin Phil. Mag. & Jour. of Science.*, 2:559–572, 1901.

[115] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

[116] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, December 2000.

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

[118] M. Elad. Sparse and redundant representations. In *From theory to applications in signal and image processing, Springer-New York*, 2010.

[119] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans. Image Processing*, 15(12):3736–3745, December 2006.

[120] I. Gkioulekas and T. Zickler. Dimensionality reduction using the sparse linear model. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[121] Hanchao Qi and Shannon Hughes. Using the kernel trick in compressive sensing: Accurate signal recovery from fewer measurements. In *IEEE Int. Conf. on Acoustics, Speech and Signal Proc.*, pages 3940 –3943, may 2011.

[122] H. V. Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa. Kernel dictionary learning. *IEEE Int. Conference on Acoustics, Speech and Signal Processing*, 2012.

[123] Julien Mairal, Francis Bach, and Jean Ponce. Task-driven dictionary learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(4):791 –804, april 2012.

[124] K. Engan, S. O. Aase, and J. H. Husoy. Multi-frame compression: Theory and design. *Signal Processing*, 80(10):2121–2140, October 2000.

[125] S. Pan and Q. Yang. A Survey on Transfer Learning. *Journal of Artificial Intelligence Research*, 22(10):1345–1359, 2009.

[126] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting Visual Category Models to New Domains. In *Proc. ECCV*, pages 213–226, 2010.

[127] B. Kulis, K. Saenko, and T. Darrell. What You Saw is Not What You Get: Domain Adaptation using Asymmetric Kernel Transforms. In *Proc. CVPR*, pages 1785–1792, 2011.

[128] R. Gopalan, R. Li, and R. Chellappa. Domain Adaptation for Object Recognition: An Unsupervised Approach. In *Proc. ICCV*, pages 999–1006, 2011.

[129] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic Flow Kernel for Unsupervised Domain Adaptation. In *Proc. CVPR*, pages 2066–2073, 2012.

[130] I.-H. Jhuo, D. Liu, D.T. Lee, and S.-F. Chang. Robust Visual Domain Adaptation with Low-Rank Reconstruction. In *Proc. CVPR*, pages 2168–2175, 2012.

[131] G. Hinton, S. Osindero, and Y. Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neur. Comp.*, 18(7):1527–1554, 2006.

[132] H. Lee, R. Grosse, R. Ranganath, and A. Ng. Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. In *Proc. ICML*, 2009.

[133] D. Lowe. Distinctive Image Features From Scale-Invariant Keypoints. *IJCV*, 60(2):91–110, 2004.

[134] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features. *Comp. Vis. Img. Under.*, 110:346–359, 2008.

[135] C. Leggetter and P. Woodland. Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models. *Computer Speech and Language*, 9(2):171–185, 1995.

[136] H. Daumé III and D. Marcu. Domain Adaptation for Statistical Classifiers. *Jour. of Arti. Intell. Res.*, 26(1):101–126, 2006.

[137] J. Zheng, M.-Y. Liu, R. Chellappa, and J. Phillips. A Grassmann Manifold-Based Domain Adaptation Approach. In *Proc. ICPR*, pages 2095–2099, 2012.

[138] Qiang Qiu, VishalM. Patel, Pavan Turaga, and Rama Chellappa. Domain Adaptive Dictionary Learning. In *Proc. ECCV*, pages 631–645, 2012.

[139] S. Shekhar, V. Patel, H. Nguyen, and R. Chellappa. Generalized Domain-Adaptive Dictionaries. In *Proc. CVPR*, 2013.

[140] L. Duan, I.W. Tsang, D. Xu, and T.-S. Chua. Domain Adaptation from Multiple Sources via Auxiliary Classifiers. In *Proc. ICML*, 2009.

[141] L. Duan, I.W. Tsang, D. Xu, and T.-S. Chua. Domain Transfer Multiple Kernel Learning. *IEEE Trans. PAMI*, 34(3):465–479, 2012.

[142] Jenny Rose Finkel and Christopher D Manning. Hierarchical Bayesian Domain Adaptation. In *Proc. NAACL*, pages 602–610, 2009.

[143] B. Manjunath and R. Chellappa. A Unified Approach to Boundary Perception: Edges, Textures, and Illusory Contours. *IEEE Trans. Neural Networks*, 4(1):96–108, 1993.

[144] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol. Extracting and Composing Robust Features with Denoising Autoencoders. In *Proc. ICML*, 2008.

[145] K. Yu, Y. Lin, and J. Lafferty. Learning Image Representations from the Pixel Level via Hierarchical Sparse Coding. In *Proc. CVPR*, pages 1713–1720, 2013.

[146] L. Bo, X. Ren, and D. Fox. Hierarchical Matching Pursuit for Image Classification: Architecture and Fast Algorithms. In *Proc. NIPS*, pages 2115–2123, 2011.

[147] L. Bo, X. Ren, and D. Fox. Multipath Sparse Coding using Hierarchical Matching Pursuit. In *Proc. CVPR*, 2013.

[148] M. Elad. *Sparse and Redundant Representations*. Springer, 2010.

[149] Julien Mairal, Francis Bach, and Jean Ponce. Task-Driven Dictionary Learning. *IEEE Trans. PAMI*, 34(4):791–804, 2012.

[150] Hien Nguyen, Vishal Patel, Nasser Nasrabadi, and Rama Chellappa. Sparse Embedding: A Framework for Sparsity Promoting Dimensionality Reduction. In *Proc. ECCV*, pages 414–427, 2012.

[151] M. Ruhnke, L. bo, D. Fox, and W. Burgard. Compact RGBD Surface Models Based on Sparse Coding. In *Proc. AAAI*, 2013.

[152] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online Dictionary Learning for Sparse Coding. In *Proc. ICML*, pages 689–696, 2009.

[153] Y-L. Boureau, J. Ponce, and Y. LeCun. A Theoretical Analysis of Feature Pooling in Visual Recognition. In *Proc. ICML*, 2010.

[154] Z. Wen and W. Yin. A Feasible Method for Optimization with Orthogonality Constraints. *Math. Prog.*, pages 1–38, 2013.

[155] G. Griffin, A. Holub, and P. Perona. Caltech-256 Object Category Dataset. Technical Report 7694, Caltech, 2007.

[156] V. Monga, N. Damera-Venkata, H. Rehman, and B. Evans. Halftoning MATLAB Toolbox. `http://users.ece.utexas.edu/~bevans/projects/halftoning/toolbox/`, 2005.

[157] J. Canny. A Computational Approach to Edge Detection. *IEEE Trans. PAMI*, 8(6):679–698, 1986.

[158] V. N. Vapnik. *Statistical Learning Theory*. John Wiley, 1998.

[159] J. Yang, R. Yan, and A. Hauptmann. Cross-Domain Video Concept Detection using Adaptive SVMs. In *Proc. MM*, 2007.

[160] M. Yang, L. Zhang, X. Feng, and D. Zhang. Fisher Discrimination Dictionary Learning for Sparse Representation. In *Proc. CVPR*, pages 543–550, 2011.

[161] Roland Memisevic. Gradient-based learning of higher-order image features. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1591–1598. IEEE, 2011.

[162] Ching Ming Wang, Jascha Shol-Dickstein, Ivana Tosic, and Bruno A Olshausen. Lie group transformation models for predictive video coding. In *Data Compression Conference (DCC), 2011*, pages 83–92. IEEE, 2011.