Learning and Mapping onto Manifolds with Applications to Patch-based Image Processing

by

Yevgen Matviychuk

B.S., Donetsk National Technical University, 2007

M.S., Donetsk National Technical University, 2008

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Electrical, Computer, and Energy Engineering

2016

This thesis entitled:
Learning and Mapping onto Manifolds
with Applications to Patch-based Image Processing
written by Yevgen Matviychuk
has been approved for the Department of Electrical, Computer, and Energy Engineering

Shannon M. Hughes

Youjian Liu

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Matviychuk, Yevgen (Ph.D., Electrical Engineering)

Learning and Mapping onto Manifolds

with Applications to Patch-based Image Processing

Thesis directed by Prof. Shannon M. Hughes

While the field of image processing has been around for some time, new applications across many diverse areas, such as medical imaging, remote sensing, astrophysics, cellular imaging, computer vision, and many others, continue to demand more and more sophisticated image processing techniques. These areas inherently rely on the development of novel methods and algorithms for their success. Many important cases in these applications can be posed as problems of reversing the action of certain linear operators. Recently, patch-based methods for image reconstruction have been shown to work exceptionally well in addressing these inverse problems, establishing new state-of-the-art benchmarks for many of them, and even approaching estimated theoretical limits of performance.

However, there is still space and need for improvement, particularly in highly specialized domains. The purpose of this thesis will be to improve upon these prior patch-based image processing methods by developing a computationally efficient way to model the underlying set of patches as arising from a low-dimensional manifold. In contrast to other works that have attempted to use a manifold model for patches, ours will rely on the machinery of kernel methods to efficiently approximate the solution. This will make our approach much more suitable for practical use than those of our predecessors. We will show experimental results paralleling or exceeding those of modern state-of-the-art image processing algorithms for several inverse problems. Additionally, near the end of the thesis, we will revisit the problem of learning a representation for the manifold from its samples and develop an improved approach for it. In contrast to prior methods for manifold learning, our kernel-based strategy will be robust to issues of learning from very few or noisy samples, and it will readily allow for interpolation along or projection onto the manifold.

Dedication

To my family.

Acknowledgements

I am deeply thankful to my advisor Prof. Shannon Hughes for her unceasing support, guidance, encouragement, and countless hours devoted to growing me as a better scientist. She set an example of excellence for me as a passionate researcher, wise mentor, and inspiring teacher. I am proud to base my future work on the solid foundation she helped me to build.

Also, I would like to express my gratitude to the members of my committee: Prof. Bradley, Prof. Chen, Prof. Corcoran, Prof. Liu, Prof. Mathys, and Prof. Meyer. Their insightful questions and comments helped me to develop a multifaceted view on research problems. I am especially thankful to Prof. Liu for serving as my thesis reader and for his valuable suggestions.

Many thanks to other fellow students at the University of Colorado, with whom I had a chance to work. Our friendships not only broadened my professional horizons but also enriched my world view in general.

Finally, I would like to thank my friends and family. No words can express my deepest gratitude for their love and support.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

1.1 Overview and Motivation

While the field of image processing has been around for some time, new applications arising across many diverse areas, such as medical imaging, remote sensing, astrophysics, cellular biology, computer vision, and many others, are now requiring an increase in image quality beyond what has been achievable with current methods. Therefore, to move forward, all these fields are increasingly calling for the design of new, more effective image processing methodology.

As a first example, we see this need in the field of medical imaging. Since the invention of Magnetic Resonance Imaging (MRI), it has quickly become an indispensable part of medical diagnostic and treatment monitoring. While previous imaging techniques, such as radiography or computed tomography, involved harmful ionizing radiation, MRI is safe for the patient and thus has become the new method of choice for medical diagnostics [95]. Furthermore, MRI is one of the few technologies that allows one to observe dynamic biological processes, e.g. the heart beating with cardiac MRI or real-time brain activity with functional MRI (fMRI) [8, 87], which makes it indispensable for studying cardiac health problems or brain function.

Unfortunately, however, there are physical and physiological limits on the speed of scanning, which have made MRI slow, and caused limited resolution, increased cost, patient discomfort, and potential blurring of the image if/when a patient moves during the procedure. Faster acquisition to avoid these problems has become possible only due to very recent advances in compressive sensing methods in image processing [136]. These techniques apply sophisticated image reconstruction

algorithms, so that only a fraction of the raw signal needs to be acquired during each scan. This eventually saves time and cost, while also helping to increase resolution. In dynamic applications such as fMRI, compressive sensing not only improves temporal resolution almost fourfold but also, rather surprisingly, increases sensitivity of activation detection with specific sampling sequences in low SNR regimes [216]. However, even with these advances, the cost of an average patient MRI in the United States exceeded $2600 in 2014, which is a financial hardship for many. Meanwhile, with conventional fMRI, for example, one still can only acquire only a single image every 2 seconds, which is much too slow for detailed temporal understanding of brain function. Hence, even more sophisticated image processing algorithms will be needed to make MRI cost-effective as a diagnostic tool, and to enable more advances in understanding in the field of neuroscience.

In the study of cellular biology, the ability to understand how cells function on the molecular level is limited by the challenges of studying them in-vivo. Cellular objects of interest, such as membrane structures, protein clusters, or even single molecules are much too small to study with most current imaging technologies. For example, the resolution of conventional microscopy is limited to $200-250$ nm by diffraction of light. Meanwhile, electron scanning microscopes allow finer imaging, but only at the cost of producing static images [51, 82]. Only the recent development of image processing methods for superresolution microscopy [150] made it possible to achieve resolutions of about 10 nm, one order of magnitude below the diffraction limit. This has allowed researchers to finally begin to study the molecular machinery of a cell in-vivo. Similarly, single-molecule and neuronal activity imaging have become possible only in the last 3-5 years, largely due to recent advances in imaging techniques [90, 180]. However, these methods are still limited, specifically by relatively low temporal resolution (e.g. on the order $0.1-0.3$ sec in localization microscopy [145]), which may not be enough to study the dynamics of rapid cellular processes.

On the other extreme of scale, modern applications in astronomy and remote sensing also require higher quality image processing algorithms than exist to date. The main challenges here are to reconstruct and enhance images taken from a satellite, an airplane, or from small remote sensing devices. However, while a high-quality image is needed for study in these applications, the

amount of data that can be transmitted from satellites or remote sensors is often very limited, and computational power onboard may be insufficient as well. For example, the amount of raw data produced by space telescopes, such as the ESA's Hershel, far exceed the available capacity of the downlink communication channel, while severely limited CPU time prevents the use of complicated compression algorithms on the satellite itself. Hence, methods that allow high quality image reconstruction from very little data, e.g. successful compressive sensing algorithms, are needed for this application [19]. As another example, the reconstruction of high quality color images of Jupiter's moon Europa, released by NASA in 2014 after being taken almost 25 years ago, was made possible only recently due to newly created imaging algorithms. Needless to say that the pursuit of new frontiers in space exploration will walk hand in hand with the development of novel more powerful image reconstruction methods.

Moreover, it is important to emphasize that image quality requirements in scientific and medical domains, such as those mentioned above, are generally much higher than in, for example, conventional photography, and they constantly increase. This further highlights the need for better and better algorithms for image reconstruction to keep up with the demands of these applications.

Furthermore, it is worth noting that recent statistical analysis indicates that existing algorithms, for example, for denoising [43, 44, 128], image registration [164, 209], and superresolution [6, 131], have not yet achieved their potential performance bounds, and thus there is still significant room for their improvement. Specifically, despite tremendous breakthroughs seen in the past decades in processing rich photographic images, many problems, even as fundamental as efficient removal of non-Gaussian noise [86] or avoiding introduction of artifacts in low noise regimes [43], remain challenging for most modern algorithms.

Thus, we have seen that there is a need for more powerful image reconstruction and enhancement algorithms across a broad spectrum of important modern scientific applications. This thesis will focus on the creation of more powerful algorithms to aid across a variety of such applications. In the next sections, we will briefly describe our approach to these modern challenges in image processing along with our main results and provide an outline for the rest of the dissertation.

1.2 Plan of Attack: The Manifold View of Image Patches

Most of the scenarios described above, such as denoising, compressive sensing reconstruction, superresolution, etc., can be posed as special instances of the general linear inverse problem; we will discuss its formulation in detail in the next chapter but here will provide a high level overview of the modern acknowledged solutions. Over the previous decades, these problems were targeted with numerous algorithmic approaches ranging from filtering in the image domain to non-linear manipulations in a coefficient domain. However, recently it was found that breaking an image into a collection of its small overlapping pieces – *patches* – and then considering each of them separately, produces exceptionally good results when the image is reassembled. In fact, patch-based methods constitute the core of many modern state-of-the-art algorithms for image reconstruction and will be the main focus of our work.

The successes of the patch-based approach in addressing inverse problems can be explained by the fact that in the mentioned applications the ability to preserve crisp contrast edges as well as details of patterns and textures are often the most desirable qualities of the algorithm. Working with smaller patches, as opposed to an image as a whole, often allows one to efficiently achieve these goals. Hence, patch-based representations allow the resulting methods to be applicable across a broad range of problems in different fields.

In this thesis, we will develop novel patch-based methods for general linear inverse problems in image processing. However, unlike most previous work in this category, we will employ an elegant description of patches with an underlying manifold model. Numerous practical methods so far, often indirectly, have imposed a low-dimensional structure on the set of image patches, and several theoretical studies have shown that it indeed can be well-modeled with a smooth non-linear manifold. A few methods have even attempted to model the patches with a manifold. However, the design of computationally tractable algorithms that make this underlying manifold assumption explicit has remained an open problem to date, which we will address in this thesis.

The purpose of this thesis will thus be to improve upon prior patch-based image processing

methods, by developing a computationally efficient way to model the underlying set of patches as arising from a low-dimensional manifold. In contrast to other work that has attempted using a manifold model for patches, ours will rely on the machinery of kernel methods to efficiently approximate the manifold. This will make our approach much more pragmatic than those of our predecessors. We will show experimental results paralleling or exceeding state-of-the-art image processing methods for several inverse problems.

Additionally, in the final chapter, we will revisit the problem of manifold learning and develop an improved approach for it. In contrast to prior work, our novel kernel-based algorithm will be robust to issues of learning from very few or noisy manifold samples and will readily allow for interpolation along or projection onto the manifold.

1.3 Main Contributions

With a broad range of applications in mind, we will present a novel intersecting manifolds model for images. In this novel model, we will assume that each image patch lies on its own manifold in the image space, which will locate an entire image at the intersection of many such manifolds corresponding to its different overlapping patches. We will see that this intersection seeking strategy is a natural way to formulate the problem of simultaneously constraining all image patches to lie on the underlying manifold model.

Next, to find a computationally efficient solution to the problem of finding the manifolds' intersection, we will develop a non-linear extension of the well-known Projections onto Convex Sets (POCS) algorithm, which is typically used for finding intersections of, for example, affine subspaces. To achieve this, we will have to carefully reformulate the POCS algorithm entirely in terms of inner products and then apply the kernel trick from machine learning to obtain its non-linear extension. To the best of our knowledge, this result has not been reported in the literature before. The model of intersecting manifolds and our efficient closed-form solution can potentially be used in problems beyond image processing, such as, for example, set extrapolation.

Then, we will focus specifically on image processing applications and will design an effective

technique for solving any linear inverse problem with our model of intersecting patch-manifolds. Essentially, our solution is a generalization of a popular kernel-based approximator of the distance to a manifold. Our method favorably compares to several specialized state-of-the-art algorithms often surpassing them in both visual quality and quantitative performance measures. This constitutes our main contribution to the field of image processing.

Finally, we will propose a method for learning and representation of manifolds in kernel-induced feature spaces. We will parameterize the manifold-approximating subspace in a similar way to many successful kernel-based algorithms, but will improve the expressive power of this description and generalize it for manifolds of arbitrary dimensions. We will show the particular suitability of our representation for problems of mapping points onto and interpolating along the manifold. It will also be extremely powerful for learning manifolds from either very few or noisy samples, which have been issues for past manifold learning algorithms.

1.4 Structure of the Thesis

We now briefly outline the structure of the dissertation and the contents of its chapters.

To put our work in context, we will start by reviewing some background information and the existing related literature in Chapter 2. Here we will first formally set up the most general form of the linear inverse problem for images. We will see that many important modern image processing applications mentioned in the beginning of this chapter reduce to this form, so we use it as the main motivational example that will direct our upcoming narrative. We will then proceed by examining its existing solutions. While acknowledging some popular time-proven global models for entire images, we will primarily focus on recently-emerged but surprisingly effective local patch-based methods. We will attempt to analyze the rich spectrum of prior patch-based algorithms ranging from strictly specialized such as [181] to more general and flexible [54] and will conclude by discussing elegant descriptions for the set of image patches with underlying manifold models.

Next, in Chapter 3, we will turn to more technical topics and review the machinery of kernel methods, which will form the leitmotif of the thesis. Specifically, we will focus in detail

on Kernel Principal Component Analysis (KPCA) as one of the most powerful existing manifold learning algorithms. It will become our tool of choice on top of which we will build our solutions. Furthermore, at this point, we will establish the notation used throughout the rest of the work.

The fourth chapter is dedicated to our first piece of original work, a method for efficiently finding an approximate intersection of many manifolds. Borrowing the idea from Kernel PCA of approximating a manifold with an affine subspace in a higher-dimensional feature space, we will consider several such subspaces to model intersecting manifolds. We will then derive a kernel-based non-linear extension of the POCS algorithm and express our solution in closed form for improved efficiency. We will show successful proof-of-concept results on several toy examples and in a problem of set extrapolation, and we will see that this setting is of interest because of its connection to patch-based image processing with overlapping patches.

In Chapter 5, we will approach the problem of finding the manifolds' intersection specifically with image processing applications in mind. Again, based on the kernel PCA representation of manifolds, we will form a functional to approximate the distance to their sought intersection. We will then minimize it with a descent algorithm in order to arrive at a point on this intersection. Unlike our closed-form method from the previous chapter that operates in the feature space, the minimization here will be carried out in the original image space directly. This will eliminate the need for solving the cumbersome error-prone preimage problem often associated with kernel methods and will guarantee the existence of a suitable solution. Furthermore, we will greatly improve upon the efficiency of our method by reusing the same simple patch manifold description learned once for all patch positions in the image. In fact, in addition to small structured texture images as in Chapter 4, we now will be able to effectively process natural photographic images of significantly larger sizes. To conclude, we will show how our practical patch manifolds intersection framework can be applied unchanged to solve any linear inverse problem in image processing. We achieve experimental results similar to or better than the state-of-the-art algorithms on each problem, even though the comparison algorithms are specifically tailored for each individual problem.

Finally, in the sixth chapter we will expose some shortcomings of the traditional kernel PCA

parameterization of the manifold approximating subspace in feature space. We will see that the minimizers of the distance to this subspace form a set of disconnected points in the original space, forcing the minimization procedure to eventually converge to one of them. Potentially this undermines our ability to accurately represent a continuous manifold and to map points onto it. To deal with this problem, especially conspicuous with manifolds of low codimensions, we present our novel kernel-based manifold learning method. It allows one to learn the structure of the manifold from only a few noisy samples and is particularly useful for finding projections onto and interpolating along non-linear manifolds. We will show encouraging results of our method outperforming other popular approaches in these applications as well as in modeling datasets for the purposes of classification and anomaly detection.

Chapter 2

Background and Literature Review

In this chapter we will provide a brief overview of existing image reconstruction methods. We will particularly focus on patch-based algorithms as they form the core of the latest state-of-the-art tools. We further discuss manifold models as an elegant and effective way to impose a structure on the set of image patches. But first, let us start by formally introducing the general problem setting that will motivate our work.

2.1 Inverse Problems in Image Processing

Many problems in image processing are often viewed as special instances of the general problem of reconstructing an unknown signal from its linear measurements. These are referred to as linear inverse problems. In its most general form, a linear inverse problem aims to reconstruct a signal $\mathbf{z}_{true}$ from its observations $\mathbf{b}$ obtained by some linear transformation $\mathbf{W}$ and possibly corrupted with additive noise $\mathbf{n}$:

$$\mathbf{b} = \mathbf{W}\mathbf{z}_{true} + \mathbf{n}. \tag{2.1}$$

Many practical applications, such as denoising and compressive sensing among others, can be expressed in the form of Eq. 2.1. Typically, however, the low column rank of the matrix $\mathbf{W}$ makes the above problem underdetermined with an infinite number of solutions. For example, the number of measurements in compressive sensing is usually assumed to be much smaller than the dimension of the signal space (the number of pixels). In inpainting, on the other hand, the masking

matrix selects only a fraction of the original image pixels and leaves out the others. In denoising, even though $\mathbf{W} = \mathbf{I}$, the identity matrix, additive noise $\mathbf{n}$ still makes the problem ill-posed.

Therefore, in order to restrict the set of possible solutions, additional assumptions on the sought image $\mathbf{z}_{true}$ should be made. They usually take the form of an underlying model, whose choice is guided by some prior knowledge about the sought image. A suitable model promotes desired qualitative characteristics in the reconstructed image, while establishing a quantitative criterion that helps to choose the optimal solution. For example, one may encourage the image to have sparsity in a wavelet basis by minimizing the ℓ^1 norm of the wavelet coefficients, or one may seek low total variation (TV) to sharpen edge transitions of the image, if those qualities are expected in the desired solution. On the other hand, methods based on exploiting similarity between image regions may be suitable if the image $\mathbf{z}_{true}$ contains repetitive patterns and textures. Needless to say that the appropriateness of the assumed model for representing a particular class of signals is a major factor that affects the overall quality of reconstruction. Finally, it directly determines what signal processing algorithms are potentially suitable, which can also be an important consideration in making this choice.

2.2 Image Models and Reconstruction Algorithms

Finding a successful solution to the inverse problem of Eq. 2.1 inherently relies upon the representational suitability of the assumed image model. In this section we will review some of the most common assumptions (such as sparsity of transform domain coefficients and similarity of small image patches) made by popular image processing algorithms to capture the characteristics of natural images. We start by considering the variational interpretation of image processing methods, as it provides a convenient way to design new and extend existing algorithms for addressing a wide range of inverse problems. Later, we will adapt this view for our solution developed in Chapter 5.

2.2.1 Variational Approach to Image Reconstruction

The criteria for successfully inverting the action of the measurement (or degradation) operator $\mathbf{W}$ in Eq. 2.1 are twofold. Foremost, one wants to find an estimate $\widehat{\mathbf{z}}$ of the original image $\mathbf{z}_{\text{true}}$, whose measurements with $\mathbf{W}$ match those given in the vector $\mathbf{b}$. This desire for data fidelity is usually expressed in terms of the Euclidean distance, thus leading to $\min_{\widehat{\mathbf{z}}} \|\mathbf{b} - \mathbf{W}\widehat{\mathbf{z}}\|_2^2$. As noted before, in most cases this least squares problem is ill-posed and has an infinite number of admissible solutions. Therefore, additional prior expectations about the sought type of images are invoked in order to select a suitable $\widehat{\mathbf{z}}$, which results in minimizing an augmented energy functional:

$$\min_{\mathbf{z}} \|\mathbf{b} - \mathbf{W}\mathbf{z}\|_2^2 + \lambda R(\mathbf{z}), \tag{2.2}$$

where λ is a positive constant.

The regularization term $R(\cdot)$ is method-dependent and often takes the form of an ℓ^p-norm of the image coefficients in some specific basis. One of the classical examples of regularized image reconstruction is Wiener filtering [206]. It can be shown that in this case $R(\cdot)$ in Eq. 2.2 becomes the ℓ^2-norm of the image's Fourier coefficients, making the method an instance of the Tikhonov regularization in the Fourier domain [147]. Unfortunately, this implicitly imposes the assumption of continuity on the found solution (not necessarily obeyed by most images), which usually results in poor reconstruction of sharp contrast edges.

To avoid the excessive image smoothing, a regularization that allows for discontinuities in $\mathbf{z}$ was introduced in the form of image total variation (TV) [169], which is defined as the ℓ^1-norm of the image gradient, $R_{TV}(\mathbf{z}) = \|\nabla\mathbf{z}\|_1$. Operating in the image domain directly, it penalizes small differences in neighboring pixels (which in denoising, for example, are supposedly caused by noise) and instead favors solutions with large homogeneously filled regions separated by sharp edges. Due to its edge-preserving properties, TV minimization has found broad applicability in denoising [39, 169], compressive sensing reconstruction [33], inpainting [40], and other problems. In compressive sensing, for example, this approach achieves perfect reconstruction of certain types of images (such as the popular Shepp-Logan phantom) from a small number of their measurements [33]; theoretical

guarantees of its convergence were further investigated in [149].

Further motivation for using the ℓ^1-norm as a regularization term in Eq. 2.2 comes from its ability to favor the recovery of *sparse* solutions [68], which made it a useful heuristic approximation for the computationally intractable ℓ^0-pseudonorm. The principal assumption of such approaches is that an appropriately chosen basis or frame concentrates essential image information in just a few high-magnitude representation coefficients while setting the others to zero. In addition to separable orthogonal wavelet bases [59, 139], numerous overcomplete directional transforms (e.g. steerable wavelet pyramids [182, 198], wedgelets [67], curvelets [34], contourlets [65], etc.) have been designed as parsimonious descriptors of essential low-level image features, such as sharp oriented edges. Furthermore, group sparsity has become a powerful idea that relies on statistical dependence between neighboring coefficients [156, 159, 200].

To invoke the sparsity assumption in reconstruction, popular reconstruction algorithms are often based on matching pursuit [140, 154], which greedily tries to build a sparse solution by adding one best basis element to the approximation at a time. Others, assuming $R(\cdot) = \|\cdot\|_1$ in Eq. 2.2, directly rely on methods for solving the resulting convex optimization problem [46, 192], which became known as basis pursuit. Moreover, a number of proximal methods [18] have been developed [12, 60, 96, 129] that arrive at a sparse solution by alternately updating $\mathbf{z}$ to better fit the vector of measurements $\mathbf{b}$ and shrinking the transform coefficients with a specific thresholding function (e.g. hard- or soft-thresholding) to zero out small entries.

In our work, we will directly associate the regularization term in Eq. 2.2 with the distance to the set of admissible images and then minimize the resulting functional with a simple descent algorithm. In our case, the set of admissible images are those whose patches all conform to a specific patch model. Therefore, we next turn our attention to the paradigm of representing images with their patches and discuss in detail some of the most popular and successful patch-based algorithms, as this local approach to image processing constitutes the inspiration for our solution.

2.2.2 Overview of Patch-based Image Models

All methods mentioned in the previous section rely on global models for entire images (e.g. in iterative thresholding, every pixel of a large image simultaneously contributes to computing a single vector of wavelet coefficients). Instead, it was found recently that treating images as collections of their small overlapping regions, often called *patches*, and modeling each of them separately produces surprisingly effective results in solving various image processing problems ranging from denoising to structural manipulations. This approach is motivated primarily by the high degree of self-similarity and redundancy often present in most natural images. The reasoning behind it can be conveniently illustrated with the following simple example.

For the purpose of explanation, let us consider images of high contrast black-and-white curves, such as those found on the hide of a zebra, for example (Fig. 2.1). We note that all sufficiently small patches (e.g. 5×5 pixels) across such images are very simple and similar to each other. Indeed, they either contain almost straight contrast edges or uniformly filled black or white areas. Thus, it is not unreasonable to expect the existence of simple models for the set of such patches.

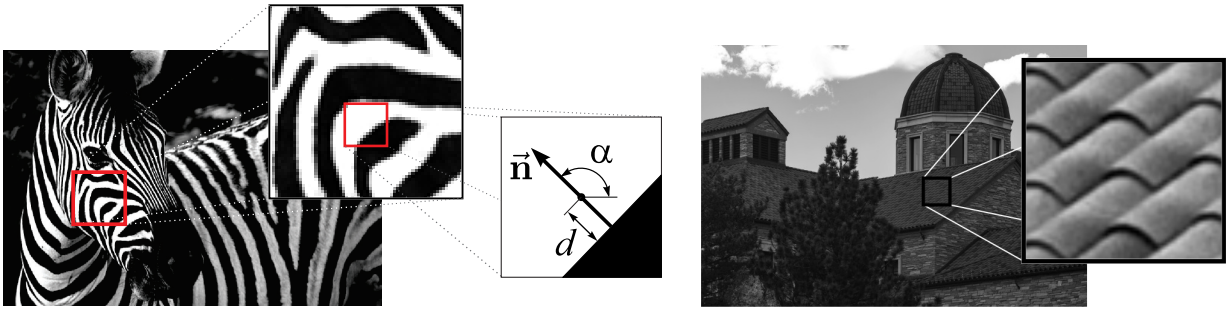


Figure 2.1: Patch-based approach to natural image processing. Left: Most small patches of high-contrast images (e.g. 5×5 pixels) contain almost straight contrast edges or uniformly filled black or white areas. A possible low-dimensional parameterization of such patches includes the distance d from the wedge to the center of the patch and the angle α between the normal to the wedge and the horizontal direction. Right: Self-similarity found in regular patterns and textures validates their representation as a collection of small constituent elements. We will use the enlarged fragments of these images in our experiments in Chapters 4 and 5.

Even though this example may seem overly simplified, it is straightforward to extend our observations to the case of more complex natural images, whose patches may also contain ridges

or gradients, as well as vary in contrast and brightness. In any case, the number of degrees of freedom in all admissible patches appears to be much smaller than their dimensionality (note that even relatively small 5×5 patches already reside in a $25-$dimensional space). Therefore, large complex images can be modeled as collections of their overlapping patches, each of which admit simpler descriptions.

This is the key idea of recently developed patch-based algorithms that show state-of-the-art results in denoising [25, 54, 217], inpainting [10, 53], compressive sensing reconstruction [71, 45], and other inverse problems [56, 57]. We will discuss their most successful representatives in detail in the next subsections.

To structure our survey of the vast number of existing patch-based image processing algorithms, we will draw a distinction between them based on the source of exemplar patches they use for reconstruction. We note that it is common, at least in the denoising literature [27, 146, 213], to refer to the methods that build a solution from modified patches of the *same* initially-given (i.e. noisy) image as internal; Non-local Means [25] and BM3D algoithms [54] are two common examples of such approaches. In contrast, external methods often rely on universal patch models derived from other exemplar images, which are cast, for example, in the form of patch dictionaries [73, 217], patch manifolds [45, 158], or learned deep network structures [89, 186]. Usually external methods are more flexible and can be more or less directly extended to a broader range of problem modalities. However, internal methods tend to surpass them, especially in processing textures, when sufficient numbers of exemplar patches are readily available in the given image [146].

Thus, we adopt a similar viewpoint in our modest attempt to capture the diversity of patch-based approaches for various image processing problems. In the next subsection we will discuss algorithms that build their solutions directly from given exemplar patches; they are often acknowledged as tools of choice for inpainting and structural editing of large complex scenes (see Fig. 2.3). Then, on the next level of generalization, we will look at methods that allow each image patch to change, often as a certain function of its neighbors. Algorithms of this kind, such as BM3D, have proven especially effective for denoising. Finally, in Section 2.2.2.3, we will discuss truly external

approaches to modeling the entire set of all admissible image patches. Here we will focus primarily on dictionaries- and neural-networks-based approaches but defer our review of manifold models for patches – the foundation of our effective solution – to the upcoming section.

2.2.2.1 Internal Algorithms for Structural Image Editing

To start, on one extreme of the spectrum of patch-based image processing algorithms, we place the methods that rely on searching for suitable patches in an input image and then directly adapting them without major changes as building blocks to form a solution [10, 53, 70, 69, 118]. They achieve results of particularly good visual quality in large scenes with complex patterns and textures, where substitution of a patch for a similar one may not be easily noticeable. However, often such algorithms are tailored for addressing only a specific problem, such as inpainting. Their extension to other inverse problems (e.g. compressive sensing reconstruction) appears to be non-trivial, if even possible, since they rely on a high-quality initial image for patch sourcing.

Historically, the algorithm for texture synthesis proposed by Efros and Leung [70] was one of the first successful methods of this kind. Given a small sample of a desired texture, the algorithm proceeds pixelwise to generate a larger piece statistically similar to the observed prior. It is assumed that each pixel is conditionally independent on the rest of the image given the values of its closest neighbors (i.e. those pixels constituting a patch). Thus, each new pixel is synthesized such that the resulting patch centered around it could be also found somewhere in the original sample.

However, growing large images by one pixel at a time could be inefficient, especially given that for most patches of structured patterns, knowledge of a few pixels in a patch completely determines the rest of them. Therefore, it was soon after proposed to copy entire patches from the initial sample texture to synthesize the resulting solution. This, in fact, raises the problem of fitting and seamlessly stitching together patches with unmatching borders. A possible solution is to make patches overlap and then carve them with an irregularly shaped cut to make their borders match as jigsaw pieces. Cuts along the minimum error boundary in the overlap were proposed by Efros and Freeman in the algorithm known as Image Quilting [69]. Kwatra et al. consider each

pixel in the overlap region to be a node in the graph with edges between adjacent nodes weighted according to the pixel differences in the two patches. Their Graphcut algorithm then cuts each patch along the minimum weight cut in the graph [118].

Simakov et al. summarized the idea of similarity between two images on the patch-level by introducing a global bi-directional patch-based similarity measure [181]. An image $\mathbf{S}$ is close to an image $\mathbf{T}$ with respect to this distance if for every patch $\mathbf{s} \in \mathbf{S}$ there is a close approximation in $\mathbf{T}$ (which ensures "coherence"), and vice versa, if every patch $\mathbf{t} \in \mathbf{T}$ is represented by a similar patch in $\mathbf{S}$ ("completeness"). Mathematically these conditions are expressed as the distance between images:

$$d_{BDS}(\mathbf{S},\mathbf{T}) = \underbrace{\frac{1}{N_{\mathbf{S}}}\sum_{\mathbf{s}\in\mathbf{S}}\min_{\mathbf{t}\in\mathbf{T}} d(\mathbf{s},\mathbf{t})}_{Completeness} + \underbrace{\frac{1}{N_{\mathbf{T}}}\sum_{\mathbf{t}\in\mathbf{T}}\min_{\mathbf{s}\in\mathbf{S}} d(\mathbf{t},\mathbf{s})}_{Coherence}, \tag{2.3}$$

where $N_{\mathbf{S}}$ and $N_{\mathbf{T}}$ are the total numbers of patches in both images and $d(\mathbf{s},\mathbf{t})$ is the Euclidean distance between two patches (please see Fig. 2.2 for an illustrative example of comparing images using this distance). The PatchMatch algorithm of Barnes et al. [10] for efficient approximate nearest neighbors search for matching patches and minimization of this similarity measure has become critical to some of the most successful structural image editing tools to date. They allow one to effectively address such high-level tasks as image retargeting, reshuffling, and completion.

Meanwhile, other successful exemplar-based algorithms were specifically developed for image inpainting, which is a problem closely related to the texture synthesis problem above. The goal here is to fill a gap in the image (caused for example by an overlaid inscription, physical damage to a photograph, or an object removed during editing) using the content from the rest of the scene. Similar to texture synthesis, these algorithms attempt to stitch together exemplar patches from elsewhere in the image to fill the unknown region. In contrast to the general problem of texture synthesis however, here the order in which unknown pixels are filled significantly affects the final result. Criminisi et al. in [53], for example, follow a technique similar to [69] and fill the gap pixelwise to match the result with patches in the reference region. They specify the filling order by assigning high priority values to those pixels lying on the continuation of essential linear structures

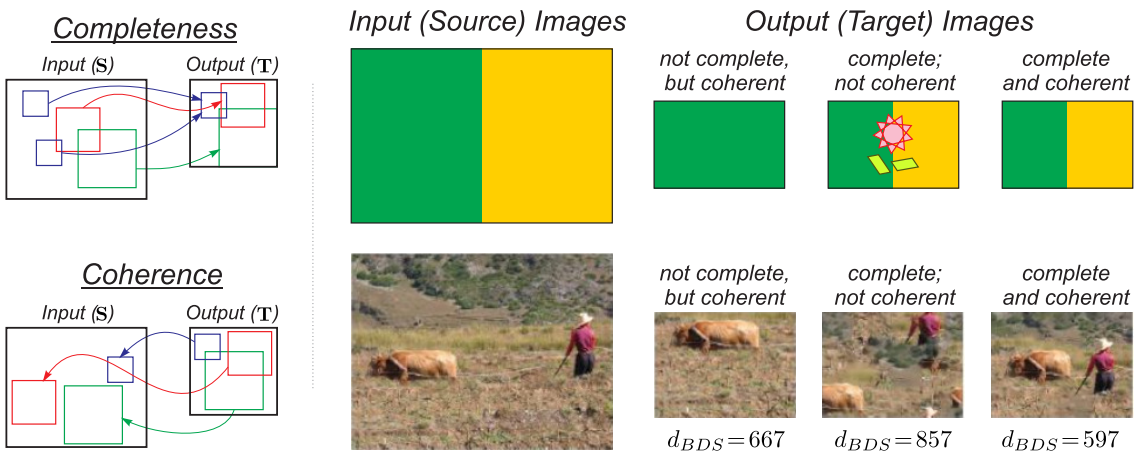


Figure 2.2: (Image from [181].) Demonstration of the concepts of completeness and coherence of two images in an example of creating an image summary. Completeness means that every patch in the source image (**S**) corresponds to a similar patch in the target image (**T**), essentially that **T** represents all parts of **S** in some way. Coherence means that every patch in the target image **T** corresponds to a similar patch in **S**, i.e. that **T** does not invent new image features that have no analog in **S**. The source (**S**) and target (**T**) images are considered equal (or close) with respect to the bi-directional similarity distance of Eq. 2.3 if any patch of one of these images can be (approximately) found in the other and vice versa.

in the image or located in the corners of the gap. Moreover, Zhou and Robles-Kelly in [214] guide their choice of the optimal filling patch not only by the values of known pixels on the border of the gap, but also consider those patches that can be potentially chosen for inpainting the neighboring pixels. This maximizes the local consistency with respect to these potential neighbors. Finally, a global criterion for inpainting was proposed by Wexler et. al. in [205], who optimize a functional equivalent to the coherence term in Eq. 2.3 with **S** and **T** being the unknown and reference regions respectively. This encourages every patch in the final inpainted region to be similar to one elsewhere in the undamaged portion of the image.

2.2.2.2 Internal Methods based on Joint Modeling of Similar Image Patches

Methods of another class, instead of explicitly borrowing existing patch exemplars, attempt to exploit the dependencies and relationships between similar patches in the same image. In this vein, local image models based primarily on a Markov random fields description were initially used for texture synthesis [215]. Similar models were soon established to address a broader class of

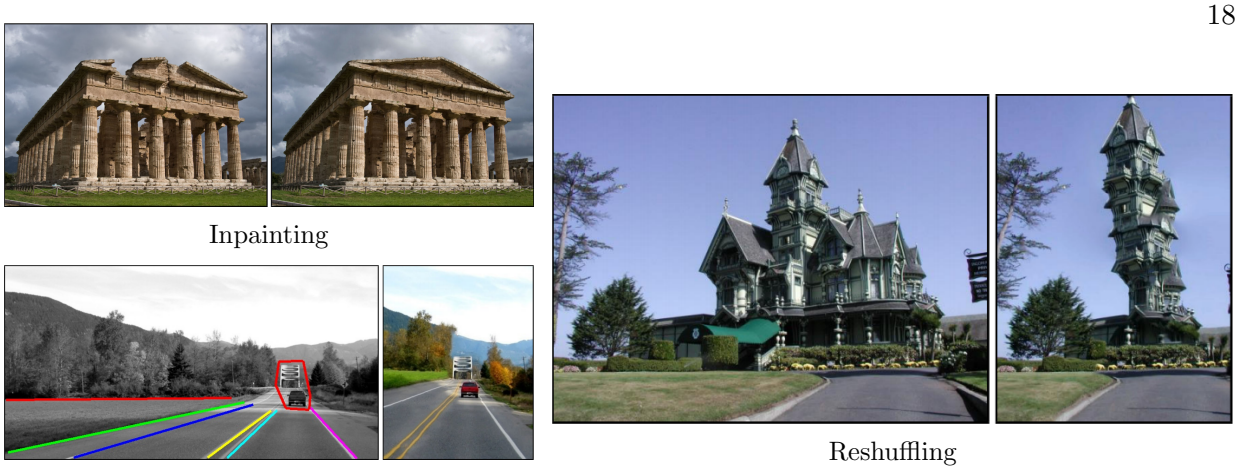


Figure 2.3: (Image from [10].) Examples of structural editing problems that can be solved by minization of Eq. 2.3. Internal algorithms like PatchMatch take full advantage of the vast set of available patches to construct a visually plausible solution.

problems, such as superresolution [80], classification [62], or change detection [155]. Furthermore, fractal methods that directly exploit image self-similarity on multiple scales (across patches of different sizes) were proposed for image compression [11] and later adopted for denoising [85]. In essence, all these approaches view each image patch as a function of its neighbors and can often be analyzed with the framework of image-dependent filtering [144].

The Non-local Means (NL-means) denoising algorithm of Buades et al. [25, 26] is a common example of such a filtering procedure [144]. It partially owes its popularity to the simple and elegant formulation: the image is denoised by adaptively averaging similar patches, which is known to be an asymptotically optimal strategy given infinitely many reference patches. Formally, for a patch $\mathbf{p}$ of an image $\mathbf{I}$, NL-means computes a denoised estimate of its central pixel $c_{\mathbf{p}}$ as:

$$\widehat{c}_{\mathbf{p}} = \sum_{\mathbf{q}\in\mathbf{I}} c_{\mathbf{q}} \cdot w\left(\mathbf{p},\mathbf{q}\right), \tag{2.4}$$

where the sum is over all patches $\mathbf{q}$ in image $\mathbf{I}$, $c_{\mathbf{q}}$ denotes the central pixel of patch $\mathbf{q}$, and the weights $w\left(\mathbf{p},\cdot\right)$ are defined as $w\left(\mathbf{p},\mathbf{q}\right) = \frac{1}{S_{\mathbf{p}}} e^{-\frac{\|\mathbf{p}-\mathbf{q}\|_2^2}{h^2}}$. This weighting scheme puts more emphasis on patches $\mathbf{q}$ that are similar to a given patch. The normalization constant $S_{\mathbf{p}} = \sum_{\mathbf{q}\in\mathbf{I}} w\left(\mathbf{p},\mathbf{q}\right)$ makes all weights for a patch $\mathbf{p}$ sum to 1, and the width parameter h depends on the noise variance.

In practice, the sum in Eq. 2.4 is computed not over all of $\mathbf{I}$ but instead over some limited predefined neighborhood (e.g. over 21×21 regions for 3×3 patches [26]). Notice, however, that the

relative positions of patches $\mathbf{q}$ used to construct the estimate are ignored by the algorithm, hence making it non-local. From this point of view, it is instructive to compare this approach with its special case, Bilateral Filtering [193]: NL-means defines pixel similarity patch-wise across possibly remotely-located image regions instead of pixel-wise in a local neighborhood as done by the latter.

NL-means produced significant improvements in denoising results over traditional non-patch-based filtering schemes, and the simplicity of its procedure stemmed numerous works proposing possible quality-improving modifications. For example, adaptive neighborhood selection and refinement of the local noise variance estimates have been considered in [24, 113, 137]. Moreover, Kervrann et al. look at iterative application of this model [113], and further discussion of its connection to diffusion processes can be found in [144, 183].

The strategy of processing groups of similar image patches together, employed by NL-means, was taken further and generalized by Dabov et al. leading to the development of state-of-the-art denoising algorithm, BM3D [54]. It operates by searching for and stacking similar image patches into three-dimensional arrays. These blocks of patches are then brought to a three-dimensional transform domain (e.g. with a separable wavelet or discrete cosine transform), where the presumed hidden structure shared by similar patches can be revealed in the form of a shared sparsity structure across their coefficients. Now, as in established denoising methods [60, 96], one may encourage transform sparsity in a noisy patch by thresholding its coefficients (according to the shared sparsity structure of similar exemplars) to reduce noise. (Please see Sections 2.2.1 and 2.2.2.3 for detailed discussions of methods that rely on sparsity of image coefficients under certain transforms and of their patches in learned dictionaries respectively.)

The patches are then returned back to the pixel domain with an inverse transform, and the entire image estimate is computed by aggregating and averaging overlapping patches. This joint filtering procedure is able to preserve and reveal even the finest details shared across many patches while effectively removing the noise. To further enhance the results of reconstruction, this basic estimate is passed as an input to a specifically designed collaborative Wiener filter on the second step of the algorithm (see Fig. 2.4).

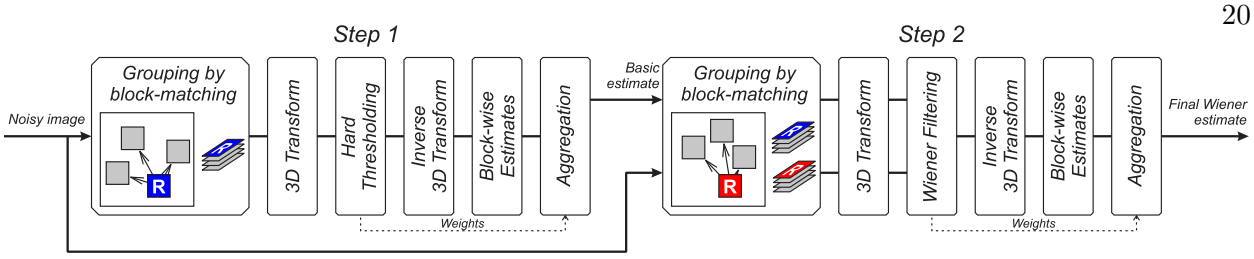


Figure 2.4: (Figure from [54].) The flowchart of the two-step BM3D algorithm. The main algorithm consists of the grouping, thresholding, and aggregation steps. The final estimate is produced by collaborative Wiener filtering to enhance the quality of denoising.

Besides denoising, this model was adapted for superior performance in deblurring, compressive sensing reconstruction, superresolution, and other image processing problems [57, 71]. In compressive sensing, for example, the estimate is recursively injected with noise and then filtered with the described algorithm to restore its features consistent with the CS measurements. In our method developed in Chapter 5, we will employ a very similar idea of projecting intermediate solutions onto the constraint subspace, yet our assumed global model for patches will effectively make our algorithm readily applicable for solving any linear inverse problem without modifications.

Let us summarize by commenting on a few other approaches that search for and jointly process the nearest neighbors of a patch in an input image. For example, Chatterjee and Milanfar [42] group similar patches in disjoint clusters and then build local linear models for each of them. Ram and Elad, on the other hand, arrange all patches in a shortest-path chain and then perform denoising on one-dimensional signals formed by the pixels along this chain [160]. We notice that an assumption of smoothness of the set of image patches underlies both these methods as well as many other algorithms mentioned so far. Indeed, while the above smooth ordering method of Ram and Elad [160] was shown to be related to the BM3D algorithm (they both apply one-dimensional transforms on stacks of patches), it can also be viewed as an instance of tracing a path on the underlying smooth patch manifold. Furthermore, we would argue that the authors in [42] implicitly construct an estimate of the nearest point on the piece-wise linearly approximated smooth set of patches. Hence, we might hope to improve upon these methods by making their implicit goals explicit in our approach by using an externally-defined manifold prior. We will focus more closely on the models of this kind in the next section.

2.2.2.3 External Models for the Entire Set of Image Patches

A weakness of all above (internal) methods, however, is that they can only reproduce functions of the patch exemplars available in the given image and do not attempt to model a collection of all suitable patches as a whole. This potentially limits their applicability in addressing problems for which initializations are not directly defined in the image domain but are given in the form of measurements instead, as in, for example, compressive sensing reconstruction. In contrast, external methods completely rely on general patch models, which they apply to treat various previously unseen images. Typically such models are derived or learned offline from a set of representative patches. Learned dictionaries, learned non-linear deep approximators, and manifold models underlie the most successful approaches of this type.

An overcomplete dictionary is a collection of vectors, called atoms, in which any valid signal of interest admits an extremely concise description. Such representations have been particularly useful for modeling sets of image patches as seen in many practical applications [74, 138, 211]. Among many existing dictionary learning methods [75, 127, 168, 199], the K-SVD [3], along with its numerous enhancements [141, 142, 161], stands out as one of the most efficient and successful algorithms. It proceeds by alternating between finding a sparse code for parsimonious representation for the collection of image patches with respect to the current dictionary and updating the dictionary elements with respect to the patch codings.

Once the dictionary is learned, all overlapping patches of an image can be modeled independently with their sparse representations and then averaged to form a solution [74]. To fully exploit the potential of this technique, however, one needs a good initialization for patches, which may not be readily available in such problems as inpainting and CS reconstruction. Alternatively, Zoran and Weiss propose to maximize the likelihood of a randomly selected patch with respect to the sparsity prior on the dictionary, i.e. the expected patch likelihood of an entire image [217]. Their general approach overcomes the challenges of treating overlapping patches and also allows for the use of other probabilistic priors. In this regard, relatively simple patch dictionaries are advantageous over

translation-invariant MRF models, which can be extremely difficult to train [166].

Invoking the sparsity assumption of dictionary coefficients geometrically characterizes the set of all admissible patches as a union of subspaces [18]. While proven successful in numerous applications, there is no evidence that this model indeed provides the most accurate representation for the set of image patches. On the other hand, recent theoretical results indicate that the structure of this set closely resembles a low-dimensional manifold, which effectively allows for smooth transitions between similar patches [36, 125, 158]. Thus, employing manifold models in image processing may pull out the unrealized potential of established image representations and will be of primary interest in our work. We will review them in detail in the next section.

Before moving on to discussing manifold models, we would like to conclude this section by mentioning a class of novel effective solutions based on the machinery of deep neural networks. Their recent remarkable successes in computer vision [116, 170] motivate their applicability for solving inverse problems as well. In fact, it has been noted that many image processing algorithms described so far can be viewed as realizations of deterministic (although complex) mappings onto the set of desired images [89, 186]. Thus, instead of characterizing the target set, one may attempt to directly learn the mapping itself, for example, in the form of a trainable network.

Taking on this strategy, plain multilayer perceptrons (MLP) were found to readily achieve state-of-the-art results in image denoising [28] and deconvolution [175], although at the cost of their excessively high complexity [203]. This pure learning strategy, however, does not make any assumptions about image statistics but rather relies on the property of MLPs to be universal approximators [103]. Furthermore, convolutional neural networks [123] – a powerful tool for modeling translation-invariant image structures [116] – were successfully applied for denoising [105, 72], superresolution [66, 204], and deconvolution [208]. These novel network designs inspired by the traditional dictionary learning techniques effectively combine the structure of established thresholding methods [60, 129] with adaptability of the deep learning framework eventually leading to significant quality improvements.

In the next section we will finally turn our attention to manifold models particularly suitable

for families of similarly structured smoothly changing high dimensional signals, of which image patches are a common example.

2.3 Manifold Models in Signal Processing

Let us start by providing a high-level motivation for the manifold representation of signals and then proceed by discussing applications of manifold models in patch-based image processing.

2.3.1 Overview and Motivation

Even though the dimensionality of modern signals constantly increases (e.g. the number of pixels in images), the inherent structure in a signal often allows for a more concise description [176]. For example, in computer vision, images of a (known) subject taken from varying positions may be described in terms of these parameters. As an illustration, consider a set of images of a bunny viewed from different angles (such as the one shown in Fig. 2.5) with a moving lighting source. Indeed, there is a bijective correspondence between such images and the values of the continuous angular parameters, which allows one to unambiguously reconstruct the image knowing the position of the camera and the lighting source relative to the subject and vice versa.

Usually, as in the above example, the number of descriptive parameters d is much smaller than the dimension of the ambient signal space D, but still such a parameterization describes the signal with sufficient accuracy. The signal itself can thus be viewed as a (non-linear, in general) mapping f from the set of parameters $\Theta \subseteq \mathbb{R}^d$ to the signal space. Assuming f is a homeomorphism, all signals $f(\theta)$ generated by this model for different values of $\theta \in \Theta$ lie on some low-dimensional manifold $\mathcal{M}$ in $\mathbb{R}^D$ [126, 201]. In other words, signals change smoothly as a function of the parameters, which is a reasonable assumption in most cases.

We note that the parameters Θ, in general, may or may not have any specific comprehensible meaning, but can be learned from the set of signal samples and reflect its geometry instead. Statistical approaches that aim to recover this or a related low-dimensional parameterization based on the training set of representative samples have found broad applicability in machine learning,

particularly for the purposes of computer vision [133], face recognition [98, 202, 210], identification of facial expressions [41, 77, 177], action recognition [1, 195], automatic lipreading and synthesis [3], human gait modeling [64], and medical image analysis [4, 84, 94, 179, 185], among others. Furthermore, application of topological methods to the analysis of large high-dimensional datasets was considered by Carlsson in [35]. Recent work of Lum et al. [134] demonstrates the superiority of this approach over standard purely statistical methods in revealing subtle but presumably meaningful dependencies in datasets of various kinds.

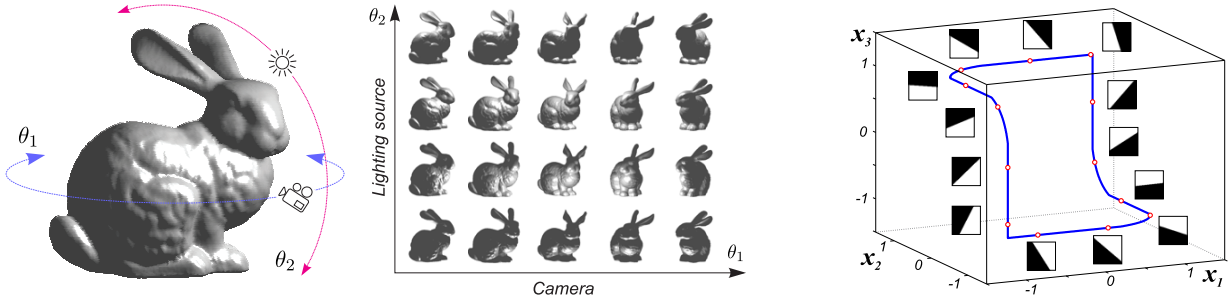


Figure 2.5: Two examples of signals that admit natural descriptions with low-dimensional manifolds. Left: The images of a bunny taken with varying positions of the camera and the light source. These modes of variability constitute a natural intrinsic parameterization of the manifold. Right: A submanifold of high-contrast 5×5 image patches (plotted for $0 \leq \alpha \leq 2\pi$, $d = 0$) embedded in $\mathbb{R}^3$ by keeping the values of only the first three pixels of each patch: x_1, x_2, and x_3. The two cusps are caused by the low-dimensional embedding; these points correspond to patches with vertical transition boundaries, i.e. $x_1 = x_2 = x_3$. In both examples the number of degrees of variability is much smaller than the dimension of the ambient spaces. Note that these sets are not closed under linear operations, which demonstrates the non-linearity of the generative mappings.

On the other hand, manifold models suggest an elegant way to impose a structure on the set of image patches. For the purpose of explanation, we again look at the example of a simple image of high-contrast black-and-white curves in Fig. 2.1. Indeed, all its patches approximated with wedges can now be easily parameterized with an angle α and a distance d. For an appropriately chosen parameter range, there is a one-to-one and onto continuous correspondence between α and d on the one hand and the set of patches on the other. In other words, all such patches belong to an underlying two-dimensional manifold that imposes mutual constraints on the pixels of each patch.

Next we discuss recent works on patch-manifold models and their applications in addressing inverse problems.

2.3.2 Manifold Models for the Set of Image Patches

Even though appearance manifolds of images are useful in machine learning and computer vision scenarios [133, 132, 148, 184], their application for solving inverse problems on the set of natural (e.g. photographic) images is hindered by the necessity of learning the image manifold from a large training set of sufficiently similar examples, which usually is not available except for very simple image classes like the sculpture faces above. Instead, working in the lower-dimensional space of image patches, as opposed to the space of entire images, dramatically reduces the size of the required training set and the model complexity overall. In principle, it allows one to describe a specific class of images by a single set of training patches [83].

Recent works study the properties of the underlying patch manifold derived from natural and synthetic images. The analysis of full probability distributions of small image patches was conducted by Lee et al. in [125]. They found that most 3×3 patches extracted from range and optical images are concentrated in compact clusters or along non-linear manifolds of intrinsic dimensionality much lower than the dimension of the ambient space. Moreover, the manifold of high-contrast patches was shown by Carlsson et al. in [36] to have the topology of a Klein bottle.

Furthermore, practical methods that directly use patch manifolds as effective priors were developed for regularization of inverse problems. The nonparametric Bayesian models of Chen et al. [45] and Gaussian mixture models [212, 217] approximate the nonlinear patch manifold as a union of linear local distributions learned, for example, with the MAP-EM algorithm [212]. They have been successfully applied to describe the manifold corresponding to a single patch. However, because of their complexity, neither extends readily to the case of several overlapping patches, although the recently proposed method of imposing coherence on overlapping patches via Markov random fields [130] is one of the first attempts at this extension. The training of such models is also extremely computationally intensive. Moreover, Kim et al. [114] model the patch manifold with kernel PCA, as we will also, inspired by its prior success in the manifold learning literature. During reconstruction, they minimize the distance to the resulting model but treat each patch separately. This

inevitably results in blocking artifacts in the reconstructed image, which are commonly reduced by overlapping and averaging the borders of neighboring patches.

In a different approach [158], Peyré regularizes inverse problems by requiring the overlapping image patches to trace a two-dimensional trajectory along the patch manifold. However, the main drawback of this method is the computational expense of optimizing over all such trajectories on the densely-sampled non-linear patch manifold.

Thus, we can see that manifold models for image patches have great potential as useful and effective regularizers for inverse problems in image processing. However, their applicability in most practical settings has been primarily hindered by the difficulty of working with non-linear patch manifolds that usually lack exact descriptions. In our work, we will propose a pragmatic solution to this problem, which we briefly outline next.

2.3.2.1 An Outline of Our Proposed Model of Intersecting Patch Manifolds

In our model for images, we will consider several manifolds, one for each patch position. We will claim that the image itself composed of many overlapping patches lies on the intersection of their corresponding manifolds. We note that the hard problem of finding manifolds' intersections was considered by Cadzow for several applications [32]. Their approach is based on iterative composite projections and converges under relatively mild conditions. However, this method is restricted to manifolds for which the projections can be easily computed via property mappings (such as sets of matrices of a specific rank or structure). Instead, we will learn the manifolds from their training samples, which allows for wider generalization of our method.

The main idea of our work is to use the kernel PCA algorithm in our manifolds intersection model as it is one of the most general and powerful known manifold learning techniques. While not necessarily the most exact way of projecting onto a manifold, kernel PCA will allow us to quickly find an approximate mapping. Approximating each individual projection with speed will then make it possible for us to locate an approximate intersection point for many manifolds simultaneously. With this strategy, we will develop two solutions, one in terms of a closed-form expression in the

kernel-induced feature space and another in the form of iterations in the original image space that will easily incorporate any additional linear constraints on the reconstructed image, if desired.

2.4 Conclusion

In this chapter we have presented an overview of methods and algorithms for solving inverse problems in image processing. Patch-based approaches have recently become particularly effective in this realm and often they account for current state-of-the-art solutions. Meanwhile, manifold models provide an elegant way to impose a structure on the set of image patches. We will use this idea in the next chapters to develop a general image processing framework based on an intersecting manifolds model of overlapping image patches. This will eventually allow us to effectively solve any linear inverse problem.

However, inferring the structure of a manifold from its samples is a difficult learning problem by itself. We will address it with the machinery of kernel methods. These methods have provided the foundation for non-linear extensions of many established linear algorithms previously and have been proven effective over time. In the next chapter, we review in detail this powerful machine learning framework that constitutes the core of our dissertation work. We will also use this opportunity to formally introduce the notation used throughout the rest of the work.

Chapter 3

Introduction to Kernel Methods in Machine Learning

This chapter serves to provide the reader with needed background on kernel methods before we describe how we will employ this machine learning tool to build our solutions later in the thesis. We start by giving a broad and general overview of kernel-based algorithms. Then we focus on specific details of Kernel PCA, as it is one of the most powerful known manifold learning methods; we will use it in our approaches. We continue by restating some of the theoretical guarantees of the Kernel PCA solution that by extension will apply to our method in Chapter 6 and then conclude by describing a memory-efficient incremental implementation of the Kernel PCA algorithm.

3.1 Overview of Kernel Methods

Kernel methods are a popular strategy in machine learning for handling difficulties imposed by nonlinearity of a problem in a computationally efficient way. Their main idea is to map data points by some non-linear transformation $\Phi : \mathbb{R}^D \to \mathcal{H}$ to a $D_{\mathcal{H}}-$dimensional feature space $\mathcal{H}$ (with $D_{\mathcal{H}} > D$), in which they can instead be analyzed with linear algorithms (see Fig. 3.2). Efficiency is gained by the fact that the images $\Phi(\mathbf{x})$ need never be computed. Instead, the space $\mathcal{H}$ is implicitly induced by a positive semi-definite kernel function $\kappa : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ that bears the meaning of a similarity measure and represents the inner products in $\mathcal{H}$ [174, 16],

$$\kappa(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}),\, \Phi(\mathbf{y}) \rangle_{\mathcal{H}} . \tag{3.1}$$

The Gaussian kernel $\kappa(\mathbf{x}, \mathbf{y}) = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2}\right\}$ parameterized by the width $\sigma > 0$ and inhomogeneous polynomial kernels $\kappa(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle + c)^d$ of degree d are two common examples of κ.

Lifting the data into a higher-dimensional feature space is indeed a powerful idea as evidenced by numerous machine learning applications that gain advantage from the kernel-based approach. For example, polynomial kernels give rise to feature spaces of monomials of powers no greater than d. They often help to easily "unfold" complex non-linear data structures making them linearly separable, for example, as shown in Fig. 3.1. For Gaussian kernels, the mapping Φ can be interpreted as mapping each point in $\mathbb{R}^n$ to an L^2 function, consisting of a "Gaussian bump" of a certain width centered at that particular data point. When mapped back to the original feature space, a solution consists of linear combinations of these Gaussians, which for example can well approximate a non-linear function in the regression problem (see the right panel in Fig. 3.1). We note that in this case, the induced L^2-feature space of functions is effectively infinite-dimensional. However, we still will be able to work with it using an elegant scheme outlined next. Finally, it is worth mentioning that kernel methods effectively generalize the notion of inner products in arbitrary spaces besides $\mathbb{R}^n$ and have been found useful for comparing complex data structures, such as strings, sequences, or trees [14].

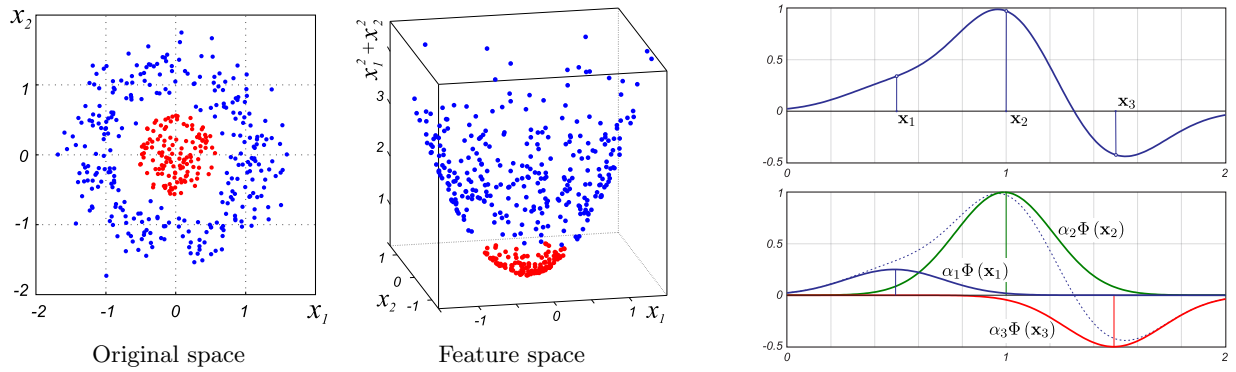


Figure 3.1: The advantage offered by kernel methods: non-linear machine learning problems are linearized in the induced feature space. Left: A dataset consisting of samples of two concentric classes can be made linearly separable in a higher-dimensional feature space (shown is an embedding of $\mathcal{H}$ in $\mathbb{R}^3$). Right: The Gaussian kernel maps data samples $\mathbf{x}_i$ into an L^2-space of Gaussians centered on them, $\Phi(\mathbf{x}_i) = \kappa(\mathbf{x}_i, \cdot)$; linear combinations of these Gaussians can be used to approximate non-linear functions.

Due to Mercer's theorem [5, 174], the feature space $\mathcal{H}$ induced by a positive semi-definitie kernel has the structure of a reproducing kernel Hilbert space. Therefore, any algorithm formulated

in terms of inner products can be adapted to operate in this space simply by substituting the values of the kernel $\kappa(\mathbf{x}, \mathbf{y})$ for the corresponding inner products $\langle \mathbf{x}, \mathbf{y} \rangle$. Eventually, this yields a non-linear solution when mapped back to the original space. This strategy, called the kernel trick, has been used to produce efficient nonlinear extensions of the Support Vector Machines, Principal Component Analysis, and Ridge Regression algorithms, among others [106, 157, 174].

In particular, Kernel Principal Component Analysis (Kernel PCA or KPCA) [173] presumes that, for an appropriate choice of Φ, a manifold in the original space approximately becomes an affine subspace in feature space. It thus learns a manifold from its samples via PCA in feature space (see Fig. 3.2). Despite this seemingly simple approach, KPCA is one of the most powerful known methods for learning the non-linear structure of a manifold from its samples. Indeed, other popular manifold learning algorithms, such as Laplacian Eigenmaps [13], Locally Linear Embedding [167], and ISOMAP [191], were shown in [93, 207] to be special cases of it. Its effectiveness has been proved in many signal processing settings. For example, besides direct application of KPCA for denoising [143], it has been used for super-resolution in [117], and to locally parameterize a patch manifold for the purpose of image deconvolution in [152]. We will use KPCA as a main building block in our model of intersecting manifolds in Chapter 4 and adopt its manifold learning capabilities to efficiently approximate and minimize the distance to the patch manifold $\mathcal{M}$ in Chapter 5.

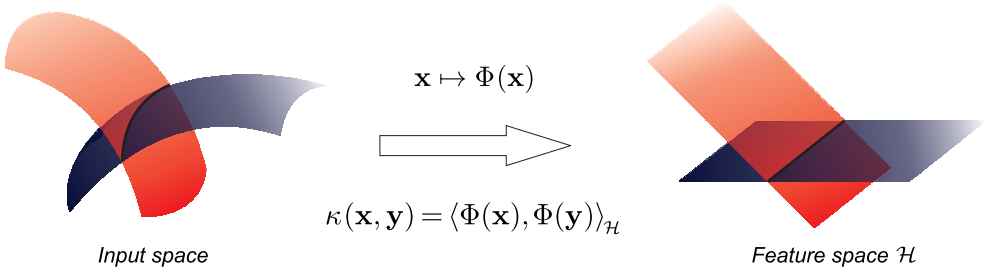


Figure 3.2: In the kernel-induced feature space $\mathcal{H}$, for an appropriate choice of kernel, manifolds can approximately become affine subspaces and can be learned with linear PCA. This results in a non-linear solution when mapped back to the input space. The reproducing property of the kernel function (Eq. 3.1) obviates the need of explicitly mapping datapoints to the feature space and accounts for the computational efficiency of the algorithm.

3.2 Kernel PCA Algorithm

Because of its importance in our later work, in this section, we will review the Kernel PCA algorithm in detail. We begin by first reviewing the conventional (linear) PCA algorithm and then apply the kernel trick to derive its non-linear kernel-based extension for the problem of manifold learning, the KPCA algorithm [173].

3.2.1 Principal Component Analysis Algorithm

Principal Component Analysis (PCA) [108] is a powerful statistical procedure for unsupervised learning that provides a concise description of the dataset in terms of its uncorrelated (orthogonal) principal components, which are the directions of maximum data variability. These components are typically ordered according to the (descending) data variance along each of them. A leading subset of them may thus be chosen to provide a lower-dimensional subspace in which the data nearly lies. This allows one to discover correlations in the data, as well as to learn a more compact representation for describing the observations. These advantages have made PCA a widely used tool for applications ranging from compression and dimensionality reduction to pattern recognition and analysis.

To formally introduce the PCA algorithm, let $\left\{\mathbf{x}_i \in \mathbb{R}^D\right\}_{i=1}^{n_\mathrm{X}}$ be a set of training data samples, which we assemble into a $D \times n_\mathrm{X}$ matrix $\mathbf{X}$ for convenience. PCA aims to find a subspace $\mathcal{U}$ parameterized by its orthonormal basis $\{\mathbf{u}_l\}_{l=1}^{d_\mathcal{U}}$ and an offset $\mathbf{m}$ that minimizes the mean-squared error of projecting the data samples onto it:

$$\begin{aligned} \min_{\mathbf{u}_1,\ldots,\mathbf{u}_{d_\mathcal{U}},\mathbf{m}} \quad & \sum_{i=1}^{n_\mathrm{X}} \left\|\mathbf{x}_i - \mathcal{P}_\mathcal{U}\left(\mathbf{x}_i\right)\right\|_2^2 \\ \text{subject to} \quad & \left\langle \mathbf{u}_k, \mathbf{u}_l \right\rangle = 0, \ for\, k \neq l, \\ & \left\|\mathbf{u}_l\right\|_2 = 1, \end{aligned} \tag{3.2}$$

where $\mathcal{P}_\mathcal{U}\left(\mathbf{x}\right) = \sum_{l=1}^{d_\mathcal{U}} \mathbf{u}_l \mathbf{u}_l^\mathrm{T}\left(\mathbf{x} - \mathbf{m}\right) + \mathbf{m}$ is an othogonal projector onto $\mathcal{U}$. Note that this formulation is equivalent to finding the vectors $\mathbf{u}_l$ that maximize the variance of projections onto them and,

therefore, "explains" the data the best by retaining the most information in $\mathbf{X}$. We will consider this alternative formulation in Section 3.3.2.

The solution to the above optimization problem is found first by noting that the optimal $\mathbf{m}$ is given by the center of the training data samples, i.e. $\mathbf{m} = \frac{1}{n_\mathrm{X}} \sum_{i=1}^{n_X} \mathbf{x}_i$. Then the optimal $\mathbf{u}_k$ can be found in closed form by diagonalization of the sample covariance matrix

$$\mathbf{C} = \frac{1}{n_\mathrm{X} - 1} \sum_{j=1}^{n_\mathrm{X}} (\mathbf{x}_j - \mathbf{m}) (\mathbf{x}_j - \mathbf{m})^\mathrm{T}.$$

Specifically, we decompose $\mathbf{C} = \mathbf{U}\mathbf{\Lambda}_\mathrm{C}\mathbf{U}^\mathrm{T}$, where $\mathbf{U}$ is the matrix of eigenvectors of $\mathbf{C}$ in its columns, i.e. the sought principal components $\mathbf{u}_l$; typically, for dimensionality reduction as well as associated problems, only the first $d_\mathcal{U}$ eigenvectors corresponding to the largest eigenvalues in $\mathbf{\Lambda}_\mathrm{C}$ are retained.

3.2.2 Kernel PCA: From Affine Subspaces to Non-Linear Manifolds

As noted in Section 3.1, to develop a non-linear kernel-based extension of the PCA algorithm, we need to formulate it entirely in terms of inner products between training samples. Unless specified otherwise, we will reuse the notation introduced in Section 3.2.1, but now, instead of working in the original space, we will be (implicitly) looking for a basis $\{\mathbf{u}_l\}_{l=1}^{d_\mathcal{U}}$ satisfying the minimum MSE condition of Eq. 3.2 in the feature space induced by a chosen kernel κ.

Let $\mathbf{x}_i \in \mathbb{R}^D$, $i = 1, \ldots, n_\mathrm{X}$ be training samples of the target manifold $\mathcal{M}$. We denote by $\Phi(\mathbf{X})$ a $D_\mathcal{H} \times n_\mathrm{X}$ matrix formed by the images of these samples $\Phi(\mathbf{x}_i) \in \mathcal{H}$ in feature space arranged in columns. We note that even though the dimension of the feature space, $D_\mathcal{H}$, may be infinite, we will never explicitly work with the matrix $\Phi(\mathbf{X})$ and introduce it only to simplify the derivation of the algorithm. Similarly to our previous definitions, we use $\mathbf{C} = \frac{1}{n_\mathrm{X}-1} \sum_{j=1}^{n_\mathrm{X}} [\Phi(\mathbf{x}_j) - \mathbf{m}] [\Phi(\mathbf{x}_j) - \mathbf{m}]^\mathrm{T}$ and $\mathbf{m} = \frac{1}{n_\mathrm{X}} \sum_{i=1}^{n_\mathrm{X}} \Phi(\mathbf{x}_i) = \frac{1}{n_\mathrm{X}} \Phi(\mathbf{X}) \mathbb{1}$ to stand for the sample covariance matrix and the mean of training samples respectively, but now in feature space; $\mathbb{1}$ denotes the $n_\mathrm{X} \times 1$ column vector of ones.

We note that for any eigenvector $\mathbf{u}_l$ of $\mathbf{C}$, one can write $\mathbf{C}\mathbf{u}_l = \mathbf{u}_l\lambda_l$, and then for any $\lambda_l \neq 0$,

$$\begin{aligned}\mathbf{u}_l &= \frac{1}{\lambda_l}\mathbf{C}\mathbf{u}_l \\ &= \frac{1}{\lambda_l}\frac{1}{n_\mathrm{X}-1}\sum_{i=1}^{n_\mathrm{X}}\left[\Phi\left(\mathbf{x}_i\right)-\mathbf{m}\right]\left[\Phi\left(\mathbf{x}_i\right)-\mathbf{m}\right]^\mathrm{T}\mathbf{u}_l \\ &= \sum_{i=1}^{n_\mathrm{X}}\left[\Phi\left(\mathbf{x}_i\right)-\mathbf{m}\right]\alpha_{i,l}, \end{aligned} \tag{3.3}$$

where $\alpha_{i,l}$ denotes the elements of a $n_\mathrm{X}\times d_\mathcal{U}$ matrix of expansion coefficients $\boldsymbol{\alpha}$, which will be defined later. Equation 3.3 implies that the eigenvectors of $\mathbf{C}$ that correspond to non-zero eigenvalues necessarily lie in the subspace spanned by the centered training samples, $\Phi\left(\mathbf{x}_i\right)-\mathbf{m}$, $i=1,\ldots,n_\mathrm{X}$. Thus, for all $i=1,\ldots,n_\mathrm{X}$, we may consider an equivalent system,

$$\left[\Phi\left(\mathbf{x}_i\right)-\mathbf{m}\right]^\mathrm{T}\mathbf{U}\boldsymbol{\Lambda}_\mathrm{C} = \left[\Phi\left(\mathbf{x}_i\right)-\mathbf{m}\right]^\mathrm{T}\mathbf{C}\mathbf{U}. \tag{3.4}$$

To express the algorithm in terms of inner products and to avoid computation of the sample covariance matrix in feature space, we consider the centered Gram (kernel) matrix $\bar{\mathbf{K}}$ with entries

$$\bar{\mathbf{K}}_{i,j} = \left\langle \Phi\left(\mathbf{x}_i\right)-\mathbf{m},\, \Phi\left(\mathbf{x}_j\right)-\mathbf{m}\right\rangle_\mathcal{H}.$$

It can be easily verified that $\bar{\mathbf{K}} = \left(\mathbf{I}-\frac{1}{n_\mathrm{X}}\mathbb{1}\,\mathbb{1}^\mathrm{T}\right)\mathbf{K}\left(\mathbf{I}-\frac{1}{n_\mathrm{X}}\mathbb{1}\,\mathbb{1}^\mathrm{T}\right)$, where the entries of the (uncentered) kernel matrix $\mathbf{K}_{i,j}=\kappa\left(\mathbf{x}_i,\,\mathbf{x}_j\right)$ are computed in the input space.

With this definition, after substituting Eq. 3.3 and the definition of the covariance matrix $\mathbf{C}$ into Eq. 3.4, the KPCA problem becomes: $\left(n_\mathrm{X}-1\right)\bar{\mathbf{K}}\boldsymbol{\alpha}\boldsymbol{\Lambda}_\mathrm{K} = \bar{\mathbf{K}}^2\boldsymbol{\alpha}$, which is then solved by eigendecomposition of $\bar{\mathbf{K}} = \bar{\boldsymbol{\alpha}}\boldsymbol{\Lambda}_\mathrm{K}\bar{\boldsymbol{\alpha}}^\mathrm{T}$. Coefficients $\boldsymbol{\alpha}$ are found as the first $d_\mathcal{U}$ eigenvectors of $\bar{\mathbf{K}}$ scaled by the reciprocal of the square root of the corresponding eigenvalues, $\boldsymbol{\alpha}_{:,l} = \frac{1}{\sqrt{\lambda_{\mathrm{K}_l}}}\bar{\boldsymbol{\alpha}}_{:,l}$, $l=1,\ldots,d_\mathcal{U}$, to achieve normalization in feature space.

To summarize, the sought subspace $\mathcal{U}$ in the feature space is then described with an orthonormal basis $\mathbf{U}$ formed by its principal components $\mathbf{U}=\left[\Phi\left(\mathbf{X}\right)-\mathbf{m}\right]\boldsymbol{\alpha}$ and the sample mean $\mathbf{m}$. We note that since $\mathbb{1}^\mathrm{T}\bar{\mathbf{K}} = \mathbb{0}^\mathrm{T}$ and thus $\mathbb{1}^\mathrm{T}\boldsymbol{\alpha} = \mathbb{0}^\mathrm{T}$, we can omit subtraction of $\mathbf{m}$ in the above expression for $\mathbf{U}$ and expand it in terms of uncentered samples $\Phi\left(\mathbf{X}\right)$ instead, $\mathbf{U}=\Phi\left(\mathbf{X}\right)\boldsymbol{\alpha}$. Please see [173] if further details of the Kernel PCA derivation are desired.

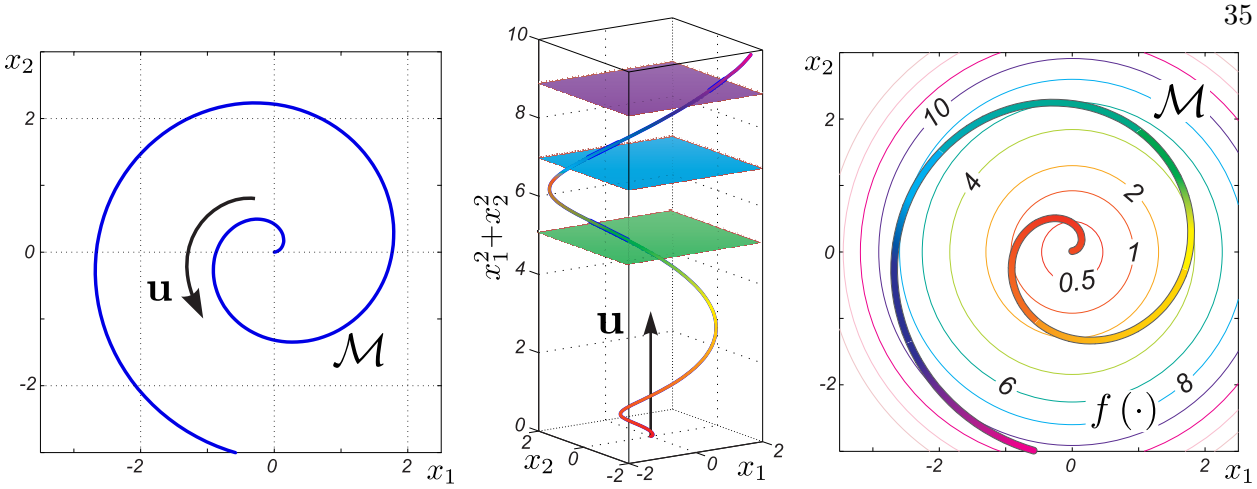
3.2.3 Typical Uses of KPCA and an Interpretation of its Solution

To better understand how Kernel PCA works, we will illustrate with a simple toy example of a spiral-shaped manifold $\mathcal{M}$ in $\mathbb{R}^2$ shown in Fig. 3.3. Here we are targeting two related problems. On the one hand, we want to find a simple representation of the spiral in terms of its low-dimensional intrinsic geometry. On the other hand, we will need to form an understanding of the manifold as it relates to the higher-dimensional ambient space in order to eventually map points in this ambient space onto it. We will show that each of these problems can be effectively addressed with KPCA in the two subsections that follow.

3.2.3.1 Learning a Low-Dimensional Representation of a Manifold

First, we want to obtain a low-dimensional (one-dimensional in this case) representation of the spiral by creating a bijective correspondence between it and a section of $\mathbb{R}$. That is, to any point on the spiral, we aim to assign a unique number continuously, similar to the color coding in Fig. 3.3. This will allow for comparison of different points on the spiral with respect to their relative position along the manifold, which is usually more informative than the simple Euclidean distance in the ambient space, which ignores the underlying manifold geometry.

In our toy example, let us explicitly define a feature space mapping Φ as $\Phi : [x_1, x_2]^{\mathrm{T}} \mapsto \left[x_1, x_2, x_1^2 + x_2^2\right]^{\mathrm{T}}$ for illustration purposes. (Note that this corresponds to a subspace of the feature space associated with the polynomial kernel of degree 2, but does not exactly match any feature space in its entirety. It is introduced here for the purpose of an illustration of feature space we can visualize easily.) This mapping lifts and unfolds the spiral in the induced three-dimensional feature space, which allows one to *approximate* it with a one-dimensional subspace (learned with PCA). We note that depending upon the actual sampling of the spiral, this principal component $\mathbf{u}$ will form a certain small angle with the positive direction of the vertical axis, $x_1^2 + x_2^2$; here for simplicity of explanation, let us assume that it aligns exactly with it and thus admits the form $\mathbf{u} = [0, 0, 1]^{\mathrm{T}}$. Now, the image of any point on the spiral has a unique corresponding projection



Input space | Feature space | Level curves of $f(\cdot) = \langle \mathbf{u}, \cdot \rangle_{\mathcal{H}}$

Figure 3.3: A toy example of learning a spiral with Kernel PCA. We aim to unwrap the spiral by identifying it with a half-axis. This problem can be addressed with PCA in a higher-dimensional feature space, which corresponds to a non-linear solution when mapped back to the original space. Note that each level curve of $f(\cdot)$ intersects the spiral only in one point effectively corresponding to its coordinate on the sought half-axis.

onto $\mathbf{u}$ in $\mathcal{H}$, so the function $\langle \mathbf{u}, \cdot \rangle_{\mathcal{H}}$ creates the desired bijective correspondence in feature space.

Now, we can look at the manifestation of this solution in the original space. By virtue of the Representer Theorem [174], inner products with $\mathbf{u}$ in the feature space can be computed as linear combinations of kernels evaluated on the training dataset, thus defining a function $f : \mathbb{R}^D \to \mathbb{R}$:

$$\langle \mathbf{u}, \cdot \rangle_{\mathcal{H}} = \left\langle \sum_{i=1}^{n_X} \alpha_i \Phi(\mathbf{x}_i), \cdot \right\rangle_{\mathcal{H}} = \sum_{i=1}^{n_X} \boldsymbol{\alpha}_i \kappa(\mathbf{x}_i, \cdot) \triangleq f(\cdot). \tag{3.5}$$

On the right panel of Fig. 3.3, we plot the level curves of $f(\cdot)$, which signify the points in the original space that have the same inner products with $\mathbf{u}$ in $\mathcal{H}$. Note that each of the concentric circles intersects the spiral only once, allowing us to effectively unwrap it with the function f.

3.2.3.2 Mapping Points onto Manifolds and the Preimage Problem

A low-dimensional representation, such as the one found above, is what is typically returned by most manifold learning algorithms [13, 167, 191]. However, they are not readily able to efficiently address our second problem – to map points onto the learned manifold. Specifically, our goal is to identify every point in the ambient space with a unique (ideally closest) position on the manifold.

To attack this problem, a popular mapping strategy was developed, within the context of a larger algorithm called kernel PCA denoising [143]. It maps a point $\mathbf{z}$ onto the manifold by first projecting its image $\Phi(\mathbf{z})$ onto the subspace $\mathcal{U}$ that represents the manifold in feature space, and then secondly seeking a point in the original space, a "preimage", $\widehat{\mathbf{z}} \in \mathbb{R}^D$ that ideally would correspond to this point in feature space, thus satisfying $\Phi(\widehat{\mathbf{z}}) = P_{\mathcal{U}}[\Phi(\mathbf{z})]$. So, for example, for our spiral above, we would first project $\Phi(\mathbf{z})$ onto the vector $\mathbf{u}$ in feature space, then seek the corresponding point in original space.

However, this brings forward another important issue: solutions found in the higher-dimensional feature space most likely lack exact preimages in the input space. Unfortunately, due to the non-invertability of the mapping Φ, the projection $P_{\mathcal{U}}[\Phi(\mathbf{z})] \in \mathcal{H}$ likely does not correspond precisely to any point in the lower-dimensional input space [102]. Indeed, in our example in Section 3.2.3.1, of all points on the KPCA subspace $\mathcal{U}$, only the origin can be mapped back to the input space directly. For any other projection, at best, only an approximate preimage can be recovered instead.

To attempt to address this problem, a variety of preimage-finding strategies exist in the literature, each of which tries to find a point in original space that will be as good a match as possible for the one in feature space. For example, early methods look for a point in the original space whose image under Φ in the feature space lies close to the desired one, i.e. they aim to solve:

$$\widehat{\mathbf{z}} = \underset{\mathbf{z}^*}{\operatorname{argmin}} \left\| \Phi(\mathbf{z}^*) - P_{\mathcal{U}}[\Phi(\mathbf{z})] \right\|_2^2 . \tag{3.6}$$

This distance can be expressed entirely in terms of inner products and minimized using descent algorithms [110, 174] or, in certain cases, with fixed-point iterations [143]. Furthermore, a possible regularization to improve stability and robustness of these methods was proposed in [2, 151]. Alternatively, Kwok et al. [119] construct a preimage by matching the mutual distances between training samples and the found solution in feature space. Moreover, learning the inverse map in a way similar to ridge regression was proposed in [7]. Other recently developed algorithms construct an isometry between the two spaces with respect to training data and thus preserve inner products [101] or establish an isomorphic relation between local Gram matrices in both spaces [104].

However, if our solution in feature space lies far from the image of the input space under the mapping Φ, any of these methods will have no option but to introduce errors in order to arrive at a possible preimage, which eventually may be a poor approximation of the desired mapping onto the manifold.

To alleviate this problem, an improved approach, Robust Kernel PCA denoising [151], explicitly requires the existence of a suitable preimage while minimizing the distance to the subspace $\mathcal{U}$ in $\mathcal{H}$. More precisely, it regularizes the problem of Eq. 3.6 with an additional functional $J_{\mathcal{U}}(\mathbf{z}) = d^2_{\mathcal{H}}(\Phi(\mathbf{z}), \mathcal{U})$. This term can still be computed easily using kernel functions and effectively serves as a tractable proxy for the true distance to the approximated manifold $\mathcal{M}$. It also does not compromise the efficiency of the kernel-based approach. Thus, we see that this elegant approach neatly combines finding of the feature space solution with preimage finding in a single step. We will be inspired by it as we develop our methods in Chapter 5. Furthermore, in Chapter 6, we will develop a novel manifold learning and representation strategy and define a similar distance approximating term, $J_{\mathcal{W}}$, which we will show to be more effective in certain cases.

3.3 Theoretical Guarantees for Kernel PCA

In this section, we will review some theoretical guarantees for the Kernel PCA result. Particularly, we will discuss convergence guarantees for Kernel PCA that establish its consistency. Furthermore, we will see how the choice of the kernel function affects the implicit regularization of the algorithm and thus determines the smoothness of the solution. Later, we will use similar insights to interpret our novel manifold learning approach developed in Chapter 6.

3.3.1 Convergence Properties of Kernel PCA

It is of practical importance to be able to estimate the rate of convergence of the KPCA algorithm for a growing number of training samples n_{X}. Here we recall the results of Shawe-Taylor et al. [178] and Blanchard [17] formulated by bounding the so-called excess error of reconstruction, which we define below.

Let $\Phi(\mathrm{X})$ be a random variable taking values in the RKHS $\mathcal{H}$ induced by some kernel κ and distributed according to a probability distribution P_{X}. Furthermore, we assume that the norm $\|\Phi(\mathrm{X})\|_{\mathcal{H}}^2$ is bounded almost surely, and the rank-one cross-product operator associated to a random element $\Phi(\mathrm{X}) \in \mathcal{H}$, $\mathbf{C}_{\mathrm{X}} = \Phi(\mathrm{X})\,\Phi(\mathrm{X})^{\mathrm{T}}$, almost surely belongs to the associated Hilbert space of Hilbert-Schmidt operators with bounded diameter (note that the expectation of $\mathbf{C}_{\mathrm{X}}$, $\mathbf{C}_1 = \mathbb{E}\left[\Phi(\mathrm{X})\,\Phi(\mathrm{X})^{\mathrm{T}}\right]$, is the covariance operator of $\Phi(\mathrm{X})$ in $\mathcal{H}$). For example, for radial basis functions kernels (such as the Gaussian kernel), while the first condition always holds, the second one translates into a requirement that, in input space, the data lies in a region of bounded diameter almost surely.

Suppose that the data samples $\mathbf{x}_i$, $i = 1, \ldots, n_{\mathrm{X}}$, are drawn such that their images $\Phi(\mathbf{x}_i)$ are distributed according to P_{X} in the feature space. We define R_d to be the (theoretical) error of representing samples of P_{X} with the optimal $d_{\mathcal{U}}-$dimensional subspace (in terms of minimum MSE). Similarly, for a finite sample set $\{\mathbf{x}_i\}_{i=1}^{n_{\mathrm{X}}}$, let $R_{d,n_{\mathrm{X}}}$ be the empirical error of its representation by the KPCA-parameterized subspace $\mathcal{U}$. Note that $R_{d,n_{\mathrm{X}}}$ equals the tail sum of the smallest $d_{\mathcal{U}} + 1, \ldots, n_{\mathrm{X}}$ eigenvalues of the data kernel matrix, $\mathbf{K} = \sum_{i,j=1}^{n_{\mathrm{X}}} \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle_{\mathcal{H}}$, while R_d similarly depends on the tail eigenvalues of the kernel integral operator, $\mathrm{K}f(\cdot) = \mathbb{E}\left[f(\mathrm{X}) \langle \Phi(\mathrm{X}), \Phi(\cdot) \rangle_{\mathcal{H}}\right] = \int f(\mathrm{X}) \langle \Phi(\mathrm{X}), \Phi(\cdot) \rangle_{\mathcal{H}}\, dP_{\mathrm{X}}$. The excess error of reconstruction is defined as the difference $R_{d,n_{\mathrm{X}}} - R_d$ and is always non-negative [178].

It was shown by Shawe-Taylor et al. [178] that with high probability the excess error can be bounded up to a scaling constant and low-order terms by:

$$R_{d,n_{\mathrm{X}}} - R_d \leq \sqrt{\frac{d_{\mathcal{U}}}{n_{\mathrm{X}}} \operatorname{tr} \mathbf{C}_2}, \tag{3.7}$$

where $\operatorname{tr} \mathbf{C}_2$ is the trace of the fourth moment operator $\mathbf{C}_2 = \mathbb{E}\left[\mathbf{C}_1 \mathbf{C}_1^{\mathrm{T}}\right]$.

Blanchard et al. [17] further tightened the inequality in Eq. 3.7 by taking into account the behavior of the eigenvalues of the fourth moment operator, which particularly improves the bound for large n_{X}. Furthermore, they introduced a relative excess bound more appropriate for growing $d_{\mathcal{U}}$ and fixed n_{X}.

3.3.2 Implicit Regularization of the KPCA Solution

It is undeniable that the choice of kernel function and its parameters is the main factor that determines the properties of the resulting non-linear solution. While κ is frequently chosen ad hoc in practice, there are certain theoretical guarantees that characterize the solutions for a given kernel [174]. We will present them here for the kernel PCA algorithm.

We start by recalling the formulation of the linear PCA problem of Eq. 3.2 but for the case of seeking a single principal component. Furthermore, we expand and reformulate the objective as $\|\mathbf{x} - \mathcal{P}_{\mathcal{U}}(\mathbf{x})\|_2^2 = \left\|(\mathbf{x}-\mathbf{m}) - \mathbf{u}\mathbf{u}^{\mathrm{T}}(\mathbf{x}-\mathbf{m})\right\|_2^2 = \|\mathbf{x}-\mathbf{m}\|_2^2 - \left\|\mathbf{u}^{\mathrm{T}}(\mathbf{x}-\mathbf{m})\right\|_2^2$, where the last equality follows from the assumption $\mathbf{u}^{\mathrm{T}}\mathbf{u} = 1$. Thus, given the set of training samples $\mathbf{x}_i$, $i = 1, \ldots, n_{\mathrm{X}}$, we may consider an equivalent optimization problem aiming to find a unit-length vector $\mathbf{u}$ that maximizes the variance of their projections on it:

$$\begin{aligned} \max_{\mathbf{u}} \quad & \sum_{i=1}^{n_{\mathrm{X}}} \left[\langle \mathbf{u}, \mathbf{x}_i \rangle - \frac{1}{n_{\mathrm{X}}} \sum_{j=1}^{n_{\mathrm{X}}} \langle \mathbf{u}, \mathbf{x}_j \rangle \right]^2 \\ \text{subject to} \quad & \|\mathbf{u}\|_2 = 1. \end{aligned} \tag{3.8}$$

As noted in Section 3.2.3, the solution to the Kernel PCA problem can be expressed through the inner products in feature space and formulated in terms of a function $f(\cdot) = \langle \mathbf{u}, \cdot \rangle_{\mathcal{H}}$, itself defined in the original space:

$$\begin{aligned} \max_{f} \quad & \sum_{i=1}^{n_{\mathrm{X}}} \left[f(\mathbf{x}_i) - \frac{1}{n_{\mathrm{X}}} \sum_{j=1}^{n_{\mathrm{X}}} f(\mathbf{x}_j) \right]^2 \\ \text{subject to} \quad & \|f\|_{\mathcal{H}} = 1. \end{aligned}$$

Then, restating the above problem, f can equivalently be regarded as a minimizer of the following function optimization problem (up to a possible constant scaling factor):

$$\begin{aligned} \min_{f} \quad & \|f\|_{\mathcal{H}}^2 \\ \text{subject to} \quad & \sum_{i=1}^{n_{\mathrm{X}}} \left[f(\mathbf{x}_i) - \frac{1}{n_{\mathrm{X}}} \sum_{j=1}^{n_{\mathrm{X}}} f(\mathbf{x}_j) \right]^2 = 1. \end{aligned}$$

The minimization is performed over the set of linear combinations of kernel functions centered on data points, $f \in \{\sum_{i=1}^{n_X} \boldsymbol{\alpha}_i \kappa(\mathbf{x}_i, \cdot) \mid \boldsymbol{\alpha} \in \mathbb{R}^{n_X}\}$, and thus reduces to solving for the expansion coefficients $\boldsymbol{\alpha}$ as discussed in Section 3.2.2.

It was shown in [174] that minimizing the norm of a function in the Reproducing Kernel Hilbert Space amounts for a certain type of implicit regularization. For example, for Gaussian kernels $\kappa(\mathbf{x}, \mathbf{y}) = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right\}$ the above norm can be represented in the Fourier domain as:

$$\|f\|_{\mathcal{H}}^2 = (2\pi)^{D/2} \int_{\Omega} \frac{|\mathcal{F}[f](\boldsymbol{\omega})|^2}{v(\boldsymbol{\omega})} d\boldsymbol{\omega},$$

where $v(\boldsymbol{\omega}) = \mathcal{F}[\kappa(\mathbf{x}-\mathbf{y})](\boldsymbol{\omega}) = |\sigma| \exp\left(-\frac{\sigma^2\boldsymbol{\omega}^2}{2}\right)$ is the Fourier transform of the kernel as a function of $(\mathbf{x}-\mathbf{y})$, and $\mathcal{F}[f](\boldsymbol{\omega})$ is the Fourier transform of f. Since $v(\boldsymbol{\omega})$ quickly decays with $\boldsymbol{\omega}$, minimizing $\|f\|_{\mathcal{H}}^2$ effectively penalizes high-frequency components in f. Furthermore, larger values of the bandwidth parameter σ correspond to more peaked $v(\boldsymbol{\omega})$, and a corresponding higher penalty on high-frequency components in f.

Now we recall that the function f will eventually be used as a sort of low-dimensional coordinate system or chart along the manifold (see Fig. 3.3), giving the relative positions of the data points along the manifold. Hence, we see that this optimization produces exactly the desired behavior for such a low-dimensional description. Kernel PCA (for this kernel choice) is an algorithm that finds the function f on the data with least high-frequency energy (out of a class of possible f), subject to a constraint that f varies its values on the data by at least a certain amount. Thus, f varies as smoothly as possible along on the manifold while still varying at least somewhat in order to describe position along the manifold. (We note that without the last constraint, the optimal f would be the function that is zero everywhere, so in effect, this constraint steers us away from trivial solution of having the coordinate system be constant along the manifold.) Moreover, the above problem setting imposes a strong penalty on quickly oscillating manifold descriptions and thus helps to avoid overfitting noisy data. To summarize then, the implicit optimization problem being solved by Kernel PCA for this kernel is guaranteed to return the smoothest f that varies adequately on the data.

3.4 Practical Workarounds Increasing the Efficiency of Kernel Methods

Finally, we will discuss important details of implementing kernel-based algorithms and Kernel PCA in particular. One of the main difficulties of using kernel methods is that most algorithms require computation and processing of the full Gram matrix. Indeed in kernel PCA, we need to decompose the $n_{\mathrm{X}} \times n_{\mathrm{X}}$ matrix $\bar{\mathbf{K}}$, which generally takes $\mathcal{O}\left(n_{\mathrm{X}}^3\right)$ operations. This complexity, which grows with the number of samples cubed, may prohibit the use of the direct algorithm on large-scale problems. Furthermore, the found solution is expressed in terms of *all* data samples, when a few might approximate the subspace nearly as well. Both these factors potentially can reduce the algorithm efficiency.

Several approaches were proposed to treat larger datasets with KPCA. Their common idea is to construct the solution iteratively using smaller data batches for each update. This is achieved, for example, by kernelizing the generalized Hebbian algorithm [92, 114] or relying on the incremental kernel SVD as its foundation [47, 48]. We will focus on this latter approach to speed up our KPCA-based manifolds intersection method in Chapter 4 and will review it in detail in Subsection 3.4.2. Inspired by these ideas, we will also develop a similar incremental extension for our novel kernel-based manifold learning algorithm in Chapter 6.

Let us start, however, with the problem of constructing reduced set expansions for more efficient subspace representation in kernel feature spaces. In the next subsection, we will outline the greedy approach for this as a successful method that will be used in our experiments later.

3.4.1 Reduced Set Expansion of the Solution

In kernel methods, solutions are typically expressed as linear combinations of images of all training samples; for example, a principal component vector in a kernel feature space is found by KPCA as $\mathbf{u} = \sum_{i=1}^{n_{\mathrm{X}}} \boldsymbol{\alpha}_i \Phi\left(\mathbf{x}_i\right)$. Such representations are often highly redundant for large number of samples (consider, for example, a case when $D_{\mathcal{H}} < n_{\mathrm{X}}$ and the vectors $\Phi\left(\mathbf{x}_i\right)$ in the feature space are linearly dependent). This inefficiency eventually leads to slower inference. Many reduced-set methods, such as those proposed in [29, 48, 79, 171, 174], aim to find parsimo-

nious expressions for the solution in terms of fewer (possibly different) expansion vectors $\widetilde{\mathbf{x}}_j$ and corresponding coefficients $\widetilde{\boldsymbol{\alpha}}_j$, $j = 1, \ldots, m$, with a goal of minimizing the representation error $E = \|\mathbf{r}\|_{\mathcal{H}}^2 = \left\|\Phi\left(\widetilde{\mathbf{X}}\right)\widetilde{\boldsymbol{\alpha}} - \Phi\left(\mathbf{X}\right)\boldsymbol{\alpha}\right\|_{\mathcal{H}}^2$. Once the new expansion vectors are selected, the optimal values of updated coefficients can be found as $\widetilde{\boldsymbol{\alpha}} = \mathbf{K}_{\widetilde{\mathrm{X}}\widetilde{\mathrm{X}}}^{-1}\mathbf{K}_{\widetilde{\mathrm{X}}\mathrm{X}}\boldsymbol{\alpha}$.

We want to emphasize that the methods described in this work (both, KPCA and our new KODA proposed later in Chapter 6) virtually admit any desired reduced set expansion technique. In our experiments, however, we use sparse greedy approximation of feature space vectors [174], whose main idea is to iteratively choose expansion vectors from the set of given samples while achieving the largest error decrease when selecting each of them, i.e. choosing $\widetilde{\mathbf{x}}_j$ that maximizes $\frac{\langle \mathbf{r}, \Phi(\widetilde{\mathbf{x}}_j)\rangle_{\mathcal{H}}}{\|\mathbf{r}\|_{\mathcal{H}}\|\Phi(\widetilde{\mathbf{x}}_j)|_{\mathcal{H}}}$. We outline this algorithm in Appendix C.

3.4.2 Incremental Algorithm for Efficient Solution Update

The applicability of algorithms that rely on manipulations of Gram matrices, whose size rapidly grows with the number of samples, is inherently limited to relatively small datasets. Indeed, in KPCA, the eigendecomposition of the kernel matrix $\mathbf{K}$ computed using all n_{X} data samples may quickly become infeasible, as its complexity generally grows as $\mathcal{O}\left(n_{\mathrm{X}}^3\right)$. However, as noted above, all samples are often not needed to express the final result with an acceptable accuracy. Thus, such solutions may be gradually updated using smaller batches of data incrementally rather than using an entire dataset at once.

In this section, we will review one such incremental approach to the problem of Gram matrix eigendecomposition. We focus on this specific operation, as it is the pivotal point in terms of complexity in both KPCA and our KODA algorithm proposed in Chapter 6 and largely determines the overall efficiency of their implementation. Specifically, to perform this eigendecomposition step, we will use the incremental updating algorithm of Chin and Suter [48]. Although this method in general involves eigendecomposition of a centered Gram matrix, we will also use a special uncentered version of it when needed in Chapter 6.

As an overview, the algorithm [48] starts with the usual eigendecomposition of a small Gram

matrix. When new data points arrive, instead of decomposing a larger combined matrix, it simply updates the existing estimates of the eigenvectors by rotating them accordingly. Compressing the representation using a small number of expansion vectors (see Section 3.4.1) keeps the memory usage low throughout the iterations.

To facilitate the discussion and simplify the notation, we present the derivation of the algorithm in the input rather than in the feature space. We note, however, that all data samples appear in the final result exclusively in the form of inner products, and thus, for the purpose of kernelizing the algorithm, they can be effectively treated throughout the derivation as images in $\mathcal{H}$.

Let $\mathbf{X}_1$ denote a $D \times n_1$ matrix whose columns are the initial data samples available on the first step of the algorithm. The corresponding sample mean is then $\mathbf{m}_1 = \mathbf{X}_1\boldsymbol{\varepsilon}_1$. Here $\boldsymbol{\varepsilon}_1$ stands for an $n_1 \times 1$ vector of coefficients; typically one assumes $\boldsymbol{\varepsilon}_1 = \frac{1}{n_1}\mathbb{1}$, but we generalize it here to allow for a reduced set expansion of the mean vector to be used as well.

The centered Gram matrix is defined as $\bar{\mathbf{K}}_{11} = \bar{\mathbf{X}}_1^{\mathrm{T}}\bar{\mathbf{X}}_1$, where $\bar{\mathbf{X}}_1 = \mathbf{X}_1 - \mathbf{m}_1\mathbb{1}^{\mathrm{T}}$ denotes the matrix of centered data samples. It can be shown that $\bar{\mathbf{K}}_{11} = \left[\mathbf{I} - \mathbb{1}\boldsymbol{\varepsilon}_1^{\mathrm{T}}\right]\mathbf{K}_{11}\left[\mathbf{I} - \boldsymbol{\varepsilon}_1\mathbb{1}^{\mathrm{T}}\right]$ with $\mathbf{K}_{11} = \mathbf{X}_1^{\mathrm{T}}\mathbf{X}_1$. Let $\boldsymbol{\alpha}_{1^r}$ be a matrix of the first r eigenvectors associated with the largest eigenvalues in the eigendecomposition $\bar{\mathbf{K}}_{11} = \boldsymbol{\alpha}_1\boldsymbol{\Sigma}_1^2\boldsymbol{\alpha}_1^{\mathrm{T}}$. We note that this factorization allows us to write the rank-r SVD representation of the centered data as:

$$\begin{aligned}\bar{\mathbf{X}}_{1^r} &= \left[\bar{\mathbf{X}}_1\boldsymbol{\alpha}_{1^r}\boldsymbol{\Sigma}_{1^r}^{-1}\right]\left[\boldsymbol{\Sigma}_{1^r}\right]\left[\boldsymbol{\alpha}_{1^r}\right]^{\mathrm{T}} \\ &= \left[\mathbf{X}_1\left(\mathbf{I} - \boldsymbol{\varepsilon}_1\mathbb{1}^{\mathrm{T}}\right)\boldsymbol{\alpha}_{1^r}\boldsymbol{\Sigma}_{1^r}^{-1}\right]\left[\boldsymbol{\Sigma}_{1^r}\right]\left[\boldsymbol{\alpha}_{1^r}\right]^{\mathrm{T}} = \mathbf{U}_{1^r}\boldsymbol{\Sigma}_{1^r}\mathbf{V}_{1^r}^{\mathrm{T}}. \end{aligned} \tag{3.9}$$

Now, suppose that n_2 new data samples $\mathbf{X}_2$ become available. Our goal is to find the eigenvectors $\boldsymbol{\alpha}_{2^r}$ of the Gram matrix of the combined and centered dataset, $\bar{\mathbf{X}}_{12} = [\mathbf{X}_1\ \mathbf{X}_2] - \mathbf{m}_{12}\mathbb{1}^{\mathrm{T}}$, where the updated mean is $\mathbf{m}_{12} = \mathbf{X}_{12}\boldsymbol{\varepsilon}_2$, and

$$\boldsymbol{\varepsilon}_2 = \frac{1}{n_1 + n_2}\begin{bmatrix} n_1\boldsymbol{\varepsilon}_1 \\ \\ \mathbb{1}_{n_2} \end{bmatrix}. \tag{3.10}$$

One may attempt to solve this problem directly by explicitly constructing and diagonalizing the combined Gram matrix $\bar{\mathbf{K}}_{12} = \bar{\mathbf{X}}_{12}^{\mathrm{T}}\bar{\mathbf{X}}_{12}$. However, this essentially discards the already found

factorization of $\bar{\mathbf{K}}_{11}$, which may be useful for this purpose. Instead, we will obtain an approximate decomposition for the combined dataset based on the given low-rank representation of the initial data, $\mathbf{X}_{1^r}$. For this, we will adjust the eigenvectors $\boldsymbol{\alpha}_{1^r}$ by effectively rotating them to account for the new information present in $\mathbf{X}_2$.

We will achieve the above goal by invoking incremental SVD [21] on the sample covariance matrix to find a basis for the column space of the centered data (which is equivalent to the sought eigenspace of its centered Gram matrix). To construct the scaled covariance matrix of the combined dataset, we use the result of Ross et al. [165], who have shown that for a concatenated data matrix $\mathbf{X}_{12} = [\mathbf{X}_1\ \mathbf{X}_2]$, $cov\left(\mathbf{X}_{12}\right) = cov\left(\mathbf{X}_1\right) + cov\left(\mathbf{X}_2\right) + \frac{n_1 n_2}{n_1+n_2}\left(\mathbf{m}_1 - \mathbf{m}_2\right)\left(\mathbf{m}_1 - \mathbf{m}_2\right)^{\mathrm{T}}$, where $cov\left(\mathbf{X}_1\right)$ and $cov\left(\mathbf{X}_2\right)$ denote (centered) covariance matrices, and $\mathbf{m}_1$ and $\mathbf{m}_2$ stand for the mean vectors of the first and second parts of the dataset respectively. By approximating $\mathbf{X}_1$ with its rank-r representation, we define $\mathbf{X}_{1^r 2} = [\mathbf{X}_{1^r}\ \mathbf{X}_2]$, $\bar{\mathbf{X}}_{1^r 2} = [\mathbf{X}_{1^r}\ \mathbf{X}_2] - \mathbf{m}_{12}\mathbb{1}^{\mathrm{T}}$ and finally obtain:

$$cov\left(\mathbf{X}_{1^r 2}\right) = \bar{\mathbf{X}}_{1^r 2}\bar{\mathbf{X}}_{1^r 2}^{\mathrm{T}} = \left[\begin{array}{c:c} \bar{\mathbf{X}}_{1^r} & \mathbf{X}_{12}\boldsymbol{\gamma} \end{array}\right]\left[\begin{array}{c:c} \bar{\mathbf{X}}_{1^r} & \mathbf{X}_{12}\boldsymbol{\gamma} \end{array}\right]^T,$$

where $\boldsymbol{\gamma}$ is an $(n_1 + n_2) \times (n_2 + 1)$ matrix defined as:

$$\boldsymbol{\gamma} = \begin{bmatrix} 0_{n_1 \times n_2} & \sqrt{\frac{n_1 n_2}{n_1+n_2}}\boldsymbol{\varepsilon}_1 \\ \mathbf{I} - \frac{1}{n_2}\mathbb{1}_{n_2}\mathbb{1}_{n_2}^{\mathrm{T}} & -\frac{1}{n_2}\sqrt{\frac{n_1 n_2}{n_1+n_2}}\mathbb{1}_{n_2} \end{bmatrix}. \tag{3.11}$$

Clearly, the bottom left block of $\boldsymbol{\gamma}$ centers $\mathbf{X}_2$ with respect to its own mean, while the rightmost column serves to compute the weighted outer product of the mean vectors' difference.

Now we are ready to use the incremental SVD algorithm [21] to decompose the combined and centered dataset $\bar{\mathbf{X}}_{1^r 2}$ similarly to the low-rank factorization of $\bar{\mathbf{X}}_{1^r}$ in Eq. 3.9. Specifically, we are going to find the directions in which the newly arrived data will rotate the existing estimate of singular vectors.

Consider decomposing the new data into its components in the subspace spanned by the singular vectors $\mathbf{U}_{1^r}$ and in that subspace's orthogonal compliment: $\mathbf{X}_{12}\boldsymbol{\gamma} = \mathbf{U}_{1^r}\mathbf{U}_{1^r}^{\mathrm{T}}\mathbf{X}_{12}\boldsymbol{\gamma} + \mathbf{U}_{1^r}^{\perp}\left(\mathbf{U}_{1^r}^{\perp}\right)^{\mathrm{T}}\mathbf{X}_{12}\boldsymbol{\gamma}$. Since, in general, the orthogonal compliment subspace may have dimension

higher than $span\left[\mathbf{U}_{1^r}^{\perp}\left(\mathbf{U}_{1^r}^{\perp}\right)^{\mathrm{T}}\mathbf{X}_{12}\boldsymbol{\gamma}\right]$, e.g. it can be infinite-dimensional in the cases of certain kernels, we may consider an equivalent partial basis $\mathbf{J}$ for it; by definition, $\mathbf{U}_{1^r}^{\mathrm{T}}\mathbf{J}=\mathbb{0}$. The dimension of $\mathbf{J}$ is $r_{\mathrm{M}} \leq (n_2+1)$, which is equal to the (numerical) rank of a certain matrix $\mathbf{M}$ defined later. Finally, we write: $\mathbf{X}_{12}\boldsymbol{\gamma} = \mathbf{U}_{1^r}\mathbf{L} + \mathbf{J}\mathbf{P}$, where $\mathbf{L} = \mathbf{U}_{1^r}^{\mathrm{T}}\mathbf{X}_{12}\boldsymbol{\gamma}$ and $\mathbf{P} = \mathbf{J}^{\mathrm{T}}\mathbf{X}_{12}\boldsymbol{\gamma}$ are the matrices of projection coefficients onto the singular vectors subspace and its orthogonal compliment respectively. With these definitions, the combined dataset can now be factorized as:

$$\bar{\mathbf{X}}_{1^r2} = \begin{bmatrix} \mathbf{U}_{1^r} & \mathbf{J} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma}_{1^r} & \mathbf{L} \\ \mathbb{0}_{r_{\mathrm{M}}\times r} & \mathbf{P} \end{bmatrix} \begin{bmatrix} \mathbf{V}_{1^r} & \mathbb{0}_{n_1\times r_{\mathrm{M}}} \\ \mathbb{0}_{r_{\mathrm{M}}\times r} & \mathbf{I}_{r_{\mathrm{M}}} \end{bmatrix}^{\mathrm{T}}. \tag{3.12}$$

Using the above representation, we can readily write the singular value decomposition of the combined dataset as $\bar{\mathbf{X}}_{1^r2} = \mathbf{U}_2\boldsymbol{\Sigma}_2\mathbf{V}_2^{\mathrm{T}}$. For this, consider the middle matrix in Eq. 3.12, which we denote $\mathbf{F}$, and decompose it with SVD, $\mathbf{F} = \mathbf{U}_F\boldsymbol{\Sigma}_F\mathbf{V}_F^{\mathrm{T}}$. Now the matrices containing the left and right singular vectors of $\bar{\mathbf{X}}_{1^r2}$ are $\mathbf{U}_2 = \begin{bmatrix} \mathbf{U}_{1^r} & \mathbf{J} \end{bmatrix}\mathbf{U}_F$ and $\mathbf{V}_2 = \begin{bmatrix} \mathbf{V}_{1^r} & \mathbb{0}_{n_1\times r_{\mathrm{M}}} \\ \mathbb{0}_{r_{\mathrm{M}}\times r} & \mathbf{I}_{r_{\mathrm{M}}} \end{bmatrix}\mathbf{V}_F$ respectively, and its singular values form the diagonal of $\boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}_F$. The best rank-$r$ approximation is obtained simply by truncating the above matrices, $\bar{\mathbf{X}}_{1^r2^r} = \mathbf{U}_{2^r}\boldsymbol{\Sigma}_{2^r}\mathbf{V}_{2^r}^{\mathrm{T}}$.

At this point, all that's left is to find the partial basis $\mathbf{J}$ and the projection coefficients $\mathbf{L}$ and $\mathbf{P}$ to form the matrix $\mathbf{F}$. We further emphasize that throughout the derivation, one needs to keep in mind that all the operations have to be performed on matrices and vectors expressed in terms of inner products to allow for kernelizing the algorithm later.

One can readily see that the matrix of projection coefficients of $\mathbf{X}_{12}\boldsymbol{\gamma}$ onto the $span\left[\mathbf{U}_{1^r}\right]$ can be expressed entirely in terms of inner products:

$$\begin{aligned} \mathbf{L} &= \mathbf{U}_{1^r}^{\mathrm{T}}\mathbf{X}_{12}\boldsymbol{\gamma} = \left[\mathbf{X}_1\left(\mathbf{I}-\boldsymbol{\varepsilon}_1\mathbb{1}^{\mathrm{T}}\right)\boldsymbol{\alpha}_{1^r}\boldsymbol{\Sigma}_{1^r}^{-1}\right]^{\mathrm{T}}\mathbf{X}_{12}\boldsymbol{\gamma} \\ &= \boldsymbol{\Sigma}_{1^r}^{-1}\boldsymbol{\alpha}_{1^r}^{\mathrm{T}}\left(\mathbf{I}-\mathbb{1}\boldsymbol{\varepsilon}_1^{\mathrm{T}}\right)\begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \end{bmatrix}\boldsymbol{\gamma}. \end{aligned} \tag{3.13}$$

Now the components contained in the new batch of data in the directions orthogonal to the subspace $\mathbf{U}_{1^r}$ (i.e. lying in its orthogonal compliment $\mathbf{U}_{1^r}^{\perp}$) become $\mathbf{H} = \mathbf{X}_{12}\boldsymbol{\gamma} - \mathbf{U}_{1^r}\mathbf{L}$. For

convenience, we define $\mathbf{H} = \mathbf{X}_{12}\boldsymbol{\eta}$ with

$$\boldsymbol{\eta} = \boldsymbol{\gamma} - \begin{bmatrix} \left(\mathbf{I} - \boldsymbol{\varepsilon}_1 \mathbb{1}^{\mathrm{T}}\right) \boldsymbol{\alpha}_{1^r} \boldsymbol{\Sigma}_{1^r}^{-1} \mathbf{L} \\ \\ \mathbb{0}_{n_2 \times (n_2+1)} \end{bmatrix}, \tag{3.14}$$

which is a matrix of size $(n_1 + n_2) \times (n_2 + 1)$.

In general, one may find a (partial) basis $\mathbf{J}$ for $\mathbf{U}_{1^r}^{\perp}$ by orthogonalizing $\mathbf{H}$, for example using the Gram-Schmidt process. However, this will require explicitly working in the ambient space of $\mathbf{H}$, which we aim to avoid in order to extend the algorithm to kernel feature spaces. Thus, instead of orthogonalizing $\mathbf{H}$ in feature space, we form an equivalent basis using the left singular vectors found by SVD similarly to Eq. 3.9. For this, we define a symmetric $(n_2 + 1) \times (n_2 + 1)$ positive semi-definite matrix $\mathbf{M}$ expressed entirely in terms of inner products as:

$$\mathbf{M} = \mathbf{H}^{\mathrm{T}}\mathbf{H} = \boldsymbol{\eta}^{\mathrm{T}} \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \boldsymbol{\eta}. \tag{3.15}$$

Next, we decompose it as $\mathbf{M} = \mathbf{Q}_{\mathrm{M}} \boldsymbol{\Delta}_{\mathrm{M}} \mathbf{Q}_{\mathrm{M}}^{\mathrm{T}}$ and retain all non-zero eigenvalues in the diagonal of $\boldsymbol{\Delta}_{\mathrm{M}}$ along with corresponding eigenvectors $\mathbf{Q}_{\mathrm{M}}$; we then form $\mathbf{J} = \mathbf{X}_{12}\boldsymbol{\eta}\mathbf{Q}_{\mathrm{M}}\boldsymbol{\Delta}_{\mathrm{M}}^{-1/2}$, $\mathbf{P} = \boldsymbol{\Delta}_{\mathrm{M}}^{1/2}\mathbf{Q}_{\mathrm{M}}^{\mathrm{T}}$.

Finally, we find and output the estimate of the eigenvectors of $\bar{\mathbf{K}}_{12}$:

$$\boldsymbol{\alpha}_{2^r} = \begin{bmatrix} \left(\mathbf{I} - \boldsymbol{\varepsilon}_1 \mathbb{1}^{\mathrm{T}}\right) \boldsymbol{\alpha}_{1^r} \boldsymbol{\Sigma}_{1^r}^{-1} & \boldsymbol{\eta}\mathbf{Q}_{\mathrm{M}}\boldsymbol{\Delta}_{\mathrm{M}}^{-1/2} \\ \\ \mathbb{0}_{n_2 \times r} \end{bmatrix} \mathbf{U}_{F^r} \boldsymbol{\Sigma}_{2^r}, \tag{3.16}$$

and also the corresponding singular values $\boldsymbol{\Sigma}_{2^r} = \boldsymbol{\Sigma}_{F^r}$.

Note that for the eigendecomposition of uncentered Gram matrices, $\boldsymbol{\varepsilon}_1 = \mathbb{0}$ and the above definitions reduce to: $\boldsymbol{\gamma} = \begin{bmatrix} \mathbb{0}_{n_1 \times n_2} \\ \\ \mathbf{I}_{n_2} \end{bmatrix}$, $\mathbf{L} = \boldsymbol{\Sigma}_{1^r}^{-1} \boldsymbol{\alpha}_{1^r}^{\mathrm{T}} \mathbf{K}_{12}$, $\mathbf{H} = \mathbf{X}_2 - \mathbf{U}_{1^r}\mathbf{L} = \mathbf{X}_{12}\boldsymbol{\eta}$ with

$$\boldsymbol{\eta} = \boldsymbol{\gamma} - \begin{bmatrix} \boldsymbol{\alpha}_{1^r} \boldsymbol{\Sigma}_{1^r}^{-1} \mathbf{L} \\ \\ 0_{n_2 \times (n_2+1)} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{\alpha}_{1^r} \boldsymbol{\Sigma}_{1^r}^{-1} \mathbf{L} \\ \\ \mathbf{I}_{n_2} \end{bmatrix}. \tag{3.17}$$

Finally, $\mathbf{M} = \mathbf{K}_{22} - \mathbf{L}^{\mathrm{T}}\mathbf{L}$. The entire algorithm for both cases is summarized in Appendix B.

To conclude, this procedure turns established batch algorithms that are based on matrix eigendecomposition into those capable of working with streams of data. In fact, it is readily applicable unchanged for incremental implementation of KPCA [48].

Its efficiency, however, is not readily obvious. Potentially, the most computationally demanding operations here are the needed decompositions of the matrices $\mathbf{M}$ and $\mathbf{F}$, which scale as $\mathcal{O}\left(n_2^3\right)$ and $\mathcal{O}\left(\left(r + r_{\mathrm{M}}\right)^3\right)$ respectively. Fortunately, these values are very small in practice relative to the overall size of the dataset making the incremental approach attractive. Indeed, the number of new samples n_2 added on each step can potentially be made as small as desired, and $r + r_{\mathrm{M}}$ is typically low if a chosen kernel well approximates the manifold with a low-dimensional subspace in feature space. Detailed analysis of the algorithm's complexity can be found in [48].

We further emphasize that the main advantage of using the incremental updating procedure instead of a single-pass algorithm is its potential memory efficiency. In its present form, the final result (the coefficients expressed by Eq. 3.10 and Eq. 3.16) is still formulated in terms of *all* old and new $n_1 + n_2$ data points, which is equivalent to running the batch algorithm on the entire dataset. However, very often in practice it is possible to compress such representations and keep the number of expansion coefficients constant throughout the iterations, thus maintaining low memory usage. As noted before, this can be easily done with the greedy approach of Section 3.4.1 that proved useful in our experiments, although any other available method is applicable for this.

3.5 Conclusion

Kernel methods have become an important framework for obtaining non-linear extensions of established linear learning algorithms. Specifically, Kernel PCA effectively provides an approximate manifold linearization in the feature space expressed in a convenient tractable form. Even though the accuracy of this representation to the true manifold inherently depends on the chosen kernel and parameters, there are certain theoretical guarantees that establish what optimal qualities the final manifold representation will have for a given choice of kernel and parameters. Furthermore, the

preimage problem, i.e. the final step of signal reconstruction in kernel methods, can be effectively addressed with modern preimage solvers and methods such as robust KPCA. Finally, additional techniques, such as reduced set expansions and incremental updating schemes can be used to efficiently extend the methods to larger datasets.

To summarize, while not necessarily exact, the kernel-based approach provides a very useful class of models and algorithms that allow one to quickly obtain good approximate solutions for many difficult non-linear problems. For example, in our problem of many intersecting manifolds, which can be extremely hard, if even possible, to solve exactly, linearization of manifolds with kernel PCA has a potential of offering a tremendous advantage, as we will see in the upcoming chapters.

Specifically, in the next chapter, we will work with the KPCA solution directly in the feature space and will use it as a starting point for kernelizing a popular algorithm of projections onto convex sets for finding their intersections. We will further improve upon this method, particularly for image processing, in Chapter 5, where we will again use the KPCA-found subspace but now as useful means to build an approximator for the distance to a manifold in the input space.

Chapter 4

A Closed-form Approximate Solution to the Problem of Intersecting Manifolds

This chapter is dedicated to setting up the general manifolds intersection problem. We will see that it is of particular importance to patch-based image processing. More specifically, if we assume that individual image patches can be accurately modeled as lying on or close to a non-linear manifold, then we will see that applying this to many overlapping patches imposes mutual constraints on the shared pixels and leads to a specific arrangement of the corresponding patch-manifold constraints in the image space. Hence, under these assumptions, an entire image honoring these patch constraints must be located at the intersection of all such manifolds.

Unfortunately, finding the exact description of the intersection of many non-linear manifolds, for example, for the purpose of mapping a point onto it, is a notoriously difficult task. Recognizing this, we will present a first efficient method to quickly find its approximate solution. For this, we will model the manifolds as linear subspaces in a kernel-induced feature space and will find their intersection with the Projection onto Convex Sets algorithm. We will derive a closed form solution in Section 4.3 and present applications of our model to patch-based image processing and extrapolation of the set of facial images in Section 4.4.

4.1 Model Description

In this section, we will formulate our intersecting manifolds model for images and provide a simple toy example to motivate and explain it. We will see that it follows naturally from the assumption that each overlapping patch is drawn from an underlying manifold.

To introduce the model, let us first consider D-pixel images, i.e. those belonging to the space $\mathbb{R}^D$. For any $p \times q$-area of the image pixel grid, there is a corresponding d-dimensional subspace of $\mathbb{R}^D$ with $d = pq$. The manifold model for a single $p \times q$ patch allows us to assume that such a patch lies on or close to a $d_{\mathcal{M}}$-dimensional non-linear manifold $\mathcal{M}$ (with $d_{\mathcal{M}} < d$) within this subspace. At the same time, the other $D - d$ pixels of the image are unconstrained by this patch, so the whole image is allowed to lie on a $(D - d) + d_{\mathcal{M}}$-dimensional manifold $\mathcal{M}_m \subset \mathbb{R}^D$. Since there is one such manifold constraint corresponding to each of M overlapping patches, the image itself lies on or close to the intersection of all these manifolds $\mathcal{M}_m$, $m = 1, \ldots, M$.

As an illustration, consider the toy example of an image with only three pixels. Such an image can be regarded as a combination of two overlapping 2-pixel patches as shown in the leftmost panel of Fig. 4.1. For the purpose of this example, suppose that each patch is restricted to lie on some 1-dimensional manifold, e.g. a unit circle in $\mathbb{R}^2$. Note that this is equivalent to constraining the whole image to lie on the side of a cylinder in $\mathbb{R}^3$. Thus, all images that conform to this model lie on the intersection of both cylinders and solve the system of non-linear equations $\begin{cases} x_1^2 + x_2^2 = 1 \\ x_2^2 + x_3^2 = 1 \end{cases}$.

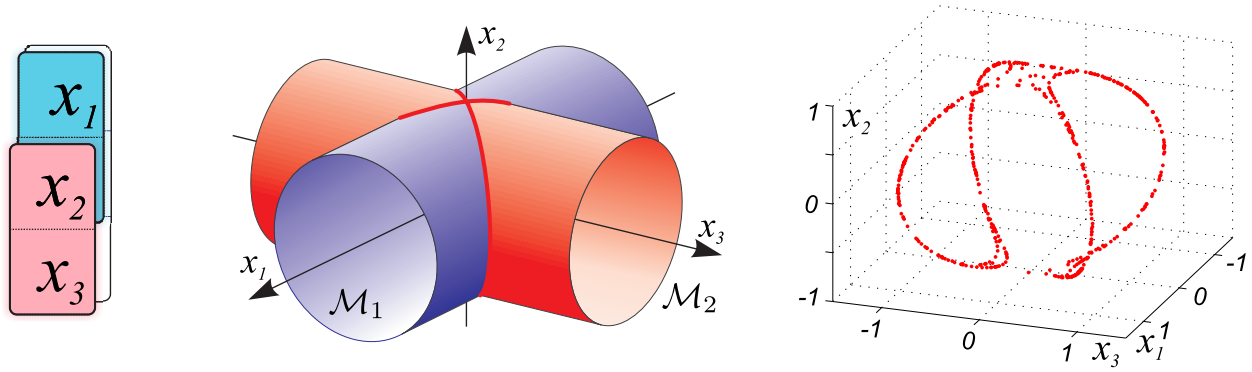


Figure 4.1: Left: Covering a three-pixel image with two overlapping patches. Center: Two cylinders in $\mathbb{R}^3$ created by constraining each of the image patches to lie on the unit circle. Right: The result of using our algorithm to map randomly-generated points (not shown) close to the nearest points on the manifolds' intersection (see Section 5.5.1).

The problem of constraining every patch of the image to come from a manifold model is then the problem of constraining the entire image to lie at the intersection of many patch manifolds.

However, this is a very difficult problem to solve exactly. In the next section, we will look at how to use kernel methods to efficiently approximate the solution to this hard problem.

Our main idea will be to employ the kernel trick to implicitly approximate our nonlinear manifolds as affine subspaces in the feature space $\mathcal{H}$. We will then use a fast algorithm for finding intersections of many affine subspaces, the Projection onto Convex Sets (POCS) algorithm, to quickly find the nearest point on their intersection in feature space; its preimage in the original space is then returned as our solution. We will thus contribute a kernel-methods-variant of POCS as a new method for finding manifolds' intersections.

4.2 Review of the Projections onto Convex Sets Algorithm

Let us continue by reviewing the method of projections onto convex sets, particularly for finding intersections of many affine subspaces. We will then adapt its ideas to efficiently find a point on the intersection of manifolds in the next section.

The general idea of the broad class of POCS algorithms is to find a point on the intersection of the desired convex sets (subspaces $\mathcal{U}_m$ in our case), $\widehat{\mathbf{z}} \in \bigcap_{m=1}^{M} \mathcal{U}_m$, by iteratively constraining the solution to each of them. In our work, we use a simple yet powerful method of parallel projections closely related to the Cimmino's method [50] and the Landweber iterations [52, 194]. Its idea is, starting with some initial guess $\mathbf{z}^{(0)}$, to find the projections $\mathcal{P}_m(\cdot)$ of the current solution onto each subspace $\mathcal{U}_m$ and average them to get the next step approximation:

$$\mathbf{z}^{(k+1)} = \sum_{m=1}^{M} w_m \mathcal{P}_m\left(\mathbf{z}^{(k)}\right), \tag{4.1}$$

where w_m are positive weights satisfying $\sum_{m=1}^{M} w_m = 1$. Note that the subspaces here are allowed to have different dimensions d_m.

The right panel of Fig. 4.2 shows a graphical interpretation of this algorithm for finding the intersection of two lines in $\mathbb{R}^2$. This procedure can be viewed as a combination of the original Cimmino's method [50], which uses reflections instead of projections, and the Kaczmarz's method [109], which circularly projects the current iterate onto each of the subspaces in turn without averaging.

Using projections in our approach will eventually allow us to express the solution in terms of inner products with training samples, which will be crucial for kernelizing the algorithm in Section 4.3.

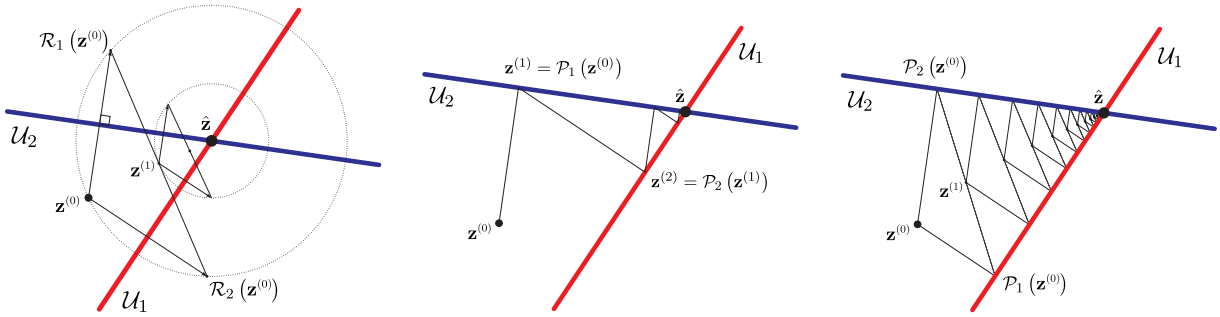


Figure 4.2: Left: The geometric interpretation of the original Cimmino's algorithm. The current iterate $\mathbf{z}^{(k)}$ and its reflections $\mathcal{R}_m\left(\mathbf{z}^{(k)}\right)$ with respect to the subspaces $\mathcal{M}_m$ lie on the hypersphere centered on the intersection set. The center of gravity found by averaging over the reflections converges to the center of the hypersphere. Middle: The method of consecutive projections of Kaczmarz [109]. Right: Finding the intersection of two affine subspaces with the chosen iterative projection algorithm of Eq. 4.1.

At this point, it is instructive to illustrate the convergence of the original Cimmino's algorithm by noting that the starting point at each iteration and the corresponding reflections lie on a hypersphere centered on the intersection of the subspaces (see the left panel in Fig. 4.2). The center of gravity of the reflections, which is effectively computed by averaging, will necessarily fall into this sphere, thus improving the solution at each step. We will discuss the convergence of the parallel projection algorithm of Eq. 4.1 in more detail in the next subsection. For a comprehensive review of feasibility algorithms and their convergence properties, please see, for example, [31] or [52] and references therein.

4.2.1 Convergence Properties of the Iterative Projection Algorithm

In this section, we examine several special cases, namely when the intersection $\bigcap_{m=1}^{M} \mathcal{U}_m$ is empty, contains a single point, or contains infinitely many points; we note that our chosen iterative scheme of Eq. 4.1 converges to a reasonable solution in each of them.

We start by expressing the orthogonal projection of a point $\mathbf{z}$ onto the subspace $\mathcal{U}_m$ as

$$\mathcal{P}_m(\mathbf{z}) = \mathbf{U}_m \mathbf{U}_m^{\mathrm{T}} \mathbf{z} + \left(\mathbf{I} - \mathbf{U}_m \mathbf{U}_m^{\mathrm{T}}\right) \mathbf{m}_m, \tag{4.2}$$

where $\mathbf{U}_m$ is the matrix whose columns form an orthonormal basis for the subspace $\mathcal{U}_m$ after eliminating the offset $\left(\mathbf{I} - \mathbf{U}_m\mathbf{U}_m^{\mathrm{T}}\right)\mathbf{m}_m$.

By substituting Eq. 4.2 into Eq. 4.1, we observe that the next-step iterate can be found as

$$\mathbf{z}^{(k+1)} = \mathbf{A}\mathbf{z}^{(k)} + \mathbf{b} \tag{4.3}$$

with $\mathbf{A} = \sum_{m=1}^{M} w_m \mathbf{U}_m\mathbf{U}_m^{\mathrm{T}}$ and $\mathbf{b} = \sum_{m=1}^{M} w_m(\mathbf{I} - \mathbf{U}_m\mathbf{U}_m^{\mathrm{T}})\mathbf{m}_m$. While obviously related to the Cimmino's algorithm (see Fig. 4.2), our chosen method of parallel projections is nothing but a variant of stationary linear Richardson iterations of the first degree – a popular fast technique for approximating solutions to systems of linear equations [163]. In contrast to the original Richardson's method for a single system, however, we will effectively apply it to solve multiple systems of equations simultaneously; our convergence analysis will nevertheless be similar.

We start by assuming that the intersection of subspaces is not empty, i.e. there is at least one point $\mathbf{z}^*$ that satisfies $\mathbf{z}^* = \mathcal{P}_m\left(\mathbf{z}^*\right)$ for all $m = 1, \ldots, M$, which after rearrangement becomes

$$\left(\mathbf{I} - \mathbf{U}_m\mathbf{U}_m^{T}\right)\mathbf{z}^* = \left(\mathbf{I} - \mathbf{U}_m\mathbf{U}_m^{T}\right)\mathbf{m}_m,\ m = 1, \ldots, M, \tag{4.4}$$

where the offset $\left(\mathbf{I} - \mathbf{U}_m\mathbf{U}_m^{T}\right)\mathbf{m}_m$ is orthogonal to the columns of $\mathbf{U}_m$.

In this case, as shown by Reich [162], our parallel POCS algorithm of Eq. 4.1 converges to the orthogonal projection of the initialization $\mathbf{z}^{(0)}$ onto the set of solutions of the system in Eq. 4.4, i.e. the intersection subspace $\bigcap_{m=1}^{M}\mathcal{U}_m$. Provided that the positive weights w_m sum to 1, their exact values do not affect the result. This is the behavior we will want, for example, for denoising, assuming that the iterations start from the noisy sample. Furthermore, when the intersection of subspaces reduces to a single point, $\mathbf{z}^* = \bigcap_{m=1}^{M}\mathcal{U}_m$, our algorithm will return it regardless of the initialization.

To analyze this result, let us now look closer at the residual error term on the k^{th} iteration of the algorithm. For this, we let $\mathbf{b}_m \triangleq \left(\mathbf{I} - \mathbf{U}_m\mathbf{U}_m^{\mathrm{T}}\right)\mathbf{m}_m$ and consider:

$$\mathbf{z}^{(k)} - \mathbf{z}^* = \sum_{m=1}^{M} w_m \left[\mathbf{U}_m\mathbf{U}_m^{\mathrm{T}}\mathbf{z}^{(k-1)} + \left(\mathbf{I} - \mathbf{U}_m\mathbf{U}_m^{\mathrm{T}}\right)\mathbf{m}_m\right] - \mathbf{z}^*$$

(4.5)

$$\begin{aligned}\mathbf{z}^{(k)} - \mathbf{z}^* &= \sum_{m=1}^{M} w_m \left[\mathbf{U}_m\mathbf{U}_m^{\mathrm{T}}\mathbf{z}^{(k-1)} + \mathbf{b}_m + \mathbf{U}_m\mathbf{U}_m^{\mathrm{T}}\left(\mathbf{z}^* - \mathbf{z}^*\right)\right] - \mathbf{z}^* \\ &= \sum_{m=1}^{M} w_m \left[\mathbf{U}_m\mathbf{U}_m^{\mathrm{T}}\left(\mathbf{z}^{(k-1)} - \mathbf{z}^*\right) + \left(\mathbf{b}_m - \left(\mathbf{I} - \mathbf{U}_m\mathbf{U}_m^{\mathrm{T}}\right)\mathbf{z}^*\right)\right] \\ &= \left[\sum_{m=1}^{M} w_m \mathbf{U}_m\mathbf{U}_m^{\mathrm{T}}\right]\left(\mathbf{z}^{(k-1)} - \mathbf{z}^*\right) = \mathbf{A}\left(\mathbf{z}^{(k-1)} - \mathbf{z}^*\right), \end{aligned} \tag{4.6}$$

where, in the third equality, we used the fact that the weights w_m sum to 1 and, in the last line, the assumption that $\left(\mathbf{I} - \mathbf{U}_m\mathbf{U}_m^{\mathrm{T}}\right)\mathbf{z}^* = \mathbf{b}_m$ for any $m = 1, \ldots, M$. We note that if the difference $\left(\mathbf{z}^{(k-1)} - \mathbf{z}^*\right)$ is aligned with every basis $\mathbf{U}_m$, i.e. if it is in $span\left[\mathbf{U}_m\right]$ for every m, then $\mathbf{U}_m\mathbf{U}_m^{\mathrm{T}}\left(\mathbf{z}^{(k-1)} - \mathbf{z}^*\right) = \mathbf{z}^{(k-1)} - \mathbf{z}^*$ for all m leading to $\mathbf{z}^{(k)} - \mathbf{z}^* = \mathbf{z}^{(k-1)} - \mathbf{z}^*$. This means that the current iterate $\mathbf{z}^{(k-1)}$ is already in the shared subspace, and it will not move.

On the other hand, if the difference $\mathbf{z}^{(k-1)} - \mathbf{z}^*$ is not exactly aligned with the intersection subspace, then $\left\|\mathbf{U}_m\mathbf{U}_m^{\mathrm{T}}\left(\mathbf{z}^{(k-1)} - \mathbf{z}^*\right)\right\| \leq \left\|\mathbf{z}^{(k-1)} - \mathbf{z}^*\right\|$ for every m, but the inequality is strict for at least one of them, thus resulting in $\left\|\mathbf{z}^{(k)} - \mathbf{z}^*\right\| < \left\|\mathbf{z}^{(k-1)} - \mathbf{z}^*\right\|$. So if $\mathbf{z}^{(k-1)}$ is not in the shared subspace, it will move closer to it on the next iteration.

The same strict inequality holds if the matrices $\mathbf{U}_m\mathbf{U}_m^{\mathrm{T}}$ do not share common principal eigenvectors reducing the intersection of the subspaces to a single point $\mathbf{z}^* \neq \mathbf{z}^{(k-1)}$. In this case, we can see by taking norms in Eq. 4.6 that the convergence rate to this unique solution is geometric in the spectral norm of the symmetric matrix $\mathbf{A}$, i.e. $\left\|\mathbf{z}^{(k)} - \mathbf{z}^*\right\| < \|\mathbf{A}\|^k \left\|\mathbf{z}^{(0)} - \mathbf{z}^*\right\|$.

Finally, in the inconsistent case, when the system of Eq. 4.4 has no solution (if it defines, for example, parallel hyperplanes or non-intersecting lines in $\mathbb{R}^3$), our iterations are known to converge to the minimizer of the weighted sum of squared distances to all subspaces [61],

$$\min_{\mathbf{z}} \sum_{m=1}^{M} w_m \left\|\mathbf{z} - \mathcal{P}_m\left(\mathbf{z}\right)\right\|^2. \tag{4.7}$$

We note that the criterion of Eq. 4.7 naturally formalizes our desire of locating a solution on the intersection of M subspaces (or at least close to each of them). Here, non-equal weights w_m may be introduced to make our solution closer to one subspace or another. Notice also that unlike the cyclic method of Kaczmarz, our chosen iterative procedure (Eq. 4.1) results in a relevant solution regardless of whether the system of Eq. 4.4 is under- or overdetermined [52]. Finally, in contrast

to the exact methods (e.g. based on Gaussian elimination), whose complexity is usually cubic in the number of equations in the system, iterations of Eq. 4.3 offer a fast way to find an *approximate* solution that can be refined by considering more steps k, if desired. For a comprehensive review of feasibility algorithms and their convergence properties, please see, for example, [31] or [52] and references therein.

Next, we will apply the principles of the POCS algorithm for finding intersections of non-linear manifolds. Specifically, we will employ kernel PCA to linearize the problem in feature space, which will allow us to compute projections onto the manifold-approximating subspaces easily, iteratively find their intersection, and finally express the solution in closed form.

4.3 Application of the Kernel Trick to the Subspace Intersection Problem

We now wish to extend the machinery for finding intersections of affine subspaces to nonlinear manifolds. We will apply the kernel trick to model each manifold with an affine subspace in feature space. Hence, the problem of finding the intersection of manifolds will be approximated by finding the intersection of subspaces in feature space.

Specifically, we will show that the solution defined by Eq. 4.1 can be written as a combination of powers of certain matrices (which embody the action of each iteration) multiplied by the vectors of the algorithm initialization. It is crucially important that these matrices and vectors can be expressed entirely in terms of inner products, so we may use the kernel trick to derive a non-linear extension of the POCS algorithm. For this, we start by iterating Eq. 4.3 and write the K-th step approximation of the solution as

$$\mathbf{z}^{(K)} = \mathbf{A}^K \mathbf{z}^{(0)} + \sum_{k=0}^{K-1} \mathbf{A}^k \mathbf{b}. \tag{4.8}$$

Now, to derive an inner product form of the above equation, we attempt to learn a description of each subspace $\mathcal{U}_m$ from its n_m training samples $\mathbf{x}_i^{(m)}$ using the PCA algorithm. Following the discussion in Section 3.2.1, the principal components of the m^{th} subspace can be expressed as $\mathbf{U}_m = \mathbf{X}_m \boldsymbol{\alpha}_m$, where $\mathbf{X}_m$ is a $D \times n_m$ matrix of samples arranged in columns and $\boldsymbol{\alpha}_m$ is an

$n_m \times d_m$ matrix of scaled eigenvectors of the centered Gram matrix $\bar{\mathbf{K}}_m$ (please see Chapter 3 for more details). With this parameterization, Eq. 4.8 can be written as a linear combination of training points with coefficients $\boldsymbol{\gamma}_m$ expressed entirely in terms of inner products:

$$\mathbf{z}^{(K)} = \sum_{m=1}^{M} \mathbf{X}_m \boldsymbol{\gamma}_m, \tag{4.9}$$

where each $\boldsymbol{\gamma}_m$ is an $n_m \times 1$ vector.

The derivation of the coefficients $\boldsymbol{\gamma}_m$ involves only algebraic manipulations and its details can be found in Appendix A. Here we just define the following block matrices and vectors, each with entries expressed entirely in terms of inner products:

- the $\sum d_m \times 1$-dimensional vector $\mathbf{h}$ with M block entries $\mathbf{h}_{[i]} = \sqrt{w_i}\boldsymbol{\alpha}_i^{\mathrm{T}}\mathbf{X}_i^{\mathrm{T}}\mathbf{z}^{(0)}$ arranged vertically;
- the $\sum d_m \times 1$-dimensional vector $\mathbf{g}$ with M block entries $\mathbf{g}_{[i]} = \sqrt{w_i}\sum_{m=1}^{M} w_m \boldsymbol{\alpha}_i^{\mathrm{T}}\mathbf{X}_i^{\mathrm{T}}\mathbf{X}_m\boldsymbol{\mu}_m$ arranged vertically, where $\boldsymbol{\mu}_m = \left(\mathbf{I} - \boldsymbol{\alpha}_m\boldsymbol{\alpha}_m^{\mathrm{T}}\mathbf{X}_m^{\mathrm{T}}\boldsymbol{X}_m\right)\frac{1}{n_m}\mathbb{1}$ is a $n_m \times 1$ vector, also computed using only inner products;
- the $\sum d_m \times \sum d_m$-dimensional matrix $\mathbf{H}$ with block entries $\mathbf{H}_{[i,j]} = \sqrt{w_i}\boldsymbol{\alpha}_i^{\mathrm{T}}\mathbf{X}_i^{\mathrm{T}}\mathbf{X}_j\boldsymbol{\alpha}_j\sqrt{w_j}$.

Intuitively, the vector $\mathbf{h}$ bears the meaning of a starting point of the iterations, the vector $\mathbf{g}$ comprises the information about the offsets of each subspace, and the matrix $\mathbf{H}$ describes pairwise relations between different subspaces. Computing powers of $\mathbf{H}$ essentially corresponds to running the iterations of the POCS algorithm.

Finally, we let the vector $\mathbf{s} = \mathbf{H}^{K-1}\mathbf{h} + \sum_{k=1}^{K-1}\mathbf{H}^{k-1}\mathbf{g}$ and denote its m^{th} block of length d_m as $\mathbf{s}_{[m]} = \mathbf{s}_{\left(\sum_{i=1}^{m-1} d_i\right)+1,\ldots,\sum_{i=1}^{m} d_i}$. Now $\boldsymbol{\gamma}_m$ can be expressed as

$$\boldsymbol{\gamma}_m = \sqrt{w_m}\boldsymbol{\alpha}_m\mathbf{s}_{[m]} + w_m\boldsymbol{\mu}_m\,. \tag{4.10}$$

Since computing the vectors $\boldsymbol{\gamma}_m$ involves only evaluation of inner products, this algorithm can be easily extended to the non-linear case by substituting the entries of inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ with corresponding values of the kernel function $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ (and similarly replacing $\left\langle \mathbf{x}_i, \mathbf{z}^{(0)} \right\rangle$ with

$\kappa\left(\mathbf{x}_i, \mathbf{z}^{(0)}\right)$). Therefore, the intersection of subspaces is sought in an implicitly induced higher-dimensional feature space, which corresponds to approximating the intersection of non-linear manifolds in the original space. The preimage $\widehat{\mathbf{z}}$ of the solution $\Phi[\mathbf{z}^{(K)}] = \sum_{m=1}^{M} \sum_{i=1}^{n_m} \Phi(\mathbf{x}_i^{(m)})\boldsymbol{\gamma}_{i,m}$ can thus be found by minimizing the Euclidean distance $\left\|\Phi(\widehat{\mathbf{z}}) - \Phi[\mathbf{z}^{(K)}]\right\|^2$ in feature space. The form of Eq. 4.9 allows one to use any of the preimage methods described in Section 3.2.3.2. The entire procedure is summarized in Algorithm 1.

We note that accurately learning the manifold geometry may require a large number of samples in the training sets $\mathbf{X}_m$, which, in turn, increases the size of the kernel matrices. Furthermore, due to the form of the final solution (Eq. 4.9), this directly affects the running time. To alleviate these problems, as discussed in Section 3.4, one can use one of the incremental KPCA algorithms [48, 100, 115] to iteratively update the parameters of the manifolds on the learning stage using smaller portions of the training dataset. Moreover, sparse (in terms of training data) approximation of the principal components in feature space can be found by any of the reduced set methods [76, 121, 153, 174]. Their main idea is to choose only a very small subset of the training points $\mathbf{x}_i^{(m)}$ such that the span of their images in the feature space well approximates the entire training set. Therefore, the principal components of the subspaces can be expressed as combinations with significantly fewer terms. Fortunately, these costly procedures need to be performed only once. Then the learned model can be reused for solving several problems with different initial conditions (for example, for processing similar images with the manifolds intersection model of overlapping patches).

To conclude, we note that our algorithm can be run with several initial conditions simultaneously by arranging them column-wise in a matrix $\mathbf{Z}^{(0)}$ instead of the vector $\mathbf{z}^{(0)}$, which can further reduce computational burden. We will experimentally evaluate the performance of our manifolds intersection finding algorithm in the next section.

Algorithm 1 Manifolds intersection algorithm

Input: Sets of training samples $\mathbf{x}_i^{(m)}$, $i = 1, \ldots, n_m$ of M manifolds, initial approximation of the solution $\mathbf{z}^{(0)}$, kernel function κ, number of iterations K.

Output: A point $\widehat{\mathbf{z}} \in \mathbb{R}^D$ that minimizes the criterion of Eq. 5.1.

for $m := 1 \ldots M$ **do** ▷ Loop over all manifolds.

  $\mathbf{K}_{i,j}^{(m,m)} \leftarrow \kappa\left(\mathbf{x}_i^{(m)}, \mathbf{x}_j^{(m)}\right)$, $i, j = 1, \ldots, n_m$ ▷ Create kernel matrices.

  $\mathbf{k}_i^{(m,z)} \leftarrow \kappa\left(\mathbf{x}_i^{(m)}, \mathbf{z}^{(0)}\right)$, $i, j = 1, \ldots, n_m$

  $\bar{\mathbf{K}}^{(m,m)} \leftarrow \left(\mathbf{I} - \frac{1}{n_m}\mathbb{1}\,\mathbb{1}^T\right)\mathbf{K}^{(m,m)}\left(\mathbf{I} - \frac{1}{n_m}\mathbb{1}\,\mathbb{1}^T\right)$ ▷ Centering; see [173].

  $[\mathbf{A}_m, \mathbf{\Lambda}_m] \leftarrow \text{EIG}\left(\bar{\mathbf{K}}^{(m,m)}\right)$ ▷ Find the eigendecomposition $\bar{\mathbf{K}}^{(m,m)} = \mathbf{A}_m\mathbf{\Lambda}_m\mathbf{A}_m^T$.

  $\boldsymbol{\alpha}_{:,i}^{(m)} \leftarrow \mathbf{A}_{:,i}^{(m)}\frac{1}{\sqrt{\mathbf{\Lambda}_{i,i}^{(m)}}}$, $i = 1 \ldots d_m$ ▷ Choose d_m leading eigenvectors and scale them.

  $\widetilde{\mathbf{h}}_m \leftarrow \sqrt{w_m}\boldsymbol{\alpha}_m\mathbf{K}^{(m,y)}$

  $\widetilde{\mathbf{g}}_m \leftarrow \mathbb{0}_{d_m}$

  for $l = 1, \ldots, M$ **do**

    $\mathbf{K}_{i,j}^{(m,l)} \leftarrow \kappa\left(\mathbf{x}_i^{(m)}, \mathbf{x}_j^{(l)}\right)$, $i = 1, \ldots n_m$, $j = 1, \ldots n_l$

    $\mathbf{c}_l \leftarrow \left(\mathbf{I} - \boldsymbol{\alpha}_l^{\mathrm{T}}\boldsymbol{\alpha}_l\mathbf{K}^{(l,l)}\right)\frac{1}{n_l}\mathbb{1}_{n_l}$

    $\widetilde{\mathbf{g}}_m \leftarrow \widetilde{\mathbf{g}}_m + \sqrt{w_m}w_l\boldsymbol{\alpha}_m\mathbf{K}^{(m,l)}\mathbf{c}_l$

    $\widehat{\mathbf{H}}_{m,l} \leftarrow \sqrt{w_m}\boldsymbol{\alpha}_m\mathbf{K}^{(m,l)}\boldsymbol{\alpha}_l^{\mathrm{T}}\sqrt{w_l}$

  end for

  $\widetilde{\mathbf{H}}_m \leftarrow \text{CONCATENATE}_{rows}\left(\widehat{\mathbf{H}}_{m,1},\ \widehat{\mathbf{H}}_{m,2},\ \ldots,\ \widehat{\mathbf{H}}_{m,M}\right)$

end for

$\mathbf{h} \leftarrow \text{CONCATENATE}_{columns}\left(\widetilde{\mathbf{h}}_1,\ \widetilde{\mathbf{h}}_2,\ \ldots,\ \widetilde{\mathbf{h}}_M\right)$

$\mathbf{g} \leftarrow \text{CONCATENATE}_{columns}\left(\widetilde{\mathbf{g}}_1,\ \widetilde{\mathbf{g}}_2,\ \ldots,\ \widetilde{\mathbf{g}}_M\right)$

$\mathbf{H} \leftarrow \text{CONCATENATE}_{columns}\left(\widetilde{\mathbf{H}}_1,\ \widetilde{\mathbf{H}}_2,\ \ldots,\ \widetilde{\mathbf{H}}_M\right)$

$\mathbf{s} \leftarrow \mathbf{H}^{K-1}\mathbf{h} + \sum_{k=1}^{K-1}\mathbf{H}^{k-1}\mathbf{g}$

for $m = 1, \ldots, M$ **do**

  $\widetilde{\mathbf{s}}_m \leftarrow \mathbf{s}_{\sum_{i=1}^{m-1}d_i+1\cdots\sum_{i=1}^{m}d_i}$

  $\boldsymbol{\gamma}_m \leftarrow \sqrt{w_m}\boldsymbol{\alpha}_m\widetilde{\mathbf{s}}_m + w_m\mathbf{c}_m$

end for

$\widehat{\mathbf{z}} \leftarrow \underset{\mathbf{z}}{\text{argmin}}\left\|\Phi(\mathbf{z}) - \sum_{m=1}^{M}\sum_{i=1}^{n_m}\Phi(\mathbf{x}_i^{(m)})\boldsymbol{\gamma}_i^{(m)}\right\|^2$ ▷ Find a preimage [101, 119, 174].

Note: In this algorithm, indexes m, l refer to entire matrices or vectors, and i, j denote their scalar entries.

4.4 Experimental Results and Discussion

In this section, we consider several applications of our manifold intersection finding method. First, we will run our algorithm on small synthetic examples to vividly demonstrate that it finds the intersections of smooth curves and surfaces. Then we will show that our method easily extends to more complex manifolds, such as those formed by sequences of smoothly changing images. We will then continue by presenting its application to solving practical problems in signal processing and data analysis. In all examples, unless otherwise stated, we use the Gaussian kernel and obtain the preimage $\widehat{\mathbf{z}}$ of the final solution by minimizing the distance $\left\|\Phi(\widehat{\mathbf{z}}) - \Phi(\mathbf{z}^{(K)})\right\|^2$ in the feature space with the gradient descent approach (see, for example, [29] or [111] for details of this preimage method).

4.4.1 Intersections of Curves and Surfaces

We start by using our algorithm to map clouds of randomly generated points onto intersections of smooth curves in $\mathbb{R}^2$ and surfaces in $\mathbb{R}^3$.

Our first example is finding the intersection of two curves shown in Fig. 4.3. To find this intersection, the curves $\mathcal{M}_1$ and $\mathcal{M}_2$ are learned from 50 samples each using KPCA with the Gaussian kernel of width $\sigma = 1$ and $d_m = 20$ and 30 respectively. In the plots in the left and middle panels of Fig. 4.3, we first apply the special case of our algorithm with $M = 1$ to a cloud of randomly generated points to show that this procedure (corresponding to mapping onto a single manifold) well approximates projections onto each of the desired curves. We use the preimage method of [143] to reconstruct the results. The same random points were then projected onto the intersection of the subspaces with our algorithm. The found solutions land on (or close) to the true intersection of the two curves in the original space as shown in the right panel of Fig. 4.3.

Figure 4.4 demonstrates a similar experiment of mapping a cloud of random points onto the intersections of two non-linear smooth surfaces in $\mathbb{R}^3$. Again, we learn each surface from its 200 samples; we use the Gaussian kernel with $\sigma = 1.1$ and $d_{\mathcal{U}} = 130$ for the cones and $\sigma = 1.2$, $d_{\mathcal{U}} = 120$

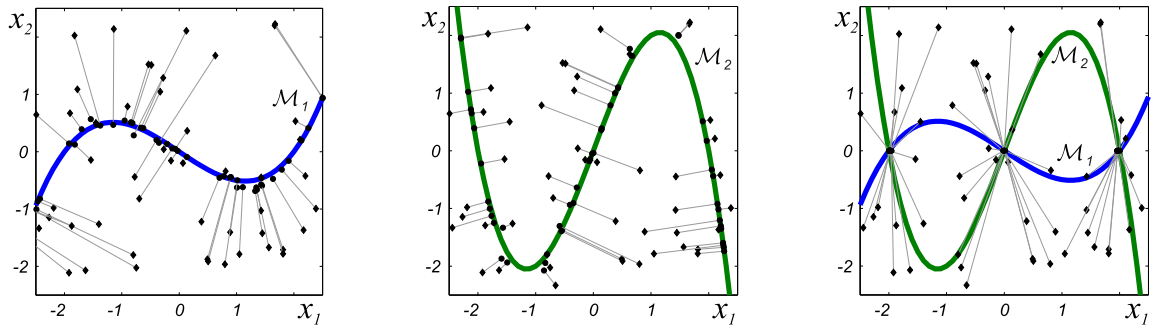


Figure 4.3: Mapping a cloud of randomly generated points onto smooth curves in $\mathbb{R}^2$ separately (left and center) and onto their intersection (right). The ground truth manifolds that we attempt to learn with kernel PCA are shown as the green and blue curves. Points are mapped close to their nearest points on the corresponding manifolds or on the manifolds' intersection.

for the sinusoidal surfaces. The found solutions lie on or close to the real intersection curves, and are also close to the initialization points $\mathbf{Z}^{(0)}$ as we desired.

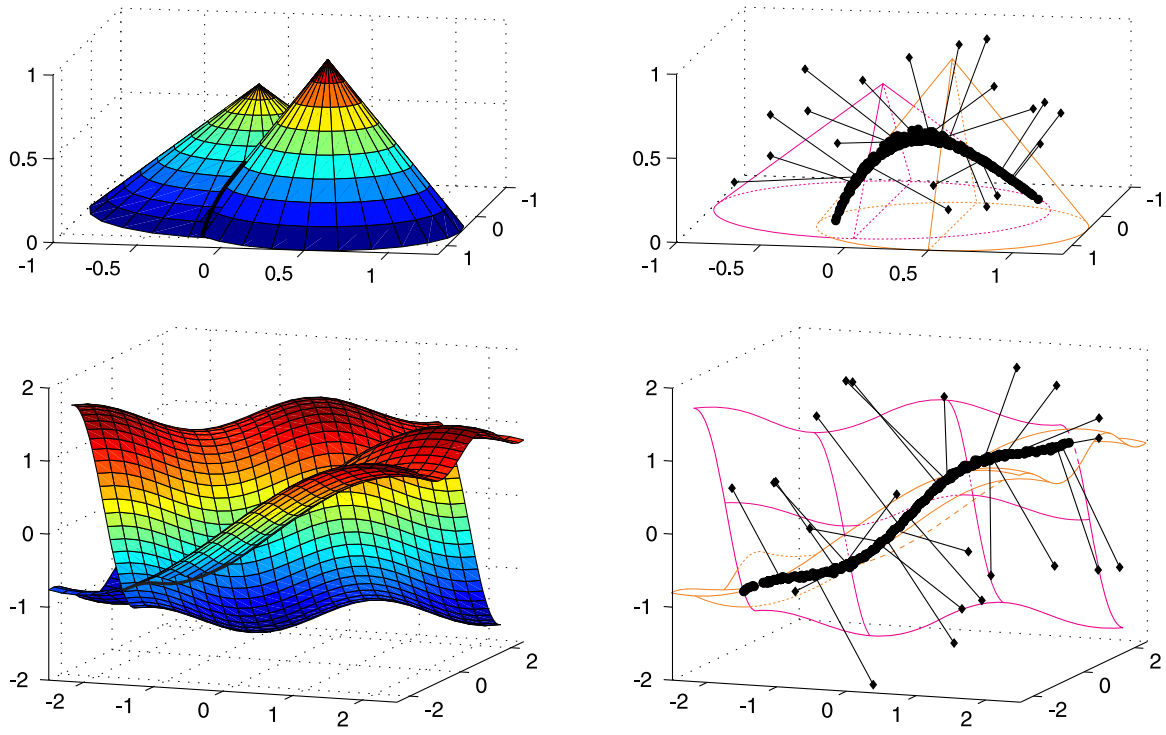


Figure 4.4: Results of finding the intersections of two surfaces in $\mathbb{R}^3$ (left). Notice how randomly generated points are mapped close to the corresponding nearest points on the manifolds and trace the sought intersection curves (right; not all starting points are shown).

4.4.2 Intersections of Image Manifolds

In our next experiments we consider a similar problem of finding intersections of two manifolds of smoothly changing images.

First, we look at the synthetic images of an object viewed from different directions and under varying lighting conditions (see Section 2.3.1). In this experiment, a sequence of 249 images of a bunny was generated with the camera moving around it in the horizontal plane while the lighting source was held fixed above. Next, the lighting source was moved in the vertical direction and another sequence of 99 images was taken with the same frontal view. These two sets of 64×64 images were then used as training samples to learn the underlying manifolds in our model (we used $\sigma = 5 \cdot 10^3$ in the Gaussian kernel and $d_1 = 50$, $d_2 = 20$ for the dimensions of the approximating subspaces). As shown in Fig. 4.5, our randomly initialized algorithm converges to a point on (or very close to) the manifolds' intersection.

Next, we run a similar experiment with a real-world dataset consisting of images of two independently moving objects, which is inspired by similar applications in computer vision. While one of the objects is held fixed in the middle of its trajectory, the other one moves around and then vice versa, thus tracing two one-dimensional manifolds in the image space. The scene is photographed with small movement intervals to ensure good sampling of the underlying manifolds (yielding 56 and 67 training images of size 256×256 pixels for the movements of each object). An image located at the sought intersection using our algorithm with $\sigma = 50 \cdot 10^3$ and $d_1 = d_2 = 35$ is shown in the bottom right corner of Fig. 4.5. As we would expect, these manifolds intersect at the point where both objects are in the middle of their trajectories.

4.4.3 Extrapolation of the Facial Images Dataset

The manifold intersection model can be particularly useful in representing signals that exhibit smooth intra- and inter-subject variations. As an example, we consider modeling and learning the underlying structure of a large set of facial images of different people showing different expressions.

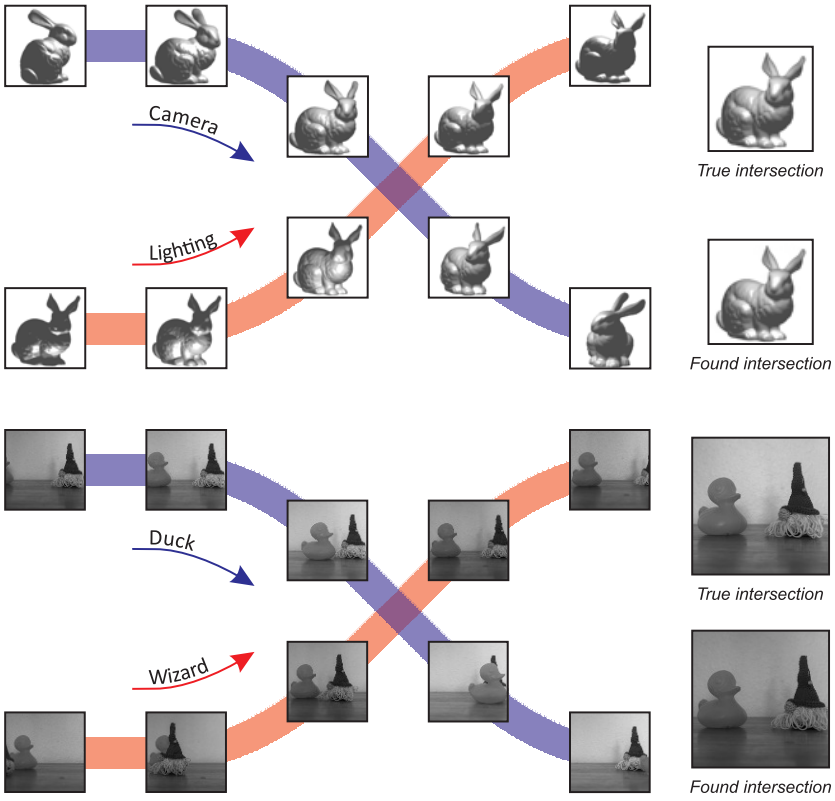


Figure 4.5: A computer-vision-inspired examples of finding intersections of manifolds of changing images with our algorithm. Top: Images of a bunny taken from different positions and with varying lighting conditions. Bottom: Images of two independently moving objects. The left part of the figure shows representative samples of both one-dimensional manifolds in each case. The points on their intersections, as found by our algorithm, are shown on the right. We see that our algorithm returns approximations that are very close to the true intersections. (True intersection images were omitted from training of the manifolds for both examples.)

While each of these aspects (namely, variations in appearance and in expressions) were shown to admit to underlying manifold models fairly well separately [132, 41, 197], modeling the entire set of facial images as a single entity encounters certain difficulties. Specifically, few images of the same person present in a dataset and characterized by subtle but meaningful variations (the emotions they express, or "content") could be easily overshadowed by the large number of significantly different images of the rest of population (which come in different "styles" defined by the appearance of specific people). Thus, important characteristic details of image "content" may be perceived as noise by a manifold learning algorithm and can be lost.

To avoid missing these elusive directions of variability in a large dataset, in our experiments,

we assume that the images of all facial expressions of one particular person, as well as the images of different people with the same expression (e.g. all smiling), lie on *separate* manifolds, which we call intra- and inter-subject manifolds respectively. Images of a specific person smiling belong to the both datasets and thus should appear at the intersection of the two manifolds (see Fig. 4.6). This observation allows us to use our intersecting manifolds model to address the problem of set extrapolation [190], for example, to estimate a smiling image of a specific person as a point on the intersection of the content and style manifolds (see Fig. 4.6).

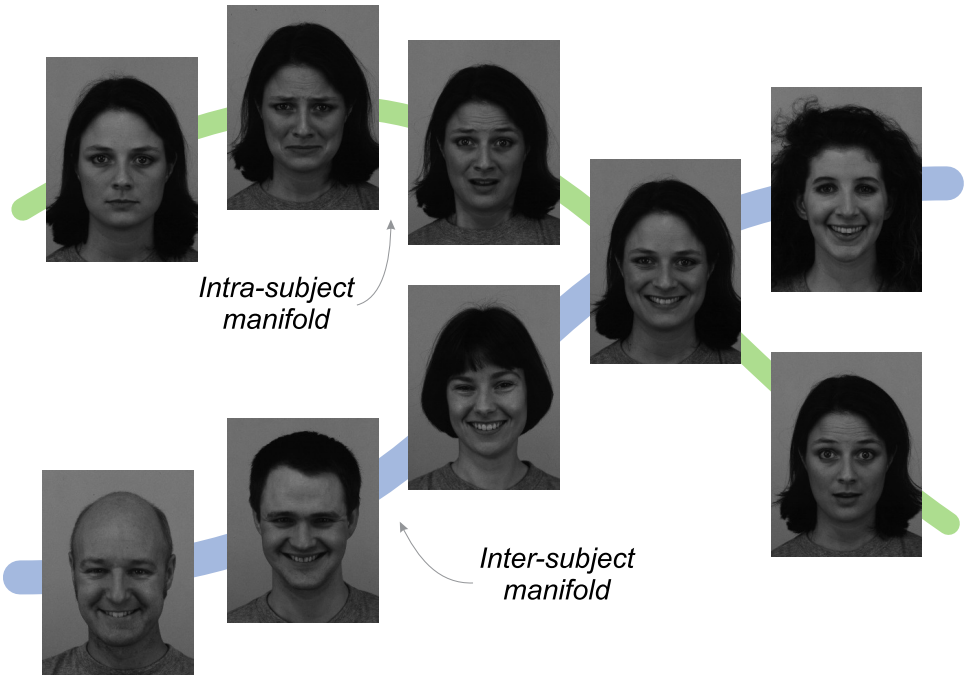


Figure 4.6: A schematic representation of the manifolds of facial images. The inter-subject manifold (showing the same "content" in different "styles") models the set of images of different people with the same facial expression (e.g. smiling). The intra-subject manifold models the set of different expressions of the same person. The subset of the images of a particular person smiling lies on the intersection of the manifolds. These images are examples from the facial expressions database [135].

For this experiment, we use the facial expression database created at the Karolinska Institutet in Sweden [135]. It contains images of seven basic facial expressions (fear, anger, disgust, smile, sadness, surprise, and neutral) each made by 70 actors photographed twice from five different angles (we use only the two frontal view images). The inter-subject manifold of smiling faces is thus learned from a training set of $2 \times (69 - 1) = 138$ images (leaving out the images of the person

we will test on), and the intra-subject manifold is learned from a set of twelve images of all different expressions (except smiling) of one particular person of interest (some examples from the training sets are shown in Fig. 4.6). Respectively, these sets of $762 \times 562-$pixel images are represented with $10-$ and $5-$dimensional subspaces in the feature space induced by the Gaussian kernel with parameter $\sigma = 8 \cdot 10^4$. We use an image of neutral expression (see Fig. 4.7) as a starting point $\mathbf{z}^{(0)}$ in our algorithm. Finally, we obtain a preimage of the solution in the original space using gradient descent initialized with the training sample, on either of the manifolds, that is closest to the solution in the feature space.

The omitted images and their found estimates are shown in the two rightmost columns in Fig. 4.7. Even though the reconstructed images are blurred and may not exactly represent the particular ground truth examples, the found solutions are a reasonable approximation to an image of a smiling face of a given person. Our manifolds intersection algorithm retains distinct personal features (such as the shape of the face and the hairstyle), which might have been lost in the diverse inter-subject set (see Fig. 4.7.c). On the other hand, it obviously introduces prominent attributes of a smile, such as an open mouth and eyes, visible teeth, and the characteristic shapes of cheek wrinkles and eyebrows. Presumably, sampling the manifolds more densely would allow us to minimize the preimage error and achieve better reconstruction quality.

4.4.4 Patch-based Denoising

To demonstrate the use of our algorithm in a practical patch-based image processing application, we describe how it can be applied for solving denoising problems. While we use this example as a proof of concept of the general manifolds intersection finding algorithm presented in this chapter, we emphasize that we will be tailoring our methodology specifically for the problem of patch-based image processing later in the thesis. This will both streamline the process, making it more computationally efficient and easier to train, and allow it to have broad applicability across the wide spectrum of other inverse problems. We will look at this closer in the next chapter.

For now, we can cover a size $P \times Q$ image with size $p \times q$ overlapping patches, assume that

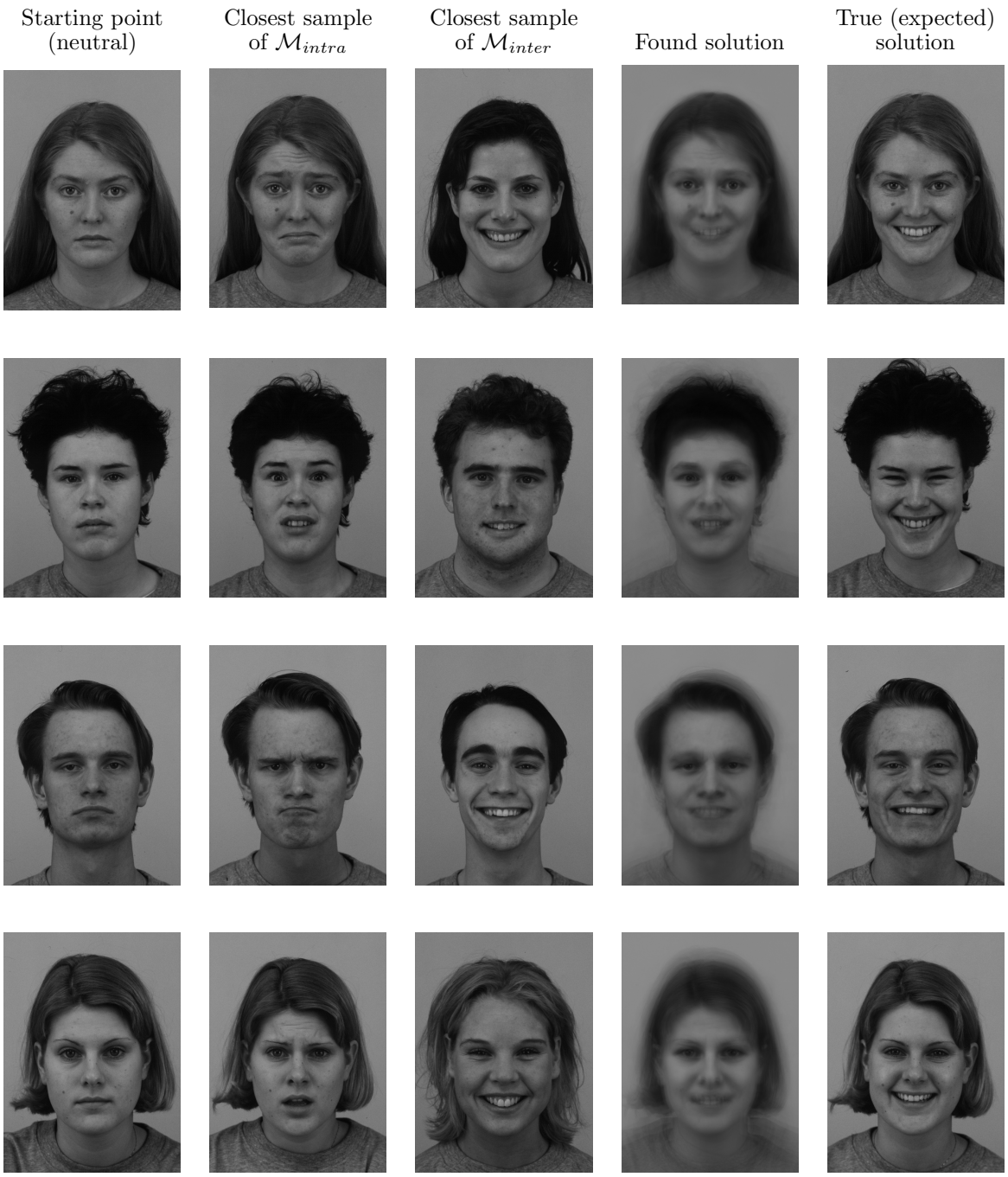


Figure 4.7: Results of approximating images of smiling faces as points on the intersection of manifolds. The iterations are initialized with an image of a neutral face (a). Panels (b) and (c) show the training images closest to the found solution on the intra- and inter-subject manifolds respectively. Our result obtained after solving the preimage problem with the gradient descent method is shown on the panel (d). Notice how the found approximation combines the distinct features of the person with the attributes of a smile. The expected (true) solution is shown on the panel (e). The samples used in this experiment are (from top to bottom): F22, F35, M21, and F26.

each of them corresponds to one manifold constraint on the entire image, and then apply our algorithm directly to map the original image onto the intersection of these manifolds. This will be a particularly useful approach for denoising, since we will be looking for the point closest to the given (noisy) image that admits to our patch model.

To run our algorithm, we would need to learn $M = (P - p + 1)(Q - q + 1)$ different manifold models for the entire PQ-dimensional image, each of dimension $PQ - pq + d_m$. This tactic works well with images of modest size, but may become computationally expensive when their dimensions grow. Therefore, to keep our approach tractable, we will decompose a large image into a set of smaller overlapping $P \times Q$ regions $\mathbf{R}_j$ and will use our algorithm from Section 4.3 to estimate each of them separately. Furthermore, to avoid edge effects, we retain only the central pixels of the (overlapping) estimates $\widehat{\mathbf{R}}_j$, whose union then forms the resulting solution. More specifically, we choose $P < 2p$, $Q < 2q$ and tile each region with the maximum number of patches (M) such that they all overlap in the middle $(2p - P) \times (2q - Q)$-pixel area (please see Fig. 4.8 for an explanatory example).

Moreover, we would like to explicitly ensure that the estimated central area admits to the manifold model. For this, after the final step of iterations, we project each region onto the subspace $\mathcal{U}_{m_c}$ in feature space that corresponds to the central patch and then reconstruct a preimage of this projection. Hence, the final solution is expressed as $\Phi(\mathbf{z}_{m_c}) = \Phi(\mathbf{X}_{m_c})\boldsymbol{\gamma}_{m_c}$, where

$$\boldsymbol{\gamma}_{m_c} = \boldsymbol{\alpha}_{m_c}\boldsymbol{\alpha}_{m_c}^T \sum_{m=1}^{M} \mathbf{K}_{m_c,m}\boldsymbol{\gamma}_m + \boldsymbol{\mu}_{m_c}, \tag{4.11}$$

and coefficients $\boldsymbol{\gamma}_m$ are given by Eq. 4.10 computed for $K \to \infty$.

In particular, in the example shown in Fig. 4.9, we cover each 9×9 image region with 25 overlapping patches of size 5×5. Thus, our algorithm determines the single central pixel in each region from the values of its 80 neighboring pixels. Proceeding in the same way for all pixels in $\mathbf{I}_{noisy}$, we find the denoised image $\widehat{\mathbf{I}}$. A more elegant solution that jointly estimates the whole image will be given in the next chapter.

To learn the manifolds in our experiments, we use training sets of $p \times q$ patches extracted

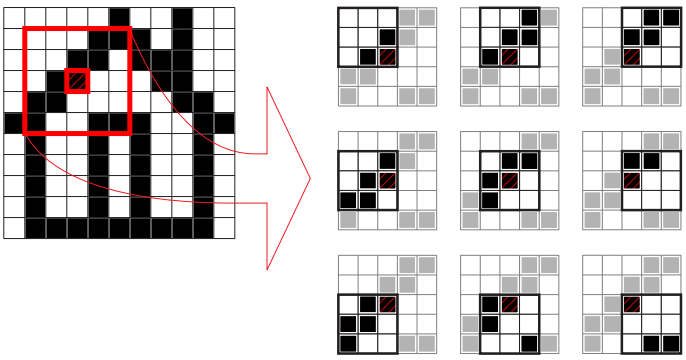


Figure 4.8: An example of the patch-based image model used for denoising. Each $P \times Q = 5 \times 5$ image region (red square on the left) is comprised of 9 overlapping $p \times q = 3 \times 3$ patches drawn from underlying manifolds. A point on the intersection of these manifolds gives an estimate of the central pixel in the region.

from exemplar images of a particular class (e.g. images of curves with contrast edges or images of a pattern). However, the form of our solution in Eq. 4.11 requires the manifolds to be defined in terms of the dimension of the ambient space, $\mathbb{R}^{PQ}$. Therefore, training patches (that are vectors in $\mathbb{R}^{pq}$) need to be extended to $P \times Q$ pixels. This reflects the fact that our model constrains the pq pixels forming the patch but allows the remaining $PQ - pq$ pixels in the image to vary freely, in order to generate a manifold constraint on the entire image (as in Fig. 4.1). However, this extension can be quite cumbersome. We consider two possible ways to overcome this problem. First, for each manifold $\mathcal{M}_m$ we define a separate set of training samples $\mathbf{X}_m \in \mathbb{R}^{PQ \times n_m}$ by augmenting the missing dimensions with gray (median-valued) pixels. Alternatively, we consider setting them equal to the pixels of the initial (noisy) image.

The result of denoising a high-contrast image corrupted with additive zero-mean Gaussian noise is shown in Fig. 4.9. In this example, we use the former type of padding with gray pixels and found that $5 - 10$ iterations ($K = 5 \ldots 10$) are enough to obtain good results. Here we use the Gaussian kernel with $\sigma = 190$ and learn the patch manifolds as $65-$dimensional subspaces in the induced feature space; all training and testing images are scaled to the range $[-1, \ldots, 1]$. Finally, we solve the preimage problem using the method of [119], which approximates the solution as a combination of nearest training samples.

For the examples in Fig. 4.10, we use Gaussian kernels with $\sigma = 75$ and $\sigma = 16$, and set the

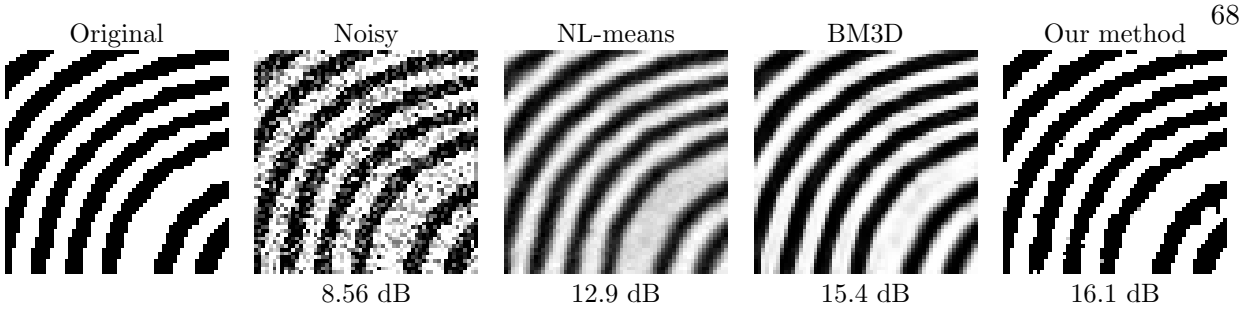


Figure 4.9: Results of denoising a high-contrast image. Numbers represent PSNR. Our algorithm preserves sharp high contrast edges of smooth curves; notice their blurring by NL-means.

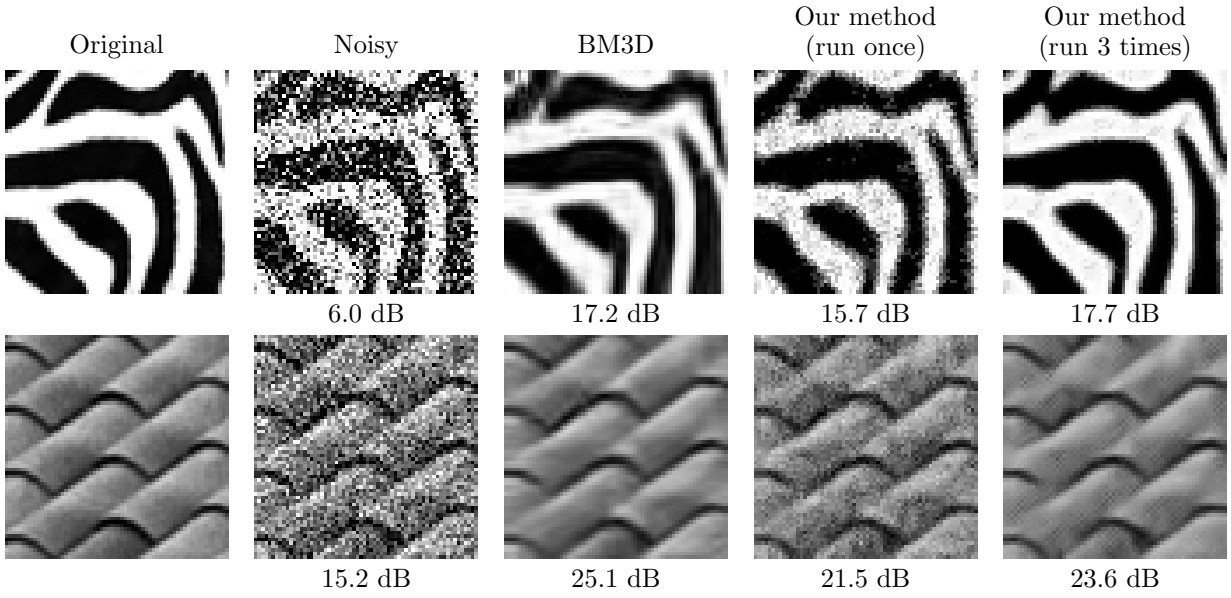


Figure 4.10: Results of denoising textures by extending the patch samples from $\mathbb{R}^{pq}$ to $\mathbb{R}^{PQ}$ with the pixels of initial (noisy) images. Numbers represent PSNR. Running our algorithm several times quickly improves the results and performs similar or slightly better than the state-of-the-art BM3D.

dimensions of the approximating subspaces to $d_{\mathcal{U}} = 55$ and $d_{\mathcal{U}} = 105$ for *Zebra* and *Roof* images respectively. We also use the second variant of padding with the pixels of initial images. This allows us to process significantly more complex natural textures from Fig. 2.1. Moreover, applying our patch manifolds intersection method iteratively multiple times (while accordingly updating the padded pixels on each run) quickly improves the results of denoising and demonstrates similar or slightly better performance than other popular patch-based image denoising methods, such as Non-local Means [25] and state-of-the-art BM3D [54].

In all our experiments, to quantitatively compare the results of different methods, we use peak signal-to-noise ratio (PSNR), defined as $PSNR = 10 \log \frac{\max \boldsymbol{I}_{i,j}^2}{\frac{1}{N}\sum(\mathbf{I}_{i,j} - \mathbf{J}_{i,j})^2}$, where $\mathbf{I}$ and $\mathbf{J}$ are the

original and denoised N-pixel images respectively. In the next chapter, we will improve the efficiency of our image denoising method and will show how to successfully apply the proposed manifolds intersection model of overlapping patches to solve other inverse problems in image processing.

4.5 Conclusion

In this chapter we proposed the model of intersecting manifolds as a novel approach to describe signals that can possess characteristics of several different classes, each modeled with underlying manifolds. The kernel trick was used to treat presumably non-linear manifolds as linear subspaces in higher-dimensional feature space and to find their intersection with a simple iterative projection algorithm, which constitutes the main contribution of this chapter. The final solution is expressed in closed form. This allows for faster algorithmic implementation and gives the possibility to simultaneously solve the problem with several different initial conditions.

The proposed manifolds intersection model can be particularly useful in representing families of signals that exhibit smooth inter- and intra-subject variations such as images of facial expressions, handwriting, or biomedical images. As an example, we described its application to the out-of-sample extension of a set of facial images. Furthermore, its applicability in a practical patch-based image processing setting was demonstrated with an effective denoising approach.

Nevertheless, implicitly operating in a higher dimensional feature space, while increasing the computational efficiency of the algorithm, entails solving a difficult preimage problem. This process inevitably introduces significant errors, which in certain cases may render the entire algorithm impractical. Therefore, instead of treating the feature space solution and preimage problems separately and sequentially, which essentially makes the final result relying on the preimage solver, in the next chapter we propose a new approach that combines finding a suitable preimage with minimization of the manifold distance criterion. This will ensure the existence of a solution in the input space and reduce the error of reconstruction. Furthermore, we will focus more closely on applying our manifolds intersection model specifically for problems in image processing and will derive a successful framework to solve any linear inverse problem with it.

Chapter 5

An Effective Application of Our Model in Patch-based Image Processing

In this chapter, we will be improving upon our manifolds intersection model in several ways. First, to avoid issues with the error introduced by the preimage estimation, we will instead change our method to only consider feature space solutions for which an appropriate preimage is actually available. For this, instead of minimizing the criterion of Eq. 4.7 defined for subspaces in feature space and then relying on a preimage method to translate it into the manifolds intersection problem in the original space, here we will explicitly aim for the found solution $\widehat{\mathbf{z}}$ to lie on or close to all M manifolds and thus satisfy

$$\min_{\mathbf{z}} \sum_{m=1}^{M} w_m d^2\left(\mathbf{z}, \mathcal{M}_m\right), \tag{5.1}$$

where $d\left(\mathbf{z}, \mathcal{M}_m\right) = \inf_{\mathbf{x} \in \mathcal{M}_m} d\left(\mathbf{z}, \mathbf{x}\right)$ is the Euclidean distance from point $\mathbf{z}$ to the m^{th} manifold. We will still rely on kernel PCA as a useful way to approximate this distance but will avoid solving the difficult preimage problem, thus increasing the effectiveness of our method.

Furthermore, we will tailor the method for finding manifolds intersection specifically for image processing. In this setting, it is worthwhile to note that we may measure the image's distance to a manifold constraint given by a single patch by simply measuring the distance from each specific patch to its manifold model. Replacing the distance from a whole image to a complicated higher-dimensional manifold with the distances from its patches to a simpler manifold will allow us to work in a smaller space and will greatly increase the efficiency of our algorithm. This will eventually eliminate some of the awkwardness in learning the manifold from samples that we saw in the previous chapter (e.g. the necessity to fill out the rest of the image with gray or noise to obtain an

appropriate training sample for each patch).

Finally, the changes we make will allow us to incorporate other linear constraints on the image into our new improved method, thus making it applicable, unchanged, to any inverse problem in image processing. We will see that despite the our method's broad generality and applicability to a broad spectrum of inverse problems, it will still perform better than or comparably to several state-of-the-art image processing methods each tailored for specific problems, including BM3D for denoising [54], spatially adaptive filtering for compressing sensing [71], and Wexler et al.'s patch-based method for image inpainting [205].

As an overview of our approach, we will try to write a functional $J_{\mathcal{U}}(\mathbf{z})$ taking images as arguments that approximately reflects the distance from the input image to the manifolds' intersection. This functional can then be used as a regularization term in any inverse problem, in the same way as other regularization functional such as total variation or ℓ_1-norm are used. However, as in the previous chapter, we will use kernel-based methods to construct $J_{\mathcal{U}}(\mathbf{z})$, so that we may again take advantage of their ability to efficiently approximate the distance to the manifolds. We note that definition of $J_{\mathcal{U}}(\mathbf{z})$ as a function of the image space ensures that feature space solutions that do not correspond to any preimage in the original image space are not considered.

5.1 Intersection of Manifolds as an Optimization Problem

In this section, we start by showing how the manifold intersection criterion can be translated into a regularization term for inverse problems. As was noted above, the assumption that the desired image $\mathbf{z}$ lies on or close to the intersection of several manifolds suggests a regularization for inverse problems that minimizes the sum of Euclidean distances to all of them (see Eq. 5.1). However, in contrast to our closed-form method for finding manifolds' intersections described in the previous chapter, here we recognize that since each manifold $\mathcal{M}_m$ is parallel to $D-d$ axes, these coordinates (i.e. those pixels not in the m^{th} patch) do not affect the distance.

To formalize this, consider M (possibly overlapping) patches of an image $\mathbf{z} \in \mathbb{R}^D$ (as before, we represent images and their patches in the form of column vectors). Let $\mathbf{E}_m, m = 1, \ldots, M$, be $d\times$

D patch extraction matrices with entries $(\mathbf{E}_m)_{i,j} = 1$ if the j^{th} pixel of an image corresponds to the i^{th} pixel of the m^{th} patch and 0 otherwise. Now the m^{th} patch of the image $\mathbf{z}$ can be written as $\mathbf{E}_m\mathbf{z}$. Therefore, the distance from the entire image to the m^{th} manifold now becomes $d(\mathbf{z}, \mathcal{M}_m) = d(\mathbf{E}_m\mathbf{z}, \mathcal{M})$, where $\mathcal{M} \subset \mathbb{R}^d$ is the lower-dimensional patch manifold, which is typically assumed to be the same across all patches. With this observation, in our example in Fig. 4.1, we could measure the distances from patches $\mathbf{E}_1\mathbf{z}$ and $\mathbf{E}_2\mathbf{z}$ to the unit circle rather then the distances from the entire three-pixel image $\mathbf{z}$ to each cylinder.

Our proposed patch-based regularization term thus becomes:

$$\min_{\mathbf{z}} \sum_{m=1}^{M} w_m d^2 (\mathbf{E}_m\mathbf{z}, \mathcal{M}) . \tag{5.2}$$

Minimizing the above equation will encourage all overlapping patches to conform to the manifold model simultaneously, finding an intersection if it exists, as we desired, or finding a point close to all manifolds otherwise. The weights $w_m \geq 0$ can be chosen to control the distances from a solution to each manifold in this latter case. We will examine how to efficiently minimize this criterion and use it to regularize the inverse problem of Eq. 2.1 in the next sections.

5.2 Finding the Intersection of the Manifolds

The difficulty of minimizing Eq. 5.2 lies in the necessity of finding the distances to the presumably non-linear manifold $\mathcal{M}$. Moreover, the geometry of the manifold is unknown in general and has to be learned from the set of training samples. Again, we will employ the kernel trick [174] as an efficient and elegant way to solve both problems. Working in the kernel-induced feature space, we will construct a functional that will serve as a proxy to the true distance to the manifold and will allow us to easily minimize the criterion of Eq. 5.2.

5.2.1 Optimization Problem in Feature Space

Given the KPCA assumption that the manifold becomes an affine $d_{\mathcal{U}}$-dimensional subspace $\mathcal{U}$ in feature space (see Fig. 3.2), the initial criterion given by Eq. 5.2 becomes equivalent to

unconstrained minimization of the following functional:

$$J_{\mathcal{U}}(\mathbf{z}) = \sum_{m=1}^{M} w_m d^2_{\mathcal{H}}\left(\Phi(\mathbf{E}_m\mathbf{z}), \mathcal{U}\right), \tag{5.3}$$

where $d^2_{\mathcal{H}}\left(\Phi(\mathbf{E}_m\mathbf{z}), \mathcal{U}\right) = \left\|\Phi(\mathbf{E}_m\mathbf{z}) - \mathrm{P}_{\mathcal{U}}(\mathbf{E}_m\mathbf{z})\right\|^2_{\mathcal{H}}$ is the squared distance from the point $\Phi(\mathbf{E}_m\mathbf{z})$ to its projection $\mathrm{P}_{\mathcal{U}}(\mathbf{E}_m\mathbf{z})$ onto the subspace $\mathcal{U}$ in feature space. We note that this functional is similar to the preimage regularization term in the Robust KPCA algorithm of Nguyen and De la Torre [151], to which it reduces for $M = 1$. Rather than finding an approximation to the optimal $\Phi(\mathbf{E}_m\mathbf{z})$ in feature space, we will minimize Eq. 5.3 over $\mathbf{z}$ directly in the space of images, thus solving both intersection and preimage problems simultaneously. This will ensure the existence of a suitable solution in the input space.

We next show how to compute the regularization criterion of Eq. 5.3. We describe the subspace $\mathcal{U}$ in the feature space with its principal components $\mathbf{U} = \Phi(\mathbf{X})\boldsymbol{\alpha}$ found with the kernel PCA algorithm (see Section 3.1) and the sample mean $\mathbf{m}$. Now, for a patch $\mathbf{E}_m\mathbf{z}$, the projection of its image $\Phi(\mathbf{E}_m\mathbf{z})$ onto the subspace $\mathcal{U}$ in feature space is:

$$\begin{aligned}
&\mathrm{P}_{\mathcal{U}}(\mathbf{E}_m\mathbf{z}) = \mathbf{U}\mathbf{U}^{\mathrm{T}}\Phi(\mathbf{E}_m\mathbf{z}) + (\mathbf{I} - \mathbf{U}\mathbf{U}^{\mathrm{T}})\mathbf{m} \qquad (5.4)\\
&= \Phi(\mathbf{X})\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\left[\Phi(\mathbf{X})\right]^{\mathrm{T}}\Phi(\mathbf{E}_m\mathbf{z})\\
&\quad + (\mathbf{I} - \Phi(\mathbf{X})\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\left[\Phi(\mathbf{X})\right]^{\mathrm{T}})\Phi(\mathbf{X})\frac{1}{n_{\mathrm{X}}}\mathbb{1}\\
&= \Phi(\mathbf{X})\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{k}_m + \Phi(\mathbf{X})\left[\mathbf{I} - \boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\left[\Phi(\mathbf{X})\right]^{\mathrm{T}}\Phi(\mathbf{X})\right]\frac{1}{n_{\mathrm{X}}}\mathbb{1}\\
&= \Phi(\mathbf{X})\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{k}_m + \Phi(\mathbf{X})\boldsymbol{\mu},
\end{aligned}$$

where $\mathbf{k}_m$ is a vector with entries $[\mathbf{k}_m]_i = \kappa(\mathbf{x}_i, \mathbf{E}_m\mathbf{z})$ for $i = 1, \ldots, n_{\mathrm{X}}$, and $\boldsymbol{\mu} = \frac{1}{n_{\mathrm{X}}}\left(\mathbf{I} - \boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{K}\right)\mathbb{1}$. The squared distance from $\Phi(\mathbf{E}_m\mathbf{z})$ to the subspace $\mathcal{U}$ in Eq. 5.3 is then:

$$\begin{aligned}
&d^2_{\mathcal{H}}\left(\Phi(\mathbf{E}_m\mathbf{z}), \mathcal{U}\right) = \left\|\Phi(\mathbf{E}_m\mathbf{z}) - \mathrm{P}_{\mathcal{U}}(\mathbf{E}_m\mathbf{z})\right\|^2\\
&= \Phi(\mathbf{E}_m\mathbf{z})^{\mathrm{T}}\Phi(\mathbf{E}_m\mathbf{z}) - 2\Phi(\mathbf{E}_m\mathbf{z})^{\mathrm{T}}\left[\Phi(\mathbf{X})\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{k}_m + \Phi(\mathbf{X})\boldsymbol{\mu}\right]\\
&+ \left[\Phi(\mathbf{X})\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{k}_m + \Phi(\mathbf{X})\boldsymbol{\mu}\right]^{\mathrm{T}}\left[\Phi(\mathbf{X})\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{k}_m + \Phi(\mathbf{X})\boldsymbol{\mu}\right]\\
&= \kappa(\mathbf{E}_m\mathbf{z}, \mathbf{E}_m\mathbf{z}) - \mathbf{k}_m^{\mathrm{T}}\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{k}_m - 2\mathbf{k}_m^{\mathrm{T}}\boldsymbol{\mu} + \boldsymbol{\mu}^{\mathrm{T}}\mathbf{K}\boldsymbol{\mu}, \qquad (5.5)
\end{aligned}$$

where we used the definition of kernel function κ and the fact that $\boldsymbol{\alpha}^{\mathrm{T}}\Phi\left(\mathbf{X}\right)^{\mathrm{T}}\Phi\left(\mathbf{X}\right)\boldsymbol{\alpha} = \mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}$.

After combining equations 5.3 and 5.5, our minimization criterion finally becomes:

$$J_{\mathcal{U}}\left(\mathbf{z}\right) \quad = \quad \sum_{m=1}^{M} w_m \left[\kappa\left(\mathbf{E}_m\mathbf{z},\, \mathbf{E}_m\mathbf{z}\right) - \mathbf{k}_m^{\mathrm{T}}\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{k}_m - 2\mathbf{k}_m^{\mathrm{T}}\boldsymbol{\mu} + \boldsymbol{\mu}^{\mathrm{T}}\mathbf{K}\boldsymbol{\mu}\right]. \tag{5.6}$$

To illustrate the utility of this functional for our purposes, consider first a simple example of a single one-dimensional manifold in $\mathbb{R}^2$, i.e. the case $M = 1$. Specifically, we consider a spiral $\begin{bmatrix} x_1 & x_2 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1/2 + r\cos\left(3.5r\right) & r\sin\left(3.5r\right) \end{bmatrix}^{\mathrm{T}}$ and generate 300 samples of it for $r = [0, \ldots, \pi]$. We learn this manifold as a 20-dimensional subspace in the feature space induced by the Gaussian kernel with parameter $\sigma = 2.5$. We then compute the values of $J_{\mathcal{U}}\left(\mathbf{z}\right)$ on a grid of points around the manifold and plot its contour lines on the left panel of Fig. 5.1. Notice that the directions of decreasing values of the resulting scalar field well approximate the directions to the closest points on the manifold. This indicates that the functional $J_{\mathcal{U}}\left(\mathbf{z}\right)$ can serve as a good proxy to the distance $d\left(\mathbf{z}, \mathcal{M}\right)$ for the purpose of minimizing Eq. 5.2. We will use it as our regularization criterion to solve inverse problems.

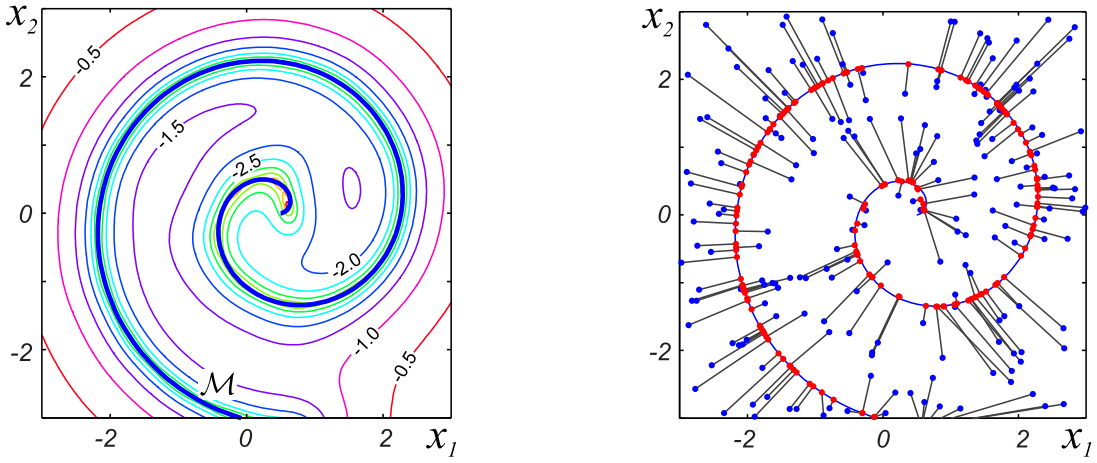


Figure 5.1: An example of minimizing the criterion of Eq. 5.3. Left: The target manifold $\mathcal{M}$ and contour lines of $J_{\mathcal{U}}\left(\mathbf{z}\right)$ in logarithmic scale. Notice how the values of $\log_{10} J_{\mathcal{U}}\left(\mathbf{z}\right)$ (numbers on the isolines) decrease towards the manifold. Right: The results of minimizing $J_{\mathcal{U}}\left(\mathbf{z}\right)$ with the gradient descent algorithm of Eq. 5.7. Randomly generated starting points $\mathbf{z}^{(0)}$ (blue) are mapped close to their nearest points on the manifold $\mathcal{M}$ (red points) using the gradient descent approach.

The criterion of Eq. 5.6 provides a simple yet powerful formulation of the manifolds intersec-

tion model. Efficiency is gained by learning each type of manifold only once and then distributing its description with matrices $\mathbf{E}_m$ to all corresponding patch positions.

Furthermore, since the terms of $J_{\mathcal{U}}$ are computed for each manifold $\mathcal{M}_m$ separately, this gives us the ability of using patches of different sizes and shapes adaptively and of choosing different kernels to achieve the best approximation of every manifold, if desired. This flexibility can be particularly useful as recent works [125, 189] have emphasized that low- and high- contrast patches can arise from very different manifolds, and in practice, allowing a variety of patch shapes and sizes when processing complex natural images has improved denoising results [55]. However, for the sake of simplicity, in the upcoming discussion we will consider all patches coming from the same manifold.

Next we will show how to minimize $J_{\mathcal{U}}(\mathbf{z})$ to conform an image with our model.

5.3 Minimizing the Regularization Term

To minimize the criterion of our patch manifolds intersection model (Eq. 5.6), we consider the steepest gradient descent algorithm as it is one of the simplest unconstrained minimization methods that yet produces excellent experimental results in Sections 5.5 and 5.6. As a special case, we also restrict our attention to the Gaussian kernel and derive a fixed-point iterative scheme for minimizing $J_{\mathcal{U}}(\mathbf{z})$.

The general idea of any iterative unconstrained minimization algorithm is to construct a minimizing sequence $\{\mathbf{z}_k\}$ that converges to an optimal point $\widehat{\mathbf{z}}$ (hopefully to $\mathbf{z}_{true}$ of Eq. 2.1 in our case) as $k \to \infty$. In particular, at every step of descent methods, the next iterate $\mathbf{z}^{(k+1)}$ is computed by moving in a direction $\mathbf{p}_k$ that minimizes the objective function $J_{\mathcal{U}}(\mathbf{z})$:

$$\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} + h_k \mathbf{p}_k, \tag{5.7}$$

where h_k are some (possibly variable) step sizes.

We assume that the kernel function is differentiable. Provided that the search direction makes an obtuse angle with the gradient of the objective function, $\left\langle \nabla J_{\mathcal{U}}\left(\mathbf{z}^{(k)}\right), \mathbf{p}^{(k)} \right\rangle < 0$, and for

properly chosen step sizes h_k, iterations of Eq. 5.7 converge to a local minimum or a saddle point of $J_{\mathcal{U}}(\mathbf{z})$. For more details on convergence properties of descent methods, please see [63] or [20].

5.3.1 Steepest Gradient Descent

First, to derive the steepest gradient descent algorithm we let $\mathbf{p}^{(k)} = -\nabla J_{\mathcal{U}}\left(\mathbf{z}^{(k)}\right)$. We note that the pixels not covered by the patch $\mathbf{E}_m\mathbf{z}$, $m = 1, \ldots, M$ can be regarded as having constant values. Therefore, for any differentiable function f,

$$\nabla_{\mathbf{z}} f\left(\mathbf{E}_m\mathbf{z}\right) = \mathbf{E}_m^{\mathrm{T}} \nabla_{\mathbf{E}_m\mathbf{z}} f\left(\mathbf{E}_m\mathbf{z}\right),$$

which effectively sets the corresponding coordinates of the gradient to 0. In what follows, to simplify the notation, we drop the index $\mathbf{z}$ and set $\nabla J_{\mathcal{U}}(\mathbf{z}) = \nabla_{\mathbf{z}} J_{\mathcal{U}}(\mathbf{z})$. Now the gradient of Eq.5.6 becomes:

$$\nabla J_{\mathcal{U}}(\mathbf{z}) = \sum_{m=1}^{M} w_m \mathbf{E}_m^{\mathrm{T}} \left[\nabla_{\mathbf{E}_m\mathbf{z}} \kappa\left(\mathbf{E}_m\mathbf{z},\, \mathbf{E}_m\mathbf{z}\right) - \mathbf{k}_m^{'} \boldsymbol{\nu}_m\right], \tag{5.8}$$

where $\boldsymbol{\nu}_m = 2\left(\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{k}_m + \boldsymbol{\mu}\right)$ and $\mathbf{k}_m^{'}$ denotes a $D \times n$ matrix with columns $\nabla_{\mathbf{E}_m\mathbf{z}} \kappa\left(\mathbf{x}_i, \mathbf{E}_m\mathbf{z}\right)$, $i = 1, \ldots, n$. Then the result of the $(k+1)$st iteration of Eq. 5.7 is found as:

$$\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} - h_k \cdot \nabla J_{\mathcal{U}}\left(\mathbf{z}^{(k)}\right). \tag{5.9}$$

For the Gaussian kernel, $\kappa\left(\mathbf{E}_m\mathbf{z},\, \mathbf{E}_m\mathbf{z}\right) = 1$ and thus Eq. 5.8 reduces to:

$$\nabla J_{\mathcal{U}}(\mathbf{z}) = -\sum_{m=1}^{M} w_m \mathbf{E}_m^{\mathrm{T}} \mathbf{k}_m^{'} \boldsymbol{\nu}_m, \tag{5.10}$$

with the i^{th} column of $\mathbf{k}_m^{'}$ given by $\left[\mathbf{k}_m^{'}\right]_{:,i} = \frac{2}{\sigma}\left[\mathbf{k}_m\right]_i \left(\mathbf{x}_i - \mathbf{E}_m\mathbf{z}\right)$.

As an example of minimizing the functional $J_{\mathcal{U}}(\mathbf{z})$ with steepest gradient descent we consider a problem of finding the nearest point on the manifold in Fig. 5.1. The results of running the algorithm initialized with a cloud of randomly generated points is shown on the right panel of Fig. 5.1. We observe that the found solutions lie on the manifold close to the initial points, as desired.

5.3.2 Fixed-point Iterative Procedure

For radial basis functions (rbf) kernels of the type $\kappa(\mathbf{x}, \mathbf{z}) = f\left(\|\mathbf{x} - \mathbf{z}\|\right)$ we can also establish an iterative fixed-point method similar to [143, 151]. In particular, for the Gaussian kernel, to derive

the recursive relation, we set the gradient in Eq. 5.10 to $\mathbb{0}$, which is the necessary and sufficient condition of an extremum of $J_{\mathcal{U}}(\mathbf{z})$, and then solve the resulting equation for $\mathbf{z}$.

Let $\mathbf{D}_m$ be an $n_{\mathrm{X}} \times n_{\mathrm{X}}$ diagonal matrix with entries $[\mathbf{D}_m]_{i,i} = [\mathbf{k}_m]_i$, and $\mathbf{X}$ denote a matrix of the training samples $\mathbf{x}_i$, $i = 1, \ldots, n_{\mathrm{X}}$ (in the input space) arranged column-wise. Then after expanding $\mathbf{k}_m^{'}$, Eq. 5.10 becomes:

$$\sum_{m=1}^{M} w_m \mathbf{E}_m^{\mathrm{T}} \mathbf{X} \mathbf{D}_m \boldsymbol{\nu}_m - \sum_{m=1}^{M} w_m \mathbf{k}_m^{\mathrm{T}} \boldsymbol{\nu}_m \mathbf{E}_m^{\mathrm{T}} \mathbf{E}_m \mathbf{z} \overset{set}{=} 0. \tag{5.11}$$

We now add and subtract $\sum_{m=1}^{M} w_m \mathbf{k}_m^{\mathrm{T}} \boldsymbol{\nu}_m \mathbf{z}$ from the above equation:

$$\sum_{m=1}^{M} w_m \mathbf{E}_m^{\mathrm{T}} \mathbf{X} \mathbf{D}_m \boldsymbol{\nu}_m - \sum_{m=1}^{M} w_m \mathbf{k}_m^{\mathrm{T}} \boldsymbol{\nu}_m \mathbf{E}_m^{\mathrm{T}} \mathbf{E}_m \mathbf{z} + \sum_{m=1}^{M} w_m \mathbf{k}_m^{\mathrm{T}} \boldsymbol{\nu}_m \mathbf{z} - \sum_{m=1}^{M} w_m \mathbf{k}_m^{\mathrm{T}} \boldsymbol{\nu}_m \mathbf{z} = 0$$

and then regroup its terms:

$$\sum_{m=1}^{M} w_m \left[\mathbf{E}_m^{\mathrm{T}} \mathbf{X} \mathbf{D}_m \boldsymbol{\nu}_m + \mathbf{k}_m^{\mathrm{T}} \boldsymbol{\nu}_m \left(\mathbf{I} - \mathbf{E}_m^{\mathrm{T}} \mathbf{E}_m \right) \mathbf{z} \right] = \sum_{m=1}^{M} w_m \mathbf{k}_m^{\mathrm{T}} \boldsymbol{\nu}_m \mathbf{z}.$$

Solving for $\mathbf{z}$ on the right-hand side eventually yields the following recursive update rule:

$$\mathbf{z}^{(k+1)} = \frac{\sum_{m=1}^{M} w_m \left[\mathbf{E}_m^{\mathrm{T}} \mathbf{X} \mathbf{D}_m \boldsymbol{\nu}_m + \mathbf{k}_m^{\mathrm{T}} \boldsymbol{\nu}_m \left(\mathbf{I} - \mathbf{E}_m^{\mathrm{T}} \mathbf{E}_m \right) \mathbf{z}^{(k)} \right]}{\sum_{m=1}^{M} w_m \mathbf{k}_m^{\mathrm{T}} \boldsymbol{\nu}_m}, \tag{5.12}$$

where both vectors $\mathbf{k}_m$ and $\boldsymbol{\nu}_m$ are evaluated at $\mathbf{z}^{(k)}$.

Iterations of Eq. 5.9 or Eq. 5.12 provide the means for finding a point on the intersection of patch manifolds close to the initialization $\mathbf{z}^{(0)}$. As we will see in Section 5.5.3, they can be readily applied for solving denoising problems, where additive noise is assumed to send an image away from such intersection. To address other linear inverse problems in their general form (i.e. $\mathbf{W} \neq \mathbf{I}$ in Eq. 2.1), we will incorporate equality constraints in our algorithm and discuss this modification in the next section.

5.4 Regularizing Inverse Problems with the Proposed Criterion

We now look at how to use the criterion of the manifolds intersection model $J_{\mathcal{U}}(\mathbf{z})$ (Eq. 5.3) as a regularization term for the inverse problem formulated in Section 2.1, i.e. we aim to solve:

$$\min_{\mathbf{z}} J_{\mathcal{U}}(\mathbf{z}) \;\; s.t. \, \mathbf{W}\mathbf{z} = \mathbf{b}. \tag{5.13}$$

5.4.1 Restriction of the Solution to the Subspace

First, consider restricting the minimization process (Eq. 5.7) by projecting $\mathbf{z}^{(k)}$ onto the constraint subspace $\mathcal{W} = \left\{\mathbf{x} \in \mathbb{R}^D \,|\, \mathbf{W}\mathbf{z} = \mathbf{b}\right\}$ on every iteration. Let $\mathcal{P}_{\mathcal{W}}(\mathbf{x}) = \left(\mathbf{I} - \mathbf{W}^\dagger\mathbf{W}\right)\mathbf{x} + \mathbf{W}^\dagger\mathbf{b}$ be the projection operator onto $\mathcal{W}$, where $\mathbf{W}^\dagger = \mathbf{W}^{\mathrm{T}}\left(\mathbf{W}\mathbf{W}^{\mathrm{T}}\right)^{-1}$ is the Moore-Penrose pseudoinverse of $\mathbf{W}$. Then we define the following iterations:

$$\mathbf{z}^{(k+1)} = \mathcal{P}_{\mathcal{W}}\left(\widetilde{\mathbf{z}}^{(k+1)}\right), \tag{5.14}$$

where $\widetilde{\mathbf{z}}^{(k+1)}$ denotes the result of computing Eq. 5.7 based on the value $\mathbf{z}^{(k)}$ either via gradient descent (Eq. 5.9) or fixed-point iterations (Eq. 5.12). Note that the above equation can be written equivalently as $\mathbf{z}^{(k+1)} = \mathcal{P}_{\mathcal{W}}\left(\mathbf{z}^{(k)} + \mathbf{q}_k\right) = \mathcal{P}_{\mathcal{W}}\left(\mathbf{z}^{(k)}\right) + \mathcal{P}_{\mathcal{W}}\left(\mathbf{q}_k\right)$ for some step $\mathbf{q}_k$. Clearly, it has the meaning of aligning the search direction with the subspace $\mathcal{W}$. Therefore, for appropriate choice of the step size, these iterations converge to a local minimum of $J_{\mathcal{U}}(\mathbf{z})$ within the subspace and solve the problem of Eq. 5.13, as desired.

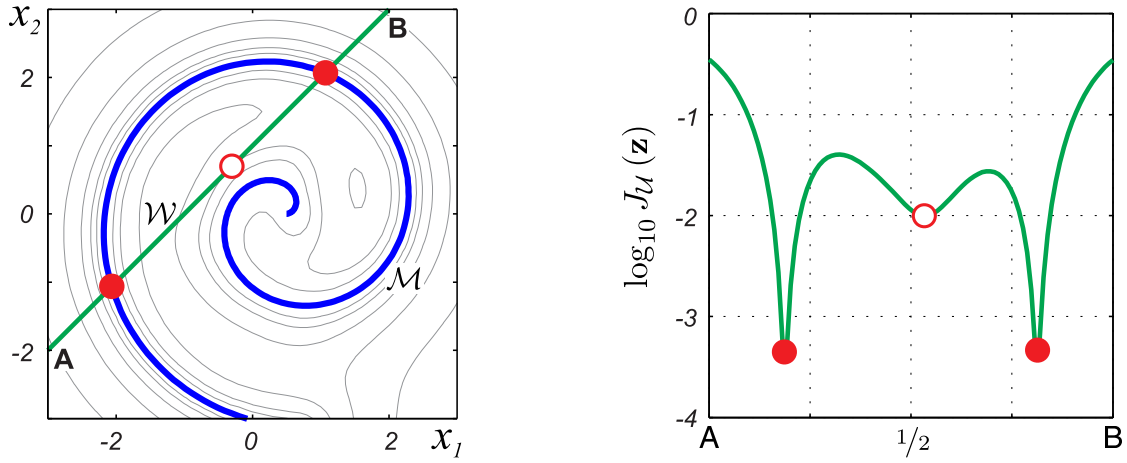


Figure 5.2: An example of regularizing an inverse problem with the manifold model. Depending on the initialization $\mathbf{z}^{(0)}$, iterations of Eq. 5.14 converge either to one of the global optima on the intersection $\mathcal{M} \cap \mathcal{W}$ (solid dots) or get trapped at the local minimum of $J_{\mathcal{U}}(\mathbf{z})$ within the constraint subspace $\mathcal{W}$ (hollow dots). In the latter case, minimizing the criterion of Eq. 5.15 with $\lambda > 0$ will set the solution closer to the manifold $\mathcal{M}$, if desired. Right panel shows the plot of the values of $J_{\mathcal{U}}(\mathbf{z})$ (in logarithmic scale) along the constraint subspace $\mathcal{W}$.

Unfortunately, the found solution is not guaranteed to lie on an intersection of all the manifolds $\mathcal{M}_m$ with the constraint subspace $\mathcal{W}$, even if such points exist (see the example on Fig. 5.2).

Iterations of Eq. 5.14 converge to a point $\widehat{\mathbf{z}}$, at which $\mathrm{P}_{\mathcal{W}}\left[\nabla J_{\mathcal{U}}\left(\widehat{\mathbf{z}}\right)\right] = \mathbf{0}$. However, the component of the gradient orthogonal to $\mathcal{W}$ may not vanish. This could mean that $\widehat{\mathbf{z}} \notin \bigcap_{m=1}^{M} \mathcal{M}_m$, and, therefore, it does not conform to our model. The same may occur when noise in the vector of measurements $\mathbf{b}$ makes the intersection set empty, $\left(\bigcap_{m=1}^{M} \mathcal{M}_m\right) \cap \mathcal{W} = \emptyset$. In either case, it may be advisable to relax adherence to the constraint subspace in order to better satisfy the model assumptions. We will address this in the next subsection.

5.4.2 Relaxation of the Constraint

In order to relax the constraint of the problem of Eq. 5.13, in addition to the criterion of the manifolds intersection model (Eq. 5.3), we require a desired solution to lie close to (but not necessarily on) the subspace $\mathcal{W}$. We propose achieving this with the following minimization:

$$\min_{\mathbf{z}} \;\; \lambda J_{\mathcal{U}}\left(\mathbf{z}\right) + \left(1-\lambda\right) d^2\left(\mathbf{z},\, \mathcal{W}\right) \tag{5.15}$$

for some regularization parameter $0 \leq \lambda \leq 1$. Indeed, in the noiseless case ($\mathbf{n} = 0$ in Eq. 2.1), under the assumptions of our model, both terms of the above equation vanish at a (global) optimum, locating the solution $\widehat{\mathbf{z}}$ at the intersection of the constraint subspace $\mathcal{W}$ with all the manifolds for any λ. When such point does not exist or is not feasible from the initialization $\mathbf{z}^{(0)}$, minimizing Eq. 5.15 still results in a reasonable solution. In this case, the parameter λ allows us to control the tradeoff between satisfying the inverse problem constraints and the model assumptions.

Minimization of Eq. 5.15 can be carried out in the way discussed before in Section 5.3. Using the expression for the projection operator $\mathcal{P}_{\mathcal{W}}$, the distance $d^2\left(\mathbf{z},\, \mathcal{W}\right) = \left\|\mathbf{z} - \mathcal{P}_{\mathcal{W}}\left(\mathbf{z}\right)\right\|^2$ after simplification becomes:

$$\begin{aligned} d^2\left(\mathbf{z},\, \mathcal{W}\right) &= \left\|\mathbf{z} - \left(\mathbf{I} - \mathbf{W}^{\dagger}\mathbf{W}\right)\mathbf{z} - \mathbf{W}^{\dagger}\mathbf{b}\right\|^2 \\ &= \mathbf{z}^{\mathrm{T}}\mathbf{W}^{\dagger}\mathbf{W}\mathbf{z} - 2\mathbf{z}^{\mathrm{T}}\mathbf{W}^{\dagger}\mathbf{b} + \mathbf{b}^{\mathrm{T}}\left(\mathbf{W}\mathbf{W}^{\mathrm{T}}\right)^{-1}\mathbf{b}. \end{aligned}$$

Then, defining $\widetilde{J}_{\mathcal{U}}\left(\mathbf{z}\right) = \lambda J_{\mathcal{U}}\left(\mathbf{z}\right) + \left(1-\lambda\right) d^2\left(\mathbf{z},\, \mathcal{W}\right)$, the gradient

$$\nabla \widetilde{J}_{\mathcal{U}}\left(\mathbf{z}\right) = \lambda \nabla J_{\mathcal{U}}\left(\mathbf{z}\right) + 2\left(1-\lambda\right)\left[\mathbf{W}^{\dagger}\mathbf{W}\mathbf{z} - \mathbf{W}^{\dagger}\mathbf{b}\right]$$

is used to compute the search direction in Eq. 5.7. We summarize our resulting algorithm with the following pseudocode (see Algorithm 2).

To conclude, we would like to comment on a connection of our method with a subgradient variant of the iterative POCS algorithm [30, 31, 52]. Often, to alleviate the computational burden of finding the exact projections onto a non-linear manifold, it is more practicable to project onto the shrinking level sets that eventually converge to a point on the sought intersection. In the most general case, this is performed (either in a circular or parallel manner, as discussed in Chapter 4) by moving in a direction opposite to a subgradient, which for differentiable objective functions essentially reduces to the steepest gradient descent method. Even though patch manifolds are not convex, and making such assumption here would be too restrictive, we still can regard the iterations of Eq. 5.7 (at least locally) as approximate projections onto the level sets of $J_{\mathcal{U}}(\mathbf{z})$.

Finally, we note that, as was shown by Blumensath in [18], as long as the measurement operator $\mathbf{W}$ satisfies a bi-Lipschitz condition on the manifold, our iterative projection algorithm converges to a near optimal solution in a fixed number of steps.

Approximating the distance to the manifolds with kernel methods significantly facilitates the problem of mapping a point onto them, whereas finding the exact solution may be intractable, if possible. Although global convergence of the proposed method can not be guaranteed due to non-convexity of the considered manifolds (see Fig. 5.2 for a pathological example of convergence to a local solution), in practical image processing applications it produces excellent results, as we demonstrate in the next section. It is worth noting also that in practice the chances of pathological situation of the Fig. 5.2 happening can be reduced by considering multiple initialization points $\mathbf{z}^{(0)}$ and then taking the best solution found from all of them.

5.5 Experiments and Discussion

Before using our model in practical image processing applications, we revisit our motivational toy example of two intersecting cylinders in $\mathbb{R}^3$, described in Section 4.1, to demonstrate that the method does indeed find the manifolds' intersection. We then apply our algorithm for denoising,

Algorithm 2 Constrained minimization of $J_{\mathcal{U}}$

Input: Set of training patches $\{\mathbf{x}_i\}_{i=1}^{n}$, measurement matrix $\mathbf{W}$, vector of measurements $\mathbf{b}$, kernel function κ, patch-extraction matrices $\{\mathbf{E}_m\}_{m=1}^{M}$ with corresponding weights w_m, regularization parameter λ, algorithm step size h, and termination conditions (e.g. maximum number of iterations K and/or minimum norm of the gradient ε).

Output: Reconstructed image $\widehat{\mathbf{z}}$.

$[\mathbf{K}, \boldsymbol{\alpha}] \leftarrow \mathrm{KPCA}\,(\mathbf{X})$ ▷ Please see Section 2.1 or [143, 48].

$\boldsymbol{\mu} \leftarrow {}^{1}\!/_{n}\left(\mathbf{I} - \boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{K}\right)\mathbb{1}$

$\mathbf{z}^{(0)} \leftarrow \mathbf{W}^{\dagger}\mathbf{b}$ ▷ Initialization (see Sec. 5.5).

$k \leftarrow 0$

while $k < K$ **and** $\|\nabla J_{\mathcal{U}}\| > \varepsilon$ **do**

  $k \leftarrow k + 1$

  $\nabla J_{\mathcal{U}} \leftarrow \mathbb{0}$

  for $m := 1 \ldots M$ **do** ▷ Loop over all patches.

    $\mathbf{p}_m \leftarrow \mathbf{E}_m\mathbf{z}^{(k-1)}$ ▷ Extract the m^{th} patch.

    for $i := 1 \ldots n$ **do**

      $[\mathbf{k}_m]_i \leftarrow \kappa\,(\mathbf{x}_i,\, \mathbf{p}_m)$

      $\left[\mathbf{k}_m^{'}\right]_{:,i} \leftarrow \nabla_{\mathbf{p}_m}\kappa\,(\mathbf{x}_i, \mathbf{p}_m)$

    end for

    $\boldsymbol{\nu}_m \leftarrow 2\left(\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{k}_m + \boldsymbol{\mu}\right)$

    $\nabla J_{\mathcal{U}_m} \leftarrow \nabla_{\mathbf{p}_m}\kappa\,(\mathbf{p}_m,\, \mathbf{p}_m) - \mathbf{k}_m^{'}\boldsymbol{\nu}_m$

    $\nabla J_{\mathcal{U}} \leftarrow \nabla J_{\mathcal{U}} + w_m\mathbf{E}_m^{\mathrm{T}}\nabla J_{\mathcal{U}_m}$

  end for

  $\nabla\widetilde{J}_{\mathcal{U}} \leftarrow \lambda\nabla J_{\mathcal{U}} + 2\,(1-\lambda)\left[\mathbf{W}^{\dagger}\mathbf{W}\mathbf{z}^{(k-1)} - \mathbf{W}^{\dagger}\mathbf{b}\right]$

  $\mathbf{z}^{(k)} \leftarrow \mathbf{z}^{(k-1)} - h\nabla\widetilde{J}_{\mathcal{U}}$ ▷ Gradient descent.

end while

return $\widehat{\mathbf{z}} = \mathbf{z}^{k}$

compressive sensing reconstruction, and inpainting of structured images whose patches well conform to a manifold model (Fig. 2.1) in this section and generalize it to entire natural images in Section 5.6.

5.5.1 Intersection of Manifolds in $\mathbb{R}^3$

As an initial illustrative proof of concept that our algorithm accurately finds the true intersection of manifolds, we use it to map a cloud of randomly generated points to their closest points on the intersection of two cylinders. The geometry of the cylinders is learned from samples of the unit circle in $\mathbb{R}^2$, which constitute the training set of two-pixel patches. The results of our algorithm accurately trace the sought intersection, as shown on the right panel of Fig. 4.1 in the previous chapter. Notice that learning the resulting non-differentiable curve directly in $\mathbb{R}^3$ would be a significantly more difficult problem requiring a larger set of higher-dimensional training samples. This clearly demonstrates the advantages of our manifolds intersection model.

5.5.2 Set-up for the Image Processing Experiments

In the upcoming three subsections, we will apply our method to a variety of inverse problems in image processing, using simple textures as test cases, before extending our method to natural photographic images in Section 5.6. First, however, we describe our criteria for choosing patch sizes and other parameter values, and the evaluation metrics to be used.

In each experiment, the patches are chosen to have sizes approximately matching the scale of pattern details or other prominent image features. We use Gaussian kernels with parameters σ chosen to make the sample means of the values in the resulting kernel matrices approximately equal to 0.5, the midpoint between minimum and maximum possible values. The dimensions of the approximating subspaces are chosen such that the first $\delta_{\mathcal{H}}$ largest eigenvalues of $\bar{\mathbf{K}}$ account for about $0.97 - 0.98$ of their total sum. While we found that our algorithm is relatively insensitive to small deviations of σ, increasing $\delta_{\mathcal{H}}$ usually leads to a noticeable performance improvement (but at increased computational cost). For the three textures used in our experiments, these guidelines result in using 5×5 patches with $\sigma = 55$, $d_{\mathcal{U}} = 75$ for the zebra texture, 5×5 patches with $\sigma = 16$,

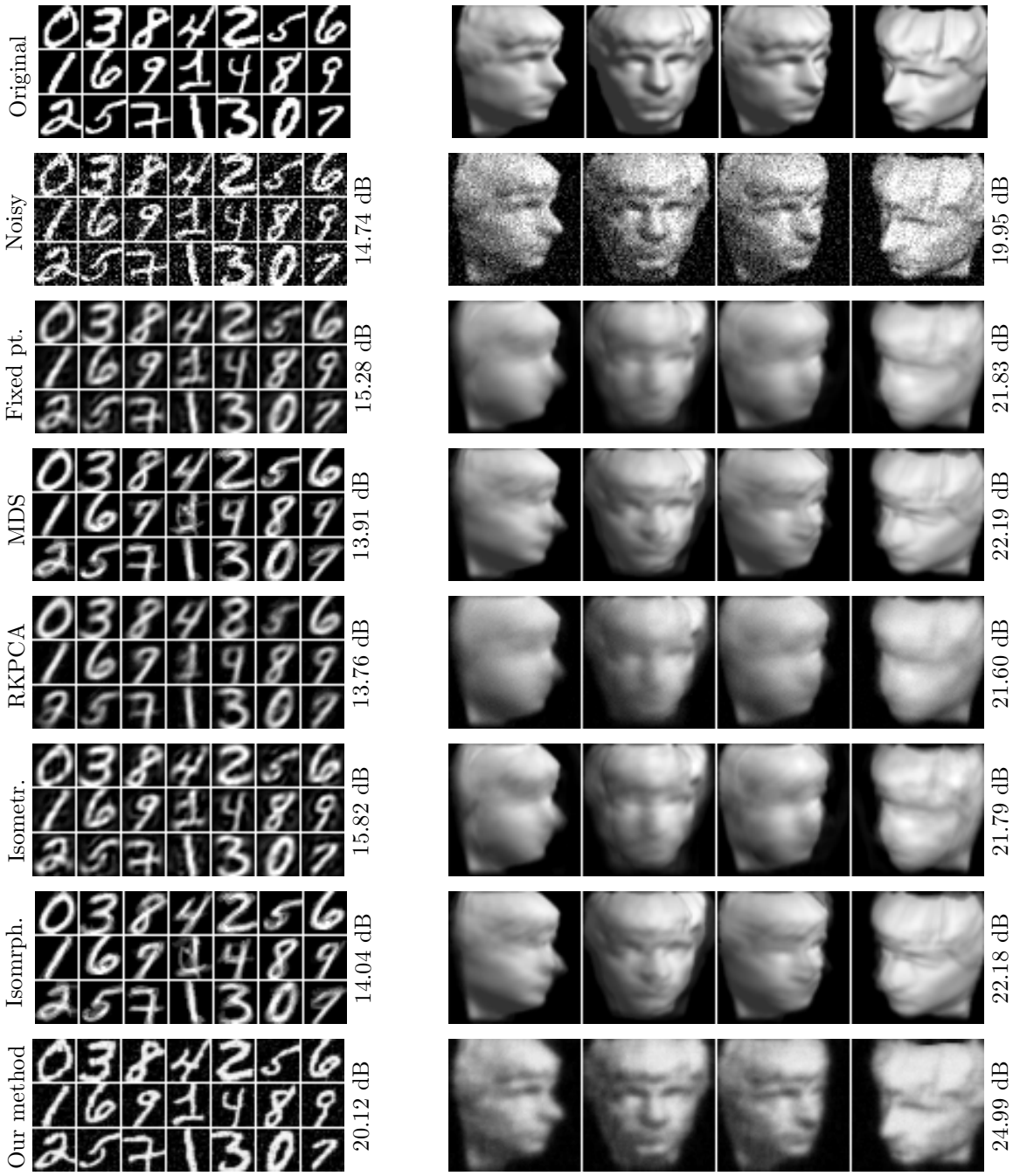


Figure 5.3: Results of denoising images of MNIST handwritten digits and a sculpture face with KPCA followed by different preimage methods: fixed-point iterations [143], MDS-based preimage [119], robust KPCA [151], and isomorphism-preserving preimage [104]. Even for relatively simple and structured images, often modeled with underlying manifolds, their patch-based representation with our model achieves noticeable improvement in reconstruction. Here the corresponding manifolds are learned from other training images (or their patches for our method). Numbers indicate PSNR.

$d_{\mathcal{U}} = 105$ for the roof tiles texture, and 9×9 patches with $\sigma = 30$, $d_{\mathcal{U}} = 75$ for the fabric texture, across all experiments. Please refer to Fig. 2.1 for the original images that we use in our experiments throughout the work. As before, for training, we use patches from similar (but different) images, i.e. another image of a zebra or a different part of the roof.

Although our final solution admits any combination of overlapping patches, in our examples we will cover each image with $L < pq$ randomly-offset grids, each segmenting the image into a layer of non-overlapping $p \times q$ patches (disregarding all partial patches along the borders). To ensure that all image pixels would be covered, we explicitly choose the layers containing the four corner patches. We then combine the patches from all L layers to obtain the M overlapping patches.

We use the gradient descent version of the algorithm (Eq. 5.9) throughout, as this method was found to converge faster (see Figs. 5.8, 5.9). To quantify the achieved performance, as in Section 4.4.4, we use peak signal-to-noise ratio (PSNR).

5.5.3 Image Denoising

In this subsection, we look at image denoising as a useful first setting for investigating the properties of our algorithm before applying it to other inverse problems. We emphasize, however, that we did not tailor our method for denoising exclusively, in contrast with the methods we will benchmark against. We thus intend these results to showcase the method's broad applicability rather than its specific performance for denoising.

To denoise a signal $\mathbf{z}_{noisy} = \mathbf{I}\mathbf{z} + \mathbf{n}$ we simply minimize Eq. 5.15 with steepest gradient descent; we initialize the iterations with $\mathbf{z}^{(0)} = \mathbf{z}_{noisy}$ and set $\lambda \approx 1$. Therefore, the algorithm is allowed to converge to the local minimum of $J_{\mathcal{U}}(\mathbf{z})$ nearest to $\mathbf{z}_{noisy}$, i.e. the nearest intersection point of the manifolds. In all experiments, noise is zero-mean additive Gaussian.

5.5.3.1 Advantages of Manifolds for Patches vs. Manifolds for Entire Images

First, we compare the performance of our model with other existing kernel-based approaches for solving inverse problems in image processing. These consider an image as arising from a mani-

fold, learned from entire image examples. In contrast, in our model, we view an image as a point on the intersection of several simpler manifolds corresponding to its overlapping patches. By analogy with our previous example from Section 5.5.1, instead of trying to learn the complex intersection of the two cylinders from samples in $\mathbb{R}^3$ (e.g. a manifold of images), we would ask each patch to conform to the circle in $\mathbb{R}^2$ (a patch manifold).

To experimentally compare these approaches, we consider two examples: images of MNIST handwritten digits [124] and images of a rotating sculpture face [191]. The images in both datasets have small sizes (20×20 and 64×64 pixels respectively) and relatively simple structure, which allows one to approximate them with underlying manifolds fairly well. We use Gaussian kernels with $\sigma = 600$ and $\sigma = 3000$ to learn them from 3961 images of digits and 694 images of sculpture faces (testing images were excluded from the training sets). Noisy images are then projected onto corresponding $50-$ and $25-$dimensional subspaces in the induced feature spaces and their denoised estimates are reconstructed with different preimage methods (Fig. 5.3).

Meanwhile, in our method, we learn manifolds for 3×3-pixel patches extracted from the same training sets. We use $\sigma = 10$, $d_{\mathcal{U}} = 12$ for MNIST and $\sigma = 5$, $d_{\mathcal{U}} = 10$ for sculpture face patches. We consider only $L = 5$ layers of patches and set $\lambda = 0.98$. Our results in Fig. 5.3 clearly demonstrate the advantages of the proposed patch-based representation which results in greatly increased PSNR. Furthermore, larger training sets are available when working with image patches. For example, while the original dataset of sculpture faces contains only 694 images, we extracted 5000 patches from them to learn the manifolds.

5.5.3.2 Denoising Natural Image Textures and Performance Analysis

We now turn our attention to natural textures whose patches conform well to manifold models (see Fig. 2.1). We will consider denoising them not only as a practical problem by itself, but also as a convenient setting to analyze the performance of our algorithm under varying conditions. Specifically, we will expose the crucial advantage of our joint patch reconstruction procedure over methods that treat each patch separately and show ways to improve its computational efficiency.

The results of denoising (using the parameters and training procedure described in Sec. 5.5.2) are presented in Fig. 5.4. In this case, learning the specific manifold structure for each texture from the training set of patches allowed us to obtain results slightly better in terms of PSNR than the BM3D algorithm [54] with enhanced visual quality. We note that unlike BM3D, we do not make any assumptions about the noise variance, but instead allow the algorithm to converge to the nearest point on the intersection of patch manifolds, thus effectively operating in a blind denoising scenario. Our method is especially effective in removing noise of high variances, where it consistently outperforms its competitors (see Fig. 5.5). In low-noise regimes, its performance is primarily determined by the accuracy of the learned model and can be improved by terminating the iterations earlier to avoid overfitting. We look at denoising of natural images in Section 5.6.

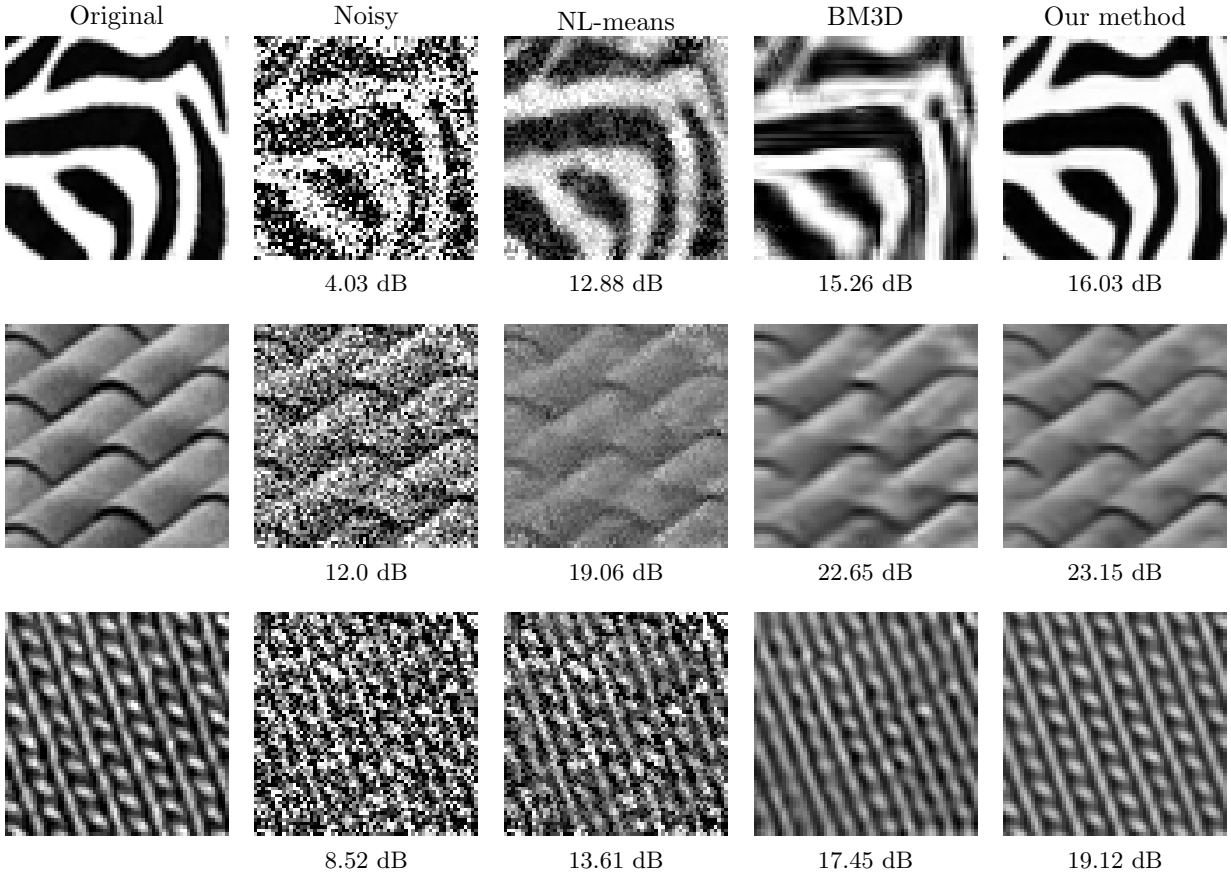


Figure 5.4: Denoising textures found in natural images. Our algorithm accurately reconstructs high contrast edges, as well as fine details of textures, and performs similarly to state-of-the-art BM3D in terms of PSNR, but with enhanced visual quality. Numbers represent corresponding PSNR.

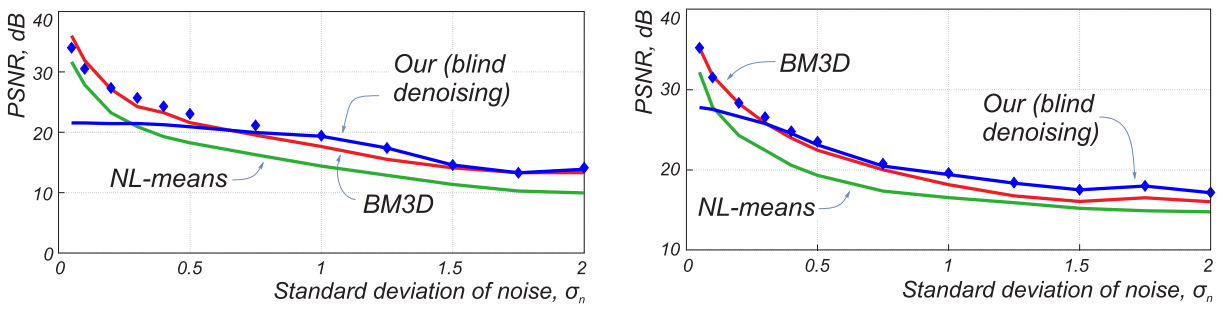


Figure 5.5: Analysis of the denoising performance under varying noise conditions for the *Zebra* (left) and *Roof* (right) images. Our method (blue line), run for 1000 iterations with unchanged parameters, does not make any assumptions about the noise variance, effectively operating as a blind denoising scheme. While it readily outperforms its competitors on high noise levels, stopping the iterations earlier to avoid overfitting leads to superior results in low-noise regimes as well (diamonds indicate results obtained by early stopping).

In the next experiment, we wish to see whether *joint* estimation of overlapping patches, as we propose, has advantages over estimating these patches individually via existing kernel-based approaches and then combining them to form the image. To compare, we consider the same set of patches as in our algorithm, but map each of them onto the patch manifold *separately* via the same kernel-based strategy (i.e. Robust KPCA, to which our algorithm reduces for a single patch). Reconstructed patches are then gathered to form the final image estimate and averaged where they overlap (see Fig. 5.7). While this scheme achieves better results compared to using non-overlapping patches ($L = 1$), estimating the patches jointly rather than separately, as we propose, achieves significant improvement. Table 5.1 further examines this comparison by applying multiple existing kernel-based denoising strategies to the patches individually. The table shows that our joint estimation strategy almost always outperforms any method of reconstructing each patch separately followed by combining them.

Moreover, we notice that, in our joint estimation, the best potential quality of reconstruction, all other conditions being equal, is achieved even when considering much fewer overlapping patches than are maximally available (see Fig. 5.6). Unless otherwise specified, in all our examples in this and the next subsections, images are covered by $L = 8$ overlapping layers of patches instead of the maximally possible 25 or 81, which significantly increases the speed of computation.

Table 5.1: Comparison of patch-based denoising performance, PSNR in dB

	Zebra, $\text{PSNR}_n = 4.1\,dB$			*Roof*, $\text{PSNR}_n = 12.0\,dB$		
	$L = 1$	$L = 8$	$L = 17$	$L = 1$	$L = 8$	$L = 17$
Fixed-point	9.93	10.15	10.16	16.24	16.42	16.42
MDS	12.39	13.94	14.00	19.26	21.18	21.33
RKPCA	12.57	13.44	13.43	20.79	22.09	22.2
Isomorph.	**12.8**	14.22	14.24	19.78	21.52	21.65
Our	12.42	**16.72**	**16.7**	**21.17**	**22.93**	**22.91**

Finally, we compare the convergence speed of the two proposed minimization methods. For this, we run the gradient descent algorithm (Eq. 5.15) as well as the fixed-point iterations (Eq. 5.12) initialized with the same noisy image. The plot of the attained PSNR as a function of the number of iterations for both methods is shown in Fig. 5.8. Results of intermediate steps (see Fig. 5.9) demonstrate that both algorithms converge to the same (or close) solution, however gradient descent does this much faster (requiring fewer iterations with the same number of kernel function evaluations per step). For this reason, we will use the gradient descent approach in all examples.

5.5.4 Compressive sensing reconstruction

In this section, we look at the application of our regularization approach to a compressive sensing problem. We apply the method of iterative projections onto the constraint subspace (Eq. 5.14) to reconstruct $64 \times 64 = 4096$ pixel images from their 400 Bernoulli random measurements (the measurement ratio is less than 10%). Although, as discussed in Section 5.4.1, the uniqueness of the solution is not guaranteed, starting the iterations with the least squares solution to Eq. 2.1, i.e. $\mathbf{z}^{(0)} = \mathbf{W}^{\dagger}\mathbf{b}$, results in excellent compressive sensing reconstruction in our experiments. Since the underlying manifold model provides an accurate description of the considered class of images, our algorithm shows state-of-the-art performance (Fig. 5.10). First, it greatly outperforms basis pursuit run on 64 8×8 non-overlapping patches, each modeled as sparse in a dictionary learned via K-SVD [3] on the training patches. This again demonstrates the advantage of using overlapping vs. non-overlapping patches. Our results further tend to have better visual quality and

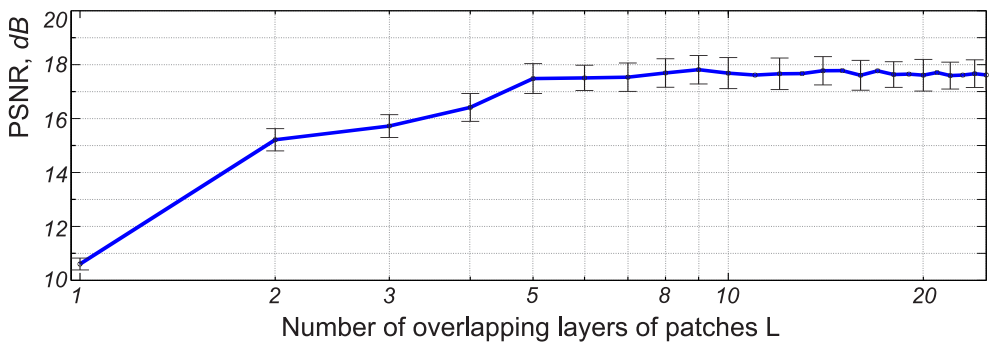


Figure 5.6: Reconstruction PSNR of the zebra image as a function of the number of layers of patches. A significant improvement is gained by using overlapping patches ($L > 1$), but no major gain could be achieved by considering more than eight layers of 5×5 patches. These results are obtained by averaging over 100 realizations of noise; error bars indicate sample standard deviations.

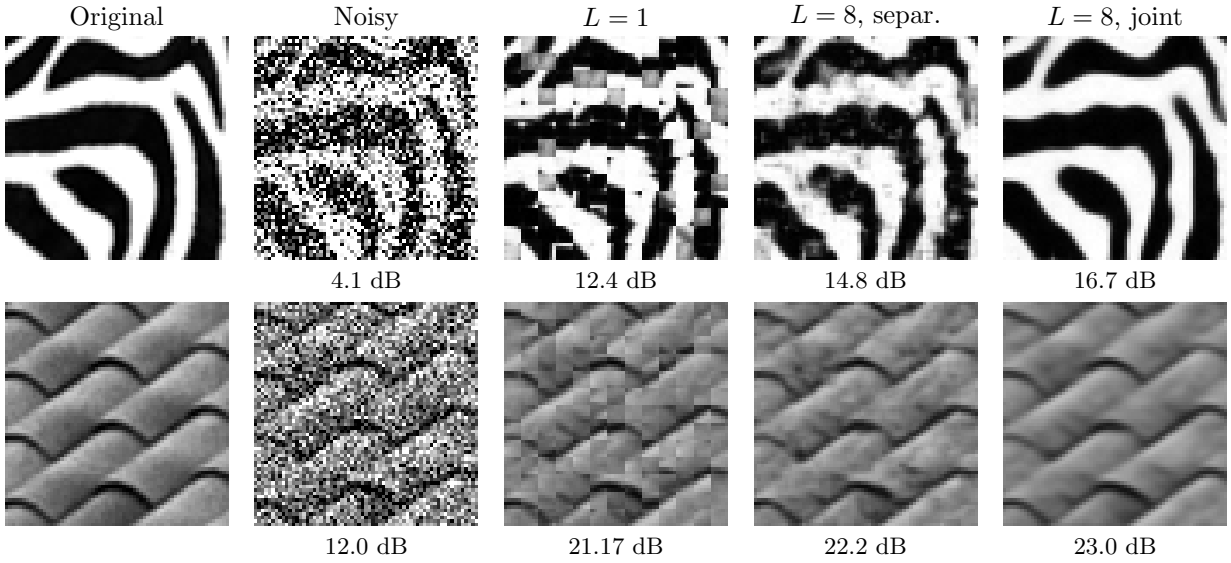


Figure 5.7: The advantages of our overlapping patch model. Using non-overlapping patches and mapping each of them onto the manifold separately ($L = 1$) results in apparent tessellation of the denoised image. The transitions can be smoothed somewhat by averaging over differently offset layers of patches to produce the final estimate. However, instead estimating all overlapping patches *jointly* on each iteration, as we propose, significantly improves the results.

higher PSNR than those obtained with recursive spatially adaptive filtering (a method based on the BM3D algorithm [71]) from the same number of (unique) low-frequency Fourier measurements. It is worth mentioning also that while most reconstruction algorithms are tailored to work with a specific class of measurement matrices (such as [71], which requires Fourier measurements), our method was derived under general assumptions of a linear measurement process and admits any measurement matrix $\mathbf{W}$.

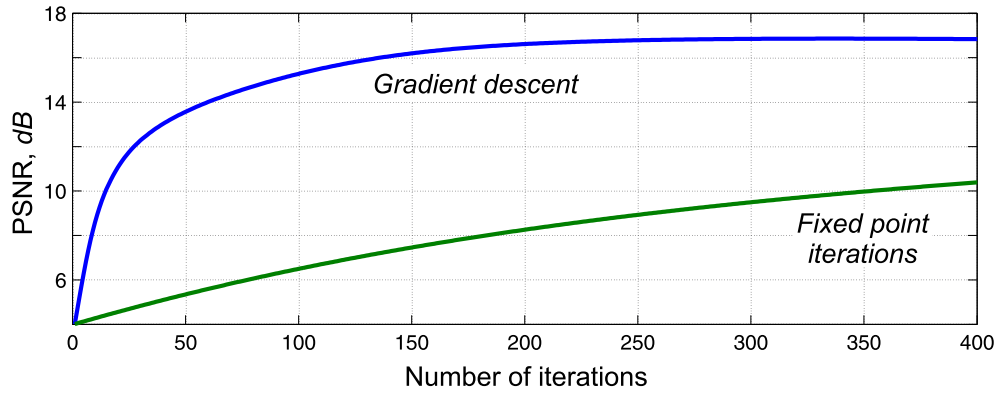


Figure 5.8: Comparison of the speed of convergence of the two proposed iteration methods. In the example of denoising the zebra image, gradient descent with fixed stepsize $h = 1$ (Eq. 5.9) achieves faster convergence than fixed-point iterations (Eq. 5.12).

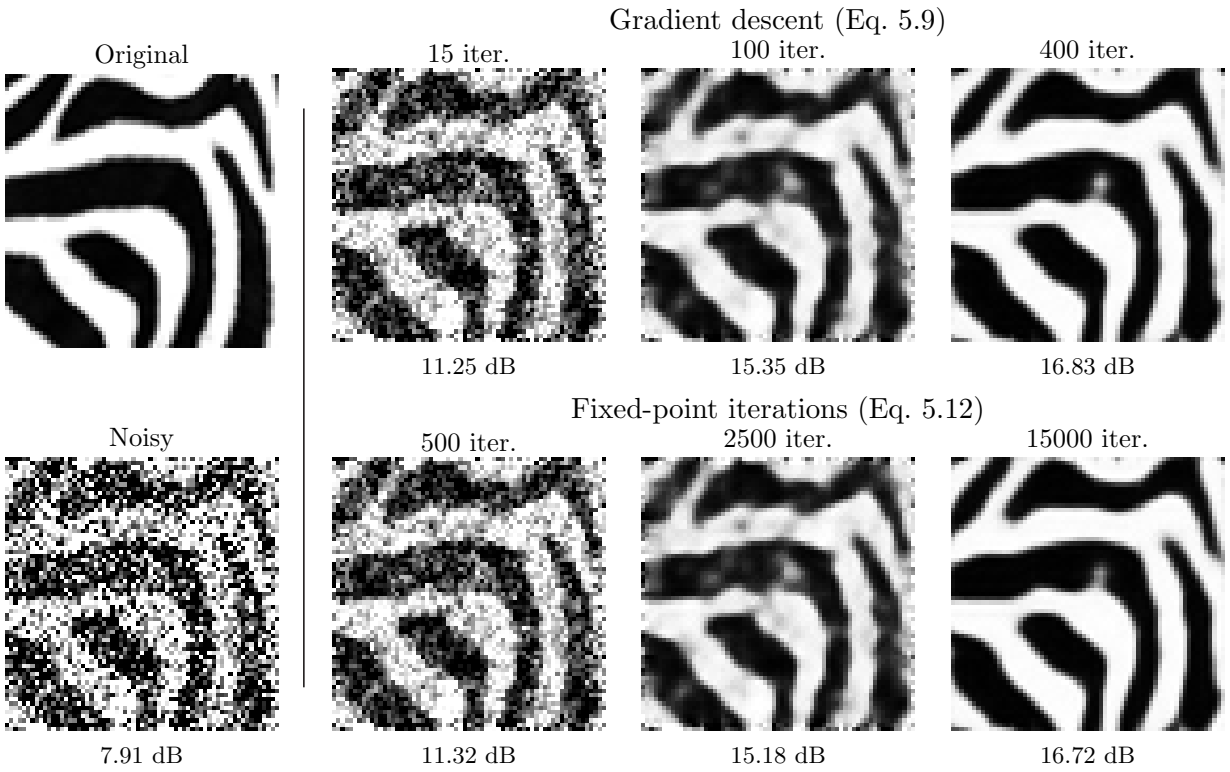


Figure 5.9: Comparison of denoising results obtained using different iterative methods. In our experiments, the gradient descent method with constant step size $h = 1$ converges to nearly the same solution much faster than the fixed-point iterations of Eq. 5.12.

5.5.5 Image inpainting

In this section, we assess the performance of our regularization term in an image inpainting task. Here we initialize the missing pixels by linearly interpolating the boundaries of the gap in both

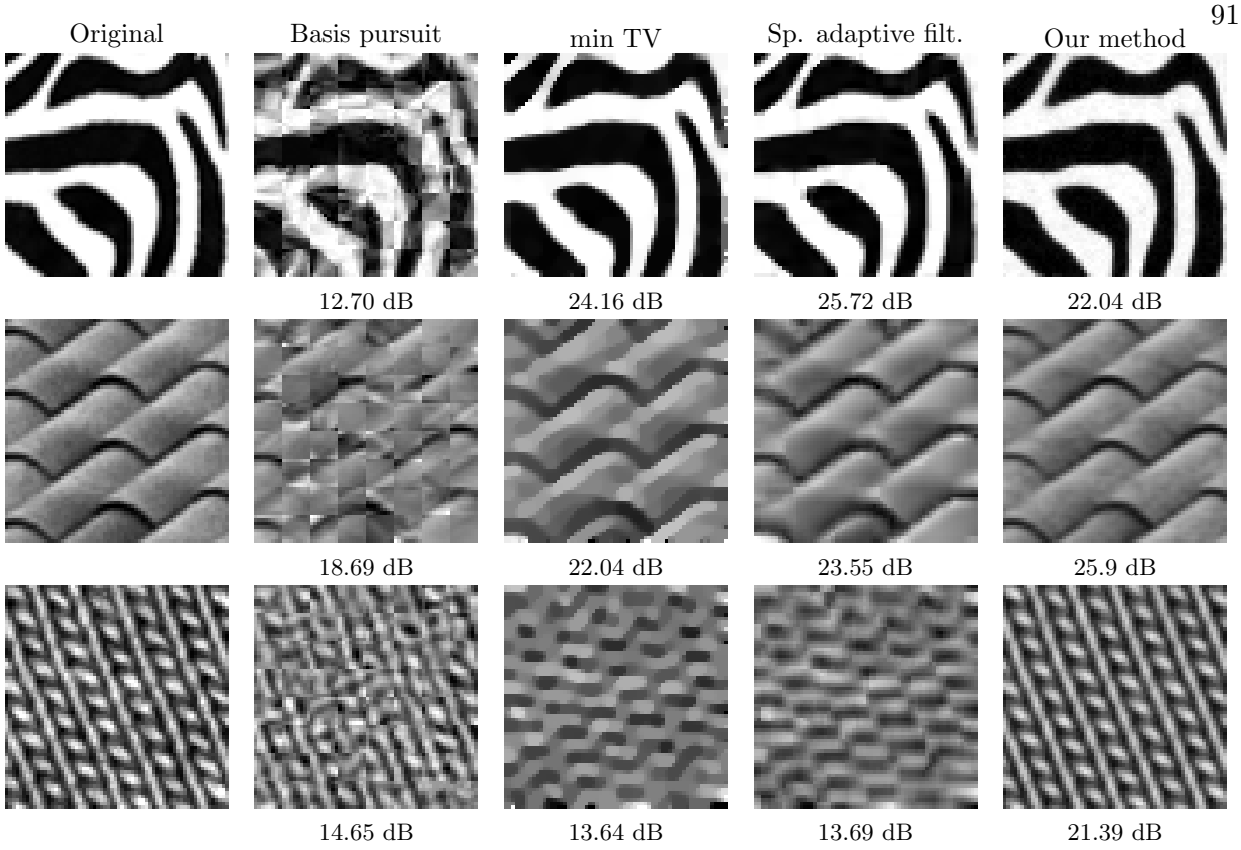


Figure 5.10: Compressive sensing reconstruction. The results of our algorithm are consistent with the learned model and nearly perfectly match the original images. Basis pursuit reconstruction is obtained from 448 Bernoulli measurements (7 separate measurements for each of 64 8×8-pixel regions); notice the tesselation artifacts resulting from this choice of measurement matrix. Spatially adaptive filtering based on the state-of-the-art BM3D algorithm [71] as well as the Total Variation minimization approach are initialized with 400 low-frequency Fourier measurements. We use 400 random Bernoulli measurements in our method. Numbers represent PSNR.

vertical and horizontal directions with successive averaging, and use Eq. 5.14 to keep the values of known pixels unchanged on each iteration. We see that our results (Fig. 5.12) are similar to or better then those of the exemplar-based method of Wexler et al. [205]. Their method maximizes global consistency between patches in the gap and the reference region and can be viewed as a special case of bidirectional similarity distance minimization for the purpose of inpainting [10, 181].

When evaluating each of the experiments just shown, and comparing our results to other methods, it is important to note that each of the comparison methods are specialized algorithms tailored for a specific inverse problem (such as BM3D for denoising, which includes a complex post-

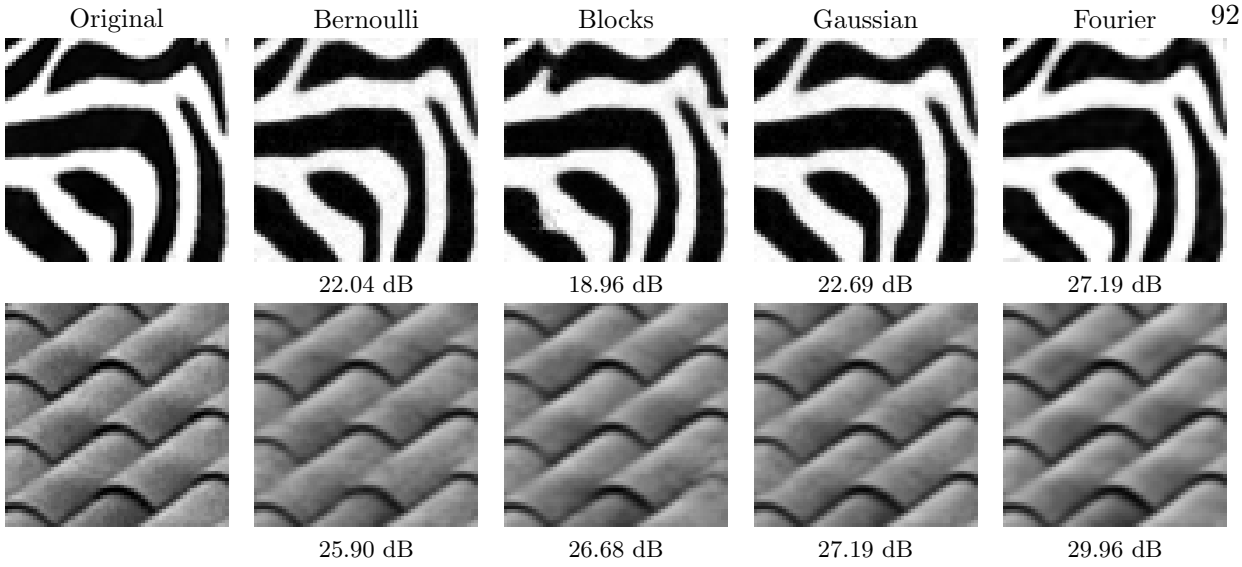


Figure 5.11: Comparison of compressive sensing results obtained with our method using different measurement matrices: 400 Bernoulli random measurements; 7 Bernoulli random measurements for each of 64 8×8 non-overlapping blocks (448 measurements total); 400 Gaussian random measurements; 422 low-frequency Fourier measurements. Numbers correspond to PSNR.

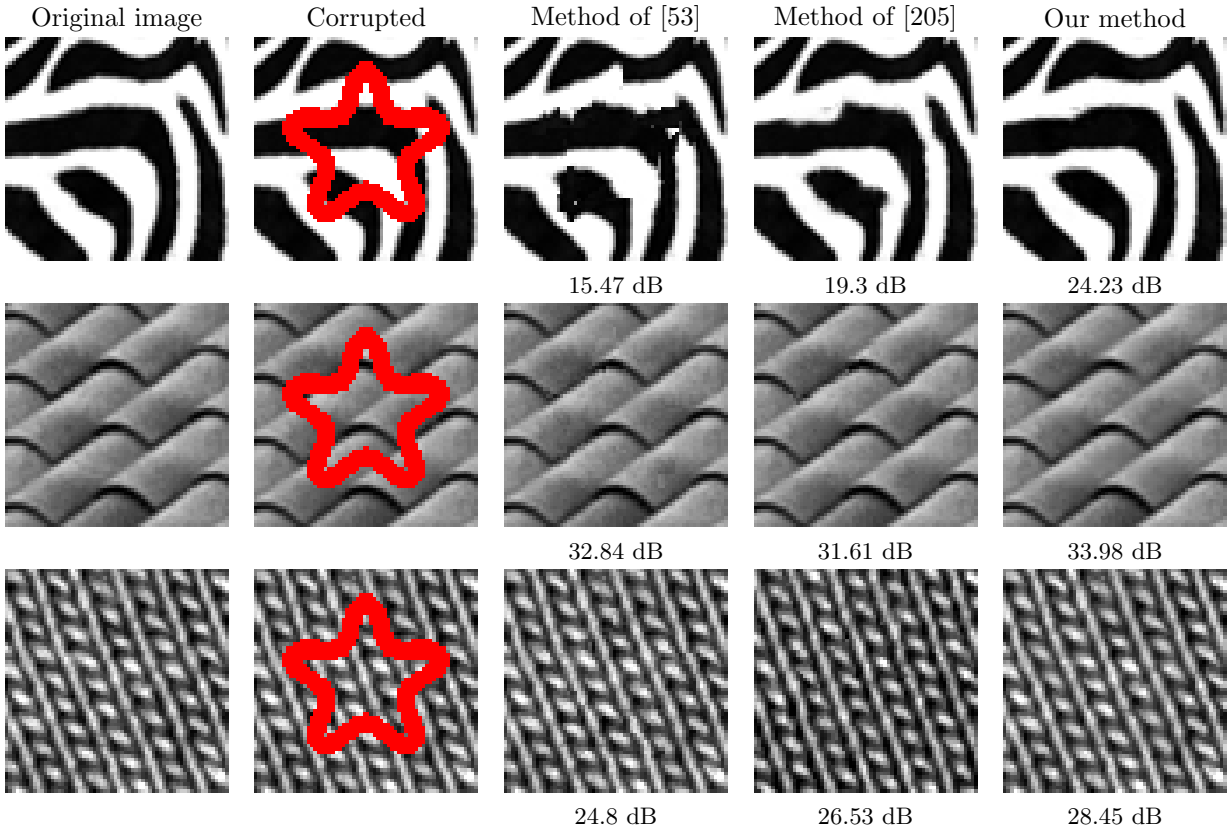


Figure 5.12: Results of image inpainting. Our algorithm outperforms other patch-based approaches of Criminisi [53] and Wexler et al. [205] with improved visual quality. Numbers represent PSNR.

processing stage with a specially designed collaborative Wiener filter). We, however, have applied *the same method without modifications* across all these different inverse problems. Hence, rather remarkably, we have remained close to or better than state-of-the-art algorithms for each different inverse problem, without the benefit of any application-specific post-processing or fine-tuning that our competitors might have.

5.6 Processing Entire Photographic Images

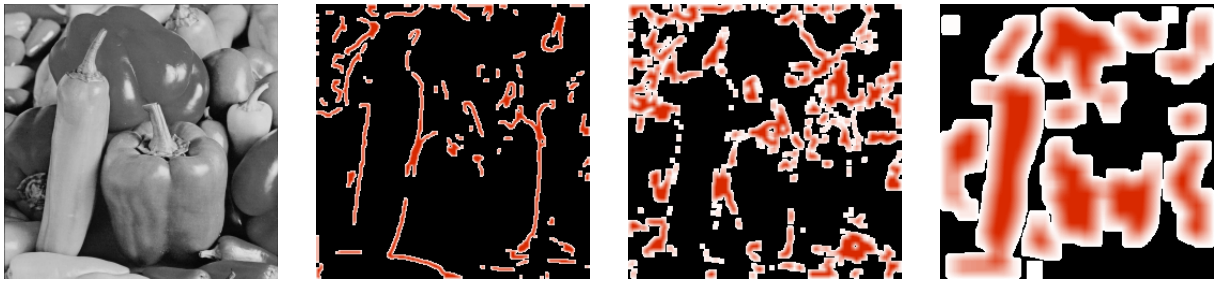
In this section we show how our flexible intersecting manifolds model can be extended beyond processing simple textures and patterns and successfully adapted to entire natural images of photographic quality. For this, we will rely on our method's ability to easily use separate manifolds for patches of different sizes. Indeed, we propose a multiscale decomposition scheme to effectively represent images with patches representing multiple levels of detail.

5.6.1 Multiscale Patch Decomposition

In all previous examples (with a notable exception of Sec. 5.5.3.1) we restricted our attention only to specific parts of natural images that contained patterns and textures. Their structure allowed us to describe the patches with relatively simple manifold models. In a more general case, patches of entire photographic images exhibit much higher variability across the dataset. This implies higher intrinsic dimension of the underlying manifold and requires one to use more training samples to accurately learn its geometry. We describe a way to overcome this obstacle and make our manifolds intersection model applicable in this setting as well.

First, we will cover an image with patches of different sizes. We will use smaller patches to reconstruct high-variance image regions, such as sharp edges and detailed patterns. On the other hand, smooth gradients and almost uniformly filled areas will be approximated with larger patches of subsequently lower variances. For this, we learn four manifolds: the manifold of 3×3-pixel patches, whose pixel variances are greater than $\theta_1 = 0.03$, and the manifolds of 5×5, 9×9, and 17×17-pixel patches with variances less than $\theta_1 = 0.03$, $\theta_2 = 0.013$, and $\theta_3 = 0.01$ respectively.

During reconstruction, we adaptively assign types of patches: on each step of the iterative algorithm, we compute local variances of every patch position in the image and cover each region with the largest possible patch that satisfies the above thresholds (see Fig. 5.13). Specifically, starting with the smallest patches, we use a particular patch $\mathbf{P}$ of the i^{th} scale (with $i = 1 \dots 4$) if its variance satisfies $\theta_i < var(\mathbf{P}) < \theta_{i-1}$, and if it does not completely cover any already chosen patches on finer scales (we assume $\theta_0 = \infty$ and $\theta_4 = 0$). Otherwise, the patch $\mathbf{P}$ is not included in computing ∇J in Eq. 5.8. To avoid oversmoothing and put more emphasis on reconstructing sharp image features, we use $\mathbf{w} = [10, 1, 1, 1]$ for the different patch scales and also weight the gradient pixel-wise by the resulting number of overlapping patches.



Original img. 3×3 5×5 17×17

Figure 5.13: Distribution of patches of different sizes and variances adaptively chosen to cover the Peppers image in the process of denoising (the final iteration is shown). Color intensity encodes the number of overlapping patches in each pixel: white – one patch, red – pq patches, black – no patches. Smaller high-variance patches are used to reconstruct sharp edges, while large 17×17 patches cover uniform smooth image regions. (9×9 patches are not shown.)

Second, to accurately learn the manifold geometry with KPCA, we would like to ensure that the manifolds are sampled uniformly. Therefore, instead of extracting training patch-samples from real-world images, we generate synthetic patches according to the parametrization of Fig. 2.1 and then vary their contrast and brightness to produce gray-scale patches. Figure 5.14 shows examples of patches generated in this way. We now apply the same unchanged method for all inverse problems, yet show that it compares favorably with specialized recent state-of-the-art methods for each.

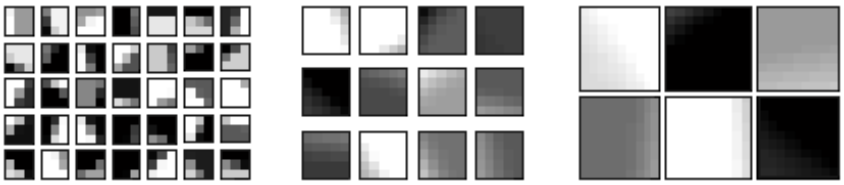


Figure 5.14: Examples of training patches used to learn the manifolds. From left to right: 3×3, 5×5, and 9×9-pixel patches; 17×17 patches are not shown. Note the decreasing pixel variance in larger patches.

5.6.2 Experimental Results on Natural Photographic Images

To test our multiscale patch-based approach for modeling complex natural images, we first consider a denoising problem. We compare against two types of other algorithms. First, as in Section 5.5.3, we consider other KPCA-based methods that can only be applied to each overlapping patch separately (with averaging to combine the results into a single image). For the most direct comparison, we break the image down into multiscale patches according to their variances as described above and train the Kernel PCA-based patch models in the same way as for our algorithm. Table 5.2 shows clearly the advantage of our *joint* estimation of the patches vs. estimating them individually via similar techniques and then combining.

Second, we compare against two recent denoising algorithms. Our performance typically is close to but falls a bit short of the state-of-the-art BM3D algorithm [54] on natural images. However, this is to be expected as our algorithm is not at all specialized for denoising while BM3D is, including e.g. specialized post-processing by Wiener filtering to boost PSNR. Notably, BM3D also assumes the noise variance is known, which our algorithm does not, so a more fair comparison might be against blind denoising approaches such as the recent Noise Clinic [122]. In any case, our results vividly demonstrate the particular suitability of the proposed method to handle noise of higher variances. While performance of other methods drops dramatically, our algorithm exhibits a certain robustness to high noise.

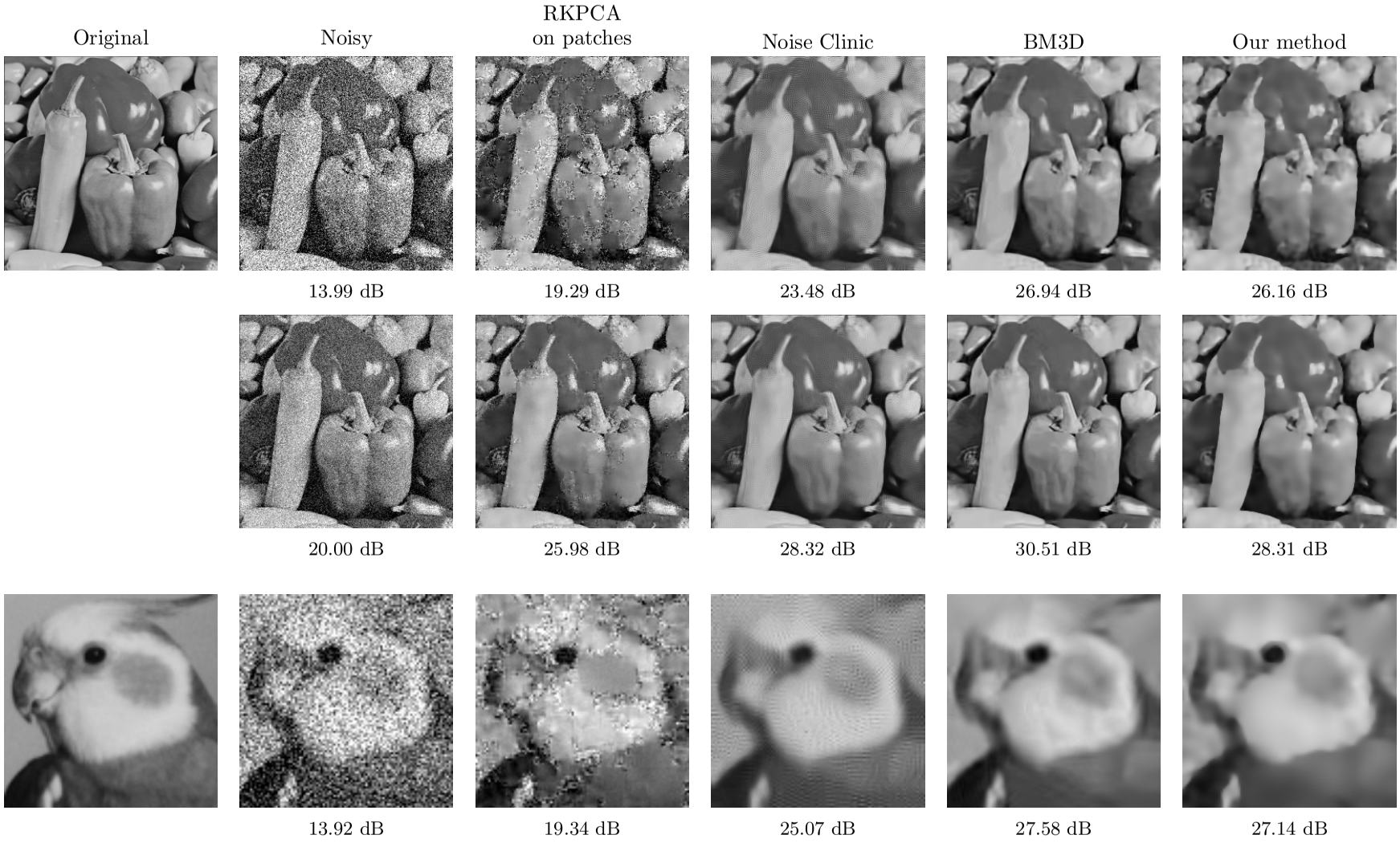


Figure 5.15: Results of denoising natural photographic images with our method described in Section 5.6.1. Based on a synthetically-generated model for patches, our method effectively handles high levels of noise. It outperforms another recently developed blind denoising algorithm, Noise Clinic [122], and approaches the state-of-the-art BM3D algorithm [54], which requires knowledge of the standard deviation of noise. Numbers represent corresponding PSNRs. Results for the *Peppers* images are shown with two different levels of noise.

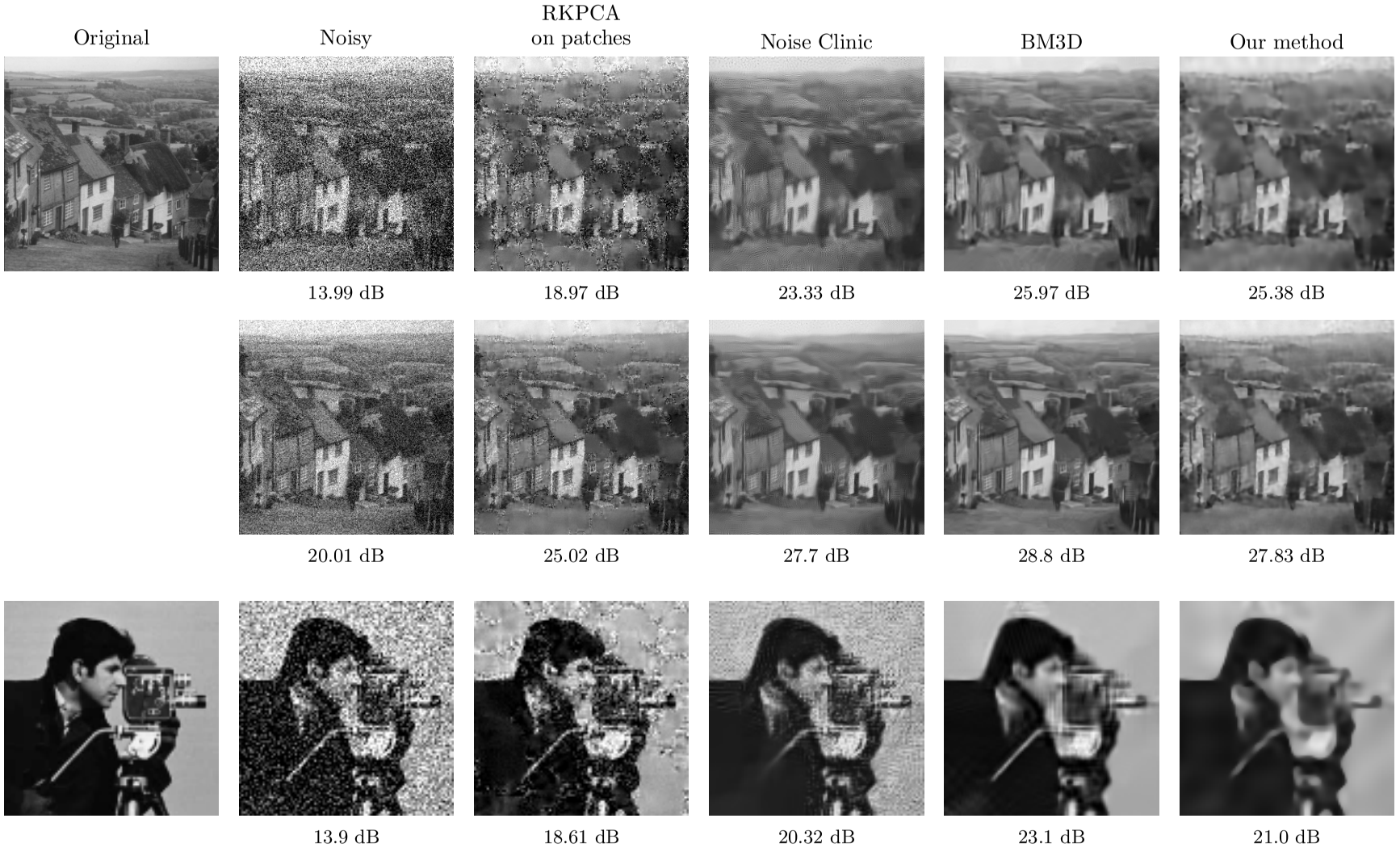


Figure 5.16: Results of denoising natural photographic images with our method described in Section 5.6.1. Based on a synthetically-generated model for patches, our method effectively handles high levels of noise. It outperforms another recently developed blind denoising algorithm, Noise Clinic [122], and approaches the state-of-the-art BM3D algorithm [54], which requires knowledge of the standard deviation of noise. Numbers represent corresponding PSNRs. Results for the *Goldhill* images are shown with two different levels of noise.

Table 5.2: Denoising performance under varying noise levels, PSNR in dB

	Noisy	KPCA-based methods					Noise Clnk.	BM3D
		Fixed pt.	MDS	RKPCA	Isomrph.	Our		
Peppers	20.00	24.78	27.74	25.98	27.84	28.31	28.32	30.51
	14.02	17.69	21.14	19.29	21.46	26.11	23.16	26.79
	10.49	14.63	18.00	16.52	18.55	23.85	19.97	24.90
Goldhill	20.01	23.82	26.49	30.62	26.58	27.83	27.7	28.80
	13.99	17.43	20.64	18.97	20.91	25.42	23.33	25.97
	10.46	14.30	17.70	16.20	18.23	23.07	20.88	24.51
Bird	19.93	24.29	27.53	25.65	27.65	30.47	29.61	31.11
	14.07	17.70	21.27	19.34	21.56	27.18	25.21	27.90
	10.41	13.81	17.23	15.71	17.76	25.35	22.72	25.75
Cam-man	19.93	23.4	26.43	24.74	26.50	23.05	25.18	27.60
	14.07	16.93	20.41	18.61	20.72	21.40	19.64	23.29
	10.41	13.35	16.81	15.32	17.36	19.56	16.2	21.07

The true merit of our algorithm, however, lies in its direct applicability to various other (constrained) inverse problems, such as compressive sensing reconstruction, where it readily outperforms established specialized methods. Results in Fig. 5.17 first demonstrate that our method achieves higher quantitative scores than the classic basis pursuit algorithm. Furthermore, compared to the recent state-of-the-art BM3D-based spatially adaptive filtering approach [56], it introduces noticeably fewer artifacts leading to improved visual quality and typically higher PSNR. Table 5.3 studies the performance of compressive sensing reconstruction (PSNRs in dB) achieved by the latter and our methods on varying number of low-frequency Fourier measurements of the *Peppers*, *Goldhill*, *Bird*, and *Cameraman* images. We see that our method outperforms [56] in 19 of 24 experiments, and that the methods are within .1 dB of each other in 3 of the other 5 experiments.

Finally, in image inpainting, our approach significantly outperforms the methods of [53] and state-of-the-art [205], as shown in Fig. 5.18.

Table 5.3: Results of compressive sensing reconstruction, PSNR in dB

Number of meas.		250	500	750	1000	1250	1500
Peppers	Our	**23.48**	26.24	**27.44**	**28.65**	**29.64**	**30.7**
	Sp.Filt.	23.19	**26.27**	27.03	27.96	28.88	29.92
Goldhill	Our	**22.15**	**25.8**	**26.82**	**27.58**	**28.23**	**28.97**
	Sp.Filt.	21.48	24.31	25.98	26.91	27.91	28.46
Bird	Our	**26.68**	**30.25**	31.25	32.39	**33.9**	**35.32**
	Sp.Filt.	26.2	30.06	**31.6**	**32.62**	33.83	34.62
Cam-man	Our	**19.05**	21.01	**21.7**	**22.27**	22.84	**23.43**
	Sp.Filt.	18.52	**21.1**	21.52	22.11	**22.85**	23.02

5.7 Conclusion

In this chapter, we proposed a unified method for regularization of any linear inverse problem in image processing based on an intersecting manifolds model of overlapping patches. We applied the kernel trick to efficiently approximate the patch-manifold in the induced feature space and combined the iterative preimage method with an intersection finding algorithm, ensuring the existence of a solution in the input space, which increased accuracy and robustness of our approach. We then proposed a Landweber-type iteration to allow this regularization term to be used with a variety of inverse problems in image processing. Finally, we also introduced a multiscale patch extension of our method for natural images.

Provided that the set of image patches is well-approximated by a manifold that can be learned in the kernel-induced feature space, our algorithm produces excellent results. Indeed, its experimental performance is close to or better than recent state-of-the-art methods for a variety of inverse problems, even though these other methods benefit from being specifically tailored to each individual problem. This shows that a manifold model of overlapping patches is an excellent choice for regularizing inverse problems in image processing.

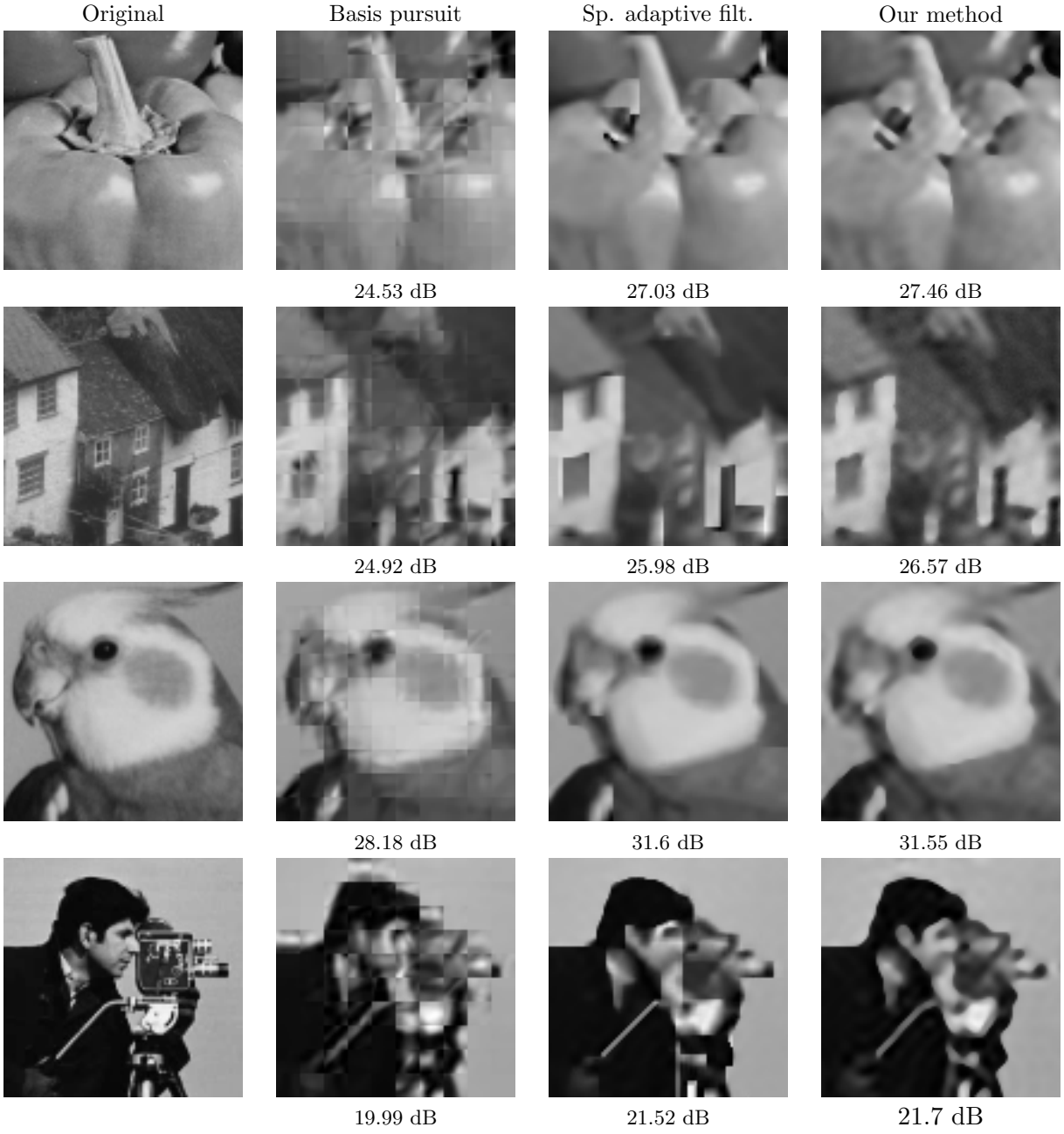


Figure 5.17: Results of compressive sensing reconstruction of natural images. Each 100×100 image is reconstructed from its 750 low-frequency Fourier measurements (measurement ratio is 7.5%). Our method achieves results similar to the spatially adaptive filtering based on the current state-of-the-art algorithm, BM3D. A traditional basis pursuit algorithm is also run on non-overlapping 8×8 image regions separately and uses a dictionary learned with KSVD. Numbers represent PSNR.

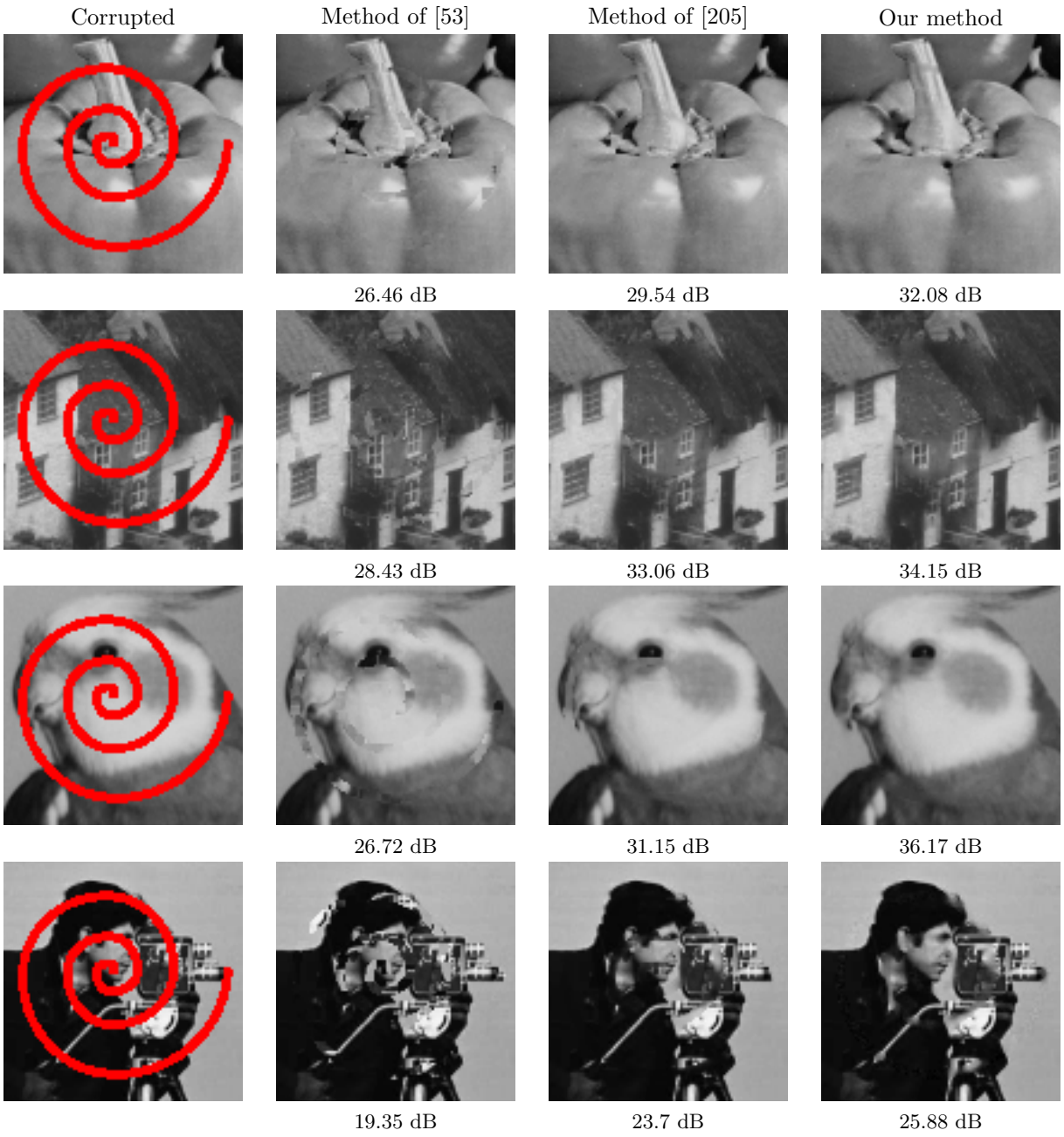


Figure 5.18: Results of inpainting natural images. Our patch-manifolds intersection method outperforms the other patch-based algorithms of Criminisi et. al. [53] and Wexler et. al. [205]. For the original images please refer to Fig. 5.17 on the left side of the page. Numbers represent PSNR.

Chapter 6

Kernel Orthogonal Direction Analysis: A New Learning Method Suited for Mapping onto Manifolds

In this chapter, we present a novel method for learning a manifold from its samples, which, by analogy with kernel PCA, we call Kernel Orthogonal Direction Analysis or KODA. Unlike in the usual manifold learning literature however, where the primary aim is to learn a lower-dimensional embedding function of the samples that reflects their organization along the manifold, our goal will be instead to learn a description of the continuous underlying manifold in the original high-dimensional space. This description will take the form of a level set of a (possibly vector-valued) function. Although we take a simple kernel-based approach not unlike kernel PCA in structure, we find that this approach excels in a variety of applications, such as interpolation along and mapping nearby points onto a manifold, where the usual manifold learning algorithms (including kernel PCA) are often applied, but with difficulty. Furthermore, it also produces its own distinctive form of dimensionality reduction that is effective in applications such as anomaly detection, classification, and ranking.

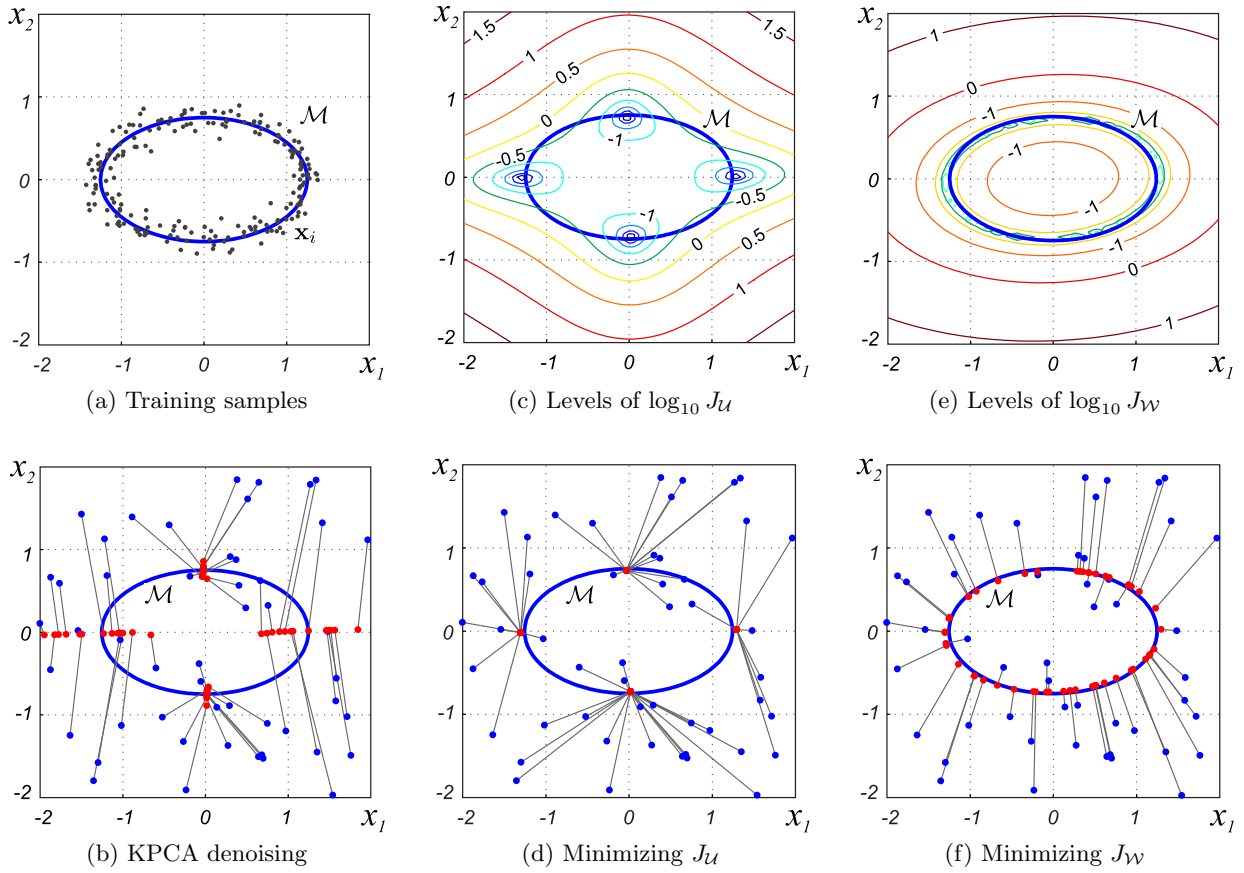
6.1 Motivation for the Approach

In the previous chapter, we have seen that minimizing the distance approximating functional $J_{\mathcal{U}}$ effectively finds an approximate manifolds intersection and shows excellent results in solving inverse problems in image processing. However, as we worked with it, we noticed that it has a noticeable shortcoming inherent to the representation of the manifold in the feature space.

Specifically, the set of minima of the energy surface of $J_{\mathcal{U}}$ is often finite. This causes the minimization algorithm to converge to one of these discrete minimizers, instead of arriving at the closest point on the continuous manifold (see Fig. 6.1). Although this does not deteriorate our performance in image processing applications, where relatively low dimensional manifolds seem to be well approximated even with a discrete (albeit large) set of minima of $J_{\mathcal{U}}$, such unwanted pathological convergence of the algorithm is especially conspicuous when dealing with manifolds of low *codimension* (i.e. where there is a small difference between the dimension of the ambient space and the intrinsic dimension of the manifold, $\bar{d}_{\mathcal{M}} = D - d_{\mathcal{M}}$).

As a motivating example for our approach, we start with the common problem of needing to denoise a sample by mapping it to the nearest point on a nearby manifold. We lack an explicit description of the manifold, but have a set of its samples from which to estimate it. Specifically, we generate $n_{\mathrm{X}} = 200$ training samples of an ellipse in $\mathbb{R}^2$ corrupted with synthetic Gaussian noise as $\begin{bmatrix} x_1 & x_2 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} a\cos(t) & b\sin(t) \end{bmatrix}^{\mathrm{T}} + \mathcal{N}\left(\mathbb{0},\, \sigma^2\right)$ for some fixed a, b, $\sigma^2 > 0$ (we use $a = 1.25$, $b = 0.75$, and $\sigma^2 = 0.1$ to produce the plots in this section) and the parameter $t \in [0 \ldots 2\pi]$. To learn this manifold, we use the homogenous quadratic kernel $\kappa(\mathbf{x},\, \mathbf{y}) = \langle \mathbf{x},\, \mathbf{y} \rangle^2$ and represent the ellipse as a one-dimensional subspace in the induced feature space by retaining only the leading eigenvector in the decomposition of the centered kernel matrix $\bar{\mathbf{K}}$.

We now look at two popular manifold mapping methods, kernel PCA denoising [143], which projects the image of a noisy point onto the principal subspace $\mathcal{U}$ in feature space and then estimates its preimage, and Robust KPCA [151], which involves minimization of $J_{\mathcal{U}}$, a functional on the original space estimating distance in feature space to the subspace $\mathcal{U}$ (please see Section 3.2.3.2 for more details about preimage methods). Studying their performance on our simple toy problem reveals significant preimage error for the first approach and puzzling behavior for the second (see Fig. 6.1). Here, even though the minimizers of $J_{\mathcal{U}}$ now do lie more or less directly on the true manifold, they comprise a finite set of only four points. Any minimization algorithm converges to one of these points, which clearly does not approximate the minimum distance mapping onto the manifold.



(a) Training samples (c) Levels of $\log_{10} J_{\mathcal{U}}$ (e) Levels of $\log_{10} J_{\mathcal{W}}$

(b) KPCA denoising (d) Minimizing $J_{\mathcal{U}}$ (f) Minimizing $J_{\mathcal{W}}$

Figure 6.1: An example of learning an ellipse from 200 noisy samples of it in $\mathbb{R}^2$. (a) The original manifold and noisy samples. (b) Results of projecting a cloud of random points onto the KPCA-parametrized subspace $\mathcal{U}$ in feature space, then finding a preimage via [143]. (c) Level curves of $\log_{10} J_{\mathcal{U}}(\cdot)$. (d) Mapping the points onto the manifold by minimizing $J_{\mathcal{U}}$ with gradient descent as done by Robust KPCA [151]. Instead of landing near the respective closest points on the manifold, iterations converge to one of the four distinct minimizers of $J_{\mathcal{U}}$ (red points). (e) Minimizers of our $J_{\mathcal{W}}$ form a continuous curve that well approximates the original manifold. (f) Minimizing $J_{\mathcal{W}}$ maps points close to their true projections.

While it can be argued that choosing a different kernel or increasing the dimension of the subspace $\mathcal{U}$ would produce better results, we have used the simple settings in this example intentionally to expose this pathology, as well as to have the ability to visualize its cause in the next section. We note that this problem is specific to the method itself and manifests itself with other kernels as well (see Section 6.5.1 for examples using the Gaussian kernel). Furthermore, we will show that the expressive power of the chosen quadratic kernel is sufficient to learn the ellipse exactly in terms of only a single vector in the feature space and, as a result, to efficiently solve the

problem of mapping points onto it.

To conclude this subsection, we have seen that parameterization of the approximating subspace with its principal components frequently leads to an approximation of a continuous manifold with a discrete set of points. In what follows, we will develop an approach to define and learn a richer subspace in the feature space that will allow us to approximate the above ellipse as a continuous one-dimensional set of points; then we will generalize our model to other manifolds.

6.1.1 View of the Problem in Feature Space

To better understand the cause of this strange behavior, let us now visualize our example of learning the ellipse from its samples directly in the feature space $\mathcal{H}$. For the two-dimensional input space, the chosen quadratic kernel is associated with the following mapping:

$$\begin{aligned} &\Phi : \mathbb{R}^2 \to \mathcal{H} \\ &\Phi : \begin{bmatrix} x_1 \\ \\ x_2 \end{bmatrix} \longmapsto \begin{bmatrix} x_1^2 \\ \\ x_2^2 \\ \\ \sqrt{2}x_1x_2 \end{bmatrix}. \end{aligned} \tag{6.1}$$

Indeed, it can be easily verified that the inner product in the feature space is expressed through the kernel function as:

$$\begin{aligned} \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_{\mathcal{H}} &= \begin{bmatrix} x_1^2 \\ \\ x_2^2 \\ \\ \sqrt{2}x_1x_2 \end{bmatrix}^{\mathrm{T}} \cdot \begin{bmatrix} y_1^2 \\ \\ y_2^2 \\ \\ \sqrt{2}y_1y_2 \end{bmatrix} \\ &= x_1^2y_1^2 + 2x_1y_1x_2y_2 + x_2^2y_2^2 \\ &= (x_1y_1 + x_2y_2)^2 = \kappa(\mathbf{x}, \mathbf{y}). \end{aligned}$$

We visualize the induced three-dimensional feature space $\mathcal{H}$ with a Cartesian 3D coordinate system in Fig. 6.2. Here, the conical surface $\mathcal{I}$ represents the image of the entire original space $\mathbb{R}^2$

under the mapping Φ, i.e. the set of points with exact preimages; the red curve on this surface is the image of the considered ellipse. Finally, the subspace $\mathcal{U}$ (the black line) depicts the one-dimensional least-squares approximation of the ellipse found with PCA in $\mathcal{H}$.

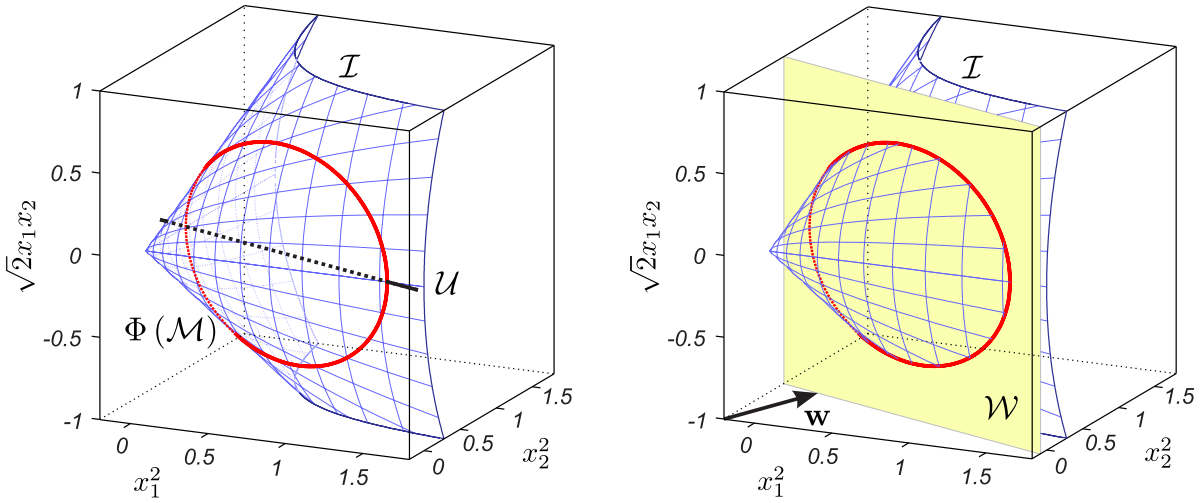


Figure 6.2: The feature space associated with the quadratic homogeneous kernel, $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle^2$ for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$. The surface $\mathcal{I}$ represents the image of the input space under the mapping Φ; $\Phi(\mathcal{M})$ is the image of the ellipse. Left: Approximation of the ellipse with a one-dimensional principal subspace $\mathcal{U}$ given by KPCA. Right: Approximation of the ellipse with a richer subspace $\mathcal{W}$. Notice that $\mathcal{U}$ intersects the image of the original space in only two pints, whereas $\mathcal{W}$ forms a continuous intersection with $\mathcal{I}$.

Observing Fig. 6.2, we see that projection of manifold samples $\Phi(\mathcal{M})$ onto $\mathcal{U}$ will work relatively well as a low-dimensional embedding strategy: each half of the ellipse is mapped continuously onto the line in the right arrangement. However, as the foundation for an inverse mapping, this subspace is a poor choice because of the relative geometry of $\mathcal{U}$ and $\mathcal{I}$. In particular, the intersection $\mathcal{U} \cap \mathcal{I}$, i.e. the set of points on $\mathcal{U}$ that have exact preimages, is just two points (corresponding to four points in $\mathbb{R}^D$ since $\Phi(\mathbf{z}) = \Phi(-\mathbf{z}), \forall \mathbf{z}$). KPCA denoising strategies consider these four points to be the only ones actually on the manifold, which eventually results in the four distinct minima of $J_{\mathcal{U}}$ observed in Fig. 6.1 and forces a minimization algorithm to map any noisy sample into one of them. Unfortunately, this pathology is not unique to this example but arises from the situation's geometry: curved, nonlinear $\mathcal{I}$ generally intersects linear $\mathcal{U}$ in a bunch of smaller disconnected pieces, much like the zero crossings of a nonlinear function, creating a correspondingly piecemeal

manifold representation in the input space (see Fig. 6.3 for examples with Gaussian kernels).

A better strategy for our interests would be to raise the dimension of the subspace so that the set $\mathcal{U} \cap \mathcal{I}$ is actually a continuous $d_{\mathcal{M}}$-dimensional manifold, guaranteeing a corresponding continuous manifold of its exact preimages in $\mathbb{R}^D$. This is less applicable for dimensionality reduction, but better for settings in which continuity of the manifold description is important. Theoretically, kernel PCA with more dimensions in $\mathcal{U}$ could accomplish this, but in practice (see Fig. 6.3), an unknown, very large, number of dimensions is required before $\mathcal{U} \cap \mathcal{I}$ even begins to approach a continuous manifold, and it is hard to predict how many dimensions it will take. Fortunately however, the intersection of a continuous D-dimensional manifold $\mathcal{I}$ with a subspace $\mathcal{W}$ of *codimension* $\bar{d}_{\mathrm{W}}$ is typically a continuous $(D-\bar{d}_{\mathrm{W}})$-dimensional manifold [91]. (Exceptions only arise when $\mathcal{W}$ does not intersect $\mathcal{I}$, or is tangential to it.) Thus, seeking a subspace $\mathcal{W}$ of *codimension* $\bar{d}_{\mathrm{W}} = D - d_{\mathcal{M}}$ will make it highly probable that the representation $\mathcal{W} \cap \mathcal{I}$ yields the desired continuous $d_{\mathcal{M}}$-dimensional manifold. In our example above, for instance, most one-dimensional subspaces $\mathcal{U}$ intersect $\mathcal{I}$ in one or two disconnected points or do not intersect it at all; on the other hand, most subspaces $\mathcal{W}$ of codimension 1 that intersect $\mathcal{I}$ define continuous one-dimensional manifolds in it.

In very small feature spaces, we can still describe a subspace of codimension $\bar{d}_{\mathrm{W}}$ using its basis vectors. However, in the high or infinite-dimensional feature spaces corresponding to most kernels of interest, the dimension needed for this type of *bottom up* strategy will be prohibitively high. Instead, we will need to employ a distinctive *top down* approach, specifying the subspace $\mathcal{W}$ via its n_{W} normals $\mathbf{W} \in \mathbb{R}^{D_{\mathcal{H}} \times n_{\mathrm{W}}}$ and offsets $\mathbf{b} \in \mathbb{R}^{n_{\mathrm{W}}}$ as $\mathcal{W} = \{\boldsymbol{\phi} \in \mathcal{H} \,|\, \mathbf{W}^{\mathrm{T}}\boldsymbol{\phi} = \mathbf{b}\}$.

This choice will prove to have myriad advantages. First, it immediately yields an explicit functional description of the learned manifold in input space. For example, choosing the single normal $\mathbf{W}^T = [1/a^2,\ 1/b^2,\ 0]$ above allows us to describe the ellipse $x_1^2/a^2 + x_2^2/b^2 = c$ exactly. Similarly, any manifold that can be described by a set of n_{W} polynomial equations of degree δ can be learned exactly using n_{W} normals in the feature space induced by the polynomial kernel of degree δ. Later, we will show how, more generally, choice of kernel determines the class of manifolds it can potentially approximate.

Thus, the proposed subspace definition will allow us to obtain a continuous description for manifolds, as opposed to the usual parameterization with principal components in feature space. Furthermore, we will show how to effectively use it as a regularization term for the preimage problem, enforcing a found preimage to lie on a desired manifold, in the flavor of the Robust KPCA algorithm [151]. But first, we proceed with developing a generalized learning procedure for learning the subspace $\mathcal{W}$ from manifold samples.

6.2 Learning the Subspace

In this section we will present our approach, Kernel Orthogonal Direction Analysis (KODA), for learning the approximating subspace $\mathcal{W}$ in terms of its normal vectors.

6.2.1 Learning $\mathcal{W}$ as an Optimization Problem

We first will establish a quantitative criterion for the desired manifold-approximating subspace $\mathcal{W}$ given samples of the manifold. In particular, we will set up an optimization problem for the normals $\mathbf{W}$ and offsets $\mathbf{b}$ and show how to solve it to find the optimal $\mathcal{W}$. Then, we will use this parameterization to compute the distance to the found subspace in terms of only inner products and will define a functional $J_{\mathcal{W}}$, similar to $J_{\mathcal{U}}$. However, in contrast to $J_{\mathcal{U}}$, our new functional will have a *continuous* set of minimizers forming a suitable representation for a manifold. Thus, we will be able to use gradient descent to minimize $J_{\mathcal{W}}$ and to map any point $\mathbf{z}$ in the input space onto the learned manifold approximation. Finally, we will show how to use this manifold learning and representation strategy for a variety of purposes including, projection onto a manifold, tracing paths on it, and classification.

We first consider a case of learning a manifold of codimension 1 from its noiseless samples (and thus will be working with a vector $\mathbf{w}$ instead of the matrix $\mathbf{W}$); we then generalize the method to easily learn manifolds of higher codimensions.

Consider a non-linear $(D-1)$-dimensional manifold $\mathcal{M} \subset \mathbb{R}^D$ given by its samples $\mathbf{x}_i \in \mathcal{M}$, $i = 1, \ldots, n_{\mathrm{X}}$. In the feature space $\mathcal{H}$ induced by an appropriate kernel function $\kappa(\cdot, \cdot)$, we will aim to

find a $\mathbf{w} \in \mathcal{H}$ and $b \in \mathbb{R}$ so that the hyperplane they describe $\mathcal{W} = \left\{\boldsymbol{\phi} \in \mathcal{H} \,|\, \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi} = \mathrm{b}\right\}$ satisfies the following two conditions. First, we want the images in feature space of all the manifold's samples $\mathbf{x}_i$ to lie in the hyperplane $\mathcal{W}$, i.e. $\mathbf{w}^T\Phi(\mathbf{x}_i) = \mathrm{b}$ for all $i = 1, \ldots, n_{\mathrm{X}}$. Second, without loss of generality, we wish to assume $\|\mathbf{w}\|_{\mathcal{H}}^2 = 1$ to remove an arbitrary degree of freedom that merely scales $\mathbf{w}$ and b without changing the described hyperplane. We note that in the case of multiple orthogonal directions this assumption translates into the requirement of their orthonormality, $\mathbf{W}^{\mathrm{T}}\mathbf{W} = \mathbf{I}$, which we make to solve the problem efficiently.

Thus far, our problem is likely underdetermined. Particularly in high- or infinite-dimensional feature spaces there will be many possible hyperplanes containing the n_{X} desired samples. To ensure the choice of a desirable hyperplane out of these, we should not only encourage $\mathcal{W}$ to contain the image of the manifold, $\Phi(\mathcal{M})$, but also to contain as little as possible of the rest of $\mathcal{I}$ – the image of the input space $\mathbb{R}^D$ under the mapping $\boldsymbol{\Phi}$. To achieve this, we generate a second set of samples $\mathbf{y}_j \in \mathbb{R}^D$, $j = 1, \ldots, n_{\mathrm{Y}}$. While we will discuss the choice of these further, for now, we ask the reader to think of these as randomly generated samples of $\mathbb{R}^D$ (e.g. from a Gaussian distribution) that are thus off-manifold samples almost surely. We then encourage desirable properties of $\mathcal{W}$, i.e. that its normal $\mathbf{w}$ aligns well with the rest of the space $\mathcal{I}$, making $\mathcal{W}$ itself avoid it, by maximizing $\sum_{j=1}^{n_{\mathrm{Y}}}\langle \mathbf{w}, \Phi\left(\mathbf{y}_j\right)\rangle^2$ subject to the two equality constraints above:

$$\begin{aligned} \max_{\mathbf{w},\mathrm{b}} \quad & \sum_{j=1}^{n_{\mathrm{Y}}}\langle \mathbf{w}, \Phi\left(\mathbf{y}_j\right)\rangle^2 \\ \text{subject to} \quad & \mathbf{w}^T\Phi(\mathbf{x}_i) = \mathrm{b}, \ \textit{for all } i = 1, \ldots, n_{\mathrm{X}}, \\ & \|\mathbf{w}\|_{\mathcal{H}}^2 = 1. \end{aligned}$$

In the noisy case, when the samples $\mathbf{x}_i$ can not be guaranteed to lie on the manifold exactly, the hard constraint $\Phi\left(\mathbf{x}_i\right) \in \mathcal{W}$ may lead to poor results. To prevent this, we interchange the objective function and the constraint $\|\mathbf{w}\|_{\mathcal{H}}^2 = 1$ in the problem and produce its equivalent formulation (provided the found solution is normalized afterwards) by minimizing $\mathbf{w}^T\mathbf{w} = ||\mathbf{w}||_{\mathcal{H}}^2$ subject to an

equality constraint on $\sum_{j=1}^{n_Y}\langle \mathbf{w}, \Phi\left(\mathbf{y}_j\right)\rangle^2$. We then relax the remaining constraint to obtain:

$$\begin{aligned} \min_{\mathbf{w},\mathrm{b}} \quad & \theta \sum_{i=1}^{n_X} \left(\mathbf{w}^{\mathrm{T}}\Phi\left(\mathbf{x}_i\right) - \mathrm{b}\right)^2 + (1-\theta)\,\mathbf{w}^{\mathrm{T}}\mathbf{w} \\ \text{subject to} \quad & \sum_{j=1}^{n_Y}\langle \mathbf{w}, \Phi\left(\mathbf{y}_j\right)\rangle^2 = \lambda, \end{aligned} \tag{6.2}$$

where the regularization parameter $0 \leq \theta \leq 1$ specifies the desired level of adherence to noisy samples. By taking the derivative of the objective function with respect to b and setting it to 0, the optimal value of b is found as $\mathrm{b} = \mathbf{w}^{\mathrm{T}}\mathbf{m}$, where $\mathbf{m}$ is the sample mean defined in Section 3.1.

We note that our proposed objective function resembles the least-squares formulation of the popular one-class Support Vectors Machines (LS-SVM) algorithm [49, 187]. One-class LS-SVM is not designed for manifold learning, but rather for classification of samples, when training examples are available for only one class. However, in its training process, instead of looking for a decision boundary, it attempts to fit a hyperplane (i.e. a subspace of codimension 1) as close as possible to the samples of interest. As we saw, this gives rise to a continuous manifold closely tracing the training samples in the input space. Like our method, the sought hyperplane is characterized in terms of its normal vector, thus resulting in an optimization problem similar to Eq. 6.2.

However, there are several essential differences between the one-class LS-SVM and KODA. First, in contrast to LS-SVM, we have designed our method to learn subspaces (and hence manifolds) of arbitrary codimensions, not just hyperplanes, which always give manifolds of codimension 1 (see Section **??**). Hence, we may, for example, learn one-dimensional curves in $\mathbb{R}^3$ just as easily as two-dimensional surfaces. Second, even more importantly, in our approach, we propose to use off-manifold samples $\mathbf{y}_j$ to build and represent the solution instead of expanding it in the limited span of only the available training samples $\mathbf{x}_i$, as done by LS-SVM (and most kernel-based algorithms for that matter). We will see in the experimental results that this choice of basis offers significant advantages: it enriches the set of normal vectors (and thus the manifold descriptions) we can potentially recover. In feature space, this means that we are not trying to build a normal to a subspace entirely out of points that should be contained in that subspace, an awkward, if not impossible, task. In the original space, for the Gaussian kernel for example, this corresponds

to being able to simply describe a manifold as a smooth level set of a sum of Gaussians centered off-manifold, rather than undertaking the impossible task of trying to describe the manifold as a smooth level set of Gaussians centered on the manifold itself. In fact, our method results in a more accurate manifold representation than the one achieved with LS-SVM (see Section 6.5.1).

In the next sections, we will show how the optimization problem of Eq. 6.2 can be efficiently solved via generalized matrix eigendecomposition. Furthermore, we will restate it in terms of two simple eigenproblems, which allows us to develop in Section 6.4 an effective incremental algorithm for processing large high-dimensional datasets.

6.2.2 Generalized Eigendecomposition for Finding W

As in all kernel methods, we need to avoid explicitly working with $\mathbf{w}$ in feature space. An advantage of introducing the additional off-manifold samples $\mathbf{y}_j$ is that they provide a natural basis in which to express $\mathbf{w}$. Other kernel methods (such as kernel SVM, PCA, and Ridge Regression) typically find the solution in the span of images of the training samples $\Phi(\mathbf{x}_i)$, which is provably where their sought solutions lie. In our problem, however, we are looking for a vector $\mathbf{w}$ *orthogonal* to a subspace containing the manifold samples, and therefore it may not necessarily lie in span $[\Phi(\mathbf{x}_i)]$. Hence, we must introduce another basis for its expansion instead.

Our intended maximization of $\sum_{j=1}^{n_Y}\langle \mathbf{w}, \Phi(\mathbf{y}_j)\rangle^2$ indicates that $\mathbf{w}$ should align well with the images $\Phi(\mathbf{y}_j)$. In fact, for $\theta = 0$, the problem of Eq. 6.2 effectively becomes kernel PCA on the (uncentered) samples $\mathbf{y}_j$. Thus, with $\boldsymbol{\nu}$ standing for a vector of expansion coefficients, we will represent $\mathbf{w}$ as:

$$\mathbf{w} = \sum_{j=1}^{n_Y} \Phi(\mathbf{y}_j)\,\boldsymbol{\nu}_j. \tag{6.3}$$

Furthermore, it can be seen that the problem of finding a subspace in terms of its orthogonal components in a high-dimensional space is often ill-posed with an infinite number of solutions. In this case, we will see that the choice of the expansion points $\mathbf{y}_j$ will implicitly specify the regularizing properties of our algorithm. This will be discussed further in Sections 6.3.2 and 6.3.3.

We now continue by reformulating our constrained minimization problem of Eq. 6.2 as the

equivalent generalized eigendecomposition and then solve it for the expansion coefficients $\boldsymbol{\nu}$. With $\mathbf{w}$ assuming the form of Eq. 6.3, we rewrite the problem of Eq. 6.2 in matrix notation as:

$$\begin{aligned} \min_{\boldsymbol{\nu}} \quad & \theta \boldsymbol{\nu}^{\mathrm{T}} \mathbf{K}_{\mathrm{XY}}^{\mathrm{T}} \left[\mathbf{I} - \frac{1}{n_{\mathrm{X}}} \mathbb{1}\mathbb{1}^{\mathrm{T}} \right] \mathbf{K}_{\mathrm{XY}} \boldsymbol{\nu} + (1-\theta)\, \boldsymbol{\nu}^{\mathrm{T}} \mathbf{K}_{\mathrm{YY}} \boldsymbol{\nu} \\ \text{subject to} \quad & \boldsymbol{\nu}^{\mathrm{T}} \mathbf{K}_{\mathrm{YY}}^{2} \boldsymbol{\nu} = \lambda, \end{aligned}$$

where the elements of kernel matrices $\mathbf{K}_{\mathrm{XY}}$ and $\mathbf{K}_{\mathrm{YY}}$ are defined as $[\mathbf{K}_{\mathrm{XY}}]_{i,j} = \kappa(\mathbf{x}_i, \mathbf{y}_j)$ and $[\mathbf{K}_{\mathrm{YY}}]_{i,j} = \kappa(\mathbf{y}_i, \mathbf{y}_j)$.

The optimal coefficients $\boldsymbol{\nu}$ can now be found by solving for the eigenvector corresponding to the smallest non-zero eigenvalue in the generalized eigenproblem:

$$\left[\theta \mathbf{A}^{\mathrm{T}} \mathbf{A} + (1-\theta)\, \mathbf{B}\right] \boldsymbol{\nu} = \mathbf{B}\mathbf{B}\boldsymbol{\nu}\boldsymbol{\Lambda}, \tag{6.4}$$

where we have defined $\mathbf{A} = [\mathbf{I} - \frac{1}{n_{\mathbf{x}}} \mathbb{1}\mathbb{1}^{\mathrm{T}}]\mathbf{K}_{\mathrm{XY}}$ and $\mathbf{B} = \mathbf{K}_{\mathrm{YY}}$ for convenience.

6.2.2.1 Practical Reformulation of the Eigenproblem

Finally, we show how the problem of finding the expansion coefficients $\boldsymbol{\nu}$ can be cast and solved practically as two simple eigendecompositions. The advantages of such reformulation are eventually twofold. First, it will allow us to avoid recovering degenerate solutions from the potentially non-empty intersection of the null-spaces of the left- and right-hand sides of Eq. 6.4. Second, it will become crucial in the development of an incremental extension of our algorithm in Section 6.4.

Let us start by assuming that the points $\mathbf{y}_j$ are sampled such that their images in feature space span an r-dimensional subspace with $r \leq D_{\mathcal{H}}$. Thus, we may restrict the above problem to this subspace since the valid solution $\mathbf{w}$ will necessarily lie in it. For this, we first compute the eigendecomposition of the Gram matrix $\mathbf{B} = \mathbf{K}_{\mathrm{YY}} = \boldsymbol{\beta}\boldsymbol{\Lambda}_B\boldsymbol{\beta}^{\mathrm{T}}$, where $\boldsymbol{\beta}$ are the first r eigenvectors corresponding to the largest eigenvalues $\boldsymbol{\Lambda}_B$ (by assumption, the remaining $r+1, \ldots, n_{\mathrm{Y}}$ eigenvalues are effectively 0). Then the matrix $\Phi(\mathbf{Y})$ with columns $\Phi(\mathbf{y}_j)$ is factorized via its SVD as:

$$\Phi(\mathbf{Y}) = \underbrace{\left[\Phi(\mathbf{Y})\, \boldsymbol{\beta}\boldsymbol{\Lambda}_B^{-1/2}\right]}_{\mathbf{U}_r} \underbrace{\left[\boldsymbol{\Lambda}_B^{1/2}\right]}_{\boldsymbol{\Sigma}_r} \underbrace{[\boldsymbol{\beta}]^{\mathrm{T}}}_{\mathbf{V}_r^{\mathrm{T}}}. \tag{6.5}$$

We now may explicitly restrict $\mathbf{w}$ to the span of the $\Phi\left(\mathbf{y}_j\right)$'s. We recall its definition given in Eq. 6.3, $\mathbf{w} = \Phi\left(\mathbf{Y}\right)\boldsymbol{\nu}$, and note that due to the form of Eq. 6.5, any portion of $\boldsymbol{\nu}$ orthogonal to $\boldsymbol{\beta}$ (i.e. lying in the null-space of $\Phi\left(\mathbf{Y}\right)$) does not affect the final $\mathbf{w}$. Therefore, we may look for an $r \times 1$-vector of coefficients $\widetilde{\boldsymbol{\nu}}$, such that $\boldsymbol{\nu} = \boldsymbol{\beta}\widetilde{\boldsymbol{\nu}}$ (and thus $\mathbf{w} = \Phi\left(\mathbf{Y}\right)\boldsymbol{\beta}\widetilde{\boldsymbol{\nu}}$) instead of solving the problem of Eq. 6.4 for $\boldsymbol{\nu}$ directly. Using these definitions, and noting that because $\Phi\left(\mathbf{Y}\right) = \Phi\left(\mathbf{Y}\right)\boldsymbol{\beta}\boldsymbol{\beta}^T$, we have $\mathbf{A} = \mathbf{A}\boldsymbol{\beta}\boldsymbol{\beta}^{\mathrm{T}}$, the problem of Eq. 6.4 becomes:

$$\left[\theta\boldsymbol{\beta}\boldsymbol{\beta}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\boldsymbol{\beta}\boldsymbol{\beta}^{\mathrm{T}} + (1-\theta)\,\boldsymbol{\beta}\boldsymbol{\Lambda}_B\boldsymbol{\beta}^{\mathrm{T}}\right]\boldsymbol{\beta}\widetilde{\boldsymbol{\nu}} = \boldsymbol{\beta}\boldsymbol{\Lambda}_B^2\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{\beta}\widetilde{\boldsymbol{\nu}}\boldsymbol{\Lambda},$$

$$\left[\theta\boldsymbol{\beta}\boldsymbol{\beta}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\boldsymbol{\beta} + (1-\theta)\,\boldsymbol{\beta}\boldsymbol{\Lambda}_B\right]\widetilde{\boldsymbol{\nu}} = \boldsymbol{\beta}\boldsymbol{\Lambda}_B^2\widetilde{\boldsymbol{\nu}}\boldsymbol{\Lambda},$$

where we used the fact that the columns of $\boldsymbol{\beta}$ are orthonormal, $\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{\beta} = \mathbf{I}$. Since we are looking for a solution in span $(\boldsymbol{\beta})$, we may project this problem onto it by multiplying both sides of the above equation with $\boldsymbol{\Lambda}_B^{-2}\boldsymbol{\beta}^{\mathrm{T}}$ on the left. This finally results in the following simple eigenproblem, which we solve for the eigenvector associated with the smallest eigenvalue:

$$\left[\theta\boldsymbol{\Lambda}_B^{-2}\boldsymbol{\beta}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\boldsymbol{\beta} + (1-\theta)\,\boldsymbol{\Lambda}_B^{-1}\right]\widetilde{\boldsymbol{\nu}} = \widetilde{\boldsymbol{\nu}}\boldsymbol{\Lambda}. \tag{6.6}$$

We also note that since $r \leq n_{\mathrm{Y}}$, the size of the above problem is likely reduced comparing to the original formulation in Eq. 6.4.

To finish, the sought $\boldsymbol{\nu}$ is obtained by scaling $\boldsymbol{\beta}\widetilde{\boldsymbol{\nu}}$ to satisfy the constraint $\|\mathbf{w}\|_{\mathcal{H}}^2 = 1$, if desired: $\boldsymbol{\nu} = \boldsymbol{\beta}\widetilde{\boldsymbol{\nu}}[\widetilde{\boldsymbol{\nu}}^{\mathrm{T}}\boldsymbol{\beta}^{\mathrm{T}}\mathbf{B}\boldsymbol{\beta}\widetilde{\boldsymbol{\nu}}]^{-1/2}$. Finally, we see that the optimal b can now be expressed entirely in terms of inner products as:

$$\mathrm{b} = \frac{1}{n_{\mathbf{x}}}\boldsymbol{\nu}^{\mathrm{T}}\mathbf{K}_{\mathrm{XY}}^{\mathrm{T}}\mathbb{1}. \tag{6.7}$$

6.2.3 Extending the Solution to Codimensions Greater Than One

To generalize our method for manifolds of arbitrary codimensions in $\mathbb{R}^D$, we specify n_{W} normals to the affine subspace $\mathcal{W}$ as columns of the matrix $\mathbf{W}$, which leads to an optimization

problem similar to Eq. 6.2:

$$\begin{aligned} \min_{\mathbf{W},\mathbf{b}} \quad & \theta \sum_{i=1}^{n_{\mathrm{X}}} \left\| \mathbf{W}^{\mathrm{T}} \Phi\left(\mathbf{x}_i\right) - \mathbf{b} \right\|^2 + (1-\theta)\ \mathrm{trace}\left[\mathbf{W}^{\mathrm{T}}\mathbf{W}\right] \\ \text{subject to} \quad & \mathrm{trace}\left[\mathbf{W}^{\mathrm{T}} \Phi\left(\mathbf{Y}\right) \Phi\left(\mathbf{Y}\right)^{\mathrm{T}} \mathbf{W}\right] = \lambda. \end{aligned} \tag{6.8}$$

The matrix $\mathbf{W} = \Phi\left(\mathbf{Y}\right)\boldsymbol{\nu} = \Phi\left(\mathbf{Y}\right)\boldsymbol{\beta}\widetilde{\boldsymbol{\nu}}$ is then found by solving the same eigenproblem of Eq. 6.4, but now for the n_{W} eigenvectors corresponding to the smallest nonzero eigenvalues arranged in the matrix $\widetilde{\boldsymbol{\nu}}$. The resulting matrix of expansion coefficients $\boldsymbol{\nu}$ is then properly normalized as above, $\boldsymbol{\nu} = \boldsymbol{\beta}\widetilde{\boldsymbol{\nu}}[\widetilde{\boldsymbol{\nu}}^{\mathrm{T}}\boldsymbol{\beta}^{\mathrm{T}}\mathbf{B}\boldsymbol{\beta}\widetilde{\boldsymbol{\nu}}]^{-1/2}$, to make the columns of $\mathbf{W}$ orthonormal as we desired. The expression for $\mathbf{b}$, now an $n_{\mathrm{W}} \times 1$ vector, remains exactly the same as in Eq. 6.7.

6.3 Analysis of the Manifold Description

In this section, we will conduct a brief analysis of the form of the solution returned by our proposed KODA algorithm, similar to the discussion of kernel PCA in Sections 3.2 and 3.3. Specifically, we will see that the manifold $\mathcal{M}$ is learned in terms of level sets of certain continuous functions g_k, which guarantees the continuity of our manifold representation. Moreover, our corresponding functional $J_{\mathcal{W}}$, that approximates distance to the manifold, will also have a smooth, continuous set of minimizers. This will help with interpolation along and mapping smoothly onto the manifold. Furthermore, we will discuss the implications of the choice of the kernel function κ and of the expansion basis $\mathbf{y}_j$.

6.3.1 Resulting Description of the Manifold

Having found optimal $\boldsymbol{\nu}$ and $\mathbf{b}$, we can describe the manifold as the solution to the system of non-linear equations:

$$g_k\left(\mathbf{z}\right) = b_k \text{ for } k = 1, \ldots, n_{\mathrm{W}}, \tag{6.9}$$

where each function $g_k(\mathbf{z})$ takes the form:

$$g_k(\mathbf{z}) = \langle \mathbf{w}_k, \Phi(\mathbf{z}) \rangle_{\mathcal{H}} = \sum_{j=1}^{n_{\mathrm{Y}}} \boldsymbol{\nu}_{j,k} \kappa\left(\mathbf{y}_j, \mathbf{z}\right). \tag{6.10}$$

Specifically, for codimension 1, the manifold is learned as the level set of a single function g. We note that, in computer graphics, this is a common way to represent continuous surfaces in three dimensions reconstructed from their point-cloud samples [37, 112]. In fact, our method bears striking similarity with the methods of RBF interpolation [37]. However, it can operate on unorganized point-cloud data, does not require estimating normals to the surface, and can be applied to reconstruct continuous manifolds of any dimension in any space. For higher codimensions $\bar{d}_{\mathrm{W}}$, our approach describes the manifold as the intersection of several such level sets, each itself a manifold of codimension 1 (see Fig. 6.7). Indeed, assuming that the codimension 1 manifolds defined by each constraint intersect appropriately, our approximation will have the right codimension in the original space. (If they do not intersect, we may end up with a lower-dimensional manifold.)

We use the functions g_k to define an analog of $J_{\mathcal{U}}$, which approximates the distance to the manifold. However, in contrast to $J_{\mathcal{U}}$, which approximated this distance as the distance to the low-dimensional subspace $\mathcal{U}$ in feature space, we now approximate it as the distance to the subspace $\mathcal{W}$ having low *codimension*:

$$\begin{aligned} J_{\mathcal{W}}(\mathbf{z}) = d^2_{\mathcal{H}}(\Phi(\mathbf{z}), \mathcal{W}) &= \left\| \mathbf{W}\mathbf{W}^{\dagger}\Phi(\mathbf{z}) - \left(\mathbf{W}^{\dagger}\right)^{\mathrm{T}}\mathbf{b} \right\|^2_{\mathcal{H}} \\ &= \left[\mathbf{W}^{\mathrm{T}}\Phi(\mathbf{z}) - \mathbf{b}\right]^{\mathrm{T}} \left[\boldsymbol{\nu}^{\mathrm{T}}\mathbf{B}\boldsymbol{\nu}\right]^{-1} \left[\mathbf{W}^{\mathrm{T}}\Phi(\mathbf{z}) - \mathbf{b}\right] \\ &= \left[\mathbf{g}(\mathbf{z}) - \mathbf{b}\right]^{\mathrm{T}} \left[\boldsymbol{\nu}^{\mathrm{T}}\mathbf{B}\boldsymbol{\nu}\right]^{-1} \left[\mathbf{g}(\mathbf{z}) - \mathbf{b}\right], \end{aligned} \tag{6.11}$$

where $\mathbf{g}(\mathbf{z}) \in \mathbb{R}^{n_{\mathrm{W}}}$ is the vector of values $g_k(\mathbf{z})$, and $\mathbf{W}^{\dagger} = \left(\mathbf{W}^{\mathrm{T}}\mathbf{W}\right)^{-1}\mathbf{W}^{\mathrm{T}}$ is the Moore-Penrose pseudoinverse of $\mathbf{W}$. The functional $J_{\mathcal{W}}$ is a simple quadratic form in $\mathbf{g}(\mathbf{z})$, which can be easily computed in the input space. We minimize it, for example with steepest gradient descent, to map a point onto our learned continuous approximation of the manifold. Using the definition of $\mathbf{g}$ in Eq. 6.10, the gradient of this functional is expressed as:

$$\nabla J_{\mathcal{W}}(\mathbf{z}) \;=\; 2\left[\mathbf{k}'_{\mathrm{Yz}}\boldsymbol{\nu} - \mathbf{k}'_{\mathrm{Xz}}\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{K}_{\mathrm{XY}}\boldsymbol{\nu}\right]\left[\boldsymbol{\nu}^{\mathrm{T}}\mathbf{B}\boldsymbol{\nu}\right]^{-1}\left[\boldsymbol{\nu}^{\mathrm{T}}\mathbf{k}_{\mathrm{Yz}} - \boldsymbol{\nu}^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}_{\mathrm{XY}}\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{k}_{\mathrm{Xz}} - \mathbf{b}\right],$$

where the vectors $\mathbf{k}_{\mathrm{Xz}}$ and $\mathbf{k}_{\mathrm{Yz}}$ are defined with the entries $\left[\mathbf{k}_{\mathrm{Xz}}\right]_i = \kappa(\mathbf{x}_i, \mathbf{z})$ and $\left[\mathbf{k}_{\mathrm{Yz}}\right]_j = \kappa(\mathbf{y}_j, \mathbf{z})$ and $\mathbf{k}'_{\mathrm{Xz}}$ and $\mathbf{k}'_{\mathrm{Yz}}$ are respectively $D \times n_{\mathrm{X}}$ and $D \times n_{\mathrm{Y}}$ matrices of derivatives with columns $\left[\mathbf{k}'_{\mathrm{Xz}}\right]_{(:,i)} =$

$\nabla_{\mathbf{z}}\kappa\left(\mathbf{x}_i,\mathbf{z}\right)$ and $\left[\mathbf{k}'_{\mathrm{Yz}}\right]_{(:,j)} = \nabla_{\mathbf{z}}\kappa\left(\mathbf{y}_j,\mathbf{z}\right)$ for $i = 1,\ldots,n_{\mathrm{X}}$, $j = 1,\ldots,n_{\mathrm{Y}}$

We now compare the functional $J_{\mathcal{W}}$ with the analogous term for KPCA, $J_{\mathcal{U}}$, which defines the squared distance to the principal subspace $\mathcal{U}$ [174] (see Section 3.3.2 and Eq. 5.6). For this, we first recall the definition of Eq. 3.5 that expresses the inner products with the k^{th} principal component of $\mathcal{U}$ as a certain function $f_k\left(\cdot\right)$ for $k = 1,\ldots,d_{\mathcal{U}}$:

$$f_k\left(\mathbf{z}\right) = \left\langle \mathbf{u}_k, \Phi\left(\mathbf{z}\right)\right\rangle_{\mathcal{H}} = \sum_{i=1}^{n_{\mathrm{X}}} \boldsymbol{\alpha}_{i,k}\kappa\left(\mathbf{x}_i,\mathbf{z}\right). \tag{6.12}$$

Using this form of the KPCA solution as a linear combination of kernels supported on data samples, the distance approximating functional $J_{\mathcal{U}}$ becomes:

$$\begin{aligned} J_{\mathcal{U}}(\mathbf{z}) &= \left\|\Phi\left(\mathbf{z}\right) - P_{\mathcal{U}}\left(\mathbf{z}\right)\right\|_{\mathcal{H}}^2 = \left\|\Phi\left(\mathbf{z}\right) - \Phi\left(\mathbf{X}\right)\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\Phi\left(\mathbf{X}\right)^{\mathrm{T}}\Phi\left(\mathbf{z}\right) - \left[\mathbf{I} - \Phi\left(\mathbf{X}\right)\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\Phi\left(\mathbf{X}\right)^{\mathrm{T}}\right]\mathbf{m}\right\|_{\mathcal{H}}^2 \\ &= \kappa\left(\mathbf{z},\mathbf{z}\right) - 2\frac{1}{n_{\mathrm{X}}}\sum_{i=1}^{n_{\mathrm{X}}}\kappa\left(\mathbf{z},\mathbf{x}_i\right) + \frac{1}{n_{\mathrm{X}}^2}\sum_{i,j=1}^{n_{\mathrm{X}}}\kappa\left(\mathbf{x}_i,\mathbf{x}_j\right) - \sum_{k=1}^{d_{\mathcal{U}}}\left[f_k\left(\mathbf{z}\right) - \frac{1}{n_{\mathrm{X}}}\sum_{i=1}^{n_{\mathrm{X}}} f_k\left(\mathbf{x}_i\right)\right]^2, \end{aligned} \tag{6.13}$$

For the Gaussian kernel, for example, $\kappa\left(\mathbf{z},\mathbf{z}\right) = 1$ for all $\mathbf{z}$. The terms in Eq. 6.13 that depend on $\mathbf{z}$ thus comprise a sum of Gaussians. Thus, we see from the form of Eq. 6.13 that $J_{\mathcal{U}}$ will necessarily have multiple discrete optima along the manifold, centered at each of the manifold data samples $\mathbf{x}_i$ for *any* finite number of principal components $d_{\mathcal{U}}$. Meanwhile, having $J_{\mathcal{W}}$'s support vectors off-manifold allows for a level set of this function to smoothly connect the various manifold samples $\mathbf{x}_i$. We compare both functionals, $J_{\mathcal{U}}$ and $J_{\mathcal{W}}$, in Fig. 6.3, where we learn a clover leaf shaped manifold using KPCA and our method.

6.3.2 Algorithm Interpretation as a Function Minimization

We now present an interpretation of our solution in terms of an equivalent function optimization problem, similar to the one conducted for KPCA in Section 3.3.2. Defining g as above, and following the approach of Chapter 4 of [174], the problem in Eq. 6.2 can easily be expressed as:

$$\begin{aligned} \min_{g\in\mathcal{A}} \quad & \theta\sum_{i=1}^{n_{\mathrm{X}}}\left[g\left(\mathbf{x}_i\right) - \frac{1}{n_{\mathrm{X}}}\sum_{j=1}^{n_{\mathrm{X}}} g\left(\mathbf{x}_j\right)\right]^2 + \left(1-\theta\right)\left\|g\right\|_{\mathcal{H}}^2 \\ \text{subject to} \quad & \sum_{j=1}^{n_{\mathrm{Y}}}\left[g\left(\mathbf{y}_j\right)\right]^2 = 1. \end{aligned} \tag{6.14}$$

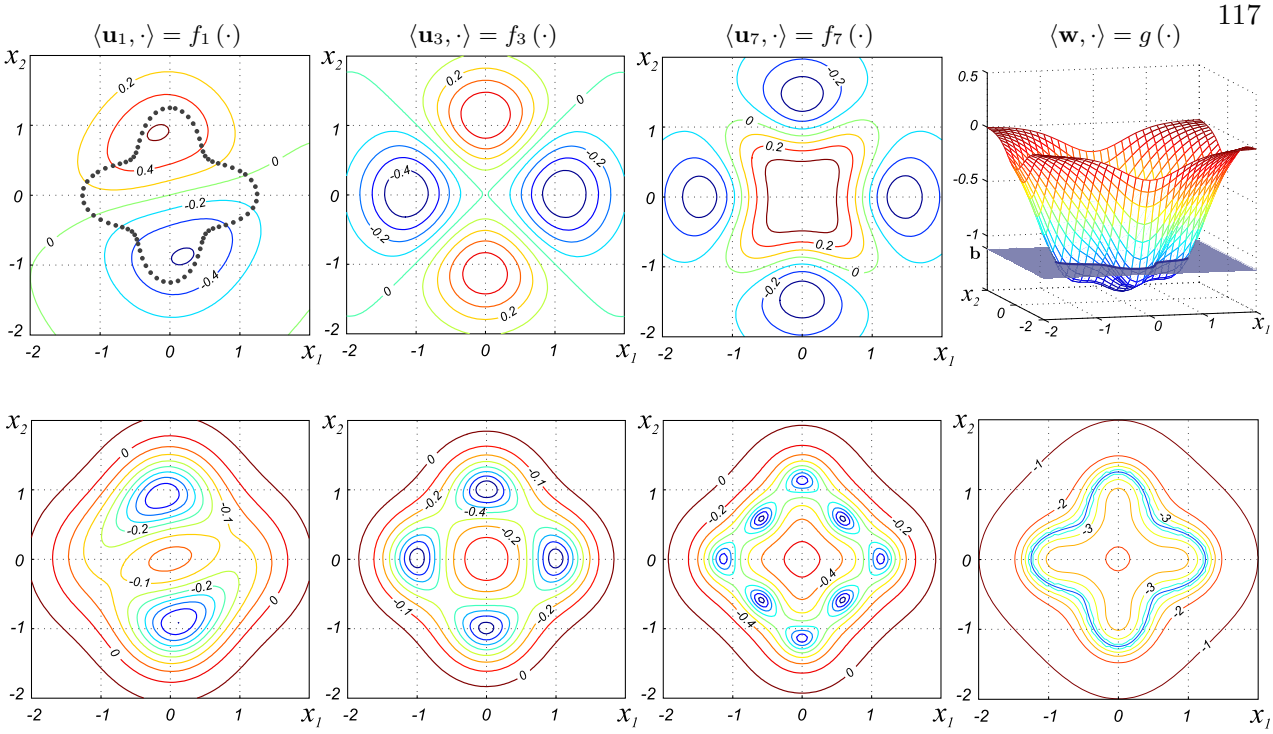


Figure 6.3: Top row: Level curves of the KPCA solutions, $f_k(\mathbf{z})$, corresponding to different principal components and the surface of $g(\mathbf{z})$ learned with our algorithm. Bottom row: Level curves of $\log_{10} J_{\mathcal{U}}$ corresponding to subspaces $\mathcal{U}$ built from the first k principal components: $f_1, \ldots, f_k$ and level curves of $\log_{10} J_{\mathcal{W}}$ resulting from our solution. Increasing the dimension of $\mathcal{U}$ results in smoother but nevertheless discontinuous approximations with several discrete minimizers of $J_{\mathcal{U}}$. The set of minimizers of $J_{\mathcal{W}}$ is continuous and accurately approximates the manifold.

where $\mathcal{A}$ is the set of all functions of the form $g(\mathbf{z}) = \sum_{j=1}^{n_Y} \boldsymbol{\nu}_j \kappa(\mathbf{y}_j, \mathbf{z})$ for $\boldsymbol{\nu} \in \mathbb{R}^{n_Y}$ and $||\cdot||_{\mathcal{H}}$ is a function norm that depends on our choice of kernel. As in other kernel methods, this $||g||^2_{\mathcal{H}}$ term acts as a kernel-dependent regularization on the function g. For example, as was noted in Section 3.3.2, for the Gaussian kernel, minimizing this norm effectively penalizes high-frequency components in the function g.

Now, examining Eq. 6.14, we see that for this kernel we can interpret our solution g as that function that can be built from Gaussians centered on the provided samples $\mathbf{y}_j$, which is most constant on the given manifold samples $\mathbf{x}_i$ and has lowest high-frequency energy overall. This last requirement encourages us to pick the least oscillatory g that satisfies our other requirements. Furthermore, larger values of the bandwidth parameter σ result in even smoother solutions and can

help to avoid overfitting noisy samples. A trivial solution (such as the zero function) is prevented by the constraint.

To compare with similar analysis of the KPCA solution (see Chapter 4 in [174]), the KPCA solution f is the minimum norm (i.e. the least oscillatory) function that varies adequately on the manifold samples. Thus, f is good for producing a low-dimensional embedding of the data samples while varying smoothly *along* the manifold, but if we'd like a function that takes a constant value along the manifold and a different one elsewhere, we need g.

Furthermore, the user's choice of θ controls the tradeoff between trying to fit all the data in a single level set of g, i.e. exactly on the estimated manifold, and having a smoother function g, i.e. a smoother manifold. It can be set according to the perceived noise level, particularly to avoid overfitting noisy data. Also, as was noted above, for $\theta = 0$, the optimization problem of Eq. 6.2 reduces to performing kernel PCA on the (uncentered) set of expansion vectors $\mathbf{y}_j$. Without the regularization, an admissible solution easily results in overfitting on the set of noisy manifold samples. Allowing both terms in Eq. 6.2 favors a smooth solution that also fits the manifold.

Finally, we note that the choice of kernel and of the expansion vectors $\mathbf{y}_j$ is critical in determining the set of possible approximations of the manifold. By selecting more closely spaced expansion samples for example, we can ensure a higher-quality approximation. In the next section, we will focus closer on the problem of choosing proper $\mathbf{y}_j$'s and then propose an efficient incremental updating algorithm to allow for using large sets of expansion points.

6.3.3 Choosing the Support Vectors for the Expansion of the Normals

As noted above, we seek the expansion of the normal vector $\mathbf{w}$ in a specifically constructed basis rather than in the span of the training manifold samples $\mathbf{x}_i$, and this constitutes one of the main differences of our method with other kernel algorithms, such as KPCA [143] or LS-SVM [49]. In this section we discuss an effective way of choosing the support vectors $\mathbf{y}_j$ for this expansion.

In our approach, we will rely on partial knowledge about $\mathcal{U}$, the best dataset-approximating subspace in the least-squares sense, which is conveniently provided by KPCA in the form of its

principal components $\mathbf{U}$. The desired vector $\mathbf{w}$ for our representation by definition has to be orthogonal to this subspace so that $\mathcal{W}$ can contain $\mathcal{U}$ and thus approximate the data well. Therefore, our strategy for selecting the $\mathbf{y}_j$'s will be to attempt to construct a partial basis for the orthogonal compliment of $\mathcal{U}$ and then to expand $\mathbf{w}$ with respect to it. We note that, in general, $\mathcal{U}^\perp$ may be infinite-dimensional, and thus constructing an exact basis for it may be impossible.

To motivate our approach for choosing the expansion vectors $\mathbf{y}_j$, let us set $\theta = 0$ and consider only the regularization term on the left-hand side of Eq. 6.4. The restricted optimization problem, in this case, reduces to a simple eigendecomposition of the kernel matrix $\mathbf{B} = \mathbf{K}_{\mathrm{YY}}$ (see Section 6.2.2), which is equivalent to performing uncentered kernel PCA on the samples $\mathbf{y}_j$. Therefore, in order to obtain a desired vector $\mathbf{w} \in \mathcal{U}^\perp$ we may aim to reverse-engineer the KPCA solution and choose the $\mathbf{y}_j$'s such that their images in feature space will make the leading eigenvectors of $\mathbf{B}$ align with the directions orthogonal to $\mathcal{U}$. Obviously, there is no unique solution to this problem: many (higher-order) distributions of $\mathbf{y}_j$ may share the same covariance operators and thus be invariant under the kernel PCA procedure. Here we propose a very simple, yet effective approach that plays well with the machinery of kernel trick, which is essential to our method.

Specifically, we will generate a number of points uniformly at random in the original space, $\mathbf{y} \sim unif\left(\mathbf{y}_{\min}, \mathbf{y}_{\max}\right)$, and then retain only a fraction of them with probability $p\left(\mathbf{y}\right)$. To align $\mathbf{y}$ with the orthogonal compliment $\mathcal{U}^\perp$, we define $p\left(\mathbf{y}\right)$ to be reciprocal to the length of the normalized projection of $\mathbf{y}$ onto $\mathcal{U}$. Furthermore, to penalize selection of points lying far from the manifold, we scale this probability by the value of their inner products with $\Phi\left(\mathbf{X}\right)\boldsymbol{\mu}$ – the offset of $\mathcal{U}$, which finally results in:

$$
\begin{aligned}
p\left(\mathbf{y}\right) &= \underbrace{\left|\frac{\left\langle \Phi\left(\mathbf{y}\right), \Phi\left(\mathbf{X}\right)\boldsymbol{\mu}\right\rangle_{\mathcal{H}}}{\left\|\Phi\left(\mathbf{y}\right)\right\|_{\mathcal{H}}\left\|\Phi\left(\mathbf{X}\right)\boldsymbol{\mu}\right\|_{\mathcal{H}}}\right|}_{Favors\ \mathbf{y}\ close\ to\ \mathcal{M}} \underbrace{\left[1 - \frac{\left\|\mathrm{P}_{\mathcal{U}}\left(\mathbf{y}\right)\right\|_{\mathcal{H}}}{\left\|\mathbf{y}\right\|_{\mathcal{H}}}\right]}_{Favors\ \Phi(\mathbf{y})\in\mathcal{U}^\perp} \\
&= \left|\frac{\left\langle \Phi\left(\mathbf{y}\right), \Phi\left(\mathbf{X}\right)\boldsymbol{\mu}\right\rangle_{\mathcal{H}}}{\left\|\Phi\left(\mathbf{y}\right)\right\|_{\mathcal{H}}\left\|\Phi\left(\mathbf{X}\right)\boldsymbol{\mu}\right\|_{\mathcal{H}}}\right| \cdot \left(1 - \sqrt{\frac{\left[\Phi\left(\mathbf{y}\right)\right]^{\mathrm{T}}\mathbf{U}\mathbf{U}^{\mathrm{T}}\Phi\left(\mathbf{y}\right)}{\left\|\Phi\left(\mathbf{y}\right)\right\|^2}}\right) \\
&= \left|\frac{\mathbf{k}_{\mathrm{Xy}}^{\mathrm{T}}\boldsymbol{\mu}}{\sqrt{\kappa\left(\mathbf{y},\mathbf{y}\right)}\sqrt{\boldsymbol{\mu}^{\mathrm{T}}\mathbf{K}_{\mathrm{XX}}\boldsymbol{\mu}}}\right| \cdot \left(1 - \sqrt{\frac{\mathbf{k}_{\mathrm{Xy}}^{\mathrm{T}}\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{k}_{\mathrm{Xy}}}{\kappa\left(\mathbf{y},\mathbf{y}\right)}}\right),
\end{aligned}
\tag{6.15}
$$

where $\mathbf{k}_{\mathrm{Xy}}$ is an $n_{\mathrm{X}} \times 1$ vector with entries $[\mathbf{k}_{\mathrm{Xy}}]_i = \kappa(\mathbf{x}_i, \mathbf{y})$. Even though $p(\mathbf{y})$ is defined in feature space, it can be easily computed in terms of inner products and kernelized. We plot examples of the level curves of $p(\mathbf{y})$ corresponding to learning the manifolds with Gaussian kernels in Fig. 6.4. The boundaries $\mathbf{y}_{\min}$ and $\mathbf{y}_{\max}$ in the initial uniform distribution of $\mathbf{y}$ can be chosen empirically to include the manifold samples and to ensure that $p(\mathbf{y})$ decays sufficiently beyond them. We summarize this procedure of generating $\mathbf{y}_j$'s in the following Algorithm.

Algorithm 3 Generation of Expansion Vectors $\mathbf{y}_j$, GETY

Input: Manifold samples $\mathbf{X}$, kernel κ, expansion coefficients $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$, number of expansion vectors to generate n_{Y}, and the boundaries $\mathbf{y}_{\min}$ and $\mathbf{y}_{\max}$.
Output: Set of expansion vectors $\mathbf{Y}$.

$\mathbf{K} \leftarrow \kappa(\mathbf{X}, \mathbf{X})$
$i \leftarrow 0$
while $i < n_{\mathrm{Y}}$ **do**
  $\mathbf{y} \leftarrow unif(\mathbf{y}_{\min}, \mathbf{y}_{\max})$ ▷ Sample $\mathbf{y}$ from a uniform distribution.
  $\mathbf{k}_{\mathrm{Xy}} \leftarrow \kappa(\mathbf{X}, \mathbf{y})$; $\mathrm{k}_{\mathrm{yy}} \leftarrow \kappa(\mathbf{y}, \mathbf{y})$
  $p(\mathbf{y}) = \left| \frac{\mathbf{k}_{\mathrm{Xy}}^{\mathrm{T}} \boldsymbol{\mu}}{\sqrt{\mathrm{k}_{\mathrm{yy}}} \sqrt{\boldsymbol{\mu}^{\mathrm{T}} \mathbf{K} \boldsymbol{\mu}}} \right| \cdot \left(1 - \sqrt{\frac{\mathbf{k}_{\mathrm{Xy}}^{\mathrm{T}} \boldsymbol{\alpha} \boldsymbol{\alpha}^{\mathrm{T}} \mathbf{k}_{\mathrm{Xy}}}{\mathrm{k}_{\mathrm{yy}}}} \right)$
  if $p(\mathbf{y}) < unif(0, 1)$ **then** ▷ Keep $\mathbf{y}$ with probability $p(\mathbf{y})$.
    $i \leftarrow i + 1$
    $\mathbf{Y}_{(:,i)} \leftarrow \mathbf{y}$ ▷ Update the resulting set.
  end if
end while

Moreover, we found that explicitly ensuring that the resulting subspace $\mathcal{W}$ will contain $\mathcal{U}$ often allows us to achieve good results of approximation faster. For this, we write its normals as $\mathbf{W} = (\mathbf{I} - \mathbf{U}\mathbf{U}^{\mathrm{T}}) \Phi(\mathbf{Y}) \boldsymbol{\nu}$ and then solve the problem of Eq. 6.2 as before. This yields the same generalized eigenproblem of Section 6.2.1 for $\boldsymbol{\nu}$, but now with new $\mathbf{A} = [\mathbf{I} - \frac{1}{n_{\mathrm{X}}} \mathbb{1}\mathbb{1}^{\mathrm{T}}][\mathbf{I} - \boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}} \mathbf{K}_{\mathrm{XX}}] \mathbf{K}_{\mathrm{XY}}$ and $\mathbf{B} = \mathbf{K}_{\mathrm{YY}} - \mathbf{K}_{\mathrm{XY}}^{\mathrm{T}} \boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}} \mathbf{K}_{\mathrm{XY}}$. The vector $\mathbf{b}$ is then: $\mathbf{b} = \mathbf{W}^{\mathrm{T}} \mathbf{m} = \boldsymbol{\nu}^{\mathrm{T}} \mathbf{K}_{\mathrm{XY}}^{\mathrm{T}} \boldsymbol{\mu}$.

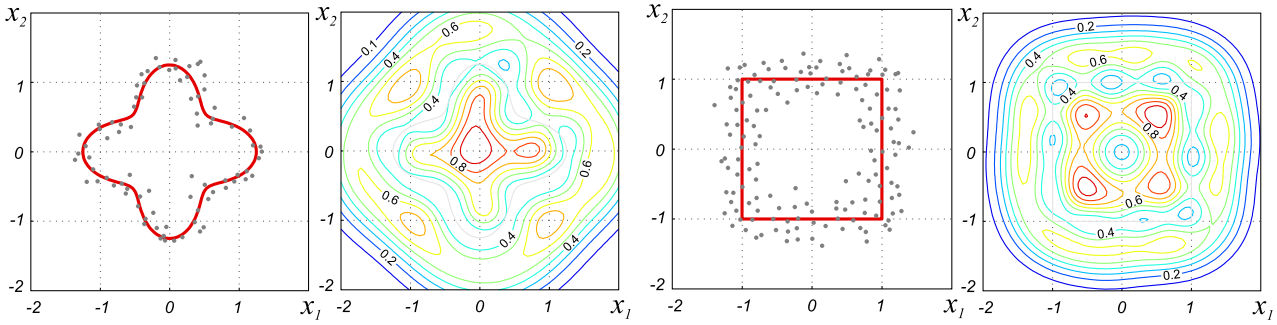


Figure 6.4: Examples of level curves of $p(\mathbf{y})$ scaled to the range $[0, 1]$ with dark red values corresponding to higher probabilities of choosing a particular expansion point. Notice that weighted "Gaussian bumps" placed in the areas where $p(\mathbf{y})$ attains higher values are likely to define functions g whose level sets will be aligned with the manifolds.

6.4 Incremental Algorithm for Efficient Processing of Large Datasets

In this section, we will develop an incremental extension of our KODA algorithm similar to the incremental KPCA of [48]. It will allow us to work with large sets of training samples $\mathbf{x}_i$ and use more support vectors $\mathbf{y}_j$, significantly improving their representational capabilities. For this, we will effectively treat the decomposition of the kernel matrix $\mathbf{K}_{\mathrm{YY}}$ in KODA as a special case of the incremental eigendecomposition procedure outlined in Section 3.4.2, although without centering. Then we will adapt the same idea for incremental updates of the expansion coefficients $\boldsymbol{\nu}$ when additional points $\mathbf{y}_j$ are supplied.

The workflow of our proposed incremental KODA is outlined in Algorithm 4. Here we assume that the training samples of the manifold are supplied successively in the form of N batches $\mathbf{X}^{(i)}$ and used to update the KPCA representation $\boldsymbol{\alpha}$ and $\boldsymbol{\varepsilon}$ (line 4) via the incremental eigendecomposition with centering. The basis $\Phi\left(\widetilde{\mathbf{X}}\right)\boldsymbol{\alpha}$ is then reorthogonalized in line 6 by projecting it onto the SVD basis to account for possible loss of orthogonality during the greedy reduced set compression (see Section 3.4.1 and Appendix C). We note that the vectors $\boldsymbol{\alpha}$ are not normalized, thus allowing us to express the singular values of $\mathbf{K}_{\mathrm{XX}} = \kappa\left(\widetilde{\mathbf{X}}, \widetilde{\mathbf{X}}\right)$ as $\boldsymbol{\Sigma} = \left(\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{K}_{\mathrm{XX}}\boldsymbol{\alpha}\right)^{1/2}$ in Algorithm 6.

Likewise, the support vectors for $\mathbf{w}$ are generated on every iteration as $\mathbf{Y}^{(i)}$ and the basis for their span is updated in $\boldsymbol{\beta}$. Note that this is achieved via an uncentered version of the eigendecomposition subroutine in line 9 of Algorithm 4. Now, using the same idea of low-rank

Algorithm 4 Incremental KODA

Input: Manifold samples arranged in N batches $\mathbf{X}^{(i)}$, kernel κ, regularization parameter θ, number of new expansion vectors to be generated on each step n_{Y} and the boundaries $\mathbf{y}_{\min}$ and $\mathbf{y}_{\max}$.
Output: Sets of expansion vectors $\widetilde{\mathbf{X}}$ and $\widetilde{\mathbf{Y}}$, representation coefficients $\boldsymbol{\alpha}$, $\boldsymbol{\varepsilon}$, $\boldsymbol{\nu}$, $\mathbf{b}$.

$\widetilde{\mathbf{X}} \leftarrow \emptyset$, $\boldsymbol{\alpha} \leftarrow \emptyset$, $\boldsymbol{\Sigma} \leftarrow \emptyset$, $\boldsymbol{\varepsilon} \leftarrow \emptyset$ ▷ Initialization.
$\widetilde{\mathbf{Y}} \leftarrow \emptyset$, $\boldsymbol{\beta} \leftarrow \emptyset$
for $i = 1, \ldots, N$ **do**
  $[\boldsymbol{\alpha}, \boldsymbol{\varepsilon}] \leftarrow \mathrm{iEIG}_{\mathrm{cnt}}\left(\widetilde{\mathbf{X}}, \boldsymbol{\alpha}, \boldsymbol{\varepsilon}, \mathbf{X}^{(i)}\right)$ ▷ KPCA. See Appendix B
  $\left[\widetilde{\mathbf{X}}, \begin{bmatrix} \boldsymbol{\alpha} & \boldsymbol{\varepsilon} \end{bmatrix}\right] \leftarrow \mathrm{RS\_EXP}\left(\begin{bmatrix} \widetilde{\mathbf{X}} & \mathbf{X}^{(i)} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\alpha} & \boldsymbol{\varepsilon} \end{bmatrix}\right)$ ▷ See Appendix C.
  $\boldsymbol{\alpha} \leftarrow \mathrm{ORTH}\left(\widetilde{\mathbf{X}}, \boldsymbol{\alpha}\right)$ ▷ Orthogonalize the basis.
  $\boldsymbol{\mu} \leftarrow \left(\mathbf{I} - \boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{K}\right)\boldsymbol{\varepsilon}$
  $\mathbf{Y} \leftarrow \mathrm{GETY}\left(\mathbf{X}, \boldsymbol{\alpha}, \boldsymbol{\mu}, \mathbf{y}_{\min}, \mathbf{y}_{\max}, n_{\mathrm{Y}}\right)$ ▷ Generate new $\mathbf{Y}$. See Algorithm 3.
  $[\boldsymbol{\beta}] \leftarrow \mathrm{iEIG}_{\mathrm{unc}}\left(\widetilde{\mathbf{Y}}, \boldsymbol{\beta}, \mathbf{Y}\right)$ ▷ Uncentered; see Sec. 3.4.2.
  $\left[\widetilde{\mathbf{Y}}, \boldsymbol{\beta}\right] \leftarrow \mathrm{RS\_EXP}\left(\begin{bmatrix} \widetilde{\mathbf{Y}} & \mathbf{Y} \end{bmatrix}, \boldsymbol{\beta}\right)$ ▷ See Appendix C.
  $\boldsymbol{\beta} \leftarrow \mathrm{ORTH}\left(\widetilde{\mathbf{Y}}, \boldsymbol{\beta}\right)$ ▷ Orthogonalize the basis.
  $\mathbf{K}_{\mathrm{YY}} \leftarrow \kappa\left(\widetilde{\mathbf{Y}}, \widetilde{\mathbf{Y}}\right)$
  $\boldsymbol{\Sigma} \leftarrow \left[\boldsymbol{\beta}^{\mathrm{T}}\mathbf{K}_{\mathrm{YY}}\boldsymbol{\beta}\right]^{1/2}$ ▷ Updated singular values.
  $\mathbf{K}_{\mathrm{XX}} \leftarrow \kappa\left(\widetilde{\mathbf{X}}, \widetilde{\mathbf{X}}\right)$, $\mathbf{K}_{\mathrm{XY}} \leftarrow \kappa\left(\widetilde{\mathbf{X}}, \widetilde{\mathbf{Y}}\right)$
  $\mathbf{A} \leftarrow \boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{K}_{\mathrm{XY}}\boldsymbol{\beta}\boldsymbol{\beta}^{\mathrm{T}}$
  $\widetilde{\boldsymbol{\nu}} \leftarrow \mathrm{EIG}_{sm}\left(\theta\boldsymbol{\Lambda}_B^{-2}\boldsymbol{\beta}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\boldsymbol{\beta} + (1-\theta)\,\boldsymbol{\Lambda}_B^{-1}\right)$ ▷ Retain the smallest eigenvectors.
  $\boldsymbol{\nu} \leftarrow \boldsymbol{\beta}\widetilde{\boldsymbol{\nu}}$, $\mathbf{b} \leftarrow \boldsymbol{\nu}^{\mathrm{T}}\mathbf{K}_{\mathrm{XY}}^{\mathrm{T}}\boldsymbol{\varepsilon}$
end for

function ORTH($\mathbf{X}$, $\boldsymbol{\alpha}$)
  $\mathbf{K} \leftarrow \kappa\left(\mathbf{X}, \mathbf{X}\right)$
  $\left[\mathbf{u}, \mathbf{s}, \mathbf{v}^{\mathrm{T}}\right] \leftarrow \mathrm{SVD}\left(\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{K}\boldsymbol{\alpha}\right)$
  return $\boldsymbol{\alpha} \leftarrow \mathbf{u}\boldsymbol{\alpha}$
end function

representation as in Eq. 6.5, the matrix $\mathbf{A}$ can be approximated (up to the number of retained principal components of $\mathcal{U}$, $d_{\mathcal{U}}$, and the numerical rank of $\Phi(\mathbf{Y})$, r) as: $\mathbf{A} = \boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{K}_{\mathrm{XY}}\boldsymbol{\beta}\boldsymbol{\beta}^{\mathrm{T}}$.

To illustrate our incremental KODA algorithm with an example, we consider learning a simple clover-leaf shaped manifold in $\mathbb{R}^2$ from its noisy samples. On each step of iterations, we generate 50 new points $\mathbf{y}_j$ for expanding the normal vector $\mathbf{w}$ and plot the values of the corresponding functionals $J_{\mathcal{W}}$ in Fig. 6.5. Note how the set of minima of $J_{\mathcal{W}}$ (dark blue lines) start to better approximate the manifold when more support vectors are added to the expansion.

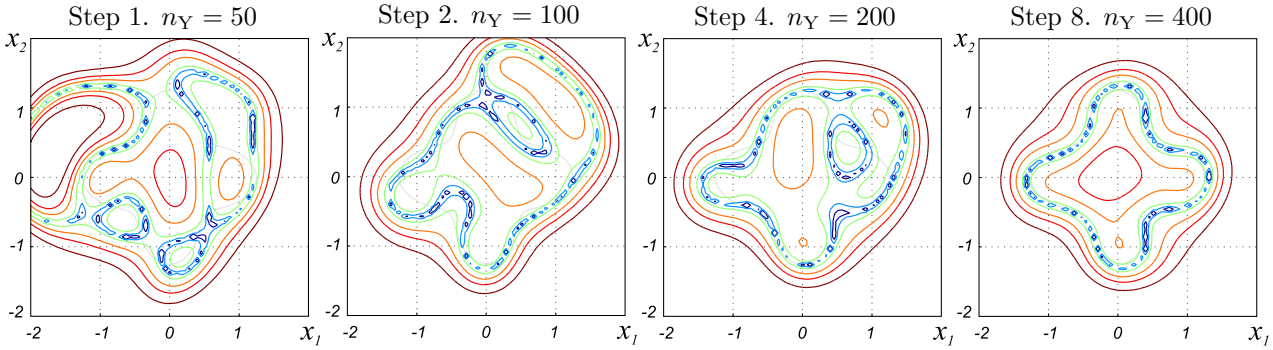


Figure 6.5: Evolution of the clover leaf-shaped manifold representation learned from noisy samples with the proposed incremental KODA algorithm (please see Fig. 6.4 for the original manifold and its samples). On each iteration, another 50 points $\mathbf{y}_j$ are generated according to $p(\mathbf{y})$ in Eq. 6.4 and added to the solution. Note how the level curves of $J_{\mathcal{W}}$ plotted here gradually approximate the desired manifold; darker blue lines correspond to lower values of $J_{\mathcal{W}}$.

Finally, we want to emphasize that our incremental KODA algorithm can be implemented with any method for parsimonious vector representation in kernel spaces, such as [29, 48, 79, 171], if desired. The greedy approximation algorithm of [174] (see Section 3.4.1 for more details), however, has shown good results in our experiments and has been used throughout this work. We also note that Algorithm 4 can be easily modified for performing all KODA-related operations *after* the entire dataset of manifold samples has been processed and the KPCA basis has been learned, as well as generalized to potentially have different numbers of data batches $\mathbf{X}^{(i)}$ and batches of generated expansion points $\mathbf{Y}^{(j)}$.

6.5 Experiments and Discussion

In this section, we will show the results of using our proposed KODA algorithm for learning manifolds from few and/or noisy samples of them, and for mapping points onto and interpolating along manifolds. We also show results of applying our method to unsupervised anomaly detection and classification with a manifold model.

In our experiments, in addition to simple geometric toy examples in two and three dimensions, we will use several publicly available real-world datasets of signals whose structure allows us to model them with underlying manifolds. In particular, we consider the set of MNIST handwritten digits [124], the Frey Face dataset [81], the 20 Newsgroups dataset [120], and the Body Attack Fitness dataset [78], which we briefly describe here.

The MNIST dataset consists of 20×20 images of handwritten digits approximately equally distributed among ten classes, one for each digit from "0" to "9"; we use only 15000 samples in total. The Frey Face dataset contains a series of 28×20 images of the same person showing different facial expressions [81]. These images were obtained from a video clip and thus can naturally be assumed to change smoothly and to be modeled with an underlying low-dimensional manifold. Both datasets are scaled to the range $[-1, \ldots, 1]$. We will use them in our mapping and interpolation examples.

The 20 Newsgroups dataset [120] is a collection of nearly 20000 text documents belonging to 20 different topics. To prepare it for our experiments, we removed all stop-words and all low-frequency words (those that appear at most three times in all documents). Furthermore, we selected only long documents containing more than 50 words. This eventually resulted in 14054 samples represented by their word-count vectors in a 12366-word dictionary. Finally, we normalized each vector to sum to one and embedded them into a 500-dimensional ambient space by multiplying with a random Gaussian matrix. This embedding significantly facilitated our experiments and is motivated by the recent results, akin to the famous Johnson-Lindenstrauss lemma [58, 107], showing that random projections of manifold samples with high probability preserve their mutual distances

and the structure of the manifold [9].

The data in our last dataset, the Body Attack Fitness dataset [78], comes in the form of time series generated by an array of ten accelerometers attached to a person performing six different fitness activities for approximately 2.5 minutes each. Each sensor records three axial components of the acceleration vector sampled at the rate of 64 Hz. We smoothed each signal with a moving average window of width 50 to reduce the effect of possible missed or outlying measurements. Finally, we converted them from acceleration to coordinate displacement vectors by computing cumulative sums twice.

The following table summarizes the statistics of each dataset, including the numbers of samples n_{X} and classes n_{cls}, and the dimensions of the ambient space D, the KPCA subspace $d_{\mathcal{U}}$ that was used, and the codimension of the underlying manifold that was assumed for our experiments $d_{\mathcal{W}}$. Unless otherwise stated, we will use the Gaussian kernel with the listed values of σ in all our examples; samples used for testing will be excluded from the training sets. We will provide more specific details about each experimental setting when necessary.

Table 6.1: Overview of the datasets' statistics and parameters used

	n_{X}	n_{cls}	D	$d_{\mathcal{U}}$	$d_{\mathcal{W}}$	σ
Entire MNIST	15000	10	400	20	8	500
MNIST – digit "2"	1929	1	400	12	3	500
Frey's Faces	1965	–	560	50	5	300
20 Newsgroups	14054	20	500	50	8	20
Body Attack Fitness	84000	6	30	50	1	1

6.5.1 Learning Manifolds from Their Samples

First, with simple toy examples, we demonstrate that our algorithm accurately reconstructs continuous curves in two and three dimensions from few and/or noisy samples of them. We plot the level curves of the approximated distance to the manifolds obtained with different methods in Fig. 6.6 and note that, unlike the KPCA parameterization, our approach results in a continuous set of minimizers where $J_{\mathcal{W}}(\mathbf{z})$ reduces to zero, i.e. where $\mathbf{g}(\mathbf{z}) = \mathbf{b}$.

For comparison, we look at the LS-SVM algorithm of [49]. Even though this algorithm was designed for solving classification problems and has several key differences with ours (see Sec. 6.2.1), it does provide a continuous representation for a manifold based on the training samples, so it is instructive to see its results next to ours. In particular, we would like to demonstrate the tremendous impact of expanding the normals $\mathbf{w}$ in terms of off-manifold samples as opposed to on-manifold samples. Note in Fig. 6.6 that the learned manifold representations are generally wider across, less smooth, and less well-defined with LS-SVM, and that we see unwanted spurious connections inside the clover leaf and the fish, resulting in a substantially larger set of distance minimizers than expected. The normal vector $\mathbf{w}$ in our solution is supported on a set of specifically-generated off-manifold points, which allows our method to produce significantly more clean and accurate manifold representations.

In the next example, we consider learning a curve in $\mathbb{R}^3$, which has codimension 2 and thus requires $n_W = 2$ constraints (see Fig. 6.7). Specifically, we generate 200 training samples of it, disturbed by additive Gaussian noise, as $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1.1\sin(t) \\ 0.9\cos(t) \\ 0.6\cos(2t) \end{bmatrix} + \mathcal{N}(0,\, 0.1\mathbf{I})$ for $t \in [0, \ldots, 2\pi]$. We then learn this curve with our algorithm by retaining two eigenvectors in $\boldsymbol{\nu}$. In Fig. 6.7, we can see that each of the two corresponding constraints separately describes a two-dimensional surface in the input space. Their intersection, satisfying both constraints, is a one-dimensional manifold that well approximates the desired curve.

Finally, we note that our method is readily applicable for an important computer graphics problem of surface reconstruction from a 3D point cloud [15]. Specifically, we use it to learn a surface of the popular Stanford bunny [196] from only 3000 randomly sampled noisy points and present our results in Fig. 6.10. Note that this problem is not the easiest for most of the existing manifold learning methods, however, our algorithm succeeds in accurately learning multiscale features of this difficult manifold and even results in a more visually pleasing outcome than the Poisson surface

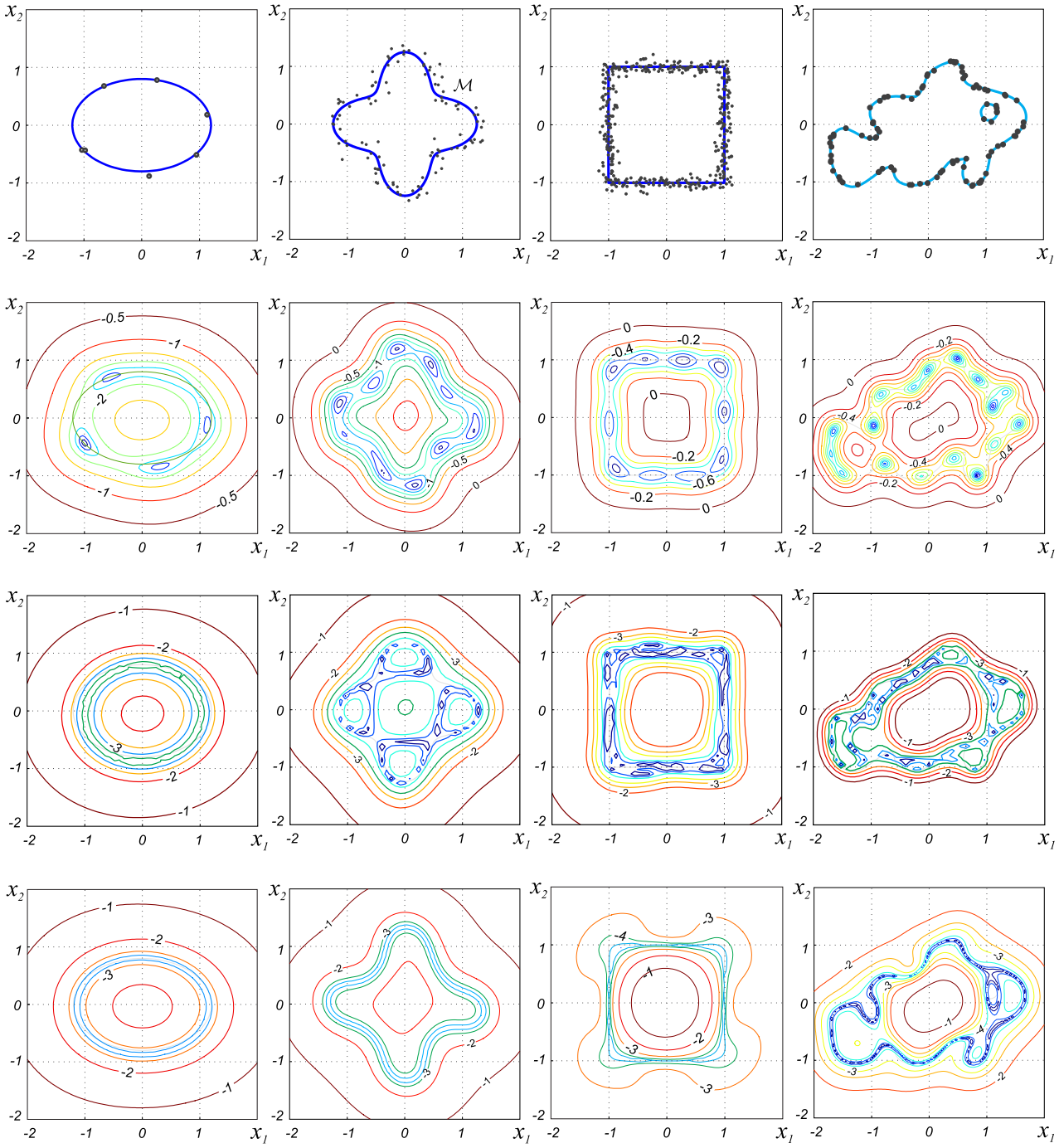


Figure 6.6: Results of learning different one-dimensional curves in $\mathbb{R}^2$. From top to bottom; the first row: Training samples of the manifolds. The second row: Level curves of $\log_{10} J_{\mathcal{U}}$. Third row: Level curves of $\log_{10} J_{LS-SVM}$. Bottom row: Level curves of our proposed $\log_{10} J_{\mathcal{W}}$. Our method results in an accurate continuous representation of the manifolds, while the KPCA parameterization suffers from local minima and poor generalization on few training samples. An alternative representation found with LS-SVM, on the other hand, produces a thick, porous manifold representation. We use the Gaussian kernel with $\sigma = 5$, 0.8, 0.25, and 0.2 for each example from left to right, respectively.

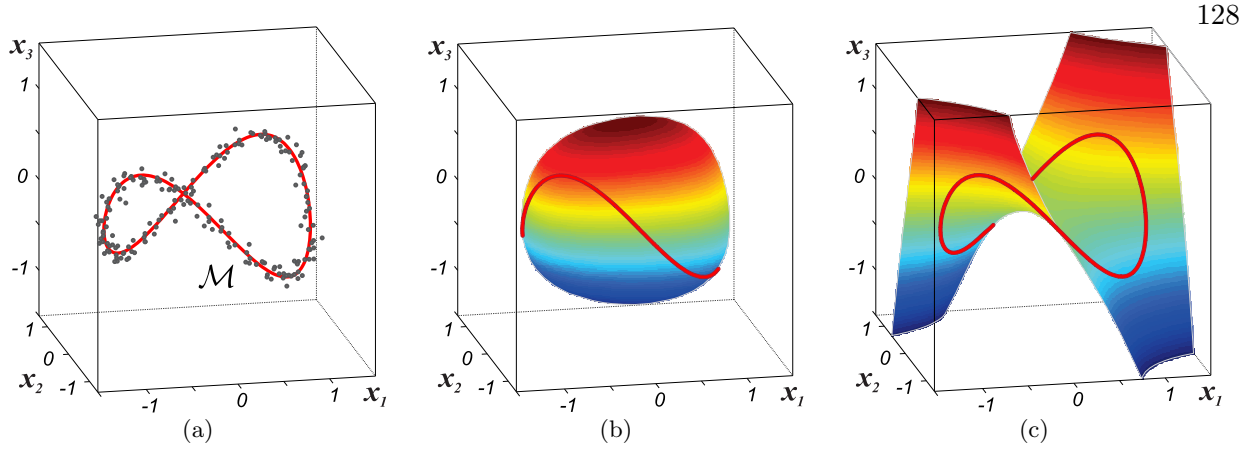


Figure 6.7: Learning a one-dimensional manifold in $\mathbb{R}^3$ with our method. (a) Training samples of a non-self-intersecting curve. (b-c) The surfaces defined by the first and second normals to the subspace $\mathcal{W}$ and associated offsets. Their intersection line (shown in red) well-approximates the desired curve. These results are obtained using the Gaussian kernel with $\sigma = 1$.

reconstruction algorithm widely used for this purpose [112].

In the following section, we discuss applications of our manifold representation as a basis for solving other important practical problems. Specifically, we will use minimization of the distance-approximating functional $J_{\mathcal{W}}$ from Section 6.3.1 for mapping points onto manifolds and finding continuous paths on them.

6.5.2 Mapping Points onto Manifolds

It is of crucial importance in many practical applications to find a mapping of an arbitrary point in the ambient space onto the surface of a manifold. For example, if the manifold is assumed to model a set of admissible clean exemplars, then adding noise to any of these exemplars will likely send them away from the manifold. Thus, one may attempt to remove the noise in an observed sample by finding a point on the surface of the manifold close to it. This is the key idea behind the KPCA denoising approach [143, 151].

However, as we have seen in the previous examples, describing manifolds with kernel PCA will in most cases result in a discontinuous representation in the input space. Thus, an algorithm like KPCA denoising that relies on minimizing $J_{\mathcal{U}}$ as a proxy to the true distance to the manifold,

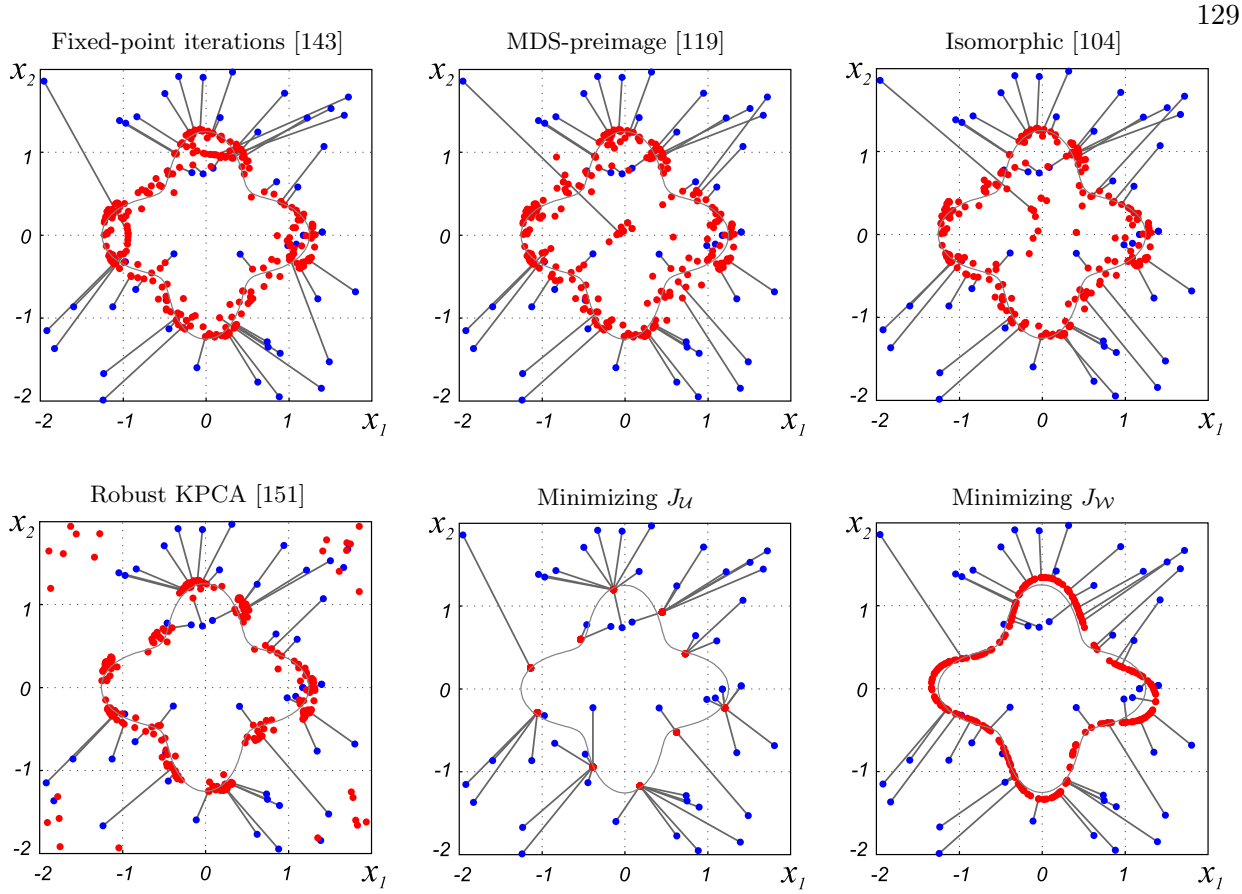


Figure 6.8: Mapping a cloud of random points (blue; not all points shown) onto the manifold using the KPCA denoising strategy [143] with various preimage methods to bring the found feature space solution back to the original space. Note that all algorithms result in points lying closer but not necessarily *on* the manifold (red). Results of minimization of the proposed functional $J_{\mathcal{W}}$ with gradient descent trace a continuous curve giving a good approximation of the initial manifold $\mathcal{M}$.

in the absence of additional regularization, will eventually converge to one of these discontinuous discrete minimizers. Instead, our proposed continuous representation with a higher-dimensional subspace $\mathcal{W}$, offers a natural solution to this problem because of the continuity of the representation it generates. To demonstrate this, we will map points onto it by minimizing $J_{\mathcal{W}}$ using gradient descent with fixed step size $h = 1$ and compare our results to the KPCA denoising strategy using a variety of popular preimage-finding methods.

As a first example, we consider the clover leaf-shaped manifold from Section 6.5.1 and create a cloud of randomly generated points around it, which imitate a set of noisy off-manifold samples

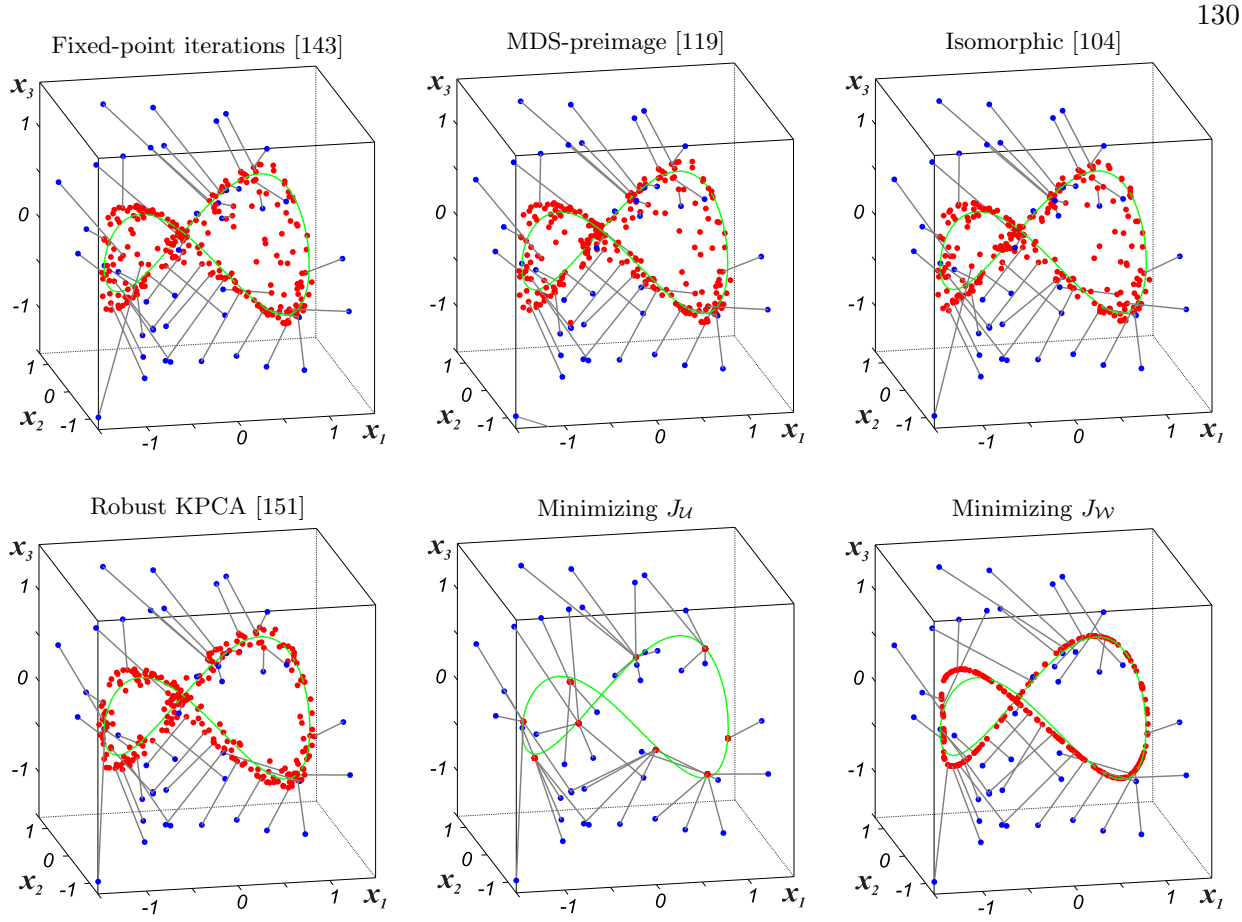
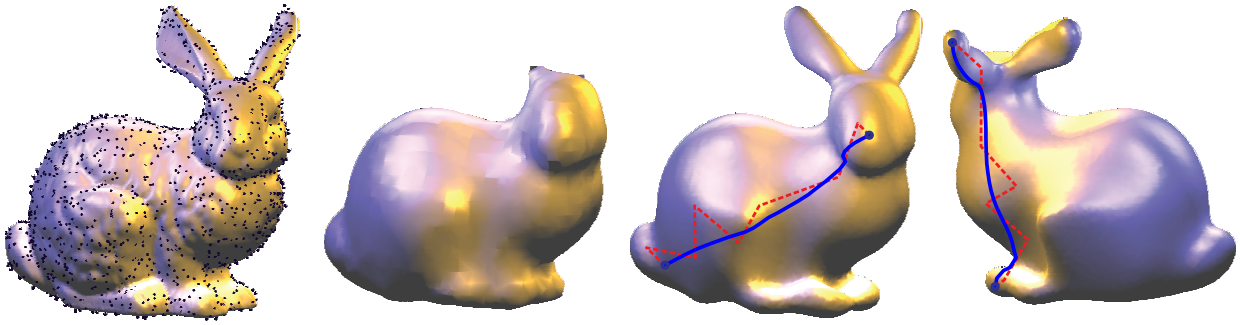


Figure 6.9: Mapping a cloud of random points (blue; not all points shown) onto the manifold using the KPCA denoising strategy [143] with various preimage methods to bring the found feature space solution back to the original space. Note that all algorithms result in points lying closer but not necessarily *on* the manifold (red). Results of minimization of the proposed functional $J_{\mathcal{W}}$ with gradient descent trace a continuous curve giving a good approximation of the initial manifold $\mathcal{M}$.

that we aim to map back to the manifold. For comparison, we run the following KPCA denoising experiment: we project the images of the noisy samples onto the KPCA-parameterized subspace $\mathcal{U}$ in the feature space (induced by the same kernel) and then reconstruct preimages of these projections using different methods: the fixed-point iterative procedure to minimize Eq. 3.6 [143], the MDS-based preimage [119], the isomorphism-preserving method of [104], and the Robust KPCA algorithm [151] regularized with the distance from the initialization. We then compare these with minimizing our $J_{\mathcal{W}}$ via gradient descent. The results of our experiments are shown in Fig. 6.8. Even though all preimage algorithms send the initialization points closer to the manifold, our results ac-



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

Figure 6.10: Learning and interpolation on a surface in $\mathbb{R}^3$. From left to right: (a) Original model and 3000 noisy samples of it and (b) the result of the Poisson surface reconstruction algorithm [112]. Panels (c-d) show our results of learning the surface and examples of tracing curves on it. Our representation with a subspace $\mathcal{W}$ leads to accurate reconstruction of important model features and allows for smooth interpolation on the manifold (blue lines). Using the KPCA parameterization and the corresponding functional $J_{\mathcal{U}}$ to approximate the distance to the manifold results in non-smooth interpolants (red dashed lines).

tually trace a continuous curve that approximates $\mathcal{M}$ fairly well. We run similar experiments for the curve of codimension 2 in $\mathbb{R}^3$, learned from noisy samples, and plot the resulting mappings in Fig. 6.9, where again our parameterization with KODA outperforms other KPCA-denoising-based strategies.

For a more realistic and practical setting, we now turn our attention to other datasets known to be well represented with underlying manifold models, such as images of handwritten digits and faces. We add synthetic zero-mean Gaussian noise to some examples from these datasets (that are not used for training) and then find their mapping onto the manifolds as described above. The reconstruction results are shown in Fig. 6.11. Our method performs comparably or slightly better in terms of PSNR than the KPCA denoising strategy (employing a variety of popular preimage algorithms to map the found feature space solution back to original space). Moreover, our results are one of the best in terms of visual quality. Note in particular how minimizing the distance to the KPCA-parameterized subspace (minimizing $J_{\mathcal{U}}$) results in the iterations converging to the same point on the manifold regardless of initialization; this does not happen with our parameterization, which corroborates its advantages.

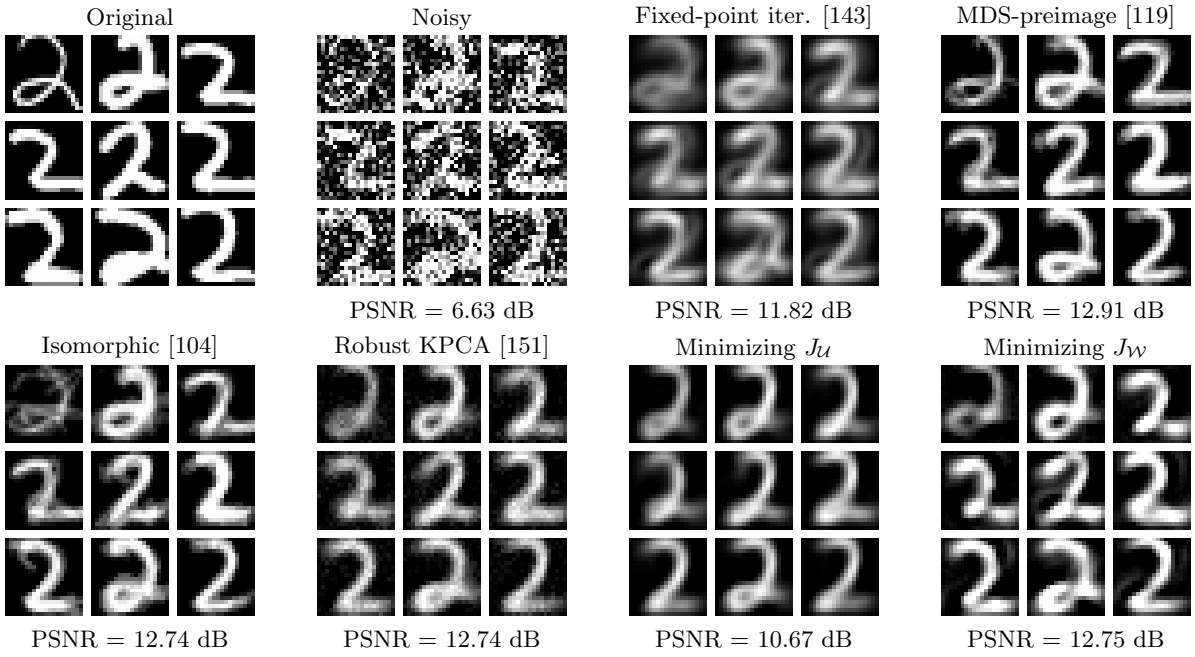


Figure 6.11: Results of denoising images of the digit "2" from the MNIST dataset with a manifold model. Different preimage methods are used to reconstruct the projections onto the KPCA subspace $\mathcal{U}$ in feature space. For comparison, noisy points are mapped onto the manifold by minimizing the $J_{\mathcal{U}}$ and $J_{\mathcal{W}}$ functionals defined on the KPCA and KODA solutions respectively. Notice how minimization of $J_{\mathcal{U}}$ results in exactly the same solutions for several different initialization points.

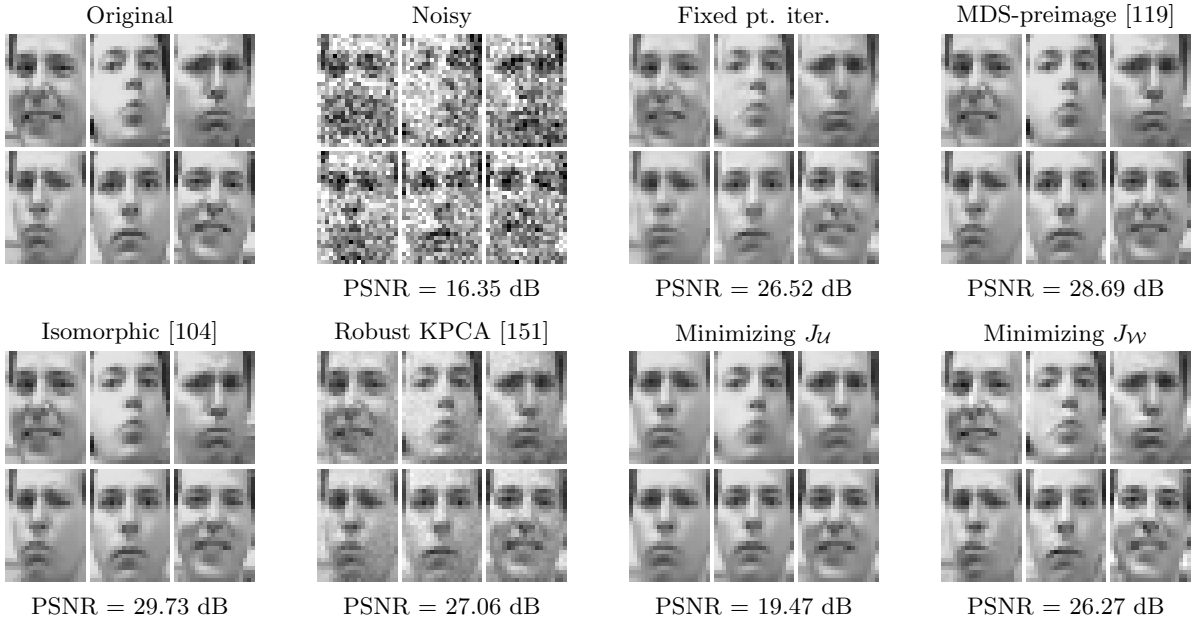


Figure 6.12: Results of the same denoising experiment as in Fig. 6.11, but for the Frey Face dataset.

6.5.3 Interpolation along a Manifold

In this section, we will expand our effective manifold mapping strategy and will address the problem of tracing paths between samples on a non-linear manifold. This setting can be regarded as a building block for many practical tasks that rely on interpolation between two points, finding geodesics, or tracing curves through multiple samples on manifolds.

For this, we consider a method known as Manifold-Snake [22]. Its idea is to approximate piece-wise linearly a curve between two points. The breakpoints $\mathbf{v}_i$, $i = 1, \ldots, n_v$ in the piecewise approximation are chosen to be equidistant and lie close to the manifold. This is done by minimizing a functional that penalizes both aspects:

$$E_{MS} = \sum_{i=1}^{n_v} \left\{ \|\mathbf{v}_{i-1} - 2\mathbf{v}_i + \mathbf{v}_{i+1}\|^2 + \lambda \|\mathbf{v}_i - \mathcal{P}_{\mathcal{M}}(\mathbf{v}_i)\|^2 \right\}, \tag{6.16}$$

where the first term of the sum reduces differences in spacing between consecutive vertices, and the second term forces all points to lie close to their projections onto the manifold, $\mathcal{P}_{\mathcal{M}}(\mathbf{v}_i)$. Here $\mathbf{v}_0$ and $\mathbf{v}_{n+1}$ are the starting and ending points respectively, which are assumed to be fixed.

Our model naturally incorporates into this problem simply by using $J_{\mathcal{W}}(\mathbf{v}_i)$ in place of $\|\mathbf{v}_i - \mathcal{P}_{\mathcal{M}}(\mathbf{v}_i)\|^2$ in the above equation and then minimizing the resulting functional. On the final step, we also minimize $J_{\mathcal{W}}(\mathbf{v}_i)$ for each node individually to assure that they lie on our approximation. For comparison, we run similar experiments but use $J_{\mathcal{U}}$ instead.

As an example, we first consider tracing a curve between two randomly chosen points on the surface of the bunny learned in Section 6.5.1. Minimizing the manifold-snake criterion effectively results in smooth paths lying exactly on the surface of the manifold as shown in Fig. 6.10. We note that the continuity of the manifold representation achieved with our learned parameterization is paramount here; for comparison, modeling the manifold with the KPCA subspace instead, results in a jagged solution with vertices mapped onto the discrete minima of $J_{\mathcal{U}}$.

In our second interpolation example, we fix two randomly chosen samples from the Frey Face dataset and then find a sequence of images that smoothly transforms one into another by tracing a path on the learned face manifold (see Fig. 6.13). We note that the reconstructed images

are inferred from the model and are not present in the original dataset, yet they provide a very good approximation to possible image dynamics. Again, the KPCA parameterization forces the path to shrink to a few distinct images (i.e. to just the distinct minimizers). To better visualize this, we plot the graphs of normalized cumulative distances along both paths, computed as $d_i^{cml} = d_{i-1}^{cml} + \frac{\|\mathbf{v}_i - \mathbf{v}_{i-1}\|}{\sum_{j=1}^{n_v+1}\|\mathbf{v}_j - \mathbf{v}_{j-1}\|}$ for $i = 1, \ldots, n_v + 1$, and $d_0^{cml} = 0$. They vividly show that the paths found by minimizing $J_{\mathcal{U}}$ in Eq. 6.16 have large discontinuities between some pairs of consecutive points, and little distance otherwise between consecutive points, while those traced on the model learned with KODA result in much smoother solutions with equally spaced nodes on the manifold.

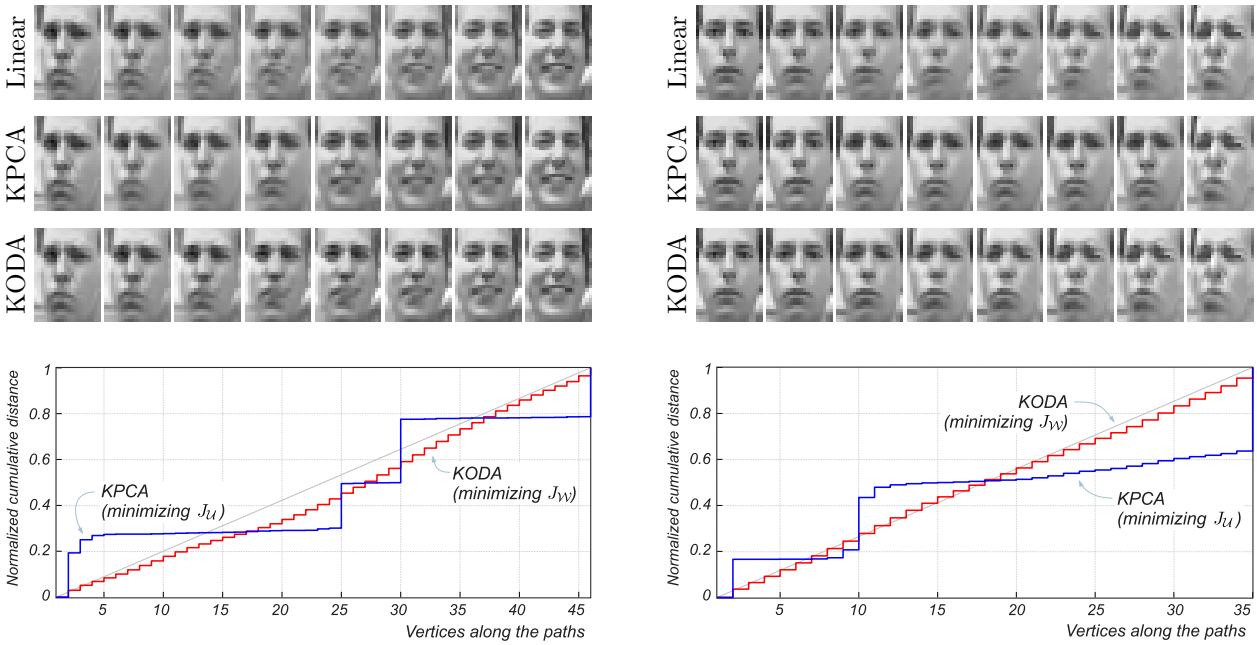


Figure 6.13: Two examples of interpolation on the learned manifold of Frey faces. Top rows: The results of linear interpolation with equidistant nodes; no underlying manifold is assumed in this case. Note the artifacts of linear superposition of the images clearly present in the middle images. Middle rows: The results of the manifold-snake approach with an underlying manifold parameterized via the KPCA subspace. Bottom rows: Our results using KODA parameterization to minimize $J_{\mathcal{W}}$ in Eq. 6.16. The graphs below represent the normalized distances between the first and the i^{th} nodes of the paths. Note how minimizing $J_{\mathcal{U}}$ (in the KPCA approach) creates large jumps between some pairs of consecutive nodes while moving others to essentially the same point. This is the result of these samples converging to the same discrete minimizer of $J_{\mathcal{U}}$. In contrast, parameterization with KODA results in much smoother interpolation with gradual differences between images.

6.5.4 Unsupervised Anomaly Detection

We note that our functions $\mathbf{g}$ actually comprise a useful and distinctive form of dimensionality reduction. Typical manifold learning methods output an estimate of intrinsic position *along* the manifold and are well-suited for, e.g. classification of different parts of the manifold. Yet, the functions $\mathbf{g}$ chart out position relative to the manifold in the ambient space, which is ideal for, e.g. on-manifold/off-manifold and outlier classification problems.

To test the performance of our embedding, we will consider two datasets, each arising from an underlying manifold corrupted with a small fraction of outliers. Here for simplicity we let the number of outliers be known, and thus hope to detect them as those points having the largest values of $J_{\mathcal{W}}$ (or $J_{\mathcal{U}}$ if the KPCA parameterization is used, as in the popular method [99]). We rank all points in the datasets according to these proximity measures and then declare the farthest points to be the sought outliers. As before, we first consider a toy example to illustrate the principles of our method (see Fig. 6.14), and then run our algorithm on the MNIST dataset of handwritten digits [124]. We learn separate manifolds for each digit from training sets containing 10% noisy samples. Results in Fig. 6.15 demonstrate that an embedding based on KODA is better at characterizing and detecting outliers than the KPCA approach [99].

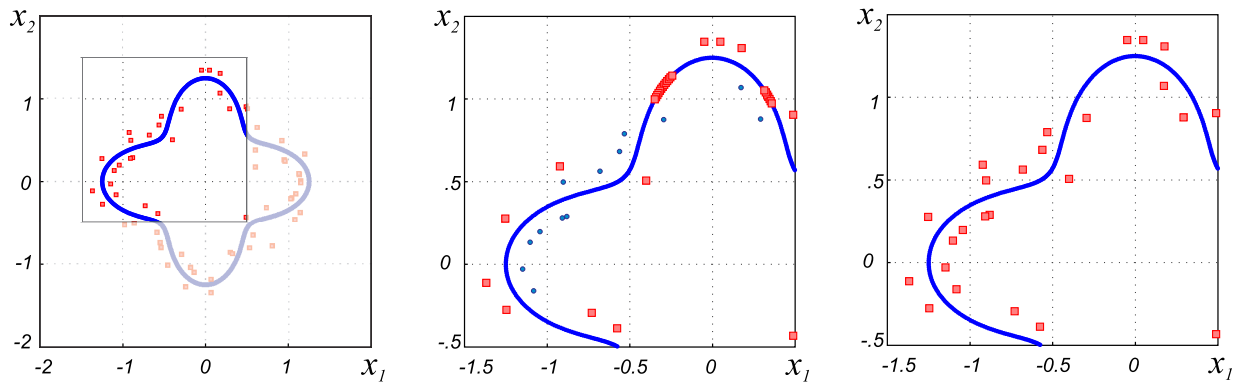


Figure 6.14: Anomaly detection. Left: The training set contains points densely sampled from the manifold (blue line), as well as 10% noisy outliers (red squares). The red squares on the two rightmost plots indicate the 10% of points that have the highest values of $J_{\mathcal{U}}$ (middle) or $J_{\mathcal{W}}$ (right). These are classified as outliers. Due to local minima of $J_{\mathcal{U}}$, some noiseless points appear to be far from the KPCA-parametrized subspace and are misclassified.

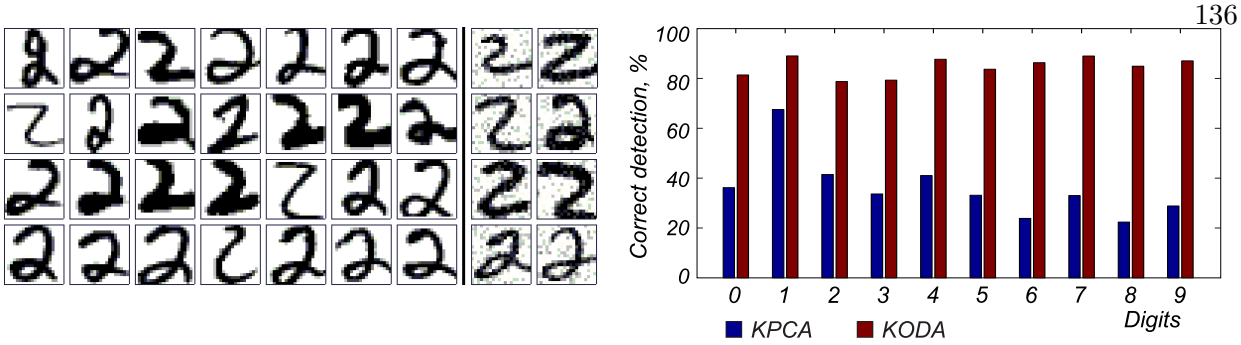


Figure 6.15: Anomaly detection. Results of detecting noisy samples in the MNIST dataset. Left: Examples of images used to learn the manifold of digit "2" including noisy samples. Right: Percentage of correctly detected outliers for each digit. Our algorithm steadily outperforms KPCA.

6.5.5 Multiclass Relevance Ranking

Finally, we consider a generalization of the previous setting to the problem of determining which of several possible manifold-modeled classes a sample most likely belongs to. A close variation of this problem, that of selecting the samples most likely belonging to a specific class, comes up commonly as the relevance ranking problem, e.g. in text categorization [38], where we aim to determine which text samples are good representatives of a given topic.

Specifically, we assume that the instances belonging to each class are expected to lie on their own (sub-)manifolds $\mathcal{M}_c$, $c = 1, \ldots, n_c$, present simultaneously in the same ambient space. We learn each of them separately from their samples. Then, given a testing set $\mathcal{Q} = \{\mathbf{x}_q\}_{q=1}^{n_\mathcal{Q}}$ of many samples belonging to different unspecified classes, our goal is to select from it only the samples of some class c. For this, we measure the distances from each $\mathbf{x}_q$ to the c^{th} manifold, $d(\mathbf{x}_q, \mathcal{M}_c)$, and then form an ordered list of samples according to it.

The exact form of the distance expression is method-dependent. For KPCA-parameterized manifolds, we use the proximity functional $J_\mathcal{U}$ to approximate the true distance. Similarly, for KODA, we use $J_\mathcal{W}$; we define analogous expressions, representing the distance squared to the subspace in feature space, for the One Class SVM [172] and LS-SVM [49] algorithms as well. Finally, we also use the distances to the centers of training samples of each class in the feature space (kernel mean, KM), $\|\mathbf{x}_q - \mathbf{m}_c\|$, for this purpose.

Given the true classes of samples, the performance of ranking is quantified using the average precision metric [38, 49], which is computed for each class as $\text{AP} = \frac{1}{n_\mathcal{Q}} \sum_{1 \le k \le n_\mathcal{Q}} \mathbf{r}_k \mathbf{p}_k$. Here $\mathbf{r}$

stands for a relevance vector with entries $\mathbf{r}_k = 1$ if the k^{th} sample in the ordered list (i.e. the k^{th} closest sample to the manifold) belongs to the class c and $\mathbf{r}_k = 0$ otherwise; $\mathbf{p}_k = \sum_{1 \leq i \leq k} \frac{\mathbf{r}_i}{k}$ is the precision at rate k. Note that $0 \leq \text{AP} \leq 1$ and that it attains its maximum if all relevant samples of the class c are ranked on top of the list (i.e. they have the lowest distances to the manifold compared to the other samples in $\mathcal{Q}$). Using average precision allows us to avoid directly comparing the distances from the same testing sample to different manifolds in possibly different feature spaces but instead gives us the means for more fair comparison of the learned manifold models themselves. For our experiment, we will compute this measure for each of the possible classes. We then give minimum, maximum, and average values of AP across all possible classes as aggregate measures of the algorithms' performance.

Table 6.2: Average precision (AP) of multiclass ranking using different manifold models for three different datasets.

		Kernel Mean	KPCA	One Class SVM	LS-SVM	KODA
MNIST	min	0.3	**0.44**	0.422	0.27	0.247
	avg	0.582	**0.766**	0.694	0.567	0.71
	max	0.918	**0.995**	0.926	0.879	**0.995**
20NG	min	0.025	0.038	0.025	0.039	**0.085**
	avg	0.065	0.086	0.066	0.064	**0.175**
	max	0.155	0.212	0.154	0.102	**0.307**
BAF	min	0.099	0.109	0.105	0.122	**0.184**
	avg	0.264	0.302	0.261	0.339	**0.369**
	max	0.662	0.639	0.738	**0.828**	0.709

For examples in this section, we use three datasets: the set of MNIST digits [124], the 20 Newsgroups dataset [120], and the Body Attack Fitness dataset [78]. We learn the ten manifolds in the MNIST dataset from 12752 training samples approximately equally distributed among the classes and use the remaining 2248 samples to form the query $\mathcal{Q}$. Similarly, in the 20-Newsgroups dataset, we use 9839 and 4215 samples for training and testing respectively, all approximately equally distributed among 20 classes. In the Body Attack Fitness dataset, we learn the manifolds for each of six classes from 3500 points randomly sampled from the available time series for each class. However, we noticed that all algorithms struggle to get good results with this dataset.

Hence, to boost their performance, for testing, we have represented each query by a sequence of 20 consecutive time samples (we consider 25 such sequences of each class). The proximity measure of a query to a manifold is computed by summing the distances from each sample in the sequence to this manifold. By doing so, we make use of temporal correlation between close samples in a query. For example, when a single sample lies close to (or even on) multiple manifolds, making its classification ambiguous, we may look at a few of its previous and next neighbors in the sequence to make a more informed decision about the query as a whole. The testing samples in all experiments were excluded from the training sets. Our results shown in Table 6.2 indicate the apparent advantages of KODA over other manifold approximating methods. The minimum, maximum, and average values are reported with respect to different classes in each dataset.

6.6 Conclusion

In this chapter we have revealed a shortcoming of the parameterization of manifold-approximating subspaces in feature space with their principal components, which becomes especially conspicuous in applications that rely on finding mappings onto the approximated manifold. We then proposed to use an alternative parameterization for the subspace defined in terms of its normal components. Furthermore, we introduced a novel method of Kernel Orthogonal Direction Analysis to efficiently learn such parameterizations. Like kernel PCA, KODA takes a simple kernel-based approach, and requires only solving a generalized eigenproblem, which can be equivalently reformulated as two simple eigenproblems. However, unlike kernel PCA, it produces a continuous representation of the manifold as a level set of its solution, and is thus extremely well-suited for problems of learning continuous manifolds from few or noisy samples of them, interpolation along a manifold, and mapping nearby points onto it. It further results in a type of dimensionality reduction that is very well-suited for anomaly detection, and is well-suited for measuring distance to the manifold, e.g. for relevance ranking problems. Finally, we have experimentally shown how KODA outperforms KPCA-based, and LS-SVM-based, approaches for these purposes, even though these other approaches are frequently used in the literature.

Chapter 7

Conclusion

Recent advances in image processing demonstrate the superiority of patch-based techniques across a wide spectrum of practical problems, ranging from denoising and compressive sensing reconstruction to structural editing. Several manifold models have been proposed as an elegant way to impose a structure on the set of image patches. Despite their successes in reconstructing single patches, most of them stumble at the necessity to simultaneously consider a large number of overlapping patches found in the same image. These observations motivated us to propose a novel manifold-based model for entire images. In our approach, we treated the constraints corresponding to overlapping image patches as separate intersecting manifolds, which led to a conclusion that the entire image lies on their intersection.

Finding intersections of many non-linear manifolds, however, is not an easy problem. Using kernel methods, we have developed two effective approaches for it. First, our kernelized version of the Projection onto Convex Sets (POCS) algorithm expressed in closed form allows one to quickly approximate the solution in a kernel-induced feature space. To our best knowledge, this efficient non-linear extension of the popular POCS algorithm has not been reported in the literature and constitutes one of several main novelties of our work. Despite its simplicity, it shows promising results in image denoising as well as in an important problem of set extrapolation.

Unfortunately, as with many other kernel-based algorithms, our kernelized POCS suffers from the necessity of solving a difficult preimage problem once the solution is located in feature space. Since this step directly affects the final result of reconstruction, we considered combining both

problems, namely finding the manifolds intersection and finding a suitable preimage for it, in a single iterative procedure. Furthermore, we have tailored it specifically for patch-based image processing, which eventually resulted in a practical framework for effectively solving any linear inverse problem without need to make modifications to the algorithm. Rather surprisingly, our universal method compares favorably with, and very often surpasses, modern highly specialized image processing algorithms, each designed to address a specific problem. For example, our results in denoising and compressive sensing of natural photographic images, as well as for textures and patterns, are comparable or slightly better than those obtained with current state-of-the-art methods, such as the BM3D algorithm for denoising. Moreover, we achieved excellent results in image inpainting, vastly outperforming popular existing algorithms. These encouraging results further corroborate our view of the effectiveness of our manifolds intersection model of overlapping patches and confidently establish it as an excellent choice for solving inverse problems in image processing.

Finally, thorough inspection has revealed an important shortcoming of the widely used kernel PCA-based strategy of mapping points onto manifolds, which essentially underlies our intersection finding algorithm as well. The probable absence of exact preimages for projections performed in the higher-dimensional feature space very often causes the iterative minimization algorithm to converge to a discrete set of disconnected points rather than to trace a continuous manifold. Even though this effect could be negligible when the intrinsic dimension of the manifold is low comparing to the ambient space – as is the case in our image processing experiments – it leads to significant errors in applications that involve working with manifolds of low *codimension*. Recognizing this, we have proposed and developed a novel manifold learning method – Kernel Orthogonal Direction Analysis. Although not unrelated to kernel PCA, it is based on an alternative parameterization of the manifold approximating subspace in the feature space, in terms of its normals rather than its basis vectors, which makes it especially suitable for mapping points onto manifolds. Unlike other popular preimage methods, our approach is able to reconstruct the continuous structure of the manifold from few or noisy samples of it and can be used for interpolation on manifolds as well as successfully solving problems of classification and anomaly detection in manifold-modeled datasets.

7.1 Possible Directions for Future Work

To project our ideas into the future, we would like to note that the iterative nature of our patch-based image processing algorithm is essential for its extension to multiple inverse problems. Nevertheless, it could be a relatively time-consuming procedure by modern standards, causing the method to lose some of its appeal for processing large images in real-time. A remedy to remove this obstacle can come in the form of deep neural networks, which offer extremely fast inference with specifically designed and trained function approximators. In particular, the recently proposed framework of deep unfolding has been proved effective in not only speeding up, but also improving, the results of many popular iterative thresholding algorithms [89, 97, 186]. Here, instead of looking at an algorithm as an iterative pursuit, its steps become successively connected layers in a trainable structure.

If the process of minimizing our distance approximating functionals $J_{\mathcal{U}}$ and $J_{\mathcal{W}}$ could be cast in a similar deep network form, for certain types of kernels (such as the Gaussian kernel) it may give rise to a class of so-called rbf-networks [23]. Being notoriously difficult to train, rbf-networks of increasing depth receive sizeably less attention in modern applications than their sigmoid-based counterparts. However, recent unexpected discoveries of adversarial examples for traditional sigmoidal classifiers [88, 188] indicate that the widely used models may not be "non-linear enough" to faithfully capture the underlying structures of datasets in high-dimensional spaces; rbf-networks notably lack this shortcoming. Thus, we foresee that the manifold representation with its proximity functionals (either $J_{\mathcal{U}}$ or $J_{\mathcal{W}}$) that we employ in our work could be very useful in providing a way to initialize deep rbf-networks and potentially facilitate their training, which is particularly important for problems of unsupervised learning.

Bibliography

[1] M. F. Abdelkader, W. Abd-Almageed, A. Srivastava, and R. Chellappa. Silhouette-based gesture and action recognition via modeling trajectories on Riemannian shape manifolds. Computer Vision and Image Understanding, 115(3):439 – 455, 2011. Special issue on Feature-Oriented Image and Video Computing for Extracting Contexts and Semantics.

[2] T. J. Abrahamsen and L. K. Hansen. Input space regularization stabilizes pre-images for kernel PCA de-noising. In Machine Learning for Signal Processing, 2009. MLSP 2009. IEEE International Workshop on, pages 1–6, 2009.

[3] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. Signal Processing, IEEE Transactions on, 54(11):4311–4322, Nov 2006.

[4] P. Aljabar, R. Wolz, and D. Rueckert. Manifold Learning for Medical Image Registration, Segmentation, and Classification. IGI Global, 2012.

[5] N. Aronszajn. Theory of reproducing kernels. Transactions of the American Mathematical Society, 68(3):337–404, 1950.

[6] S. Baker and T. Kanade. Limits on super-resolution and how to break them. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(9):1167–1183, Sep 2002.

[7] G. H. Bakır, J. Weston, and B. Schölkopf. Learning to find pre-images. In Advances in Neural Information Processing Systems, pages 449–456. MIT Press, 2004.

[8] P. A. Bandettini. What's new in neuroimaging methods? Annals of the New York Academy of Sciences, 1156(1):260–293, 2009.

[9] R. G. Baraniuk and M. B. Wakin. Random projections of smooth manifolds. In Foundations of Computational Mathematics, pages 941–944, 2006.

[10] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: a randomized correspondence algorithm for structural image editing. ACM Transactions on Graphics (Proc. SIGGRAPH), 28(3):24:1–24:11, July 2009.

[11] M. F. Barnsley and L. P. Hurd. Fractal Image Compression. AK Peters, 1993.

[12] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM Journal on Imaging Sciences, 2(1):183–202, Mar 2009.

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

[14] A. Ben-Hur and W. S. Noble. Kernel methods for predicting protein-protein interactions. In ISMB (Supplement of Bioinformatics), pages 38–46, 2005.

[15] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, J. A. Levine, A. Sharf, and C. T. Silva. State of the art in surface reconstruction from point clouds. In Eurographics 2014 - State of the Art Reports. The Eurographics Association, 2014.

[16] J. C. M. Bermudez, P. Honeine, J.-Y. Tourneret, and C. Richard. Kernel-based nonlinear signal processing. In Signals and Images: Advances and Results in Speech, Estimation, Compression, Recognition, Filtering, and Processing, chapter 2. 2015.

[17] G. Blanchard, O. Bousquet, and L. Zwald. Statistical properties of kernel principal component analysis. Machine Learning, 66(2-3):259–294, March 2007.

[18] T. Blumensath. Sampling and reconstructing signals from a union of linear subspaces. IEEE Transactions on Information Theory, 57(7):4660–4671, 2011.

[19] J. Bobin, J.-L. Starck, and R. Ottensamer. Compressed sensing in astronomy. IEEE Journal of Selected Topics in Signal Processing, 2:718–726, November 2008.

[20] S. Boyd and L. Vandenberghe. Convex Optimization. Cambridge University Press, New York, NY, USA, 2004.

[21] M. Brand. Incremental singular value decomposition of uncertain data with missing values. In ECCV (1), volume 2350 of Lecture Notes in Comp. Science, pages 707–720. Springer, 2002.

[22] C. Bregler and S. M. Omohundro. Nonlinear image interpolation using manifold learning. In Advances in Neural Information Processing Systems 7, pages 973–980. MIT Press, 1995.

[23] D. S. Broomhead and D. Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Complex Systems, 2:321–355, March 1988.

[24] T. Brox and D. Cremers. Iterated nonlocal means for texture restoration. In Scale Space and Variational Methods in Computer Vision, First International Conference, SSVM 2007, Proceedings of, pages 13–24, May–June 2007.

[25] A. Buades, B. Coll, and J. M. Morel. A review of image denoising algorithms, with a new one. Multiscale Modeling & Simulation, 4(2):490–530, 2005.

[26] A. Buades, B. Coll, and J.-M. Morel. Non-local means denoising. Image Processing On Line, 1, 2011.

[27] H. C. Burger, C. Schuler, and S. Harmeling. Learning how to combine internal and external denoising methods. In Pattern Recognition, volume 8142 of Lecture Notes in Computer Science, pages 121–130. Springer Berlin Heidelberg, 2013.

[28] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with BM3D? In IEEE Conference on Computer Vision and Pattern Recognition, pages 2392–2399, June 2012.

[29] C. J. C. Burges. Simplified support vector decision rules. In Proceedings of 13th International Conference on Machine Learning, pages 71–77, San Mateo, CA, 1996. Morgan Kaufmann.

[30] D. Butnariu, Y. Censor, P. Gurfil, and E. Hadar. On the behavior of subgradient projections methods for convex feasibility problems in Euclidean spaces. SIAM Journal on Optimization, 19(2):786–807, 2008.

[31] C. L. Byrne. Applied iterative methods. Ak Peters Series. Wellesley, MA: AK Peters, 2008.

[32] J. A. Cadzow. Signal enhancement – a composite property mapping algorithm. Acoustics, Speech and Signal Processing, IEEE Transactions on, 36(1):49–62, 1988.

[33] E. J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. Information Theory, IEEE Transactions on, 52(2):489–509, February 2006.

[34] E.J. Candes and D.L. Donoho. Curvelets: A Surprisingly Effective Nonadaptive Representation for Objects with Edges. Technical report (Stanford University. Dept. of Statistics). Department of Statistics, Stanford University, 1999.

[35] G. Carlsson. Topology and data. Bulletin of the American Mathematical Society, 46(2):255–308, Apr 2009.

[36] G. Carlsson, T. Ishkhanov, V. Silva, and A. Zomorodian. On the local behavior of spaces of natural images. International Journal of Computer Vision, 76(1):1–12, January 2008.

[37] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, SIGGRAPH '01, pages 67–76, New York, NY, USA, 2001. ACM.

[38] S. Chakrabarti. Mining the Web: Discovering Knowledge from Hypertext Data. Morgan Kaufmann, Amsterdam, 2003.

[39] A. Chambolle. An algorithm for total variation minimization and applications. Journal of Mathematical Imaging and Vision, 20(1-2):89–97, January 2004.

[40] Tony F. Chan, Jianhong Shen, and Hao-Min Zhou. Total variation wavelet inpainting. Journal of Mathematical Imaging and Vision, 25(1):107–125, 2006.

[41] Y. Chang, C. Hu, R. Feris, and M. Turk. Manifold based analysis of facial expression. Image Vision Comput., 24(6):605–614, June 2006.

[42] P. Chatterjee and P. Milanfar. Clustering-based denoising with locally learned dictionaries. Image Processing, IEEE Transactions on, 18(7):1438–1451, July 2009.

[43] P. Chatterjee and P. Milanfar. Is denoising dead? IEEE Transactions on Image Processing, 19(4):895–911, April 2010.

[44] P. Chatterjee and P. Milanfar. Practical bounds on image denoising: From estimation to information. IEEE Transactions on Image Processing, 20(5):1221–1233, 2011.

[45] M. Chen, J. Silva, J. Paisley, C. Wang, D. Dunson, and L. Carin. Compressive sensing on manifolds using a nonparametric mixture of factor analyzers: Algorithm and performance bounds. IEEE Transactions on Signal Processing, 58(12):6140 –6155, Dec 2010.

[46] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. SIAM Journal on Scientific Computing, 20(1):33–61, Jan 1998.

[47] T.-J. Chin, K. Schindler, and D. Suter. Incremental kernel SVD for face recognition with image sets. In Automatic Face and Gesture Recognition, 2006, 7th International Conference on, pages 461–466, April 2006.

[48] T.-J. Chin and D. Suter. Incremental kernel principal component analysis. Image Processing, IEEE Transactions on, 16(6):1662–1674, June 2007.

[49] Y.-S. Choi. Least squares one-class support vector machine. Pattern Recognition Letters, 30(13):1236–1240, 2009.

[50] G. Cimmino. Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari. La Ricerca Scientifica, 16(9):326–333, 1938.

[51] M. Coelho, N. Maghelli, and I. M. Tolic-Norrelykke. Single-molecule imaging in vivo: the dancing building blocks of the cell. Integr. Biol., 5:748–758, 2013.

[52] P.L. Combettes. The foundations of set theoretic estimation. Proceedings of the IEEE, 81(2):182–208, 1993.

[53] A. Criminisi, P. Pérez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. IEEE Transactions on Image Processing, 13(9):1200–1212, 2004.

[54] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-D transform-domain collaborative filtering. Image Processing, IEEE Trans. on, 16(8):2080–2095, 2007.

[55] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. BM3D image denoising with shape-adaptive principal component analysis. In Proc. Workshop on Signal Processing with Adaptive Sparse Structured Representations (SPARS'09), Saint Malo, France, Apr 2009.

[56] A. Danielyan, A. Foi, V. Katkovnik, and K. Egiazarian. Spatially adaptive filtering as regularization in inverse imaging: compressive sensing, upsampling, and super-resolution. Super-Resolution Imaging, 2010.

[57] A. Danielyan, V. Katkovnik, and K. Egiazarian. BM3D frames and variational image deblurring. IEEE Transactions on Image Processing, 21(4):1715–1728, 2012.

[58] S. Dasgupta and A. Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. Random Struct. Algorithms, 22(1):60–65, January 2003.

[59] I. Daubechies. Ten Lectures on Wavelets. Society for Industrial and Applied Math., 1992.

[60] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. Communications on Pure and Applied Mathematics, 57(11):1413–1457, 2004.

[61] A.R. De Pierro and Iusem A.N. A parallel projection method for finding a common point of a family of convex sets. Pesquisa Operacional, 5(1):1–20, Jul. 1985.

[62] D. DeMenthon, M. V. Stuckelberg, and D. Doermann. Hidden Markov models for images. In ICPR, pages 147–150, 2000.

[63] J. E. Dennis, Jr. and R. B. Schnabel. Numerical Methods for Unconstrained Optimization and Nonlinear Equations (Classics in Applied Mathematics, 16). Society for Industrial and Applied Mathematics, 1996.

[64] M. Ding and G. Fan. Multilayer joint gait-pose manifolds for human gait motion modeling. IEEE Transactions on Cybernetics, 45(11):2413–2424, 2015.

[65] M.N. Do and M. Vetterli. The contourlet transform: an efficient directional multiresolution image representation. Image Processing, IEEE Transactions on, 14(12):2091–2106, 2005.

[66] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. ArXiv e-prints, December 2015.

[67] D. L. Donoho. Wedgelets: nearly minimax estimation of edges. The Annals of Statistics, 27(3):859–897, 06 1999.

[68] D. L. Donoho. For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution. Communications on Pure and Applied Mathematics, 59(6):797–829, 2006.

[69] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH'01, pages 341–346, New York, NY, USA, 2001. ACM.

[70] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In Proceedings of the International Conference on Computer Vision, volume 2 of ICCV '99, pages 1033–1038, Washington, DC, USA, 1999. IEEE Computer Society.

[71] K. Egiazarian, A. Foi, and V. Katkovnik. Compressed sensing image reconstruction via recursive spatially adaptive filtering. In ICIP 2007, volume 1, pages 549–552, Sept 2007.

[72] D. Eigen, D. Krishnan, and R. Fergus. Restoring an image taken through a window covered with dirt or rain. In ICCV'13, pages 633–640, 2013.

[73] M. Elad and M. Aharon. Image denoising via learned dictionaries and sparse representation. In CVPR, pages 17–22, 2006.

[74] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. Trans. Img. Proc., 15(12):3736–3745, December 2006.

[75] K. Engan, S. O. Aase, and J. H. Husoy. Method of optimal directions for frame design. In Acoustics, Speech, and Signal Processing, 1999, IEEE International Conference on, volume 5, pages 2443–2446. IEEE, 1999.

[76] Y. Engel, S. Mannor, and R. Meir. The kernel recursive least-squares algorithm. Signal Processing, IEEE Transactions on, 52(8):2275–2285, 2004.

[77] D. A. Fidaleo. G-folds: an appearance-based model of facial gestures for performance driven facial animation. PhD thesis, Los Angeles, CA, USA, 2003.

[78] K. Forster, D. Roggen, and G. Troster. Unsupervised classifier self-calibration through repeated context occurences: Is there robustness against sensor displacement to gain? In Wearable Computers, 2009. ISWC '09. International Symposium on, pages 77–84, Sept 2009.

[79] V. Franc and V. Hlaváč. Greedy algorithm for a training set reduction in the kernel methods. In Computer Analysis of Images and Patterns, volume 2756 of Lecture Notes in Computer Science, pages 426–433. Springer Berlin Heidelberg, 2003.

[80] W. T. Freeman, T. R. Jones, and E. C. Pasztor. Example-based super-resolution. Computer Graphics and Applications, IEEE, 22(2):56–65, March 2002.

[81] B. J. Frey, A. Colmenarez, and T. S. Huang. Mixtures of local linear subspaces for face recognition. In IEEE Conference on Computer Vision and Pattern Recognition, pages 32–37. IEEE Computer Society, 1998.

[82] J. C. Gebhardt, D. M. Suter, R. Roy, Z. W. Zhao, A. R. Chapman, S. Basu, T. Maniatis, and X. S. Xie. Single-molecule imaging of transcription factor binding to DNA in live mammalian cells. Nature Methods, 10(5):421–426, May 2013.

[83] D. Geman and A. Koloydenko. Invariant statistics and coding of natural microimages. In IEEE Workshop on Statist. Computat. Theories of Vision, Fort Collins, 1998.

[84] S. Gerber, T. Tasdizen, P.T. Fletcher, S. Joshi, and R.T. Whitaker. Manifold modeling for brain population analysis. Medical Image Analysis, 14(5):643–53, 10 2010.

[85] M. Ghazel, G.H. Freeman, and E.R. Vrscay. Fractal image denoising. Image Processing, IEEE Transactions on, 12(12):1560–1578, Dec 2003.

[86] R. Giryes and M. Elad. Sparsity-based Poisson denoising with dictionary learning. IEEE Transactions on Image Processing, 23:5057–5069, December 2014.

[87] G. H. Glover. Overview of functional magnetic resonance imaging. Neurosurgery Clinics of North America, 22:133–139, 2011.

[88] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. ArXiv e-prints, December 2014.

[89] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In International Conference on Machine Learning, pages 399–406, 2010.

[90] C. Grienberger and A. Konnerth. Imaging calcium in neurons. Neuron, 73(5):862 – 885, 2012.

[91] V. Guillemin and A. Pollack. Differential Topology. Mathematics Series. Prentice-Hall, 1974.

[92] S. Günter, N. N. Schraudolph, and S.V.N. Vishwanathan. Fast iterative kernel principal component analysis. Journal of Machine Learning Research, 8:1893–1918, December 2007.

[93] J. Ham, D. D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. In Proceedings of the twenty-first international conference on Machine learning, ICML'04, pages 47–54, New York, NY, USA, 2004. ACM.

[94] J. Hamm, D. H. Ye, R. Verma, and C. Davatzikos. Gram: A framework for geodesic registration on anatomical manifolds. Medical Image Analysis, 14(5):633–642, 2010.

[95] V. Hartwig, G. Giovannetti, N. Vanello, M. Lombardi, L. Landini, and S. Simi. Biological effects and safety in magnetic resonance imaging: A review. International Journal of Environmental Research and Public Health, 6(6):1778, 2009.

[96] K. K. Herrity, A. C. Gilbert, and J. A. Tropp. Sparse approximation via iterative thresholding. In IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2006, volume 3, pages 624–627, May 2006.

[97] J. R. Hershey, J. Le Roux, and F. Weninger. Deep unfolding: Model-based inspiration of novel deep architectures. ArXiv e-prints, September 2014.

[98] J. Ho, M.-H. Yang, and D. Kriegman. Video-based face recognition using probabilistic appearance manifolds. In Proc. IEEE Conference on Computer Vision and Pattern Recognition, pages 313–320, 2003.

[99] H. Hoffmann. Kernel PCA for novelty detection. Pattern Recognition, 40(3):863–874, 2007.

[100] P. Honeine. Online kernel principal component analysis: A reduced-order model. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 34(9):1814–1826, Sept 2012.

[101] P. Honeine and C. Richard. A closed-form solution for the pre-image problem in kernel-based machines. Journal of Signal Processing Systems, 65(3):289–299, December 2011.

[102] P. Honeine and C. Richard. Preimage problem in kernel-based machine learning. Signal Processing Magazine, IEEE, 28(2):77–88, 2011.

[103] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. Neural Netw., 2(5):359–366, July 1989.

[104] D. Huang, Y. Tian, and F. De la Torre. Local isomorphism to solve the pre-image problem in kernel methods. In IEEE International Conference on Computer Vision and Pattern Recognition, pages 2761–2768. IEEE, 2011.

[105] V. Jain and S. Seung. Natural image denoising with convolutional networks. In Advances in Neural Information Processing Systems 21, pages 769–776. 2009.

[106] F. Jäkel, B. Schölkopf, and F. A. Wichmann. A tutorial on kernel methods for categorization. Journal of Mathematical Psychology, 51(6):343 – 358, 2007.

[107] W. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In Conference in modern analysis and probability, volume 26 of Contemporary Mathematics, pages 189–206. American Mathematical Society, 1984.

[108] I. T. Jolliffe. Principal Component Analysis. Springer, second edition, October 2002.

[109] S. Kaczmarz. Approximate solution of systems of linear equations. International Journal of Control, 57(6):1269–1271, 1993.

[110] M. Kallas, P. Honeine, C. Richard, C. Francis, and H. Amoud. Non-negative pre-image in machine learning for pattern recognition. In Proc. 19th European Conference on Signal Processing, pages 931–935, Barcelona, Spain, 29 Aug. - 2 Sept. 2011.

[111] M. Kallas, P. Honeine, C. Richard, C. Francis, and H. Amoud. Non-negativity constraints on the pre-image for pattern recognition with kernel machines. Pattern Recognition, 46(11):3066–3080, November 2013.

[112] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In Proceedings of the Fourth Eurographics Symposium on Geometry Processing, pages 61–70, 2006.

[113] C. Kervrann and J. Boulanger. Local adaptivity to variable smoothness for exemplar-based image regularization and representation. International Journal of Computer Vision, 79(1):45–69, 2008.

[114] K. I. Kim, M. O. Franz, and B. Schölkopf. Iterative kernel principal component analysis for image modeling. IEEE Trans. Pattern Analysis and Machine Intelligence, 27(9):1351–1366, 2005.

[115] S. Kimura, S. Ozawa, and S. Abe. Incremental kernel PCA for online learning of feature space. In International Conference on Computational Intelligence for Modelling, Control and Automation, 2005, volume 1, pages 595–600, Nov 2005.

[116] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In Advances in NIPS 25, pages 1097–1105. 2012.

[117] I. K. Kwang, M. O. Franz, and B. Schölkopf. Iterative kernel principal component analysis for image modeling. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(9):1351–1366, September 2005.

[118] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. ACM Transactions on Graphics, SIGGRAPH 2003, 22(3):277–286, July 2003.

[119] J. T.-Y. Kwok and I. W.-H. Tsang. The pre-image problem in kernel methods. Neural Networks, IEEE Transactions on, 15(6):1517–1525, nov. 2004.

[120] K. Lang. Newsweeder: Learning to filter netnews. In Proceedings of the Twelfth International Conference on Machine Learning, pages 331–339, 1995.

[121] N. D. Lawrence, M. Seeger, and R. Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In Advances in Neural Information Processing Systems, number 15, pages 609–616. MIT Press, 2003.

[122] M. Lebrun, M. Colom, and J.-M. Morel. The Noise Clinic: a blind image denoising algorithm. Image Processing On Line, 5:1–54, 2015.

[123] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural Comput., 1(4):541–551, December 1989.

[124] Y. Lecun and C. Cortes. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

[125] A. B. Lee, K. S. Pedersen, and D. Mumford. The nonlinear statistics of high-contrast patches in natural images. International Journal on Computer Vision, 54(1-3):83–103, August 2003.

[126] J. M. Lee. Introduction to Smooth Manifolds. Graduate Texts in Mathematics. Springer, 2003.

[127] S. Lesage, R. Gribonval, F. Bimbot, and L. Benaroya. Learning unions of orthonormal bases with thresholded singular value decomposition. In IEEE International Conf. on Acoustics, Speech, and Signal Processing, ICASSP 2005., volume 5, pages 293–296, March 2005.

[128] A. Levin and B. Nadler. Natural image denoising: Optimality and inherent bounds. In IEEE International Conference on Computer Vision and Pattern Recognition, pages 2833–2840. IEEE Computer Society, 2011.

[129] Y. Li and S. Osher. Coordinate descent optimization for ℓ_1 minimization with application to compressed sensing; a greedy algorithm. Inverse Problems and Imaging, 3(3):487–503, 2009.

[130] D. Lin and J. Fisher. Manifold guided composite of Markov random fields for image modeling. In IEEE Conf. on Computer Vision and Pattern Recognition, pages 2176–2183, June 2012.

[131] Z. Lin, J. He, X. Tang, and C.-K. Tang. Limits of learning-based superresolution algorithms. International Journal of Computer Vision, 80(3):406–420, 2008.

[132] H.-M. Lu, Y. Fainman, and R. Hecht-Nielsen. Image manifolds. In Applications of Artificial Neural Networks in Image Processing III, Proceedings of SPIE, pages 52–63, Bellingham, WA, USA, 1998. SPIE.

[133] Y. M. Lui. Advances in matrix manifolds for computer vision. Image and Vision Computing, 30(6-7):380–388, June 2012.

[134] P. Y. Lum, G. Singh, A. Lehman, T. Ishkanov, M. Vejdemo-Johansson, M. Alagappan, J. Carlsson, and G. Carlsson. Extracting insights from the shape of complex data using topology. Scientific Reports, 3, February 2013.

[135] D. Lundqvist, A. Flykt, and A. Öhman. The Karolinska Directed Emotional Faces – KDEF. [CD-ROM], 1998.

[136] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly. Compressed sensing MRI. Signal Processing Magazine, IEEE, 25(2):72–82, March 2008.

[137] M. Mahmoudi and G. Sapiro. Fast image and video denoising via nonlocal means of similar neighborhoods. IEEE Signal Processing Letters, 12(12):839–842, December 2005.

[138] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In IEEE 12th International Conference on Computer Vision, pages 2272–2279. IEEE, 2009.

[139] S. Mallat. A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way. Academic Press, 3rd edition, 2008.

[140] S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. Signal Processing, IEEE Transactions on, 41(12):3397–3415, Dec 1993.

[141] M. Marsousi, K. Abhari, P. Babyn, and J. Alirezaie. An adaptive approach to learn overcomplete dictionaries with efficient numbers of elements. IEEE Transactions on Signal Processing, 62(12):3272–3283, 2014.

[142] R. Mazhar and P. D. Gader. EK-SVD: Optimized dictionary design for sparse representations. In IEEE 19th International Conference on Pattern Recognition, pages 1–4. IEEE, 2008.

[143] S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and de-noising in feature spaces. In Proceedings of the 1998 conference on Advances in NIPS, pages 536–542, Cambridge, MA, USA, 1999. MIT Press.

[144] P. Milanfar. A tour of modern image filtering: New insights and methods, both practical and theoretical. Signal Processing Magazine, IEEE, 30(1):106–128, Jan 2013.

[145] P. P. Mondal. Temporal resolution in fluorescence imaging. Frontiers in Molecular Biosciences, 1(11), 2014.

[146] I. Mosseri, M. Zontak, and M. Irani. Combining the power of internal and external denoising. 2014 IEEE International Conference on Computational Photography (ICCP), 0:1–9, 2013.

[147] A. Murli, L. D'Amore, and V. De Simone. The Wiener filter and regularization methods for image restoration problems. In 10th International Conference on Image Analysis and Processing (ICIAP 1999), pages 394–399, September.

[148] S. K. Nayar, H. Murase, and S. A. Nene. Parametric appearance representation. In Early Visual Learning, pages 131–160. Oxford University Press, 1996.

[149] D. Needell and R. Ward. Stable image reconstruction using total variation minimization. SIAM Journal on Imaging Sciences, 6(2):1035–1058, 2013.

[150] A. Neice. Methods and limitations of subwavelength imaging. volume 163 of Advances in Imaging and Electron Physics, pages 117 – 140. Elsevier, 2010.

[151] M. H. Nguyen and F. De la Torre. Robust kernel principal component analysis. In Advances in Neural Information Processing Systems 21, pages 1185–1192. Curran Associates, Inc., 2009.

[152] J. Ni, P. Turaga, V. M. Patel, and R. Chellappa. Example-driven manifold priors for image deconvolution. volume 20, pages 3086–3096, Nov. 2011.

[153] M. Ouimet and Y. Bengio. Greedy spectral embedding. In Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics, pages 253–260. Omni Press Madison, WI, Jan. 2005.

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

[155] T. Pécot and C. Kervrann. Patch-based Markov models for change detection in image sequence analysis. In Proc. Int. Workshop on Local and Non-Local Approximation in Image Processing (LNLA'08), pages 1–6, Lausanne, Switzerland, August 2008.

[156] T. Peleg, Y. C. Eldar, and M. Elad. Exploiting statistical dependencies in sparse representations for signal recovery. IEEE Transactions on Signal Processing, 60(5):2286–2303, 2012.

[157] F. Perez-Cruz and O. Bousquet. Kernel methods and their potential use in signal processing. Signal Processing Magazine, IEEE, 21(3):57–65, May 2004.

[158] G. Peyré. Manifold models for signals and images. Computer Vision and Image Understanding, 113(2):249–260, February 2009.

[159] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. Image Processing, IEEE Transactions on, 12(11):1338–1351, Nov 2003.

[160] I. Ram, M. Elad, and I. Cohen. Image processing using smooth ordering of its patches. IEEE Transactions on Image Processing, 22(7):2764–2774, 2013.

[161] N. S. Rao and F. Porikli. A clustering approach to optimize online dictionary learning. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 1293–1296. IEEE, 2012.

[162] S. Reich. A limit theorem for projections. Linear and Multilinear Algebra, 13(3):281–290, 1983.

[163] L. F. Richardson. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 210(459-470):307–357, 1911.

[164] D. Robinson and P. Milanfar. Fundamental performance limits in image registration. IEEE Transactions on Image Processing, 13(9):1185–1199, Sept 2004.

[165] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental Learning for Robust Visual Tracking. International Journal of Computer Vision, 77(1-3):125–141, May 2008.

[166] S. Roth and M. J. Black. Fields of experts. International Journal of Computer Vision, 82(2):205–229, 2009.

[167] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. Science, 290:2323–2326, 2000.

[168] R. Rubinstein, A. M. Bruckstein, and M. Elad. Dictionaries for sparse representation modeling. Proceedings of the IEEE, 98(6):1045–1057, June 2010.

[169] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. Physica D: Nonlinear Phenomena, 60(1-4):259–268, Nov 1992.

[170] J. Schmidhuber. Deep learning in neural networks: An overview. Neural Networks, 61:85–117, 2015. Published online 2014; based on TR arXiv:1404.7828 [cs.NE].

[171] B. Schölkopf, P. Knirsch, A. Smola, and C. Burges. Fast approximation of support vector kernel expansions, and an interpretation of clustering as approximation in feature spaces. In Mustererkennung 1998 — 20. DAGM-Symposium, Informatik aktuell, Berlin, 1998. Springer.

[172] 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), July 2001.

[173] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput., 10(5):1299–1319, Jul. 1998.

[174] B. Schölkopf and A. J. Smola. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA, USA, 2001.

[175] C. J. Schuler, H. C. Burger, S. Harmeling, and B. Schölkopf. A machine learning approach for non-blind image deconvolution. In IEEE Conference on Computer Vision and Pattern Recognition, pages 1067–1074, June 2013.

[176] H. S. Seung and D. D. Lee. The manifold ways of perception. Science, 290(5500):2268–2269, December 2000.

[177] C. Shan, S. Gong, and P. W. McOwan. Appearance manifold of facial expression. In IEEE ICCV workshop on Human-Computer Interaction (HCI), 2005.

[178] J. Shawe-Taylor, C. K. I. Williams, N. Cristianini, and J. Kandola. On the eigenspectrum of the Gram matrix and the generalization error of kernel-pca. IEEE Transactions on Information Theory, 51(7):2510–2522, 2005.

[179] F. Shi, P.-T. Yap, Y. Fan, J. H. Gilmore, W. Lin, and D. Shen. Construction of multi-region-multi-reference atlases for neonatal brain MRI segmentation. NeuroImage, 51(2):684–693, 2010.

[180] A. Shivanandan, H. Deschout, M. Scarselli, and A. Radenovic. Challenges in quantitative single molecule localization microscopy. FEBS Letters, 588(19):3595 – 3602, 2014. SI: Single molecule techniques - Applications in biology.

[181] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. Summarizing visual data using bidirectional similarity. In IEEE International Conference on Computer Vision and Pattern Recognition, pages 1–8, June 2008.

[182] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger. Shiftable multiscale transforms. Information Theory, IEEE Transactions on, 38(2):587–607, March 1992.

[183] A. Singer, Y. Shkolnisky, and B. Nadler. Diffusion interpretation of nonlocal neighborhood filters for signal denoising. SIAM J. Imaging Sciences, 2(1):118–139, January 2009.

[184] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. Journal of the Optical Society of America, 4(3):519–524, March 1987.

[185] R. Souvenir and R. Pless. Isomap and nonparametric models of image deformation. In WACV/MOTION, pages 195–200. IEEE Computer Society, 2005.

[186] P. Sprechmann, A. M. Bronstein, and G. Sapiro. Learning efficient sparse and low rank models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(9):1821–1833, 2015.

[187] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. Neural Processing Letters, 9(3):293–300, 1999.

[188] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. ArXiv e-prints, December 2013.

[189] K. M. Taylor and F. G. Meyer. A random walk on image patches. SIAM J. Imaging Sciences, 5(2):688–725, 2012.

[190] J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. Neural Comput., 12(6):1247–1283, June 2000.

[191] J. B. Tenenbaum, V. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. Science, 290(5500):2319–2323, 2000.

[192] R. Tibshirani. Regression shrinkage and selection via the lasso: a retrospective. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 73(3):273–282, June 2011.

[193] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In Computer Vision, 1998. Sixth International Conference on, pages 839–846, Jan 1998.

[194] H. J. Trussel and M. R. Civanlar. The Landweber iteration and projection onto convex sets. IEEE Transactions on Acoustics, Speech, and Signal Processing, 33:1632–1634, 1985.

[195] P. Turaga and R. Chellappa. Locally time-invariant models of human activities using trajectories on the Grassmannian. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 2435–2441, 2009.

[196] G. Turk and M. Levoy. Zippered polygon meshes from range images. In Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '94, pages 311–318, New York, NY, USA, 1994. ACM.

[197] M. A. Turk and A. P. Pentland. Eigenfaces for recognition. Journal of Cognitive Neuroscience, 3(1):71–86, 1991.

[198] M. Unser, N. Chenouard, and D. Van De Ville. Steerable pyramids and tight wavelet frames in L2(R(d)). IEEE Transactions on Image Processing, 20(10):2705–2721, 2011.

[199] R. Vidal, Yi Ma, and S. Sastry. Generalized principal component analysis (GPCA). In IEEE Conference on Computer Vision and Pattern Recognition, volume 1, pages I–621–I–628, June 2003.

[200] M. J. Wainwright and E. P. Simoncelli. Scale mixtures of gaussians and the statistics of natural images. In Advances in Neural Information Processing Systems 12, pages 855–861. MIT Press, 2000.

[201] M.B. Wakin and Rice University. The Geometry of Low-dimensional Signal Models. Rice University, 2007.

[202] R. Wang, S. Shan, X. Chen, and W. Gao. Manifold-manifold distance with application to face recognition based on image set. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR, pages 1–8, 2008.

[203] Y.-Q. Wang and J.-M. Morel. A note on size of denoising neural networks. IEEE Transactions on Neural Networks and Learning Systems, 2015.

[204] Z. Wang, Y. Yang, Z. Wang, S. Chang, W. Han, J. Yang, and T. S. Huang. Self-tuned deep super resolution. ArXiv e-prints, abs/1504.05632, April 2015.

[205] Y. Wexler, E. Shechtman, and M. Irani. Space-time completion of video. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 29(3):463–476, 2007.

[206] N. Wiener. Extrapolation, interpolation, and smoothing of stationary time series with engineering applications. M.I.T. paperback series. Cambridge, Mass. Technology Press of the Massachusetts Institute of Technology, 1964. First published during the war as a classified report to Section D2, National Defense Research Committee.

[207] C. K. I. Williams. On a connection between kernel PCA and metric multidimensional scaling. Machine Learning, 46(1-3):11–19, 2002.

[208] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In Advances in NIPS 27, pages 1790–1798. 2014.

[209] M. Xu, H. Chen, and P. K. Varshney. Ziv-Zakai bounds on image registration. IEEE Transactions on Signal Processing, 57(5):1745–1755, May 2009.

[210] M.-H. Yang. Face recognition using extended ISOMAP. In International Conference on Image Processing, volume 2, pages II–117–II–120 vol.2, 2002.

[211] G. Yu, G. Sapiro, and S. Mallat. Image modeling and enhancement via structured sparse model selection. In IEEE International Conference on Image Processing, pages 1641–1644. IEEE, 2010.

[212] G. Yu, G. Sapiro, and S. Mallat. Solving inverse problems with piecewise linear estimators: From Gaussian mixture models to structured sparsity. IEEE Transactions on Image Processing, 21(5):2481–2499, 2012.

[213] H. Yue, X. Sun, J. Yang, and F. Wu. Image denoising by exploring external and internal correlations. IEEE Transactions on Image Processing, 24(6):1967–1982, 2015.

[214] J. Zhou and A. Robles-Kelly. Image inpainting based on local optimisation. In Proceedings of the 2010 20th International Conference on Pattern Recognition, ICPR '10, pages 4440–4443, Washington, DC, USA, 2010. IEEE Computer Society.

[215] S. C. Zhu, Y. Wu, and D. Mumford. Filters, random fields and maximum entropy (FRAME) – towards a unified theory for texture modeling. International Journal on Computer Vision, 27(2):1–20, 1998.

[216] X. Zong, J. Lee, A. J. Poplawsky, S.-G. Kim, and J. C. Ye. Compressed sensing fMRI using gradient-recalled echo and EPI sequences. NeuroImage, 92:312–321, 2014.

[217] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In IEEE Int. Conference on Computer Vision, pages 479–486, Nov 2011.

Appendix A

Derivation of Equations 4.9 and 4.10

Here we show that the coefficients $\boldsymbol{\gamma}_m$ in Eq. 4.9 can be expressed as given by Eq. 4.10.

First, let us define $\mathbf{V}_m = \sqrt{w_m}\mathbf{U}_m = \sqrt{w_m}\mathbf{X}_m\boldsymbol{\alpha}_m$ to simplify the notation. Then, for $k > 0$, $\mathbf{A}^k = \left(\sum_{m=1}^{M} \mathbf{V}_m \mathbf{V}_m^{\mathrm{T}}\right)^k$ can be represented as a product of block vectors and matrices with $i, j = 1, \ldots, M$:

$$\mathbf{A}^k = \sum_{m_1=1}^{M} \cdots \sum_{m_k=1}^{M} \mathbf{V}_{m_1}\mathbf{V}_{m_1}^T\mathbf{V}_{m_2}\mathbf{V}_{m_2}^T \cdots \mathbf{V}_{m_k}\mathbf{V}_{m_k}^T$$

$$= \begin{bmatrix} \cdots & \mathbf{V}_i & \cdots \end{bmatrix} \begin{bmatrix} & \vdots & \\ \cdots & \mathbf{V}_i^T\mathbf{V}_j & \cdots \\ & \vdots & \end{bmatrix}^{k-1} \begin{bmatrix} \vdots \\ \mathbf{V}_j^T \\ \vdots \end{bmatrix}.$$

Using the notation for the matrix $\mathbf{H}$ and the vectors $\mathbf{h}$ and $\mathbf{g}$ introduced in Section 4.3, we can now write $\mathbf{A}^k = \sum_{i=1}^{M}\sum_{j=1}^{M} \mathbf{V}_i \mathbf{H}_{[i,j]}^{k-1} \mathbf{V}_j^T$, and also

$$\mathbf{A}^k\mathbf{z}^{(0)} = \sum_{m=1}^{M} \mathbf{V}_m \mathbf{H}_{[m,:]}^{k-1}\mathbf{h},$$

$$\mathbf{A}^k\mathbf{b} = \sum_{m=1}^{M} \mathbf{V}_m \mathbf{H}_{[m,:]}^{k-1}\mathbf{g},$$

where $\mathbf{H}_{[m,:]}^{k-1} = \left[\mathbf{H}_{[m,1]}^{k-1}, \mathbf{H}_{[m,2]}^{k-1}, \cdots, \mathbf{H}_{[m,M]}^{k-1}\right]$ denotes the m^{th} block row of the matrix $\mathbf{H}^{k-1}$.

Then rewriting Eq. 4.8 for the K^{th} step approximation of the solution, we have

$$\begin{aligned}
\mathbf{z}^{(K)} &= \mathbf{A}^K \mathbf{z}^{(0)} + \sum_{k=1}^{K-1} \mathbf{A}^k \mathbf{b} + \mathbf{b} \\
&= \sum_{m=1}^{M} \mathbf{V}_m \mathbf{H}_{[m,:]}^{K-1} \mathbf{h} + \sum_{k=1}^{K-1} \sum_{m=1}^{M} \mathbf{V}_m \mathbf{H}_{[m,:]}^{k-1} \mathbf{g} + \mathbf{b} \\
&= \sum_{m=1}^{M} \mathbf{V}_m \left\{ \mathbf{H}_{[m,:]}^{K-1} \mathbf{h} + \sum_{k=1}^{K-1} \mathbf{H}_{[m,:]}^{k-1} \mathbf{g} \right\} + \mathbf{b}. \qquad \text{(A.1)}
\end{aligned}$$

Furthermore, we expand the vector $\mathbf{b}$ in terms of training samples as:

$$\begin{aligned}
\mathbf{b} &= \sum_{m=1}^{M} w_m \left(\mathbf{I} - \mathbf{U}_m \mathbf{U}_m^T \right) \mathbf{m}_m \\
&= \sum_{m=1}^{M} w_m \left(\mathbf{I} - \mathbf{X}_m \boldsymbol{\alpha}_m \boldsymbol{\alpha}_m^T \mathbf{X}_m^T \right) \mathbf{X}_m \frac{1}{n_m} \mathbb{1} \\
&= \sum_{m=1}^{M} w_m \mathbf{X}_m \left(\mathbf{I} - \boldsymbol{\alpha}_m \boldsymbol{\alpha}_m^T \mathbf{X}_m^T \mathbf{X}_m \right) \frac{1}{n_m} \mathbb{1} \\
&= \sum_{m=1}^{M} w_m \mathbf{X}_m \boldsymbol{\mu}_m \,,
\end{aligned}$$

Using the notation for the vector $\mathbf{s}$ and the matrix $\mathbf{V}_m$, Eq. A.1 becomes:

$$\begin{aligned}
\mathbf{z}^{(K)} &= \sum_{m=1}^{M} \sqrt{w_m} \mathbf{X}_m \boldsymbol{\alpha}_m \mathbf{s}_{[m]} + \sum_{m=1}^{M} w_m \mathbf{X}_m \boldsymbol{\mu}_m \\
&= \sum_{m=1}^{M} \mathbf{X}_m \boldsymbol{\gamma}_m \,,
\end{aligned}$$

where $\boldsymbol{\gamma}_m = \sqrt{w_m} \boldsymbol{\alpha}_m \mathbf{s}_{[m]} + w_m \boldsymbol{\mu}_m$.

Appendix B

Incremental Eigendecomposition Algorithms

Here we present two versions of the incremental eigendecomposition algorithm of Gram matrices, without and with centering of datasamples; the latter one is from [48]. All operations are performed exclusively in terms of inner products, thus allowing us to use this incremental strategy in the KPCA and our KODA algorithms with the kernel trick.

Algorithm 5 Incremental Eigendecomposition of a Gram Matrix without Centering, iEIG_{unc}

Input: Set of n_1 original samples $\mathbf{X}_1$, r unscaled eigenvectors $\boldsymbol{\alpha}_{1^r}$, set of n_2 new samples $\mathbf{X}_2$.
Output: Update eigenvectors $\boldsymbol{\alpha}_{2^r}$, eigenvaues $\boldsymbol{\Sigma}_{2^r}$.

$\mathbf{K}_{11} \leftarrow \kappa(\mathbf{X}_1, \mathbf{X}_1)$ ▷ Compute the kernel matrices.
$\mathbf{K}_{12} \leftarrow \kappa(\mathbf{X}_1, \mathbf{X}_2)$
$\mathbf{K}_{22} \leftarrow \kappa(\mathbf{X}_2, \mathbf{X}_2)$
$\boldsymbol{\Sigma}_{1^r} \leftarrow \left(\boldsymbol{\alpha}_{1^r}^{\mathrm{T}} \mathbf{K}_{11} \boldsymbol{\alpha}_{1^r}\right)^{1/2}$
$\mathbf{L} \leftarrow \boldsymbol{\Sigma}_{1^r}^{-1} \boldsymbol{\alpha}_{1^r}^{\mathrm{T}} \mathbf{K}_{12}$
$\mathbf{M} \leftarrow \mathbf{K}_{22} - \mathbf{L}^{\mathrm{T}} \mathbf{L}$
$[\mathbf{Q}_{\mathrm{M}}, \boldsymbol{\Delta}_{\mathrm{M}}] \leftarrow \text{EIG}(\mathbf{M})$ ▷ Full eigendecomposition of $\mathbf{M}$.
$\left(\mathbf{U}_F, \boldsymbol{\Sigma}_F, \mathbf{V}_F^{\mathrm{T}}\right) \leftarrow \text{SVD}\left(\begin{bmatrix} \boldsymbol{\Sigma}_{1^r} & \mathbf{L} \\ \mathbb{0}_{r_{\mathrm{M}} \times r} & \boldsymbol{\Delta}_{\mathrm{M}}^{1/2} \mathbf{Q}_{\mathrm{M}}^{\mathrm{T}} \end{bmatrix}\right)$
$\boldsymbol{\Sigma}_{2^r} \leftarrow [\boldsymbol{\Sigma}_F]_{1:r,1:r}$ ▷ Keep first r rows and columns.
$\boldsymbol{\alpha}_{2^r} \leftarrow \begin{bmatrix} \boldsymbol{\alpha}_{1^r} \boldsymbol{\Sigma}_{1^r}^{-1} & -\boldsymbol{\alpha}_{1^r} \boldsymbol{\Sigma}_{1^r}^{-1} \mathbf{L} \mathbf{Q}_{\mathrm{M}} \boldsymbol{\Delta}_{\mathrm{M}}^{-1/2} \\ \mathbb{0}_{n_2 \times r} & \mathbf{Q}_{\mathrm{M}} \boldsymbol{\Delta}_{\mathrm{M}}^{-1/2} \end{bmatrix} [\mathbf{U}_F]_{:,1:r} \, \boldsymbol{\Sigma}_{2^r}$

Algorithm 6 Incremental Eigendecomposition of a Centered Gram Matrix from [48], iEIG_{cnt}

Input: Set of n_1 original samples $\mathbf{X}_1$, r unscaled eigenvectors $\boldsymbol{\alpha}_{1^r}$, expansion coefficients for the mean vector $\boldsymbol{\varepsilon}_1$, set of n_2 new samples $\mathbf{X}_2$.

Output: Update eigenvectors $\boldsymbol{\alpha}_{2^r}$, eigenvaues $\boldsymbol{\Sigma}_{2^r}$, and expansion coefficients for the mean $\boldsymbol{\varepsilon}_2$.

$\mathbf{K}_{11} \leftarrow \kappa\left(\mathbf{X}_1, \mathbf{X}_1\right)$ ▷ Compute the kernel matrices.

$\mathbf{K}_{12} \leftarrow \kappa\left(\mathbf{X}_1, \mathbf{X}_2\right)$

$\mathbf{K}_{22} \leftarrow \kappa\left(\mathbf{X}_2, \mathbf{X}_2\right)$

$\boldsymbol{\Sigma}_{1^r} \leftarrow \left(\boldsymbol{\alpha}_{1^r}^{\mathrm{T}} \mathbf{K}_{11} \boldsymbol{\alpha}_{1^r}\right)^{1/2}$

$\boldsymbol{\gamma} \leftarrow \begin{bmatrix} \mathbb{0}_{n_1 \times n_2} & \sqrt{\frac{n_1 n_2}{n_1+n_2}}\boldsymbol{\varepsilon}_1 \\ \mathbf{I} - \frac{1}{n_2} 1_{n_2} 1_{n_2}^{\mathrm{T}} & -\frac{1}{n_2}\sqrt{\frac{n_1 n_2}{n_1+n_2}} 1_{n_2} \end{bmatrix}$

$\mathbf{L} \leftarrow \boldsymbol{\Sigma}_{1^r}^{-1} \boldsymbol{\alpha}_{1^r}^{\mathrm{T}} \left(\mathbf{I} - \mathbb{1}\boldsymbol{\varepsilon}_1^{\mathrm{T}}\right) \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \end{bmatrix} \boldsymbol{\gamma}$

$\boldsymbol{\eta} \leftarrow \boldsymbol{\gamma} - \begin{bmatrix} \left(\mathbf{I} - \boldsymbol{\varepsilon}_1 \mathbb{1}^{\mathrm{T}}\right) \boldsymbol{\alpha}_{1^r} \boldsymbol{\Sigma}_{1^r}^{-1} \mathbf{L} \\ \mathbb{0}_{n_2 \times (n_2+1)} \end{bmatrix}$

$\mathbf{M} \leftarrow \boldsymbol{\eta}^{\mathrm{T}} \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \boldsymbol{\eta}$

$[\mathbf{Q}_{\mathrm{M}}, \boldsymbol{\Delta}_{\mathrm{M}}] \leftarrow \text{EIG}\left(\mathbf{M}\right)$ ▷ Full eigendecomposition of $\mathbf{M}$.

$\left(\mathbf{U}_F, \boldsymbol{\Sigma}_F, \mathbf{V}_F^{\mathrm{T}}\right) \leftarrow \text{SVD}\left(\begin{bmatrix} \boldsymbol{\Sigma}_{1^r} & \mathbf{L} \\ \mathbb{0}_{r_{\mathrm{M}} \times r} & \boldsymbol{\Delta}_{\mathrm{M}}^{1/2} \mathbf{Q}_{\mathrm{M}}^{\mathrm{T}} \end{bmatrix}\right)$

$\boldsymbol{\Sigma}_{2^r} \leftarrow \left[\boldsymbol{\Sigma}_F\right]_{1:r,1:r}$ ▷ Keep first r rows and columns.

$\boldsymbol{\alpha}_{2^r} \leftarrow \begin{bmatrix} \left(\mathbf{I} - \boldsymbol{\varepsilon}_1 \mathbb{1}^{\mathrm{T}}\right) \boldsymbol{\alpha}_{1^r} \boldsymbol{\Sigma}_{1^r}^{-1} & \boldsymbol{\eta} \mathbf{Q}_{\mathrm{M}} \boldsymbol{\Delta}_{\mathrm{M}}^{-1/2} \\ \mathbb{0}_{n_2 \times r} \end{bmatrix} \left[\mathbf{U}_F\right]_{:,1:r} \boldsymbol{\Sigma}_{2^r}$

$\boldsymbol{\varepsilon}_2 \leftarrow \frac{1}{n_1+n_2} \begin{bmatrix} n_1 \boldsymbol{\varepsilon}_1 & 1_{n_2} \end{bmatrix}^{\mathrm{T}}$ ▷ Update the mean vector coefficients.

Appendix C

Greedy Reduced Set Expansion Algorithm

Here we present a greedy approach to forming a basis and constructing a compressed representation of vectors expanded in a kernel-induced feature space. In the following algorithm we define $\mathcal{J} \subseteq \{1, \ldots, n_{\mathrm{X}}\}$ to be the index set for chosen samples and use this notation in $\mathbf{K}_{\mathcal{J}\mathcal{J}}$ to denote the rows and columns of the total kernel matrix $\mathbf{K} = \kappa(\mathbf{X}, \mathbf{X})$ corresponding to these samples. We also note that for efficiency, the inverse $\mathbf{K}^{inv}_{\mathcal{J}\mathcal{J}} = \mathbf{K}^{-1}_{\mathcal{J}\mathcal{J}}$ is typically computed via rank-one updates after adding a new sample to the expansion on each iteration (lines 7–9 of the algorithm).

Algorithm 7 Greedy RS Expansion from [174]

Input: Set of original expansion vectors $\{\mathbf{x}_i\}_{i=1}^{n_\mathrm{X}}$, vector of expansion coefficients $\boldsymbol{\alpha}$, kernel function κ, and termination conditions (e.g., maximum number of new expansion vectors m_{max} and/or desired representation error ϵ_{max}).

Output: Indexes of support samples for sparse representation $\mathcal{J}$, vector of new expansion coefficients $\widetilde{\boldsymbol{\alpha}}$.

$\mathbf{K} \leftarrow \kappa(\mathbf{X}, \mathbf{X})$ ▷ Compute the kernel matrix.

$\mathcal{J} \leftarrow \emptyset$, $\mathcal{I} \leftarrow \{1, \ldots, n_\mathrm{X}\}$ ▷ Initialization.

$\widehat{\boldsymbol{\alpha}} \leftarrow \emptyset$, $\mathbf{K}_{\mathcal{J},\mathcal{J}}^{inv} \leftarrow 1$

$E \leftarrow \boldsymbol{\alpha}^\mathrm{T}\mathbf{K}\boldsymbol{\alpha}$

while $E \geq \epsilon_{max}$ **and** $|\mathcal{J}| < m_{max}$ **do**

  $j \leftarrow \underset{i \in \mathcal{I}}{\operatorname{argmax}} \left\{ \frac{\widetilde{\boldsymbol{\alpha}}^\mathrm{T}\mathbf{K}_{\mathcal{J},i} - \boldsymbol{\alpha}^\mathrm{T}\mathbf{K}_{:,i}}{\sqrt{E \cdot \mathbf{K}_{i,i}}} \right\}$

  $\mathbf{d} \leftarrow \mathbf{K}_{\mathcal{J},\mathcal{J}}^{inv}\mathbf{K}_{\mathcal{J},j}$

  $\delta \leftarrow \mathbf{K}_{j,j} - \mathbf{d}^\mathrm{T}\mathbf{K}_{\mathcal{J},:}$

  $\mathbf{K}_{\mathcal{J},\mathcal{J}}^{inv} \leftarrow \frac{1}{\delta} \begin{bmatrix} \boldsymbol{\delta}\mathbf{K}_{\mathcal{J}\mathcal{J}}^{inv} + \mathbf{d}\mathbf{d}^\mathrm{T} & \mathbf{d} \\ -\mathbf{d}^\mathrm{T} & 1 \end{bmatrix}$

  $\widetilde{\boldsymbol{\alpha}} \leftarrow \begin{bmatrix} \widetilde{\boldsymbol{\alpha}} + \frac{1}{\delta}\mathbf{d}\left[\mathbf{d}^\mathrm{T}\mathbf{K}_{\mathcal{J},:} - \mathbf{K}_{j,:}\right]\boldsymbol{\alpha} \\ -\frac{1}{\delta}\left[\mathbf{d}^\mathrm{T}\mathbf{K}_{\mathcal{J},:} - \mathbf{K}_{j,:}\right]\boldsymbol{\alpha} \end{bmatrix}$

  $\mathcal{J} \leftarrow \mathcal{J} \cup \{j\}$, $\mathcal{I} \leftarrow \mathcal{I} \setminus \{j\}$

  $E \leftarrow \widetilde{\boldsymbol{\alpha}}^\mathrm{T}\mathbf{K}_{\mathcal{J},\mathcal{J}}\widetilde{\boldsymbol{\alpha}} - 2\widetilde{\boldsymbol{\alpha}}^\mathrm{T}\mathbf{K}_{\mathcal{J},:}\boldsymbol{\alpha} + \boldsymbol{\alpha}^\mathrm{T}\mathbf{K}\boldsymbol{\alpha}$

end while