Title
Nonlocal Variational Methods in Image and Data Processing

Author
Zhu, Wei

Publication Date
2017

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Nonlocal Variational Methods in Image and Data Processing

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Mathematics

by

Wei Zhu

2017

ABSTRACT OF THE DISSERTATION

Nonlocal Variational Methods in Image and Data Processing

by

Wei Zhu
Doctor of Philosophy in Mathematics
University of California, Los Angeles, 2017
Professor Stanley J. Osher, Chair

In this dissertation, two nonlocal variational models for image and data processing are presented: nonlocal total variation (NLTV) for unsupervised hyperspectral image classification, and low dimensional manifold model (LDMM) for general image and data processing problems. Both models utilize the nonlocal patch-based structures in natural images and data, and modern optimization techniques are used to solve the corresponding variational problems. The proposed algorithms achieve state-of-the-art results on various image and data processing problems, in particular unsupervised hyperspectral image classification and image or data interpolation.

First, a graph-based nonlocal total variation method is proposed for unsupervised classification of hyperspectral images (HSI). The variational problem is solved by the primal-dual hybrid gradient (PDHG) algorithm. By squaring the labeling function and using a stable simplex clustering routine, an unsupervised clustering method with random initialization can be implemented. The effectiveness of the proposed algorithm is illustrated on both synthetic and real-world HSI, and numerical results show that the proposed algorithm outperforms other standard unsupervised clustering methods such as spherical K-means, nonnegative matrix factorization (NMF), and the graph-based Merriman-Bence-Osher (MBO) scheme.

Next, we present a novel low dimensional manifold model for general image processing problems. LDMM is based on the fact that the patch manifolds of many natural images have low dimensional structures. Based on this observation, the dimension of the patch

manifold is used as a regularization to recover the image. The key step in LDMM is to solve a Laplace-Beltrami equation over a point cloud, and it is tackled by the point integral method (PIM). The point integral method enforces the sample point constraints correctly and yields better results than the standard graph Laplacian. LDMM can be used for various image processing problems, and it achieves state-of-the-art results for image inpainting from random subsampling.

Lastly, we present an alternative way to solve the Laplace-Beltrami equation in LDMM. Although the point integral method correctly enforces the sample point constraints and achieves excellent results for image inpainting, the resulting linear system is on the patch domain, and hundreds of linear systems need to be solved each iteration. This causes LDMM to be computationally infeasible for large images and high dimensional data. An alternative way is to discretize the Laplace-Beltrami operator with the weighted graph Laplacian (WGL). After such discretization, we only need to solve one symmetric sparse linear system per iteration of manifold update for image inpainting. Moreover, semi-local patches that incorporate coordinate information of the patches are used in the weight update, which leads to a faster convergence of LDMM. Numerical experiments on normal image, hyperspectral image, and high dimensional scientific data interpolation demonstrate the effectiveness of the algorithm.

The dissertation of Wei Zhu is approved.

Lieven Vandenberghe

Luminita Aura Vese

Wotao Yin

Stanley J. Osher, Committee Chair

University of California, Los Angeles

2017

To my family, mentors, friends, and collaborators . . .

who believe in me and give me guidance.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank above all my advisor, Stanley Osher, for his patient guidance and strong support. Stan has been the most important person in my academic life, and he is with out a doubt the best mentor one could ever ask for. He is exceptionally knowledgeable in so many research areas, and always navigates me in the right direction. Stan, thank you for enlightening me for the past few years, and your guidance and insights made possible the research that is presented in this dissertation.

I would also like to thank my other doctoral committee members, Lieven Vandenberghe, Luminita Vese, and Wotao Yin, for their time and valuble feedback. Besides their suggestions and advice in this dissertation, I have also had the pleasure to learn from them through attending their lectures and reading their papers. They demonstrate the ideals of top scientific researchers that I will strive to become. I am also grateful to the instructors of the courses that I have taken during my graduate study at UCLA. The wisdom they have imparted upon me has helped me develop a solid foundation in mathematics for future research.

I express my deep gratitude to two of my great friends and collaborators, Zuoqiang Shi and Da Kuang, who are both extemely brilliant and diligent researchers in their own fields. The valuble insights and advice they provided have been so important in my graduate study and research. I also thank other mentors and collaborators of mine, Andrea Bertozzi, Jocelyn chanussot, Ke Yin, Jing Qin, Wenzhi Liao, and Dominique Zosso. Surrounding myself with hard-working and brilliant researchers like them has helped shape me into a mathematician today.

VITA

2012	B.S. Mathematics, Tsinghua University
2013–2017	Graduate Teaching Assistant, UCLA
2015–2017	Graduate Research Assistant, UCLA
2016 Summer	Undergraduate Research Project Mentor, UCLA REU
2017	(Expected) Ph.D. Mathematics, University of California, Los Angeles

PUBLICATIONS

Unsupervised Classification in Hyperspectral Imagery With Nonlocal Total Variation and Primal-Dual Hybrid Gradient Algorithm by W. Zhu, V. Chayes, A. Tiard, S. Sanchez, D. Dahlberg, A. Bertozzi, S. Osher, D. Zosso, D. Kuang; IEEE Transactions on Geoscience and Remote Sensing, 2017, Volume 55, Issue 5, Page 2786-2798.

Pre-processing and Classification of Hyperspectral Imagery via Selective Inpainting by V. Chayes, K. Miller, R. Bhalerao, J. Luo, W. Zhu, A. Bertozzi, W. Liao, S. Osher; 2017 IEEE International Conference on Acoustics, Speech and Signal Processing.

Weighted Nonlocal Laplacian on Interpolation from Sparse Data by Z. Shi, S. Osher, W. Zhu; Journal of Scientific Computing, 2017, Page 1-14.

Low dimensional manifold model for image processing by S. Osher, Z. Shi, W. Zhu; *accepted in SIAM Journal on Imaging Sciences.*

Low dimensional manifold model in hyperspectral image reconstruction by Z. Shi, W. Zhu, S. Osher; *submitted*.

Low Dimensional Manifold Model with Semi-local Patches by Z. Shi, S. Osher, W. Zhu; *submitted*.

Scientific Data Interpolation with Low Dimensional Manifold Model by W. Zhu, B. Wang, R. Barnard, C. Hauck, F. Jenko, S. Osher; *submitted*.

CHAPTER 1

Introduction

Most image processing problems, whether they be denoising, inpainting, segmentation, or deblurring, are ill-posed in nature since there could be infinitely many solutions from partially observed information. In order to solve these ill-posed problems, some prior knowledge of the given image is required. Usually this prior information is given as the regularization in a variational model. Therefore, an important task in the field of mathematical image processing is to devise a proper regularizer for a wide class of natural images.

Among all variational models in the realm of image processing, total variation (TV) is probably the most pioneering and widely used technique. Total variation is originally introduced in [ROF92] by Rudin, Osher, and Fatemi for image denoising:

$$\min_u E(u) = \|\nabla u\|_{L^1} + \lambda \|u - f\|_2^2,$$

where $\|\nabla u\|_{L^1}$ is the TV semi-norm of the image u, and f is the observed noisy image. The parameter λ can be adjusted to give higher priority to the TV-regularizing term, or the data fidelity term $\|u - f\|_2^2$. It is well known that total variation has the advantage of preserving edges, which is always preferable because edges are significant features in the image, and usually indicate boudaries of objects. Despite its good performance of restoring "cartoon" part of the image, TV based methods fail to achieve satisfactory results when texture, or repetative structures, are present in the image.

Nonlocal patch-based methods are proposed to address the aforementioned problem. These methods were first introduced by Buades, Coll, and Morel [BCM05] as a nonlocal filter for image denoising. They were later formulated in a variational framework by Gilboa and Osher [GO09a]. Patch-based techniques exploiting similarity and redundancy of local patches

have also been extensively studied in [PBC08, FAC09] for general image processing problems. Nonlocal image processing produces much better results than its local counterpart because theoretically any pixel in the image can interact with any other, which better preserves texture and fine details.

Among different patch-based nonlocal methods, the manifold model is attracting more and more attention, along with the development of manifold learning algorithms. The basic assumption in the manifold model is that the patches concentrate around a low dimensional smooth manifold. This assumption is verified in studies in image processing and computer vision [Pey08, Pey09, LPM03]. The fact that the patch set of a natural image samples a smooth low dimensional manifold makes the dimension of the patch manifold a perfect candidate for the regularizer in a variational model.

This dissertation consists of the study and application of various nonlocal variational models in different image processing tasks. It is organized as follows.

In Chapter 2, a graph-based nonlocal total variation (NLTV) method is proposed for unsupervised classification of hyperspectral images (HSI). The variational problem, which involves a NLTV regularizer and a fidelity term, is solved by the primal-dual hybrid gradient (PDHG) algorithm. By squaring the labeling function in the fidelity term and using a stable simplex clustering routine, an unsupervised clustering method with random pixel initialization is implemented. Numerical experiments demonstrate that the proposed algorithm consistenly outperforms other standard unsupervised clustering methods on various datasets. This chapter is based on the work [ZCT17].

In Chapter 3, a low dimensional manifold model (LDMM) is presented for general image processing problems. The key observation for LDMM is that the patch set of a natural image samples an underlying low dimensional patch manifold. As a result, the dimension of the patch manifold is used as a regularizer in variational forms for various image processing problems. The key step in the algorithm involves solving a Laplace-Beltrami equation on an unstructured point cloud, which is solved via the point integral method (PIM). Numerical results show that LDMM achieves state-of-the-art results for various image processing

problems, especially image inpainting from random sampling with a significant number of missing pixels. This chapter is mainly based on [OSZ16].

The goal of Chapter 4 is to present a faster way of solving the Laplace-Beltrami equation on the point cloud so that the algorithm is more feasible for large-scale image and high dimensional data interpolation. The way to do that is to discretize the Laplace-Beltrami operator using the weighted graph Laplacian (WGL). Unlike the traditional graph Laplacian, WGL does not sacrifice the continuity of the interpolating function on the sampled sets, therefore propagating the sampled pixel information corretly into its vicinity. Moreover, semi-local patches incorporating the pixel coordinate information are used so that much fewer iterations of manifold update are required for convergence. Numerical experiments showing the reconstruction of normal images, hyperspectral images, and high dimensional scientific data from subsampling are presented to illustrate the effectiveness of the method. This chapter is based on [SOZ17, SOZ16, SZO16, ZWB].

CHAPTER 2

Nonlocal Total Variation in Unsupervised Hyperspectral Image Classification

2.1 Introduction

Hyperspectral imagery (HSI) is an important domain in the field of remote sensing with numerous applications in agriculture, environmental science, mineralogy, and surveillance [Cha03]. Hyperspectral sensors capture information of intensity of reflection at different wavelengths, from the infrared to ultraviolet. They take measurements 10-30nm apart, and up to 200 layers for a single image. Each pixel has a unique spectral signature, which can be used to differentiate objects that cannot be distinguished based on visible spectra, for example: invisible gas plumes, oil or chemical spills over water, or healthy from unhealthy crops.

The majority of HSI classification methods are either *unmixing* methods or *clustering* methods. Unmixing methods extract the information of the constitutive materials (the *endmembers*) and the abundance map [BPD12, GV14, JQ09, GKP15]. Clustering methods do not extract endmembers; instead, they return the spectral signatures of the centroids of the clusters. Each centroid is the mean of the signatures of all the pixels in a cluster. However, when it is assumed that most of the pixels are dominated mostly by one endmember, i.e. in the absence of partial volume effects [SBB07], which is usually the case for high-resolution HSI, these two types of methods are expected to give similar results [GKP15]. The proposed nonlocal total variation (NLTV) method for HSI classification in this chapter is a clustering method.

Much work has been carried out in the literature in both the unmixing and the clustering categories. HSI unmixing models can be characterized as linear or nonlinear. In a linear unmixing model (LUM), each pixel is approximated by a linear combination of the endmembers. When the linear coefficients are constrained to be nonnegative, it is equivalent to nonnegative matrix factorization (NMF), and good unsupervised classification results have been achieved in [JQ09, GV14, GKP15] using either NMF or hierarchical rank-2 NMF (H2NMF). Despite the simplicity of LUM, the assumption of a linear mixture of materials has been shown to be physically inaccurate in certain situations [DTR14]. Researchers are starting to expand aggressively into the much more complicated nonlinear unmixing realm [HPG14], where nonlinear effects such as atmospheric scattering are explicitly modeled. However, most of the work that has been done for nonlinear unmixing so far is supervised in the sense that prior knowledge of the endmember signatures is required [BPD12]. Discriminative machine learning methods such as support vector machine (SVM) [MB04, CB05, FBC08] and relevance vector machine (RVM) [DE07, Foo08, MZ11] based approaches have also been applied to hyperspectral images, but they are also supervised methods since a training set is needed to learn the classifiers.

On the contrary, graph-based clustering methods implicitly model the nonlinear mixture of the endmembers. This type of method is built upon a weight matrix that encodes the similarity between the pixels, which is typically a sparse matrix constructed using the distances between the spectral signatures. Graph-cut problems for graph segmentation have been well-studied in the literature [SM00, SW97, SB09, BS10]. In 2012, Bertozzi and Flenner proposed a diffuse interface model on graphs with applications to classification of high dimensional data [BF12]. This idea has been combined with the Merriman-Bence-Osher (MBO) scheme [MBO94] and applied to multi-class graph segmentation [GMB14, mul14] and HSI classification [HSB15, MSB14]. The method in [BF12] minimizes a graph version of the Ginzburg-Landau (GL) functional, which consists of the Dirichlet energy of the labeling function and a double-well potential, and uses Nyström extension to speed up the calculation of the eigenvectors for inverting the graph Laplacian. This graph-based method performed well compared to other algorithms in the detection of chemical plumes in hyperspectral video

sequences [HSB15, MSB14]. However, the GL functional is non-convex due to its double-well term, which may cause the algorithm to get stuck in local minima. This issue can be circumvented by running the algorithm multiple times with different initial conditions and hand-picking the best result.

The two methods presented in this chapter are unsupervised graph-based clustering techniques. Instead of minimizing the GL functional, which has been proved to converge to the total variation (TV) semi-norm, these algorithms minimize the NLTV semi-norm of the labeling functions $\|\nabla_w u_l\|_{L^1}$ directly. A detailed explanation of the nonlocal operator ∇_w and the labeling function u_l will be provided in Section 2.2 and Section 2.3. The L^1 regularized convex optimization problem is solved by the primal-dual hybrid gradient (PDHG) algorithm, which avoids the need to invert the graph Laplacian. We also introduce the novel idea of the quadratic model and a stable simplex clustering technique, which ensures that anomalies converge to their own clusters and makes random endmember initialization possible in the proposed algorithm. The direct usage of the NLTV semi-norm makes the proposed clustering methods more accurate than other methods when evaluated quantitatively on HSI with ground-truth labels, and the quadratic model with stable simplex clustering is a completely new addition to the field of HSI classification.

The following of this chapter is organized as follows: in Section 2.2 background is provided on total variation and nonlocal operators. Two NLTV models (linear and quadratic) and a stable simplex clustering method are presented in Section 2.3. Section 2.4 provides a detailed explanation on the application of the PDHG algorithm to solving the convex optimization problems in the linear and quadratic models. Section 2.5 presents the numerical results and a sensitivity analysis on the key model parameters. Section 2.6 presents the conclusions.

2.2 Total Variation and Nonlocal Operators

Total variation (TV) method was introduced by Rudin et al in 1992 [ROF92] and has been applied to various image processing tasks [CEP05]. Its advantage is that one can preserve the edges in the image when minimizing $\|\nabla u\|_{L^1}$ (TV semi-norm). The total variation model

is:

$$\min_u E(u) = \|\nabla u\|_{L^1} + \lambda S(u).$$

The parameter λ can be adjusted to give higher priority to the TV-regularizing term, or the data fidelity term $S(u)$.

Despite its huge success in image processing, the total variation method is still a local method. More specifically, the gradient of a pixel is calculated using its immediate adjacent pixels. It is known that local image processing techniques fail to produce satisfactory results when the image has repetitive structures, or intrinsically related objects in the image are not spatially connected. To address this problem, Buades et al proposed a nonlocal means method based on patch distances for image denoising [BCM05]. Gilboa and Osher [GO09a] later formalized a systematic framework for nonlocal image processing. Nonlocal image processing produces much better results because theoretically any pixel in the image can interact with any other, which better preserves texture and fine details.

In HSI classification, clusters can have elements that are not spatially connected. Thus it is necessary to develop a nonlocal method of gradient calculation. We provide a review of nonlocal operators in the rest of this section. Note that the model is continuous, and the weights are not necessarily symmetric [ZTO15].

Let Ω be a region in $\mathbb{R}^n$, and $u : \Omega \to \mathbb{R}$ be a real function. In the model for HSI classification, Ω is the domain of the pixels, and $u : \Omega \to [0, 1]$ is the labeling function of a cluster. The larger the value of $u(x)$, the more likely that pixel x would be classified in that cluster. The nonlocal derivative is:

$$\frac{\partial u}{\partial y}(x) := \frac{u(y) - u(x)}{d(x, y)}, \quad \text{for all } x, y \in \Omega,$$

where d is a positive distance between x and y. In the context of hyperspectral images, $d(x, y)$ provides a way to measure the similarity between pixels x and y. Smaller $d(x, y)$ implies more resemblance between these two pixels. The nonlocal weight is defined as $w(x, y) = d^{-2}(x, y)$.

The nonlocal gradient $\nabla_w u$ for $u \in L^2(\Omega)$ can be defined as the collection of all partial

derivatives, which is a function from Ω to $L^2(\Omega)$, i.e. $\nabla_w u \in L^2(\Omega, L^2(\Omega))$:

$$\nabla_w u(x)(y) = \frac{\partial u}{\partial y}(x) = \sqrt{w(x,y)}(u(y) - u(x)).$$

The standard L^2 inner products on Hilbert spaces $L^2(\Omega)$ and $L^2(\Omega, L^2(\Omega))$ are used in the definition. More specifically, for $u_1, u_2 \in L^2(\Omega)$ and $v_1, v_2 \in L^2(\Omega, L^2(\Omega))$,

$$\langle u_1, u_2 \rangle := \int_\Omega u_1(x) u_2(x) dx,$$
$$\langle v_1, v_2 \rangle := \int_\Omega \int_\Omega v_1(x)(y) v_2(x)(y) dy dx.$$

The nonlocal divergence div_w is defined as the negative adjoint of the nonlocal gradient:

$$\text{div}_w v(x) := \int_\Omega \sqrt{w(x,y)} v(x)(y) - \sqrt{w(y,x)} v(y)(x) dy.$$

At last, a standard L^1 and L^∞ norm is defined on the space $L^2(\Omega, L^2(\Omega))$:

$$\|v\|_{L^1} := \int_\Omega \|v(x)\|_{L^2} dx = \int_\Omega \left| \int_\Omega |v(x)(y)|^2 dy \right|^{\frac{1}{2}} dx,$$
$$\|v\|_{L^\infty} := \sup_x \|v(x)\|_{L^2}.$$

2.3 Two NLTV Models for Unsupervised HSI classification

In this section, two NLTV models are explained for unsupervised classification of HSI. The linear model runs faster in each iteration, but it requires a more accurate centroid initialization. The quadratic model runs slower in each iteration, but it is more robust with respect to the centroid initialization. Moreover, the quadratic model converges faster if the initialization is not ideal.

2.3.1 Linear Model

We extend the idea from [CP10] to formulate a linear model for classification on HSI. The linear model seeks to minimize:

$$\begin{aligned} E_1(u) &= \|\nabla_w u\|_{L^1} + \langle u, f \rangle \\ &= \sum_{l=1}^k \|\nabla_w u_l\|_{L^1} + \sum_{l=1}^k \int u_l(x) f_l(x) dx, \end{aligned} \tag{2.1}$$

where $u = (u_1, u_2, \ldots, u_k) : \Omega \to \mathbb{K}^k$ is the labeling function, k is the number of clusters, $\mathbb{K}^k = \{(x_1, x_2, \ldots, x_k) | \sum_{i=1}^k x_i = 1, x_i \geq 0\}$ is the unit simplex in $\mathbb{R}^k$, and $\nabla_w u = (\nabla_w u_1, \ldots, \nabla_w u_k)$ such that $\|\nabla_w u\|_{L^1} = \sum_{l=1}^k \|\nabla_w u_l\|_{L^1}$. $f_l(x)$ is the error function defined as $f_l(x) = \frac{\lambda}{2} |g(x) - c_l|_\mu^2$, where $g(x)$ and c_l are the spectral signatures of pixel x and the l-th centroid, which is initially either picked randomly from the HSI or generated by any fast unsupervised centroid extraction algorithm (e.g. H2NMF, K-means.) The distance in the definition of $f_l(x)$ is a linear combination of cosine distance and Euclidean distance:

$$|g(x) - c_l|_\mu = 1 - \frac{\langle g(x), c_l \rangle}{\|g(x)\|_2 \|c_l\|_2} + \mu \|g(x) - c_l\|_2, \quad \mu \geq 0.$$

In HSI processing, the cosine distance is generally used because it is more robust to atmospheric interference and topographical features [ZZW12]. The reason why the Euclidean distance is also used is that sometimes different classes have very similar spectral angles, but vastly different spectral amplitudes (e.g. "dirt" and "road" in the Urban dataset, which is illustrated in Section 2.5.) This is called the linear model since the power of the labeling function u_l in (2.1) is one.

The intuition of the model is as follows: In order to minimize the fidelity term in (2.1), a small $u_l(x)$ is required if $f_l(x)$ is large, while no such requirement is needed if $f_l(x)$ is relatively small. This combined with the fact that $(u_1(x), \ldots, u_l(x))$ lies on a unit simplex implies that $u_l(x)$ would be the largest term if pixel x is mostly similar to the l-th centroid c_l. Meanwhile, the NLTV regularizing term $\sum_{l=1}^k \|\nabla_w u_l\|_{L^1}$ ensures that pixels similar to each other tend to have analogous values of u. Therefore a classification of pixel x can be obtained by choosing the index l that has the largest value $u_l(x)$.

Now we discuss how to discretize (2.1) for numerical implementation.

2.3.1.1 Weight Matrix

Following the idea from [GO09a], the patch distance is defined as:

$$d_\sigma(x, y) = \int_\Omega G_\sigma(t) \left| g(x + t) - g(y + t) \right|^2 dt,$$

where G_σ is a Gaussian of standard deviation σ. To build a sparse weight matrix, we take a patch P_i around every pixel i, and truncate the weight matrix by constructing a k-d tree [FBF77] and searching the m nearest neighbors of P_i. k-d tree is a space-partitioning data structrue that can significantly reduce the time cost of nearest neighbor search [Bro14]. We employ a randomized and approximate version of this algorithm [ML09] implemented in the open source VLFeat package [1]. The weight is binarized by setting all nonzero entries to one. In the experiments, patches of size 3×3 are used, and m is set to 10. Note that unlike RGB image processing, the patch size for HSI does not have to be very large. The reason is that while low dimensional RGB images require spatial context to identify pixels, high dimensional hyperspectral images already encode enough information for each pixel in the spectral dimension. Of course, a larger patch size that is consistent with the spatial resolution of the HSI will still be preferable when significant noise is present.

2.3.1.2 The Labeling Function and the Nonlocal Operators

The labeling function, $u = (u_1, u_2, \ldots, u_k)$, is discretized as a matrix of size $r \times k$, where r is the number of pixels in the hyperspectral image, and $(u_l)_j$ is the l-th labeling function at j-th pixel; $(\nabla_w u_l)_{i,j} = \sqrt{w_{i,j}}((u_l)_j - (u_l)_i)$ is the nonlocal gradient of u_l; $(\mathrm{div}_w v)_i = \sum_j \sqrt{w_{i,j}} v_{i,j} - \sqrt{w_{j,i}} v_{j,i}$ is the divergence of v at i-th pixel; and the discrete L^1 and L^∞ norm of $\nabla_w u_l$ are defined as: $\|\nabla_w u_l\|_{L^1} = \sum_i \left(\sum_j (\nabla_w u_l)_{i,j}^2 \right)^{\frac{1}{2}}$, and $\|\nabla_w u_l\|_{L^\infty} = \max_i \left(\sum_j (\nabla_w u_l)_{i,j}^2 \right)^{\frac{1}{2}}$.

The next issue to address is how to minimize (2.1) efficiently. The convexity of the energy functional E_1 allows us to consider using convex optimization methods. First-order primal-dual algorithms have been successfully used in image processing with L^1 type regularizers [CP10, ZC08, ZWC08, EZC10]. We use the primal-dual hybrid gradient (PDHG) algorithm. The main advantage is that no matrix inversion is involved in the iterations, as opposed to general graph Laplacian methods. The most expensive part of the computation comes from sparse matrix multiplications, which are still inexpensive due to the fact that only $m = 10$ nonzero elements are kept in each row of the nonlocal weight matrix.

[1] `http://www.vlfeat.org`

Algorithm 1 Linear Model

1: Initialization of centroids: Choose $(c_l)_{l=1}^k$ (randomized or generated by unsupervised centroid extraction algorithms).
2: Initialization of parameters: Choose $\tau, \sigma > 0$ satisfying $\sigma\tau\|\nabla_w\|^2 \leq 1$, $\theta = 1$
3: Initial iterate: Set $u^0 \in \mathbb{R}^{r\times k}$ and $p^0 \in \mathbb{R}^{(r\times r)\times k}$ randomly, set $\bar{u}^0 = u^0$, $u_{hard} = threshold(u^0)$
4: **while** not converge **do**
5: Minimize energy E_1 using PDHG algorithm
6: $u_{hard} = threshold(u)$
7: Update $(c_l)_{l=1}^k$
8: **end while**

We then address centroid updates and stopping criteria for the linear model. The concept of centroid updates is not uncommon; in fact, the standard K-means algorithm consists of two steps: first, it assigns each point to a cluster whose mean yields the least within-cluster sum of squares, then it re-calculates the means from the centroids, and terminates when assignments no longer change[Mac03]. Especially for data-based methods, re-calculating the centroid is essential for making the algorithm less sensitive to initial conditions and more likely to find the "true" clusters.

After solving (2.1) using the PDHG algorithm, the output u will be thresholded to u_{hard}. More specifically, for every $i \in \{1, 2, \ldots, r\}$, the largest element among $((u_1)_i, (u_2)_i, \cdots, (u_k)_i)$ is set to 1, while the others are set to 0, and we claim the i-th pixel belongs to that particular cluster. Then the l-th centroid is updated by taking the mean of all the pixels in that cluster. The process is repeated until the difference between two consecutive u_{hard} drops below a certain threshold. The pseudocode for the proposed linear model on HSI is listed in Algorithm 1.

Before ending the discussion of the proposed linear model, we point out its connection to the piecewise constant Mumford-Shah model for multi-class graph segmentation [MS89]. Assume that the domain Ω of the HSI is segmented by a contour Φ into k disjoint regions,

$\Omega = \cup_{l=1}^{k} \Omega_l$. The piecewise constant Mumford-Shah energy is defined as:

$$E_{MS}(\Phi, \{c_l\}_{l=1}^{k}) = |\Phi| + \lambda \sum_{l=1}^{k} \int_{\Omega_l} |g(x) - c_l|^2 \, dx, \tag{2.2}$$

where $|\Phi|$ is the length of the contour. To illustrate the connection between (2.1) and (2.2), consider the "local" version of (2.1), which essentially replaces the NLTV regularizer $\|\nabla_w u_l\|_{L^1}$ with its local counterpart :

$$E_1^{\text{loc}}(u) = \sum_{l=1}^{k} \|\nabla u_l\|_{L^1} + \sum_{l=1}^{k} \int u_l(x) f_l(x) dx. \tag{2.3}$$

Assume that the labeling function u_l is the characteristic function of Ω_l. Then $\int u_l(x) f_l(x) dx$ is equal to $\int_{\Omega_l} |g(x) - c_l|^2 \, dx$ up to a multiplicative constant. Moreover, the total variation of a characteristic function of a region equals the length of its boundary, and hence $|\Phi| = \sum_{l=1}^{k} \|\nabla u_l\|_{L^1}$. So the linear model (2.1) can be viewed as a nonlocal convex-relaxed version of Mumford-Shah model. We also note that the linear energy (2.1) has been studied in [HSB15]. But in their work, the authors used a graph-based MBO method to minimize (2.1) instead of the PDHG algorithm, and the difference of the numerical performances can be seen in Section 2.5.

2.3.2 Quadratic Model

2.3.2.1 Intuition

The aforementioned linear model performs very well when the centroids are initialized by accurate centroid extraction algorithms. As shown in Section 2.5, the linear model can have a significant boost to the accuracy of other algorithms if the centroid extraction algorithm is reasonable, without sacrificing speed. However, if centroids are not extracted accurately, or if random initialization is used, the segmenting results are no longer reliable, and the algorithm takes far more iterations to converge to a stable classification.

To reduce the times of centroid updates and merge similar clusters automatically and simultaneously, the following quadratic model is proposed:

$$E_2(u) = \sum_{l=1}^{k} \|\nabla_w u_l\|_{L^1} + \sum_{l=1}^{k} \int u_l^2(x) f_l(x) dx. \tag{2.4}$$

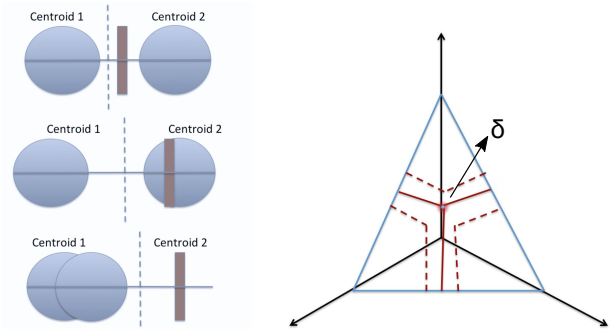


Figure 2.1: The first figure shows the "pushing" mechanism of the quadratic model. The horizontal line represents the unit simplex in $\mathbb{R}^2$. Signatures from cluster A_1 are colored blue, and signatures from cluster A_2 are colored brown. The vertical dashed bar is generated by a stable simplex clustering method, and it thresholds the points on the simplex into two categories.
The second figure shows the stable simplex clustering. Every grid point δ on the simplex generates a simplex clustering. We want to choose a δ such that there are very few data points falling into the "Y-shaped region".

Similar as before, $u = (u_1, u_2, \ldots, u_k) : \Omega \to \mathbb{K}^k$ is the labeling function, k is the number of clusters, $\mathbb{K}^k$ is the unit simplex in $\mathbb{R}^k$, and $f_l(x)$ is the error function.

Note that the only difference between (2.1) and (2.4) is that the power of the labeling function u_l here is two. The intuition for this is as follows:

Consider for simplicity a hyperspectral image with a ground truth of only two clusters, A_1 and A_2. Suppose the randomized initial centroids are chosen such that $c_1 \approx c_2 \in A_1$; or, that the two random initial pixels are of very similar spectral signatures and belong to the same ground truth cluster.

Let x be a pixel from A_2. Then $0 \ll |g(x) - c_1|^2 \approx |g(x) - c_2|^2$. When (2.1) is applied, the fidelity term $\langle u, f \rangle$ does not change when $u(x)$ moves on the simplex in $\mathbb{R}^2$, and thus pixels of A_2 will be scattered randomly on the simplex. After thresholding, an approximately equal number of pixels from cluster A_2 will belong to clusters C_1 and C_2, so the new centroids $\tilde{c}_1$ and $\tilde{c}_2$ that are the means of the spectral signatures of the current clusters will once again be approximately equal.

This situation changes dramatically when (2.4) is minimized:

- Observe that the fidelity term in E_2 is minimized for a pixel $x \in A_2$ when $u_1(x) \approx$

$u_2(x) \approx \frac{1}{2}$. Therefore, the pixels of cluster A_2 will be "pushed" toward the center of the simplex once E_2 is minimized.

- With a stable simplex clustering method (explained in Section 2.3.2.2), the clusters are divided such that all of these pixels in the center belong to either C_1 or C_2; without loss of generality suppose they belong to C_2. Then the updated centroid $\tilde{c}_1$ is essentially c_1, while the updated centroid $\tilde{c}_2$ is a linear combination of the spectral signature of members belonging to A_1 and A_2, and thus quite different from the original c_2.

- After minimizing the energy E_2 again, pixels from A_1 will be clustered in C_1, and pixels from A_2 will be pushed to C_2. Therefore, the clustering will be finished in just two steps in theory. See Figure 2.1 for a graphical illustration.

The quadratic model not only reduces the number of iterations needed to find the "true" clustering because of its capability of anomaly detection, but it allows for random initialization as well, making it a more robust technique.

2.3.2.2 Stable Simplex Clustering

As mentioned above, the quadratic model pushes anomalies into the middle of the unit simplex. Therefore it would be ill-conceived to simply classify the pixels based on the largest component of the labeling function $u(x) = (u_1(x), u_2(x), \ldots, u_k(x))$. Instead, a stable simplex clustering method has to be used.

The concept behind the stable simplex clustering is to choose a division that puts all the data points in the "middle" of the unit simplex into a single cluster. Figure 2.1 demonstrates this in the simple two-cluster case. Also refer to section 2.3.2.1 for explanation of the "pushing" process. The idea to accomplish this goal is inspired by [GKP15]. We first create a grid on a k-dimensional simplex, where k is the number of clusters, and each grid point δ generates a simplex clustering. Then a δ is searched to minimize the energy $g(\delta)$:

$$g(\delta) = -\log(\prod_{l=1}^{k} F_l(\delta)) + \eta \exp(G(\delta)),$$

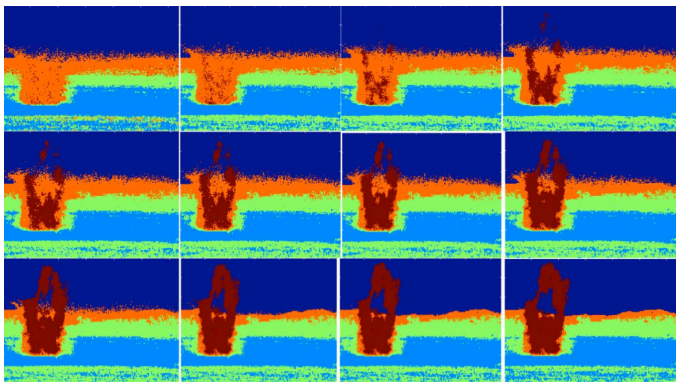


Figure 2.2: Quadratic model and stable simplex clustering on the plume dataset. The chemical plume (brown) is perfectly detected in 12 iterations.

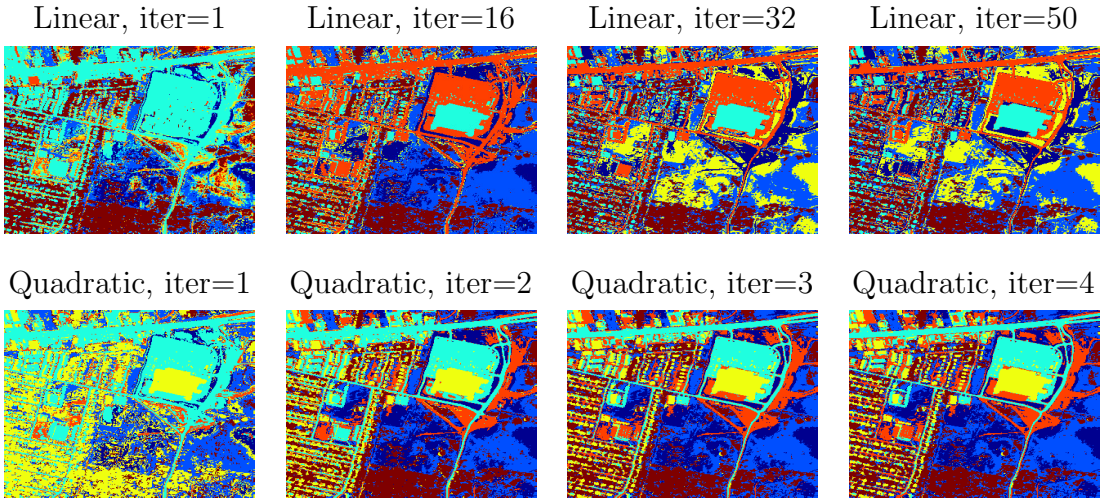


Figure 2.3: Linear vs Quadratic Model on the Urban dataset with the same centroid initialization. To produce essentially identical results, the Linear model (first row) took 50 iterations of centroid updates, and the Quadratic model (second row) took just 4 iterations.

where $F_l(\delta)$ is the percentage of data points in cluster l, and $G(\delta)$ is the percentage of data points on the edges near the division, i.e. the "Y-shaped region" in Figure 2.1. The first term in $g(\delta)$ rewards keeping clusters approximately of the same size, ensuring no skewed data from clusters far too small. And the second term rewards sparsity of points in the intermediate region. The constant η is chosen to be large enough such that stability has a bigger weight in the energy.

Algorithm 2 shows the quadratic model using stable simplex clustering. Figure 2.2

Algorithm 2 Quadratic Model with Stable Simplex Clustering

1: Initialization of centroids: Choose $(c_l)_{l=1}^k$ (randomized or generated by unsupervised centroid extraction algorithms).
2: Initialization of parameters: Choose $\tau, \sigma > 0$ satisfying $\sigma\tau\|\nabla_w\|^2 \leq 1$, $\theta = 1$
3: Initial iterate: Set $u^0 \in \mathbb{R}^{r\times k}$ and $p^0 \in \mathbb{R}^{(r\times r)\times k}$ randomly, set $\bar{u}^0 = u^0$,
4: **while** not converge **do**
5: Minimize energy E_2 using PDHG algorithm
6: $u_{hard} = threshold(u)$ with stable simplex clustering
7: Update $(c_l)_{l=1}^k$
8: **end while**

demonstrates how this detected the chemical plumes in a frame with background centroids pre-calculated and random initialization for the final centroid. Notice that no plume is detected in the first iteration. But by the twelfth iteration, the gas plume is nearly perfectly segmented.

Finally, we present the comparison between the results of the linear model and the quadratic model on the Urban dataset with identical random pixel initialization in Figure 2.3. The linear model took about 50 iterations to converge, and the quadratic model only took 4 iterations.

2.4 Primal-Dual Hybrid Gradient Algorithm

In this section, a detailed explanation is provided on the application of the PDHG algorithm [CP10, ZC08, ZWC08, EZC10] to minimizing E_1 (2.1) and E_2 (2.4) in the previous section. A review of the algorithm is provided in a more general setting to contextualize the extension to nonlocal model for hyperspectral imagery.

Algorithm 3 Primal-Dual Hybrid Gradient (PDHG) Algorithm

1: Initialization: Choose $\tau, \sigma > 0$, $\theta \in [0, 1]$, $(x^0, y^0) \in X \times Y$, and set $\bar{x}^0 = x^0$

2: **while** not converge **do**

3: $\quad y^{n+1} = (I + \sigma \partial F^*)^{-1}(y^n + \sigma K \bar{x}^n)$

4: $\quad x^{n+1} = (I + \tau \partial G)^{-1}(x^n - \tau K^* y^{n+1})$

5: $\quad \bar{x}^{n+1} = x^{n+1} + \theta(x^{n+1} - x^n)$

6: $\quad n = n + 1$

7: **end while**

2.4.1 A Review of PDHG Algorithm

Consider the following convex optimization problem:

$$\min_{x \in X} \{F(Kx) + G(x)\}, \tag{2.5}$$

where X and Y are finite-dimensional real vector spaces, F and G are proper convex lower semi-continuous functions $F : Y \to [0, \infty]$, $G : X \to [0, \infty]$, and $K : X \to Y$ is a continuous linear operator with the operator norm $\|K\| = \sup\{\|Kx\| : x \in X, \|x\| \leq 1\}$. The primal-dual formulation of (2.5) is the saddle-point problem:

$$\min_{x \in X} \max_{y \in Y} \{\langle Kx, y \rangle - F^*(y) + G(x)\}, \tag{2.6}$$

where F^* is the convex conjugate of F defined as $F^*(y) = \sup_x \langle x, y \rangle - F(x)$.

The saddle-point problem (2.6) is then solved using the iterations of Algorithm 3 from [CP10].

In Algorithm 3, $(I + \lambda \partial f)^{-1}(x)$ is the proximal operator of f, which is defined as:

$$(I + \lambda \partial f)^{-1}(x) = \text{prox}_{\lambda f}(x) = \arg\min_y f(y) + \frac{1}{2\lambda}\|y - x\|_2^2.$$

It has been shown in [CP10] that $O(1/N)$ (where N is the number of iterations) convergence can be achieved as long as σ, τ satisfy $\sigma\tau\|K\|^2 \leq 1$.

2.4.2 Primal-Dual Iteraions to Minimize E_1 and E_2

Recall from Section 2.3 that the discretized linear and quadratic energy E_1 and E_2 are:

$$\begin{aligned} E_1(u) &= \sum_{l=1}^{k} \|\nabla_w u_l\|_{L^1} + \sum_{l=1}^{k}\sum_{i=1}^{r} (u_l)_i (f_l)_i, \\ &= \|\nabla_w u\|_{L^1} + \langle u, f\rangle, \\ E_2(u) &= \sum_{l=1}^{k} \|\nabla_w u_l\|_{L^1} + \sum_{l=1}^{k}\sum_{i=1}^{r} (u_l)_i^2 (f_l)_i, \\ &= \|\nabla_w u\|_{L^1} + \langle u, f \odot u\rangle, \end{aligned}$$

where $u = (u_1, u_2, \ldots, u_k)$ is a nonegative matrix of size $r \times k$, with each row of matrix u summing to one, and $f \odot u$ denotes the pointwise product between two matrices f and u. After adding an indicator function δ_U, minimizing E_1 and E_2 are equivalent to solving (2.7) and (2.8):

$$\min_u \|\nabla_w u\|_{L^1} + \langle u, f\rangle + \delta_U(u), \tag{2.7}$$

$$\min_u \|\nabla_w u\|_{L^1} + \langle u, f \odot u\rangle + \delta_U(u) \tag{2.8}$$

where $U = \{u = (u_1, u_2, \ldots, u_k) \in \mathbb{R}^{r\times k} : \sum_{l=1}^{k} (u_l)_i = 1, \forall i = 1, \ldots, r, (u_l)_i \geq 0\}$, and δ_U is the indicator function on U. More specifically:

$$\delta_U(u) = \begin{cases} 0 & \text{if } u \in U, \\ \infty & \text{otherwise.} \end{cases} \tag{2.9}$$

By comparing (2.7), (2.8) and (2.5), we can set $K_1 = K_2 = \nabla_w$, $F_1(q) = F_2(q) = \|q\|_{L^1}$, $G_1(u) = \langle u, f\rangle + \delta_U(u)$, and $G_2(u) = \langle u, f \odot u\rangle + \delta_U(u)$. The convex conjugate of F_1 (and F_2) is $F_1^*(p) = F_2^*(p) = \delta_P(p)$, where the set $P = \{p \in \mathbb{R}^{(r\times r)\times k} : \|p_l\|_\infty \leq 1\}$.

Next, we derive the closed forms of the proximal operators $(I + \sigma\partial F_{1,2}^*)^{-1}$ and $(I + \tau\partial G_{1,2})^{-1}$ so that Algorithm 3 can be implemented efficiently to minimize E_1 and E_2.

$$\begin{aligned} &(I + \sigma\partial F_{1,2}^*)^{-1}(\tilde{p}) = (I + \sigma\partial\delta_P)^{-1}(\tilde{p}) \\ &= \arg\min_p \delta_P(p) + \frac{1}{2\sigma}\|p - \tilde{p}\|_2^2 = \text{proj}_P(\tilde{p}), \end{aligned} \tag{2.10}$$

Algorithm 4 Primal-Dual Iterations for the Linear Model

1: **while** not converge **do**

2: $p^{n+1} = \text{proj}_P(p^n + \sigma \nabla_w \bar{u}^n)$

3: $u^{n+1} = \text{proj}_U(u^n + \tau \text{div}_w p^{n+1} - \tau f)$

4: $\bar{u}^{n+1} = u^{n+1} + \theta(u^{n+1} - u^n)$

5: $n = n + 1$

6: **end while**

where $\text{proj}_P(\tilde{p})$ is the projection of $\tilde{p}$ onto the closed convex set P.

$$
\begin{aligned}
&(I + \tau \partial G_1)^{-1}(\tilde{u}) = \arg\min_u \langle u, f \rangle + \delta_U(u) + \frac{1}{2\tau}\|u - \tilde{u}\|_2^2 \\
&= \arg\min_{u \in U} \|u - \tilde{u} + \tau f\|_2^2 = \text{proj}_U(\tilde{u} - \tau f). \qquad (2.11) \\
&(I + \tau \partial G_2)^{-1}(\tilde{u}) = \arg\min_u \left\langle u, \frac{\tau}{2}\mathcal{A}u \right\rangle + \tau \delta_U(u) + \frac{1}{2}\|u - \tilde{u}\|_2^2 \\
&= \arg\min_{u \in U} \frac{1}{2}\langle u, (I + \tau\mathcal{A})u \rangle - \langle u, \tilde{u} \rangle + \frac{1}{2}\langle \tilde{u}, (I + \tau\mathcal{A})^{-1}\tilde{u} \rangle \\
&= \arg\min_{u \in U} \frac{1}{2}\|(I + \tau\mathcal{A})^{\frac{1}{2}}u - (I + \tau\mathcal{A})^{-\frac{1}{2}}\tilde{u}\|_2^2, \qquad (2.12)
\end{aligned}
$$

where $\mathcal{A} : \mathbb{R}^{r \times k} \to \mathbb{R}^{r \times k}$ is a linear operator defined as $\frac{1}{2}\mathcal{A}u = f \odot u$. Therefore $\mathcal{A}$ is a positive semidefinite diagonal matrix of size $rk \times rk$. It is worth mentioning that the matrix $(I + \tau\mathcal{A})$ is diagonal and positive definite, and hence it is trivial to compute its inverse and square root. Problem (2.12) can be solved as a preconditioned projection onto the unit simplex $\mathbb{K}^k$, and the solution will be explained in Section 2.4.3.

Combining (2.10,2.11,2.12) and Algorithm 3, we have the primal-dual iterations for minimizing E_1 (Algorithm 4) and E_2 (Algorithm 5).

Before moving on to explaining how to solve (2.12), we specify the two orthogonal projections proj_P and proj_U in Algorithm 4: Let $\tilde{p} = \text{proj}_P(p)$, where $p = (p_l)_{l=1}^k \in \mathbb{R}^{(r \times r) \times k}$. Then for every $i \in \{1, 2, \ldots, r\}$ and every $l \in \{1, 2, \ldots, k\}$, the i-th row of $\tilde{p}_l$ is the projection of the i-th row of p_l on to the unit ball in $\mathbb{R}^r$. Similarly, if $\tilde{u} = \text{proj}_U(u)$, then for every $i \in \{1, 2, \ldots, r\}$, $((\tilde{u}_1)_i, (\tilde{u}_2)_i, \ldots, (\tilde{u}_k)_i)$ is the projection of $((u_1)_i, (u_2)_i, \ldots, (u_k)_i)$ onto the unit simplex $\mathbb{K}^k$ in $\mathbb{R}^k$.

Algorithm 5 Primal-Dual Iterations for the Quadratic Model

1: **while** not converge **do**

2: $p^{n+1} = \text{proj}_P(p^n + \sigma \nabla_w \bar{u}^n)$

3: Update u^{n+1} as in (2.12), where $\tilde{u} = u^n + \tau \text{div}_w p^{n+1}$

4: $\bar{u}^{n+1} = u^{n+1} + \theta(u^{n+1} - u^n)$

5: $n = n + 1$

6: **end while**

2.4.3 Preconditioned Projection onto the Unit Simplex

This section is dedicated to solving (2.12). It is easy to see that the rows of u in (2.12) are decoupled, and the only problem that needs to be solved is:

$$\min_{u \in \mathbb{R}^k} \delta_{\mathbb{K}^k}(u) + \frac{1}{2}\|Au - y\|^2, \tag{2.13}$$

where $A = \text{diag}(a_1, a_2, \ldots, a_k)$ is a positive definite diagonal matrix of size $k \times k$, $\mathbb{K}^k$ is the unit simplex in $\mathbb{R}^k$, and $y \in \mathbb{R}^k$ is a given vector.

Theorem 2.4.1. *The solution $u = (u_1, u_2, \ldots, u_k)$ of (2.13) is:*

$$u_i = \max\left(\frac{a_i y_i - \lambda}{a_i^2}, 0\right), \tag{2.14}$$

where λ is the unique number satisfying:

$$\sum_{i=1}^{k} \max\left(\frac{a_i y_i - \lambda}{a_i^2}, 0\right) = 1 \tag{2.15}$$

Proof. Problem (2.13) is equivalent to:

$$\min_{\sum_{i=1}^k u_i = 1} \delta_{\mathbb{R}_+^k(u)} + \frac{1}{2}\|Au - y\|_2^2, \tag{2.16}$$

where $\mathbb{R}_+^k = \{u \in \mathbb{R}^k : u_i \geq 0\}$ is the nonnegative quadrant of $\mathbb{R}^k$. The Lagrangian of (2.16) is:

$$\mathcal{L}(u, \lambda) = \sum_{i=1}^{k} \left(\frac{1}{2}|a_i u_i - y_i|^2 + \delta_{\mathbb{R}_+}(u_i) + \lambda u_i\right) - \lambda.$$

If u^* is a soluton of (2.16), KKT conditions [BV04] imply that there exists a λ such that:

$$u^* = \arg\min_u \mathcal{L}(u, \lambda) = \arg\min_{u_i \geq 0} \sum_{i=1}^{k} \frac{1}{2} a_i^2 \left(u_i + \frac{\lambda - a_i y_i}{a_i^2} \right)^2.$$

Therefore $u_i^* = \max\left(\frac{a_i y_i - \lambda}{a_i^2}, 0\right)$. Meanwhile, the primal feasibility requires:

$$\sum_{i=1}^{k} u_i^* = \sum_{i=1}^{k} \max\left(\frac{a_i y_i - \lambda}{a_i^2}, 0\right) = 1.$$

□

The most computationally expensive part of solving (2.15) is sorting the sequence $(a_i y_i)_{1 \leq i \leq k}$ of length k, which is trivial since k, the number of clusters, is typically a small number.

2.5 Numerical Results

2.5.1 Comparison Methods and Experimental Setup

All experiments were run on a Linux machine with Intel core i5, 3.3Hz with 2GB of DDR3 RAM. The following unsupervised algorithms have been tested:

1. **(Spherical) K-means**: Built in MatLab Code.
2. **NMF**: Non-negative Matrix Factorization [KP11].
3. **H2NMF**: Hierarchical Rank-2 Non-negative Matrix Factorization [GKP15].
4. **MBO**: Graph Merriman-Bence-Osher scheme [MSB14, HSB15]. The code is run for 10 times on each dataset, and the best result is chosen.
5. **NLTV2**: Nonlocal Total Variation, quadratic model with random pixel initialization.
6. **NLTV1(H2NMF/K-means)**: Nonlocal Total Variation, linear model with endmembers/centroids extracted from H2NMF/K-means.

Every algorithm can be initialized via the same procedure as that in "K-means++"[AV07], and the name "Algorithm++" is used if the algorithm is initialized in such a way. For example, "NLTV2++" means nonlocal total variation, quadratic model with "K-means++" initialization procedure.

The algorithms are compared on the following datasets:

1. **Synthetic Dataset:** This dataset[2] contains five endmembers and 162 spectral bands. The 40,000 abundance vectors were generated as a sum of Gaussian fields. The dataset was generated using a Generalized Bilinear Mixing Model (GBM):

$$y = \sum_{i=1}^{p} a_i e_i + \sum_{i=1}^{p-1} \sum_{j=i+1}^{p} \gamma_{ij} a_i a_j e_i \odot e_j + n,$$

   where γ_{ij} are chosen uniformly and randomly in the interval $[0, 1]$, n is the Gaussian noise, with an SNR of 30 dB, and a_i satisfies: $a_i \geq 0$, and $\sum_{i=1}^{p} a_i = 1$.

2. **Salinas-A Dataset:** Salinas-A scene[3] was a small subscene of Salinas image, which was acquired by the AVIRIS sensor over Salinas Valley. It contains 86×83 pixels and 204 bands. The ground truth includes six classes: broccoli, corn, and four types of lettuce.

3. **Urban Dataset:** The Urban dataset[4] is from HYperspectral Digital Imagery Collection Experiment (HYDICE), which has 307×307 pixels and contains 162 clean spectral bands. This dataset only has six classes of material: road, dirt, house, metal, tree, and grass.

4. **San Diego Airport Dataset:** The San Diego Airport (SDA) dataset[5] is provided by the HYDICE sensor. It comprises 400×400 pixels and contains 158 clean spectral

[2]Available at `http://www.math.ucla.edu/~weizhu731/`

[3]Available at `http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral\_Remote\_Sensing\_Scenes`

[4]Available at `http://www.agc.army.mil/.`

[5]Available at `http://www.math.ucla.edu/~weizhu731/`

bands. There are seven types of material: trees, grass, three types of road surfaces, and two types of rooftops [GKP15]. The RGB image with cluster labels are shown in Figure 2.7.

5. **Chemical Plume Dataset:** The chemical plume dataset[6] consists of frames taken from a hyperspectral video of the release of chemical plumes provided by the John Hopkins University Applied Physics Laboratory. The image has 128×320 pixels, with 129 clean spectral bands. There was no ground truth provided for this data, so a segmentation of four classes is assumed: chemical plume, sky, foreground, and mountain. A fifth cluster is added so that the noise pixels would not interfere with the segmentation [HSB15].

6. **Pavia University Dataset:** The Pavia University dataset is collected by the ROSIS sensor. It contains 103 clean spectral bands and 610×340 pixels, and comprises 9 classes of material.

7. **Indian Pines Dataset:** The Indian Pines dataset was acquired by AVIRIS sensor and consists of 145×145 pixels, with 200 clean spectral bands. The available ground truth is labeled into 16 classes.

8. **Kennedy Space Center Dataset:** This dataset was gathered by the NASA AVIRIS sensor over the Kennedy Space Center, Florida. A subscene of the western shore of the center is used in the numerical experiment. 12 classes of different materials are reported in the datacube of size $512 \times 365 \times 176$.

K-means and NMF are non-parametric, and the parameter setups of H2NMF and the MBO scheme are described in [GKP15] and [HSB15, MSB14]. The key parameters λ and μ in the NLTV models are determined in the following way:

1. λ is chosen such that the data fidelity term is around 10 times larger than the NLTV regularizing term $\|\nabla_w u\|_{L^1}$.

[6] Available at `http://www.math.ucla.edu/~weizhu731/`

Table 2.1: Key parameters used for different datasets

Datasets	λ	μ	**Datasets**	λ	μ
Synthetic	10^{-1}	10^{-4}	**Plume**	10^{7}	10^{-2}
Urban	10^{6}	10^{-5}	**Pavia**	10^{6}	10^{-8}
Salinas-A	10^{4}	10^{-4}	**Pines**	10^{6}	10^{-9}
SDA	10^{6}	10^{-7}	**KSC**	10^{6}	10^{-8}

2. μ is chosen such that the Euclidean distances between different endmembers are roughly 10 times smaller than the cosine distances.

Table 2.1 displays the parameters chosen for the numerical experiments. The large variance of the parameter scales results from the variety of image sizes and scales. A sensitivity analysis over the parameters is presented in Section 2.5.7.

2.5.2 Synthetic Dataset and Salinas-A Dataset

All the algorithms are first tested on the synthetic dataset. The classification results are shown in Table 2.2 and Figure 2.4. Both NLTV algorithms have better overall accuracy than all of the other methods, although they took a longer time to converge. The qudratic model classified the image almost perfectly.

The visual classification results and overall accuracies of the Salinas-A dataset are shown in Figure 2.5 and Table 2.2. Both NLTV methods performed at higher accuracy compared to other methods. The linear model improved the result of K-means by incorporating spatial information of the dataset, and the quadratic model only took 4 iterations to converge.

2.5.3 Urban DataSet

There was no ground-truth provided for the Urban HSI. A structured sparse algorithm [ZWX14] (which is different from all of the testing algorithms) has been used to initialize a

Table 2.2: Comparison Of Numerical Results on the Synthetic and Salinas-A Datasets

Algorithm	Synthetic		Salinas-A	
	Run-Time	**Accuracy**	**Run-Time**	**Accuracy**
K-means++	2s	90.98%	0.9s	79.92%
NMF++	9s	80.99%	1.0s	64.47%
H2NMF	2s	72.02%	1.5s	70.08%
MBO++	21s	84.49%	7.8s	68.62%
NLTV2++	29s	99.93%	1.6s	83.69%
NLTV1(K-means)	29s	95.96%	3.4s	83.75%

Table 2.3: Comparison of Numerical Results on the Urban Dataset

Algorithm	Run-Time	Accuracy
K-means	7s	75.20%
NMF	87s	55.70%
H2NMF	7s	85.96%
MBO	92s	78.86%
NLTV2	96s	92.14%
NLTV1(H2NMF)	17s	91.56%

ground truth, which is then corrected pixel by pixel to provide a framework for numerical analysis of accuracy. As this "ground truth" was hand-corrected, it does not necessarily represent the most accurate segmentation of the image; however, it provides a basis for quantitative comparison.

After running all the algorithms that are compared to create six clusters, we noticed

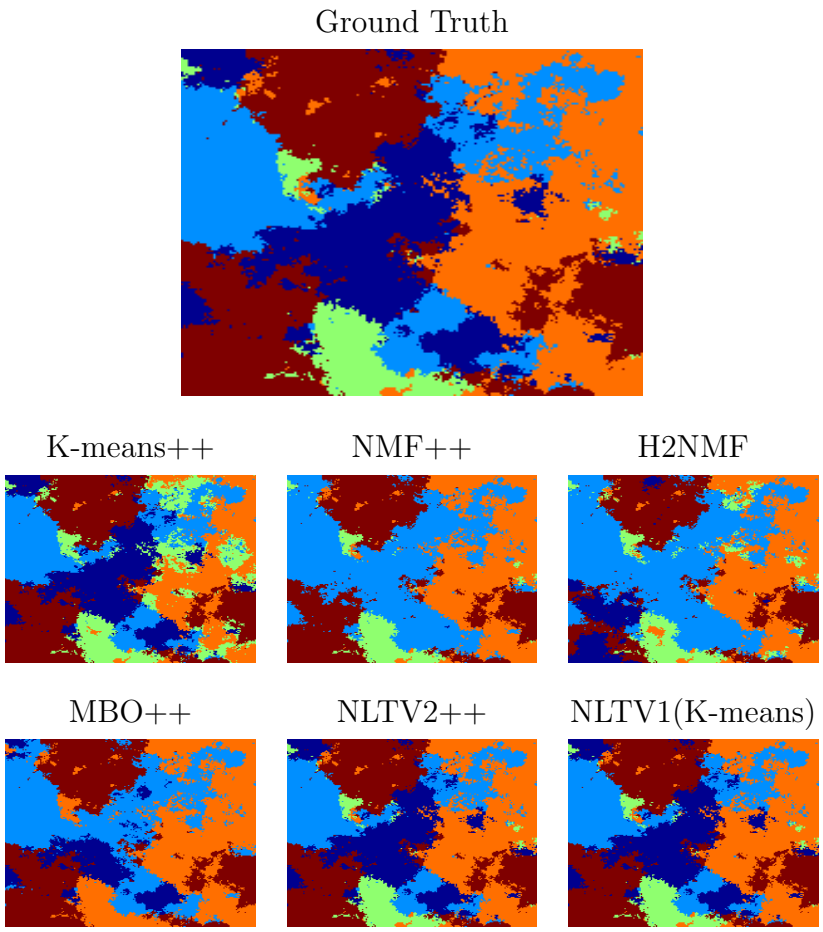


Figure 2.4: Clustering results for the synthetic dataset generated by 5 endmembers. The first image on the left is the ground truth, and the remaining six images are the clustering results of the corresponding algorithms.

that they all split "grass" into two different clusters (one of them corresponds to a mixture of grass and dirt), while treating "road" and "metal" as the same. To obtain a reliable overall accuracy of the classification results, the two "grass" clusters are combined in every algorithm, hence obtaining the classification results for 5 clusters, which are "grass", "dirt", "road+metal", "roof", and "tree".

The overall classification accuracies and run-times are displayed in Table 2.3. As can be seen, the proposed NLTV algorithms performed consistently better with comparable run-time. It is easier to see visually in Figure 2.6 that the NLTV algorithm performed best of the five algorithms tested; specifically, the NLTV algorithm alone distinguished all of the

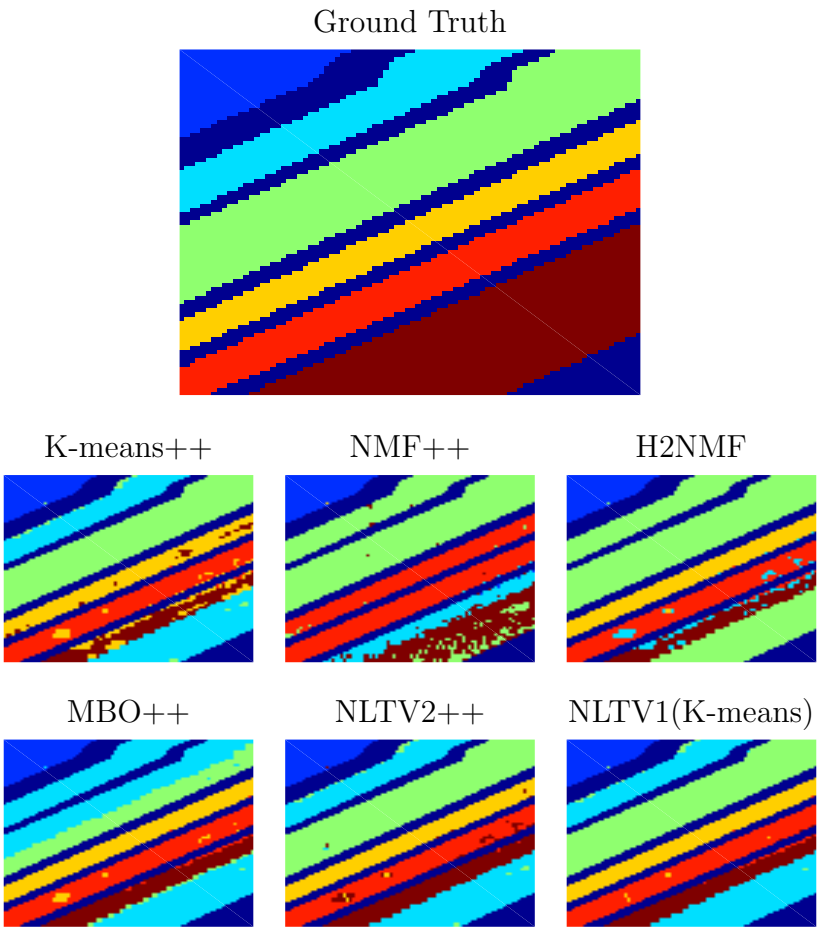


Figure 2.5: Clustering results for the Salina-A dataset. The first image on the left is the ground truth, and the remaining six images are the clustering results of the corresponding algorithms.

dirt beneath the parking lot and the intricacies of the road around the parking lot. The total variation regularizer also gives the segmented image smoother and more distinct edges, allowing easier human identification of the clusters.

2.5.4 San Diego Airport Dataset

The classification results and computational run-times are shown in Figure 2.7 and Table 2.4. No ground truth classification is available for this HSI, but after examining the spectral signatures of various pixels in the scene, we managed to pinpoint some errors that were common for each algorithm. We will not go into detail about the NMF and H2NMF algorithms, which clearly do not perform well on this dataset. K-means obtained some decent results,

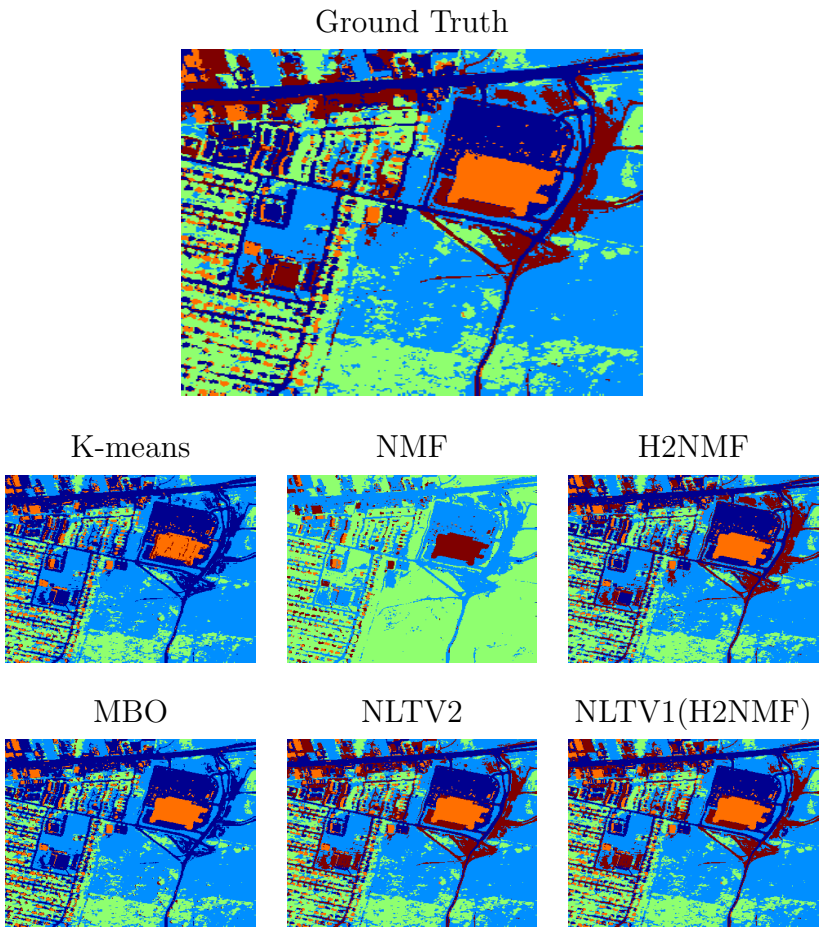


Figure 2.6: Clustering results for the Urban dataset. Five clusters including rooftops, grass, trees, dirt, and "road+metal" are generated by the algorithms.

but splitted the rooftops of the four buildings on the bottom right of the image into two distinct clusters, and failed to separate two different road types (cluster 5 and 6). The MBO scheme failed on two accounts: it did not properly segment two different road surfaces (cluster 6 and 7), and did not account for the different rooftop types (cluster 3 and 4). The linear NLTV model with H2NMF initialization is significantly more accurate than H2NMF and MBO. It successfully picked out two different types of roof (cluster 3 and 4), two different types of road (cluster 6 and 7), although the other type of road (cluster 5) is mixed with one type of roof (cluster 3). The best result was obtained by using the NLTV quadratic model with random initialization, with the only problem that tree and grass (clusters 1 and 2) are mixed together. However, the mixing of grass and tree is actually the case for all the other

Table 2.4: Run-Times for the San Diego Airport (SDA), Chemical Plume (Plume), Pavia University (Pavia), Indian Pines (Pines), and Kennedy Space Center (KSC) Datasets

Algorithm	SDA	Plume	Pavia	Pines	KSC
K-means	9s	2s	26s	10s	47s
NMF	4s	2s	120s	19s	135s
H2NMF	13s	2s	12s	4s	24s
MBO	329s	18s	1020s	198s	754s
NLTV2	43s	23s	299s	64s	561s
NLTV1(H2NMF)	17s	18s	132s	21s	188s

algorithms. This means that NLTV quadratic model alone was able to identify six of the seven clusters correctly.

2.5.5 Chemical Plume Dataset

Analyzing images for chemical plumes is more difficult because of its diffusive nature. All the algorithms are run on the image before it was denoised and the results are shown in Figure 2.8. The unmixing methods such as NMF and H2NMF do not perform satisfactorily on this dataset. MBO++, K-means++, and NLTV2++ can all properly identify the chemical plume. Note that NLTV with H2NMF as centroid initialization outperforms H2NMF as a classification method. We have to point out that the NLTV quadratic model is not so robust with respect to the centroid initialization even with a "K-means++" type procedure on this dataset. But this is also the case for all the other testing algorithms. The MBO scheme, which was specifically designed for this dataset [HSB15], does seem to have the highest robustness among all the algorithms.

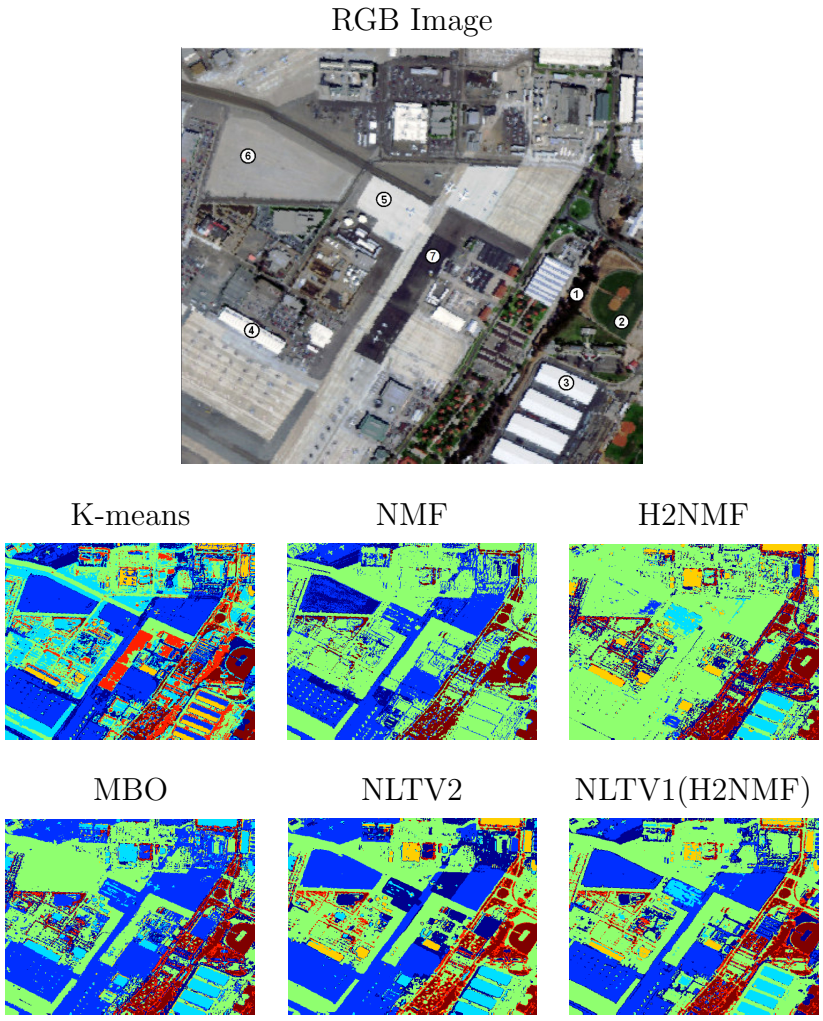


Figure 2.7: Clustering results for the San Diego Airport dataset. The first image on the left is the RGB image, and the remaining six images are the clustering results of the corresponding algorithms.

2.5.6 Pavia University, Indian Pines, and Kennedy Space Center Dataset

The Pavia University (9 clusters), Indian Pines (16 clusters), and Kennedy Space Center (12 clusters) datasets are frequently used to test supervised classification algorithms. To save space, we only report the numerical overall accuracies in Table 2.5. As can be seen, all the competing unsupervised algorithms performed poorly on these three datasets. Different clusters were merged and same clusters were splitted in various fashions by all the algorithms, which rendered the numerical accuracies no longer reliable.

The computational run-times of these three datasets are listed in Table 2.4. Unfor-

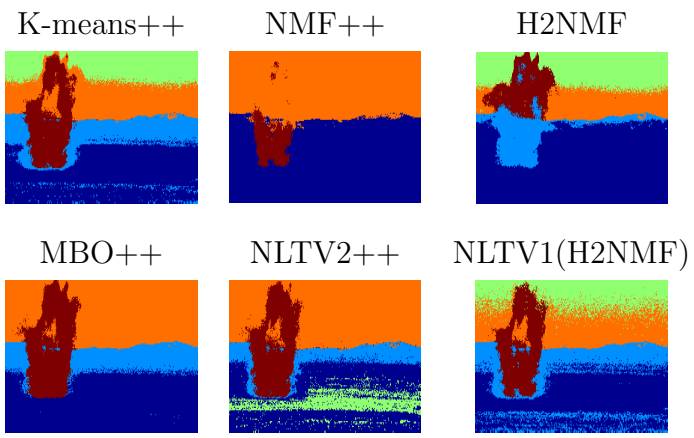


Figure 2.8: Clustering results for the Chemical Plume dataset.

Table 2.5: Comparison of Overall Accuracies on the Pavia University, Indian Pines, and Kennedy Space Center Datasets

Algorithm	**Pavia**	**Pines**	**KSC**
K-means++	42.31%	38.99%	41.73%
NMF++	54.97%	38.84%	37.07%
H2NMF	43.75%	36.78%	37.07%
MBO++	50.04%	36.49%	41.85%
NLTV, H2NMF init	42.83%	36.22%	41.41%
NLTV++	44.01%	42.35%	41.48%

tunately, when the number of clusters is increasing, the computational complexity of the quadratic model grows exponentially. The reason is that the number of grid points (δ in Figure 2.1) on the unit simplex grows exponentially as the dimension of the simplex increases. Therefore, when the number of clusters is large enough (greater than 10), the stable simplex clustering will become the most time-consuming part of the quadratic model. On these three datasets, we sacrificed the accuracy of the quadratic model by creating a coarser mesh on the unit simplex.

The reason why NLTV, as well as all the other competing unsupervised algorithms, performed poorly on these three datasets is two-fold. First, when the number of classes is too large in a HSI covering a large geographic location, the variation of spectral signatures within the same class cannot be neglected when compared to the difference between the constitutive materials, especially when the endmembers themselves are similar. As a result, the unsupervised algorithms tend to split a ground-truth cluster with large variation in spectral signatures and merge clusters with similar centroids or endmembers. Second, there might exist more distinct materials in the image than reported in the ground truth. Therefore the algorithms might detect those unreported materials because no labeling has been used in these unsupervised algorithms. Thus we can conclude that NLTV, as well as other unsupervised methods reported in this chapter, is not suitable for such images at current stage. Modifying the NLTV algorithm to work for such datasets would be the direction of future work.

2.5.7 Sensitivity Analysis over Key Model Parameters

Finally, a sensitivity analysis is provided over the parameters λ and μ in the NLTV models. As mentioned in Section 2.5.1, λ and μ are chosen to balance the scale of the regularizing and fidelity terms or the cosine and Euclidean distances. Figure 2.9 displays the robustness of the NLTV algorithm on the Synthetic, Urban, and Salinas-A datasets with respect to λ and μ within the variance of two magnitudes. Centroid initialization remains identical as λ and μ are changing. It is clear that the NLTV algorithm is fairly robust with respect to λ on all three datasets. The algorithm is also relatively robust with respect to μ on the Synthetic and Salinas-A datasets. As for the Urban dataset, a significant decay in accuracy can be observed as μ increases. This phenomenum is due to the fact that larger μ causes Euclidean distance to be the dominant one, which is not ideal with the presence of atmospheric interference in the Urban dataset. Smaller μ also leads to lower accuracy in the Urban dataset, which results from the similarity of "road" and "dirt" clusters measured in cosine distance. Overall, a reasonable robustness with respect to the key parameters λ and μ can be concluded on these three tests.

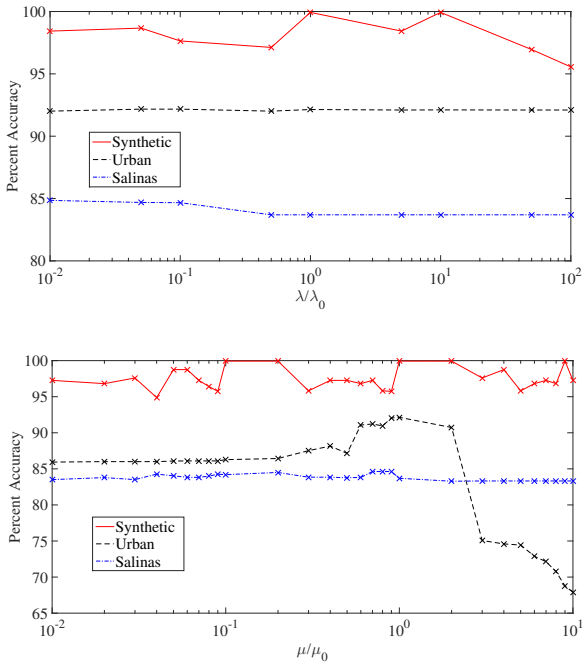


Figure 2.9: This figure shows the robustness of the NLTV algorithm with respect to λ and μ. Centroid initialization remains identical as λ and μ are changing. λ_0 and μ_0 are the optimal values specified in Section 2.5.1. The overall accuracies of the Synthetic, Urban, and Salinas-A datasets are displayed.

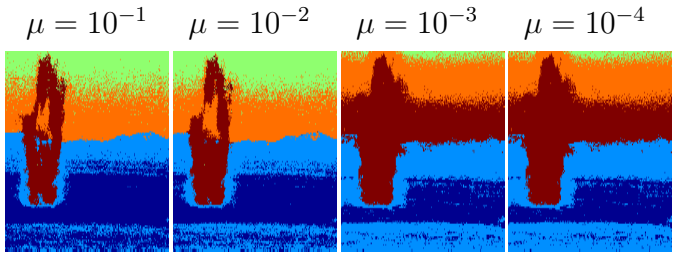


Figure 2.10: The sensitivity of the NLTV algorithm with respect to μ in the plume dataset. All the tests used the same centroid initialization (H2NMF).

Similar robustness can be observed on other datasets except for the Chemical Plume. Figure 2.10 shows the sensitivity of the result with respect to μ. All the centroids are initialized using H2NMF, and vastly different results occurred as μ changes. This could be due to the presence of significant noise.

2.6 Conclusion

In this chapter we present the framework for a nonlocal total variation method for unsupervised HSI classification, which is solved with the primal-dual hybrid gradient algorithm. A linear and a quadratic version of this model are developed; the linear version updates more quickly and can refine results produced by a centroid extraction algorithm, and the quadratic model with stable simplex clustering method provides a robust means of classifying HSI with randomized pixel initialization.

The algorithm is tested on both a synthetic and seven real-world datasets, with promising results. The proposed NLTV algorithm consistently performed with highest accuracy on synthetic and urbanized datasets such as Urban, Salinas-A, and the San Diego Airport, both producing smoother results with easier visual identification of segmentation, and distinguishing classes of material that other algorithms failed to differentiate. The NLTV algorithm also performed well on anomaly detection scenarios like the Chemical Plume datasets; with proper initialization, it performed on par with the Merriman-Bence-Osher scheme developed specifically for this dataset. However, NLTV, as well as other unsupervised algorithms, failed to achieve satisfactory results on datasets with a relatively large number of clusters. The run-times of the NLTV algorithms are generally comparable to the other methods, and the consistent higher accuracy on different types of datasets suggests that this technique is a more robust and precise means of classifying hyperspectral images with a moderate number of clusters.

CHAPTER 3

Low Dimensional Manifold Model in Image Processing

3.1 Introduction

Many image processing problems can be formalized as the recovery of an image $f \in \mathbb{R}^{m\times n}$ from a set of noisy linear measurements

$$y = \Phi f + \varepsilon \tag{3.1}$$

where ε is the noise, and the operator, Φ, typically accounts for some degradation to the image, for instance, blurring, missing pixels or downsampling, so that the measured data y only captures partial information of the original image f. It is an ill-posed problem to recover the original image from partial information. In order to solve this ill-posed problem, one needs to have some prior knowledge of f. Usually, this prior information comes in the form of different regularizations. With the help of regularizations, many image processing problems are formulated as variational problems.

One of the most widely used regularizations is the total variation mentioned in the previous chapter. In the TV model, the following optimization problem is solved:

$$\min_f \|f\|_{TV} + \frac{\mu}{2}\|y - \Phi f\|_{L^2}^2, \tag{3.2}$$

where $\|f\|_{TV} = \int |\nabla f(x)|dx$. It is well known that TV based model can restore the "cartoon" part of the image very well, while the performance on the texture part of the image is not ideal.

The nonlocal methods are another class of techniques widely used in image processing. In nonlocal methods, the local derivatives are replaced by their nonlocal counterpart explained

in Chapter 2:

$$\nabla_w u(x,y) = \sqrt{w(x,y)}(u(x) - u(y)) \tag{3.3}$$

where w is a weight function defined as

$$w(x,y) = \exp\left(-\int G(s)|u(x+s) - u(y+s)|^2 ds\right), \tag{3.4}$$

where G is a Gaussian. Using the nonlocal derivatives, the nonlocal total variation model is given as following

$$\min_f \|\nabla_w f\|_{L^1} + \frac{\mu}{2}\|y - \Phi f\|_{L^2}^2, \tag{3.5}$$

where

$$\|\nabla_w(f)\|_{L^1} = \sum_x \left(\sum_y w(x,y)(f(x) - f(y))^2\right)^{1/2}. \tag{3.6}$$

As mentioned in Chapter 2, the nonlocal model recovers textures very well.

In this chapter, inspired by the nonlocal method and the manifold model of image [Pey09], we present a low dimensional manifold model (LDMM) for image processing. Consider an image $f \in \mathbb{R}^{m\times n}$. For any pixel (i,j), where $1 \le i \le m, 1 \le j \le n$, let this pixel constitute the left-top pixel in an $s_1 \times s_2$ patch and denote this patch as p_{ij}. Let $\mathcal{P}(f)$ denote the collection of all such patches, i.e.,

$$\mathcal{P}(f) = \{p_{i,j} : (i,j) \in \Theta \subset \{1,2,\cdots,m\} \times \{1,2,\cdots,n\}\}. \tag{3.7}$$

where Θ is an index set such that the union of the patch set $\mathcal{P}(f)$ covers the whole image. There are many ways to choose Θ. For example, we can choose $\Theta = \{1,2,\cdots,m\} \times \{1,2,\cdots,n\}$ or $\Theta = \{1, s_1+1, 2s_1+1, \cdots, m\} \times \{1, s_2+1, 2s_2+1, \cdots, n\}$. This freedom may be used to accelerate the computation in LDMM.

Notice that the patch set $\mathcal{P}(f)$ can be seen as a point set in $\mathbb{R}^d$ with $d = s_1 s_2$. The basic assumption in LDMM is that $\mathcal{P}(f)$ samples a low-dimensional smooth manifold $\mathcal{M}(f)$ embedded in $\mathbb{R}^d$, which is called the patch manifold of f. It was revealed that for many classes of images, this assumption holds true [Pey08, Pey09]. Based on this assumption, one

natural regularization involves the dimension of the patch manifold. We want to recover the original image such that the dimension of its patch manifold is as small as possible. This idea formally gives the following optimization problem:

$$\min_f \dim(\mathcal{M}(f)) + \lambda\|y - \Phi f\|_2^2. \tag{3.8}$$

For a natural image, the patch manifold usually is not a single smooth manifold. It may be a set of several manifolds with different dimensions, corresponding to different patterns of the image. In this case, the dimension of the patch manifold, $\dim(\mathcal{M}(f))$, becomes a function and we use the integration of $\dim(\mathcal{M}(f))$ over $\mathcal{M}$ as the regularization,

$$\min_f \int_{\mathcal{M}} \dim(\mathcal{M}(f))(\boldsymbol{x})d\boldsymbol{x} + \lambda\|y - \Phi f\|_2^2, \tag{3.9}$$

where $\dim(\mathcal{M}(f))(\boldsymbol{x})$ is the dimension of the patch manifold of f at $\boldsymbol{x}$. Here $\boldsymbol{x}$ is a point in $\mathcal{M}(f)$ which is also a patch of f.

The remaining problem is how to compute $\dim(\mathcal{M}(f))(\boldsymbol{x})$ for a given image f at given patch $\mathcal{P}f(\boldsymbol{x})$. Fortunately, using some basic tools in differential geometry, we find that the dimension of a smooth manifold embedded in $\mathbb{R}^d$ can be calculated by a simple formula

$$\dim(\mathcal{M})(\boldsymbol{x}) = \sum_{j=1}^{d} |\nabla_{\mathcal{M}}\alpha_i(\boldsymbol{x})|^2$$

where α_i is the coordinate function. That is, for any $\boldsymbol{x} = (x_1, \cdots, x_d) \in \mathcal{M} \subset \mathbb{R}^d$,

$$\alpha_i(\boldsymbol{x}) = x_i. \tag{3.10}$$

Using this formula, the optimization problem (3.8) can be reformulated as

$$\min_{\substack{f\in\mathbb{R}^{m\times n},\\ \mathcal{M}\subset\mathbb{R}^d}} \quad \sum_{i=1}^{d} \|\nabla_{\mathcal{M}}\alpha_i\|_{L^2(\mathcal{M})}^2 + \lambda\|y - \Phi f\|_2^2, \qquad \text{subject to:} \quad \mathcal{P}(f) \subset \mathcal{M}. \tag{3.11}$$

where

$$\|\nabla_{\mathcal{M}}\alpha_i\|_{L^2(\mathcal{M})} = \left(\int_{\mathcal{M}} \|\nabla_{\mathcal{M}}\alpha_i(\boldsymbol{x})\|^2 d\boldsymbol{x}\right)^{1/2}. \tag{3.12}$$

The optimization problem (3.11) is solved by an alternating direction iteration. First, we fix the manifold $\mathcal{M}$, and update the image f. Then the image is fixed and we update the

manifold. This process is repeated until convergence. In this two-step iteration, the second step is relatively easy. It is done by directly applying the patch operator on the image f.

The first step is more difficult. To update the image, we need to solve the following Laplace-Beltrami equations over the manifold.

$$\begin{cases} -\Delta_{\mathcal{M}} u(\boldsymbol{x}) + \mu \sum_{\boldsymbol{y}\in\Omega} \delta(\boldsymbol{x}-\boldsymbol{y})(u(\boldsymbol{y}) - v(\boldsymbol{y})) = 0, & \boldsymbol{x} \in \mathcal{M} \\ \frac{\partial u}{\partial \mathbf{n}}(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \partial\mathcal{M}. \end{cases} \tag{3.13}$$

where $\mathcal{M}$ is a manifold, $\partial\mathcal{M}$ is the boundary of $\mathcal{M}$, $\mathbf{n}$ is the out normal of $\partial\mathcal{M}$. δ is the Dirac-δ function in $\mathcal{M}$, and v is a given function in $\mathcal{M}$.

The explicit form of the manifold $\mathcal{M}$ is not known. We only know a set of unstructured points which samples the manifold $\mathcal{M}$ in high dimensional Euclidean space. It is not easy to solve this Laplace-Beltrami equation over this unstructured high dimensional point set. A graph Laplacian is usually used to approximate the Laplace-Beltrami operator on point cloud. However, from the point of view of numerical PDE, the graph Laplacian was found to be an inconsistent method due to the lack of boundary correction. This is also confirmed by our numerical simulations, e.g. (Figure 3.2(d)).

Instead, in this chapter, we use the point integral method (PIM) [LSS, SSa, SSb] to solve the Laplace-Beltrami equation over the point cloud. In PIM, the discretized linear system of the Laplace-Beltrami equation (3.13) is given as:

$$\frac{|\mathcal{M}|}{N} \sum_{j=1}^{N} R_t(\boldsymbol{x}_i, \boldsymbol{x}_j)(u_i - u_j) + \mu t \sum_{j=1}^{N} \bar{R}_t(\boldsymbol{x}_i, \boldsymbol{x}_j)(u_j - v_j) = 0 \tag{3.14}$$

where $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$ samples $\mathcal{M}$, $v_j = v(\boldsymbol{x}_j)$ and $|\mathcal{M}|$ is the volume of the manifold $\mathcal{M}$. R_t and $\bar{R}_t$ are kernel functions to be explained in Section 3.4.1.

Compared with the graph Laplacian which is widely used in the machine learning and nonlocal methods, a boundary term is added in PIM based on an integral approximation of the Laplace-Beltrami operator. With the help of this boundary term, the values in the retained pixels correctly spread to the missing pixels which gives much better recovery (Figure 3.2(c)).

The rest of the chapter is organized as follows. The patch manifold is analyzed in Section

2. Several examples are given to show that it usually has a low dimensional structure. In Section 3, we introduce the low dimensional manifold model. The numerical methods including the point integral method are discussed in Section 4. In Section 5, we compare the performance of LDMM and classical nonlocal methods and carefully explain the differences between the two methods. Numerical results are shown in Section 6. Concluding remarks are made in Section 7.

3.2 Patch manifold

In this section, we analyze the patch manifold and show several examples. We consider a discrete image $f \in \mathbb{R}^{m\times n}$. For any $(i,j) \in \{1,2,\dots,m\} \times \{1,2,\dots,n\}$, we define a patch $p_{ij}(f)$ as a 2D patch of size $s_1 \times s_2$ of the original image f, and the pixel (i,j) is the top-left corner of the rectangle of size $s_1 \times s_2$. The patch set $\mathcal{P}(f)$ is defined as the collection of all patches:

$$\mathcal{P}(f) = \{p_{ij}(f) : (i,j) \in \{1,2,\dots,m\} \times \{1,2,\dots,n\}\} \subset \mathbb{R}^d, \quad d = s_1 \times s_2. \tag{3.15}$$

The patch set $\mathcal{P}(f)$ has a trivial 2D parameterization which is given as $(i,j) \mapsto p_{ij}(f)$. In this sense, the patch set is locally a 2D sub-manifold embedded in $\mathbb{R}^d$. However, this parameterization is globally not injective and typically leads to high curvature variations and self-intersections in real applications.

For a given image f, the patch set $\mathcal{P}(f)$ gives a point cloud in $\mathbb{R}^d$. Many studies reveal that this point cloud is usually close to a smooth manifold $\mathcal{M}(f)$ embedded in $\mathbb{R}^d$. This underlying smooth manifold is called the *patch manifold* associated with f, denoted as $\mathcal{M}(f)$. Figure 3.1 gives a diagram shows the relation between patch set, trival parameterization and the patch manifold.

The patch set and patch manifold have been well studied in the literature. Lee et al. studied the patch set for discretized images with 3×3 patches [LPM03]. Their results were refined later by Carlsson et al. in [CIS08] where they perform a simplicial approximation of the manifold. Peyré studied several models of local image manifolds for which an explicit

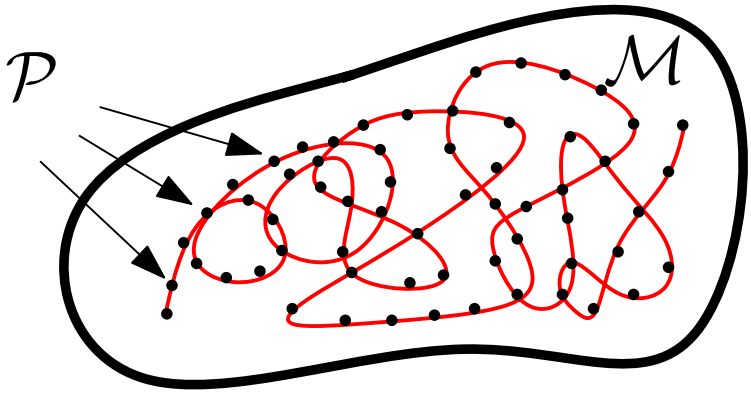


Figure 3.1: Diagram of patch set $\mathcal{P}$ (black points), trivial parameterization (red curve) and patch manifold $\mathcal{M}$.

parameterization is available [Pey08, Pey09]. One of the most important feature of the patch manifold is that it is close to a low-dimensional manifold for many natural images. This can be demonstrated through the following examples.

If f is a C^2 function which corresponds to a smooth image. Using Taylor's expansion, $p_x(f)$ can be well approximated by a linear function

$$p_x(f)(y) \approx f(x) + (y - x) \cdot \nabla f(x). \tag{3.16}$$

This fact implies that $\mathcal{M}(f)$ is close to a 3D manifold.

If f is a piecewise constant function which corresponds to a cartoon image, then, any patch, $p_x(f)$, is well approximated by a straight edge patch. Each patch is parametrized by the location and the orientation of the edge. This suggests that for a piecewise constant function, $\mathcal{M}(f)$ is also close to a 2D manifold.

If f is a oscillatory function corresponding to a texture image. We assume that f can be represented as

$$f(x) \approx a(x) \cos \theta(x), \tag{3.17}$$

where $a(x)$ and $\theta(x)$ are both smooth functions. For each pixel x, the patch $p_x(f)$ can be well approximated by

$$p_x(f) \approx a_L(y) \cos \theta_L(y), \tag{3.18}$$

where $a_L(x)$ and $\theta_L(x)$ are linear approximations of $a(x)$ and $\theta(x)$ at x, i.e.,

$$a_L(y) = a(x) + (y - x) \cdot \nabla a(x), \quad \theta_L(y) = \theta(x) + (y - x) \cdot \nabla \theta(x).$$

This means that the patch manifold $\mathcal{M}(f)$ is approximately a 6-dimensional manifold.

These simple examples show that for many images, smooth, cartoon and texture, the patch manifold is approximately a low dimensional manifold. Then one natural idea is to recover the original image by looking for the patch manifold with the lowest dimension. This idea leads to the low dimensional manifold model introduced in the next section.

3.3 Low dimensional manifold model

Based on the discussion in the previous section, we know that an important feature of the patch manifold is its low dimensionality. One natural idea is to use the dimension of the patch manifold as the regularization to recover the original image. In the low dimensional manifold model, we want to recover the image f such that the dimension of its patch manifold $\mathcal{M}(f)$ is as small as possible. This idea formally gives an optimization problem:

$$\min_{\substack{f \in \mathbb{R}^{m \times n}, \\ \mathcal{M} \subset \mathbb{R}^d}} \dim(\mathcal{M}), \qquad \text{subject to:} \quad y = \Phi f + \varepsilon, \quad \mathcal{P}(f) \subset \mathcal{M} \tag{3.19}$$

where $\dim(\mathcal{M})$ is the dimension of the manifold $\mathcal{M}$.

However, this optimization problem is not mathematically well defined, since we do not know how to compute $\dim(\mathcal{M})$ with given $\mathcal{P}(f)$. Next, we will derive a simple formula for $\dim(\mathcal{M})$ using some basic knowledge of differential geometry.

3.3.1 Calculation of $\dim(\mathcal{M})$

Here we assume $\mathcal{M}$ is a smooth manifold embedded in $\mathbb{R}^d$. First, we introduce some notations. Since $\mathcal{M}$ is a smooth submanifold isometrically embedded in $\mathbb{R}^d$, it can be locally parametrized as,

$$\boldsymbol{x} = \psi(\boldsymbol{\gamma}) : U \subset \mathbb{R}^k \to \mathcal{M} \subset \mathbb{R}^d, \tag{3.20}$$

where $k = \dim(\mathcal{M})$, $\boldsymbol{\gamma} = (\gamma^1, \cdots, \gamma^k)^t \in \mathbb{R}^k$ and $\boldsymbol{x} = (x^1, \cdots, x^d)^t \in \mathcal{M}$.

Let $\partial_{i'} = \frac{\partial}{\partial \gamma^{i'}}$ be the tangent vector along the direction $\gamma^{i'}$. Since $\mathcal{M}$ is a submanifold in $\mathbb{R}^d$ with induced metric, $\partial_{i'} = (\partial_{i'}\psi^1, \cdots, \partial_{i'}\psi^d)$ and the metric tensor is:

$$g_{i'j'} =< \partial_{i'}, \partial_{j'} >= \sum_{l=1}^{d} \partial_{i'}\psi^l \partial_{j'}\psi^l. \tag{3.21}$$

Let $g^{i'j'}$ denote the inverse of $g_{i'j'}$, i.e.,

$$\sum_{l'=1}^{k} g_{i'l'}\, g^{l'j'} = \delta_{i'j'} = \begin{cases} 1, & i' = j', \\ 0, & i' \neq j'. \end{cases} \tag{3.22}$$

For any function u on $\mathcal{M}$, let $\nabla_{\mathcal{M}} u$ denote the gradient of u on $\mathcal{M}$,

$$\nabla_{\mathcal{M}} u = \sum_{i',j'=1}^{k} g^{i'j'} \partial_{j'} u \; \partial_{i'}. \tag{3.23}$$

We can also view the gradient $\nabla_{\mathcal{M}} u$ as a vector in the ambient space $\mathbb{R}^d$ and let $\nabla^j_{\mathcal{M}} u$ denote the u component of the gradient $\nabla_{\mathcal{M}} u$ in the ambient coordinates, i.e.,

$$\nabla^j_{\mathcal{M}} u = \sum_{i',j'=1}^{k} \partial_{i'}\psi^j g^{i'j'} \partial_{j'} u, \quad j = 1, \cdots, d. \tag{3.24}$$

Let $\alpha_i,\ i = 1, \cdots, d$ be the coordinate functions on $\mathcal{M}$, i.e.

$$\alpha_i(\boldsymbol{x}) = x_i, \quad \forall \boldsymbol{x} = (x_1, \cdots, x_d) \in \mathcal{M} \tag{3.25}$$

Then, we have the following formula

Theorem 3.3.1. *Let $\mathcal{M}$ be a smooth submanifold isometrically embedded in $\mathbb{R}^d$. For any $\boldsymbol{x} \in \mathcal{M}$,*

$$\dim(\mathcal{M}) = \sum_{j=1}^{d} \|\nabla_{\mathcal{M}} \alpha_j(\boldsymbol{x})\|^2$$

Proof. First, following the definition of $\nabla_{\mathcal{M}}$, we have

$$\begin{aligned}
\sum_{j=1}^{d} \|\nabla_{\mathcal{M}} \alpha_j\|^2 =& \sum_{i,j=1}^{d} \nabla^i_{\mathcal{M}} \alpha_j \nabla^i_{\mathcal{M}} \alpha_j \\
=& \sum_{i,j=1}^{d} \left(\sum_{i',j'=1}^{k} \partial_{i'} \psi^i g^{i'j'} \partial_{j'} \alpha_j \right) \left(\sum_{i'',j''=1}^{k} \partial_{i''} \psi^i g^{i''j''} \partial_{j''} \alpha_j \right) \\
=& \sum_{j=1}^{d} \sum_{i',j',i'',j''=1}^{k} \left(\sum_{i=1}^{d} \partial_{i'} \psi^i \partial_{i''} \psi^i \right) g^{i'j'} g^{i''j''} \partial_{j'} \alpha_j \partial_{j''} \alpha_j \\
=& \sum_{j=1}^{d} \sum_{j',i'',j''=1}^{k} \left(\sum_{i'=1}^{k} g_{i'i''} g^{i'j'} \right) g^{i''j''} \partial_{j'} \alpha_j \partial_{j''} \alpha_j \\
=& \sum_{j=1}^{d} \sum_{j',i'',j''=1}^{k} \delta_{i''j'} g^{i''j''} \partial_{j'} \alpha_j \partial_{j''} \alpha_j \\
=& \sum_{j=1}^{d} \sum_{j',j''=1}^{k} g^{j'j''} \partial_{j'} \alpha_j \partial_{j''} \alpha_j.
\end{aligned}$$

The second equality comes from the definition of the gradient on $\mathcal{M}$, (3.24). The third and fourth equalities are due to (3.21) and (3.22) respectively.

Notice that

$$\partial_{j'} \alpha_j = \frac{\partial}{\partial \gamma^{j'}} \alpha_j(\psi(\boldsymbol{\gamma})) = \partial_{j'} \psi^j. \tag{3.26}$$

It follows that

$$\begin{aligned}
\sum_{j=1}^{d} \|\nabla_{\mathcal{M}} \alpha_j\|^2 =& \sum_{j=1}^{d} \sum_{j',j''=1}^{k} g^{j'j''} \partial_{j'} \psi^j \partial_{j''} \psi^j \\
=& \sum_{j',j''=1}^{k} g^{j'j''} \left(\sum_{j=1}^{d} \partial_{j'} \psi^j \partial_{j''} \psi^j \right) \\
=& \sum_{j',j''=1}^{k} g^{j'j''} g_{j'j''} \\
=& \sum_{j'=1}^{k} \delta_{j'j'} = k = \dim(\mathcal{M})
\end{aligned} \tag{3.27}$$

□

Using Theorem 3.3.1 , the optimization problem (3.19) can be rewritten as

$$\min_{\substack{f\in\mathbb{R}^{m\times n},\\ \mathcal{M}\subset\mathbb{R}^d}} \quad \sum_{i=1}^{d} \|\nabla_{\mathcal{M}}\alpha_i\|^2_{L^2(\mathcal{M})} + \lambda\|y-\Phi f\|_2^2, \qquad \text{subject to:} \quad \mathcal{P}(f)\subset\mathcal{M}. \tag{3.28}$$

where

$$\|\nabla_{\mathcal{M}}\alpha_i\|_{L^2(\mathcal{M})} = \left(\int_{\mathcal{M}} \|\nabla_{\mathcal{M}}\alpha_i(\boldsymbol{x})\|^2 d\boldsymbol{x}\right)^{1/2}. \tag{3.29}$$

This is the optimization problem we need to solve.

3.4 Numerical method

The optimization problem (3.28) is highly nonlinear and nonconvex. We propose an iterative method to solve it approximately. In the iterations, first the manifold is fixed, the image and the coordinate functions are computed. Then the manifold is updated using the new image and coordinate functions. More specifically:

- With a guess of the manifold $\mathcal{M}^n$ and a guess of the image f^n satisfying $\mathcal{P}(f^n)\subset\mathcal{M}^n$, compute the coordinate functions $\alpha_i^{n+1}, i=1,\cdots,d$ and f^{n+1},

$$(f^{n+1},\alpha_1^{n+1},\cdots,\alpha_d^{n+1}) = \arg\min_{\substack{f\in\mathbb{R}^{m\times n},\\ \alpha_1,\cdots,\alpha_d\in H^1(\mathcal{M}^n)}} \quad \sum_{i=1}^{d} \|\nabla_{\mathcal{M}^n}\alpha_i\|^2_{L^2(\mathcal{M}^n)} + \lambda\|y-\Phi f\|_2^2, \tag{3.30}$$

$$\text{subject to:} \quad \alpha_i(p_x(f^n)) = p_x^i(f),$$

  where $p_x^i(f)$ is the ith element of patch $p_x(f)$.

- Update $\mathcal{M}$ by setting

$$\mathcal{M}^{n+1} = \left\{(\alpha_1^{n+1}(\boldsymbol{x}),\cdots,\alpha_d^{n+1}(\boldsymbol{x})) : \boldsymbol{x}\in\mathcal{M}^n\right\}. \tag{3.31}$$

- Repeat these two steps until convergence.

In the above iteration, the manifold is easy to update. The key step is to solve (3.30), which is an optimization problem with linear constraints. We use the Bregman iterations [OBG05] to enforce the constraints in (3.30) which gives the following algorithm:

- Update $(f^{n+1,k+1}, \boldsymbol{\alpha}^{n+1,k+1})$ by solving

$$
\begin{aligned}
&(f^{n+1,k+1}, \alpha_1^{n+1,k+1}, \cdots, \alpha_d^{n+1,k+1}) \\
&= \arg \min_{\substack{\alpha_1,\cdots,\alpha_d \in H^1(\mathcal{M}^n), \\ f\in\mathbb{R}^{m\times n}}} \sum_{i=1}^{d} \|\nabla \alpha_i\|^2_{L^2(\mathcal{M}^n)} + \mu \|\boldsymbol{\alpha}(\mathcal{P}(f^n)) - \mathcal{P}(f) + d^k\|_{\mathrm{F}}^2 + \lambda \|y - \Phi f\|_2^2,
\end{aligned}
$$

  where

$$
\boldsymbol{\alpha}(\mathcal{P}(f^n)) = \begin{pmatrix} \alpha_1(\mathcal{P}(f^n)) \\ \alpha_2(\mathcal{P}(f^n)) \\ \vdots \\ \alpha_d(\mathcal{P}(f^n)) \end{pmatrix} \in \mathbb{R}^{d\times N}, \quad N = |\mathcal{P}(f^n)|,
$$

  and $\alpha_i(\mathcal{P}(f^n)) = (\alpha_i(x))_{x\in\mathcal{P}(f^n)}, i = 1\cdots, d$ are N dimensional row vectors. $\mathcal{P}(f)$ is also a $d \times N$ matrix, with each column represents one patch in $\mathcal{P}(f)$. $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm.

- Update d^{k+1},

$$
d^{k+1} = d^k + \boldsymbol{\alpha}^{n+1,k+1}(\mathcal{P}(f^n)) - \mathcal{P}(f^{n+1,k+1}).
$$

To further simplify the algorithm, we use the idea of split Bregman iteration [GO09b] to update f and α_i sequentially.

- Solve $\alpha_i^{n+1,k+1},\ i = 1, \cdots, d$ with fixed $f^{n+1,k}$,

$$
\begin{aligned}
&(\alpha_1^{n+1,k+1}, \cdots, \alpha_d^{n+1,k+1}) \\
&= \arg \min_{\alpha_1,\cdots,\alpha_d \in H^1(\mathcal{M}^n)} \sum_{i=1}^{d} \|\nabla \alpha_i\|^2_{L^2(\mathcal{M}^n)} + \mu \|\boldsymbol{\alpha}(\mathcal{P}(f^n)) - \mathcal{P}(f^{n+1,k}) + d^k\|_{\mathrm{F}}^2.
\end{aligned} \tag{3.32}
$$

- Update $f^{n+1,k+1}$,

$$
f^{n+1,k+1} = \arg \min_{f\in\mathbb{R}^{m\times n}} \quad \lambda \|y - \Phi f\|_2^2 + \mu \|\boldsymbol{\alpha}^{n+1,k+1}(\mathcal{P}(f^n)) - \mathcal{P}(f) + d^k\|_{\mathrm{F}}^2. \tag{3.33}
$$

- Update d^{k+1},

$$
d^{k+1} = d^k + \boldsymbol{\alpha}^{n+1,k+1}(\mathcal{P}(f^n)) - \mathcal{P}(f^{n+1,k+1}).
$$

Algorithm 6 LDMM Algorithm - Continuous version

Require: Initial guess of the image f^0, $d^0 = 0$.

Ensure: Restored image f.

1: **while** not converge **do**

2: **while** not converge **do**

3: With fixed manifold $\mathcal{M}^n$, for $i = 1, \cdots, d$, solving

$$\alpha_i^{n+1,k+1} = \arg\min_{\alpha_i \in H^1(\mathcal{M}^n)} \quad \|\nabla_{\mathcal{M}^n}\alpha_i\|^2_{L^2(\mathcal{M}^n)} + \mu\|\alpha_i(\mathcal{P}(f^n)) - \mathcal{P}_i(f^{n+1,k}) + d_i^k\|^2. \tag{3.35}$$

4: Update $f^{n+1,k+1}$,

$$f^{n+1,k+1} = \arg\min_{f \in \mathbb{R}^{m\times n}} \quad \lambda\|y - \Phi f\|_2^2 + \mu\|\boldsymbol{\alpha}^{n+1,k+1}(\mathcal{P}(f^n)) - \mathcal{P}(f) + d^k\|_{\mathrm{F}}^2 \tag{3.36}$$

5: Update d^{k+1},

$$d^{k+1} = d^k + \boldsymbol{\alpha}^{n+1,k+1}(\mathcal{P}(f^n)) - \mathcal{P}(f^{n+1,k+1}).$$

6: **end while**

7: Set $f^{n+1} = f^{n+1,k}$, $\boldsymbol{\alpha}^{n+1} = \boldsymbol{\alpha}^{n+1,k}$

8: Update $\mathcal{M}$

$$\mathcal{M}^{n+1} = \left\{(\alpha_1^{n+1}(\boldsymbol{x}), \cdots, \alpha_d^{n+1}(\boldsymbol{x})) : \boldsymbol{x} \in \mathcal{M}^n\right\}. \tag{3.37}$$

9: **end while**

Notice that in (3.32), $\alpha_i^{n+1,k+1},\ i = 1, \cdots, d$ can be solved separately,

$$\alpha_i^{n+1,k+1} = \arg\min_{\alpha_i \in H^1(\mathcal{M}^n)} \quad \|\nabla\alpha_i\|^2_{L^2(\mathcal{M}^n)} + \mu\|\alpha_i(\mathcal{P}(f^n)) - \mathcal{P}_i(f^{n+1,k}) + d_i^k\|^2 \tag{3.34}$$

where $\mathcal{P}_i(f^n)$ is the ith row of matrix $\mathcal{P}(f^n)$.

Summarizing the discussion above, we have Algorithm 6 to solve the optimization problem (3.28) in LDMM.

In Algorithm 6, the most difficult part is to solve the following type of optimization

problem

$$\min_{u\in H^1(\mathcal{M})} \quad \|\nabla_{\mathcal{M}} u\|^2_{L^2(\mathcal{M})} + \mu \sum_{\boldsymbol{y}\in P} |u(\boldsymbol{y}) - v(\boldsymbol{y})|^2, \tag{3.38}$$

where u can be any α_i, $\mathcal{M} = \mathcal{M}^n$, $P = \mathcal{P}(f^n)$ and $v(\boldsymbol{y})$ is a given function on P.

By a standard variational approach, we know that the solution of (3.38) can be obtained by solving the following PDE

$$\begin{cases} -\Delta_{\mathcal{M}} u(\boldsymbol{x}) + \mu \sum_{\boldsymbol{y}\in P} \delta(\boldsymbol{x}-\boldsymbol{y})(u(\boldsymbol{y}) - v(\boldsymbol{y})) = 0, & \boldsymbol{x} \in \mathcal{M} \\ \frac{\partial u}{\partial \mathbf{n}}(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \partial\mathcal{M}, \end{cases} \tag{3.39}$$

where $\partial\mathcal{M}$ is the boundary of $\mathcal{M}$ and $\mathbf{n}$ is the out normal of $\partial\mathcal{M}$. If $\mathcal{M}$ has no boundary, $\partial\mathcal{M} = \emptyset$.

The remaining problem is to solve the PDE (3.39) numerically. Notice that, we do not know the analytical form of the manifold $\mathcal{M}$. Instead, we know $\mathcal{P}(f^n)$ is a sample of the manifold $\mathcal{M}$, and we need to solve (3.39) on this unstructured point set $\mathcal{P}(f^n)$. We use the point integral method (PIM) [LSS, SSa, SSb] to solve (3.39) on $\mathcal{P}(f^n)$.

3.4.1 Point Integral Method

The point integral method was recently proposed to solve elliptic equations over a point cloud. For the Laplace-Beltrami equation, the key observation in the point integral method is the following integral approximation:

$$\int_{\mathcal{M}} \Delta_{\mathcal{M}} u(\boldsymbol{y}) \bar{R}_t(\boldsymbol{x},\boldsymbol{y}) d\boldsymbol{y} \approx -\frac{1}{t}\int_{\mathcal{M}} (u(\boldsymbol{x}) - u(\boldsymbol{y})) R_t(\boldsymbol{x},\boldsymbol{y}) d\boldsymbol{y} + 2\int_{\partial\mathcal{M}} \frac{\partial u(\boldsymbol{y})}{\partial \mathbf{n}} \bar{R}_t(\boldsymbol{x},\boldsymbol{y}) d\tau_{\boldsymbol{y}}, \tag{3.40}$$

where $t > 0$ is a parameter and

$$R_t(\boldsymbol{x},\boldsymbol{y}) = C_t R\left(\frac{|\boldsymbol{x}-\boldsymbol{y}|^2}{4t}\right). \tag{3.41}$$

$R : \mathbb{R}^+ \to \mathbb{R}^+$ is a positive C^2 function integrable over $[0, +\infty)$, and C_t is the normalizing factor

$$\bar{R}(r) = \int_r^{+\infty} R(s) ds, \quad \text{and } \bar{R}_t(\boldsymbol{x},\boldsymbol{y}) = C_t \bar{R}\left(\frac{|\boldsymbol{x}-\boldsymbol{y}|^2}{4t}\right). \tag{3.42}$$

We usually set $R(r) = e^{-r}$, then $\bar{R}_t(\boldsymbol{x}, \boldsymbol{y}) = R_t(\boldsymbol{x}, \boldsymbol{y}) = C_t \exp\left(\frac{|\boldsymbol{x}-\boldsymbol{y}|^2}{4t}\right)$ are Gaussians.

Next, we give a brief derivation of the integral approximation (3.40) in the Euclidean space. Here we assume $\mathcal{M}$ is an open set in $\mathbb{R}^d$. For a general submanifold, the derivation follows from the same idea but is technically more involved. Interested readers are referred to [SSa]. Thinking of $\bar{R}_t(\boldsymbol{x}, \boldsymbol{y})$ as test functions, and integrating by parts, we have

$$\begin{aligned} &\int_{\mathcal{M}} \Delta u(\boldsymbol{y}) \bar{R}_t(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} \\ =&-\int_{\mathcal{M}} \nabla u \cdot \nabla \bar{R}_t(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} + \int_{\partial \mathcal{M}} \frac{\partial u}{\partial \mathbf{n}} \bar{R}_t(\boldsymbol{x}, \boldsymbol{y}) d\tau_{\boldsymbol{y}} \\ =&\frac{1}{2t} \int_{\mathcal{M}} (\boldsymbol{y}-\boldsymbol{x}) \cdot \nabla u(\boldsymbol{y}) R_t(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} + \int_{\partial \mathcal{M}} \frac{\partial u}{\partial \mathbf{n}} \bar{R}_t(\boldsymbol{x}, \boldsymbol{y}) d\tau_{\boldsymbol{y}}. \end{aligned} \tag{3.43}$$

The Taylor expansion of the function u tells us that

$$u(\boldsymbol{y}) - u(\boldsymbol{x}) = (\boldsymbol{y}-\boldsymbol{x}) \cdot \nabla u(\boldsymbol{y}) - \frac{1}{2}(\boldsymbol{y}-\boldsymbol{x})^T \mathbf{H}_u(\boldsymbol{y})(\boldsymbol{y}-\boldsymbol{x}) + O(\|\boldsymbol{y}-\boldsymbol{x}\|^3),$$

where $\mathbf{H}_u(\boldsymbol{y})$ is the Hessian matrix of u at $\boldsymbol{y}$. Note that $\int_{\mathcal{M}} \|\boldsymbol{y}-\boldsymbol{x}\|^n R_t(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} = O(t^{n/2})$. We only need to estimate the following term:

$$\begin{aligned} &\frac{1}{4t} \int_{\mathcal{M}} (\boldsymbol{y}-\boldsymbol{x})^T \mathbf{H}_u(\boldsymbol{y})(\boldsymbol{y}-\boldsymbol{x}) R_t(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} \\ = &\frac{1}{4t} \int_{\mathcal{M}} (\boldsymbol{y}_i - \boldsymbol{x}_i)(\boldsymbol{y}_j - \boldsymbol{x}_j) \partial_{ij} u(\boldsymbol{y}) R_t(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} \\ = &-\frac{1}{2} \int_{\mathcal{M}} (\boldsymbol{y}_i - \boldsymbol{x}_i) \partial_{ij} u(\boldsymbol{y}) \partial_j \left(\bar{R}_t(\boldsymbol{x}, \boldsymbol{y})\right) d\boldsymbol{y} \\ = &\frac{1}{2} \int_{\mathcal{M}} \partial_j (\boldsymbol{y}_i - \boldsymbol{x}_i) \partial_{ij} u(\boldsymbol{y}) \bar{R}_t(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} + \frac{1}{2} \int_{\mathcal{M}} (\boldsymbol{y}_i - \boldsymbol{x}_i) \partial_{ijj} u(\boldsymbol{y}) \bar{R}_t(\boldsymbol{y}, \boldsymbol{x}) d\boldsymbol{y} \\ &-\frac{1}{2} \int_{\partial \mathcal{M}} (\boldsymbol{y}_i - \boldsymbol{x}_i) \mathbf{n}_j \partial_{ij} u(\boldsymbol{y}) \bar{R}_t(\boldsymbol{x}, \boldsymbol{y}) d\tau_{\boldsymbol{y}} \\ = &\frac{1}{2} \int_{\mathcal{M}} \Delta u(\boldsymbol{y}) \bar{R}_t(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} - \frac{1}{2} \int_{\partial \mathcal{M}} (\boldsymbol{y}_i - \boldsymbol{x}_i) \mathbf{n}_j \partial_{ij} u(\boldsymbol{y}) \bar{R}_t(\boldsymbol{x}, \boldsymbol{y}) d\tau_{\boldsymbol{y}} + O(t^{1/2}). \end{aligned} \tag{3.44}$$

The second summand in the last line is $O(t^{1/2})$. Although its $L_\infty(\mathcal{M})$ norm is of constant order, its $L^2(\mathcal{M})$ norm is of the order $O(t^{1/2})$ due to the fast decay of $w_t(\boldsymbol{x}, \boldsymbol{y})$. Therefore, we have Theorem 3.4.1 from combining equations (3.43) and (3.44).

Theorem 3.4.1. *If $u \in C^3(\mathcal{M})$ is a function on $\mathcal{M}$, then we have for any $\boldsymbol{x} \in \mathcal{M}$,*

$$\|r(u)\|_{L^2(\mathcal{M})} = O(t^{1/4}). \tag{3.45}$$

where

$$r(u) = \int_{\mathcal{M}} \Delta u(\boldsymbol{y}) \bar{R}_t(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} + \frac{1}{t} \int_{\mathcal{M}} (u(\boldsymbol{x}) - u(\boldsymbol{y})) R_t(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} - 2 \int_{\partial \mathcal{M}} \frac{\partial u(\boldsymbol{y})}{\partial \mathbf{n}} R_t(\boldsymbol{x}, \boldsymbol{y}) d\tau_{\boldsymbol{y}}.$$

The detailed proof can be found in [SSa].

Using the integral approximation (3.40), we get an integral equation to approximate the original Laplace-Beltrami equation (3.39),

$$\int_{\mathcal{M}} (u(\boldsymbol{x}) - u(\boldsymbol{y})) R_t(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} + \mu t \sum_{\boldsymbol{y} \in P} \bar{R}_t(\boldsymbol{x}, \boldsymbol{y})(u(\boldsymbol{y}) - v(\boldsymbol{y})) = 0 \tag{3.46}$$

This integral equation has no derivatives, and is easy to discretize over the point cloud.

3.4.2 Discretization

Next, we discretize the integral equation (3.46) over the point set $\mathcal{P}(f^n)$. To simplify the notation, we denote the point cloud as X. Notice that the point cloud $X = \mathcal{P}(f^n)$ in the n-th iteration.

Assume that the point set $X = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$ samples the submanifold $\mathcal{M}$ and it is uniformly distributed. The integral equation can be discretized very easily as:

$$\frac{|\mathcal{M}|}{N} \sum_{j=1}^{N} R_t(\boldsymbol{x}_i, \boldsymbol{x}_j)(u_i - u_j) + \mu t \sum_{j=1}^{N} \bar{R}_t(\boldsymbol{x}_i, \boldsymbol{x}_j)(u_j - v_j) = 0 \tag{3.47}$$

where $v_j = v(\boldsymbol{x}_j)$ and $|\mathcal{M}|$ is the volume of the manifold $\mathcal{M}$.

We can rewrite (3.47) in the matrix form.

$$(\boldsymbol{L} + \bar{\mu} \bar{\boldsymbol{W}}) \boldsymbol{u} = \bar{\mu} \bar{\boldsymbol{W}} \boldsymbol{v}. \tag{3.48}$$

where $\boldsymbol{v} = (v_1, \cdots, v_N)$ and $\bar{\mu} = \frac{\mu t N}{|\mathcal{M}|}$. $\boldsymbol{L}$ is a $N \times N$ matrix which is given as

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}, \tag{3.49}$$

where $\boldsymbol{W} = (w_{ij}),\ i, j = 1, \cdots, N$ is the weight matrix and $\boldsymbol{D} = \mathrm{diag}(d_i)$ with $d_i = \sum_{j=1}^{N} w_{ij}$. $\bar{\boldsymbol{W}} = (\bar{w}_{ij}),\ i, j = 1, \cdots, N$ is also a weight matrix. From (3.47), the weight matrices are

$$w_{ij} = R_t(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad \bar{w}_{ij} = \bar{R}_t(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{P}(f^n), \quad i, j = 1, \cdots, N. \tag{3.50}$$

Remark 3.4.1. *The discretization* (4.25) *is based on the assumption that the point set* $\mathcal{P}(f^n)$ *is uniformly distributed over the manifold such that the volume weight of each point is* $|\mathcal{M}|/N$. *If* $\mathcal{P}(f^n)$ *is not uniformly distributed, PIM actually solves an elliptic equation with variable coefficients [Shi] where the coefficients are associated with the distribution.*

Combining the point integral method within Algorithm 6, finally we get the algorithm in LDMM, Algorithm 7. In Algorithm 7, the number of split Bregman iterations is set to be 1 to simplify the computation.

3.5 Comparison with nonlocal methods

At first sight, LDMM is similar to nonlocal methods such as NLTV. But actually, they are very different. First, LDMM is based on minimizing the dimension of the patch manifold. The dimension of the patch manifold can be used as a general regularization in image processing. In this sense, LDMM is more systematic than nonlocal methods.

The other important difference is that the formulation of LDMM is continuous while the nonlocal methods use a graph based approach. In a graph based approach, a weighted graph is constructed which links different pixels x, y over the image with a weight $w(x, y)$. On this graph, the discrete gradient is defined.

$$\forall x, y, \quad \nabla_w f(x, y) = \sqrt{w(x, y)}(f(y) - f(x)). \tag{3.51}$$

Typically, in the nonlocal method, the following optimization problem is solved.

$$\min_{f \in \mathbb{R}^{m \times n}} \frac{1}{2}\|y - \Phi f\|^2 + \lambda J_{w(f)}(f). \tag{3.52}$$

where $J_w(f)$ is a regularization term related with the graph. Most frequently used $J_w(f)$ are the L^2 energy

$$J_w(f) = \left(\sum_{x,y} w(x, y)(f(x) - f(y))^2 \right)^{1/2} \tag{3.53}$$

and the nonlocal total variation

$$J_w(f) = \sum_x \left(\sum_y w(x, y)(f(x) - f(y))^2 \right)^{1/2}. \tag{3.54}$$

Algorithm 7 LDMM Algorithm

Require: Initial guess of the image f^0, $d^0 = 0$.

Ensure: Restored image f.

1: **while** not converge **do**

2: Compute the weight matrix $W = (w_{ij})$ from $\mathcal{P}(f^n)$, where $i, j = 1, \cdots, N$ and $N = |\mathcal{P}(f^n)|$ is the total number of points in $\mathcal{P}(f^n)$,

$$w_{ij} = R_t(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad \bar{w}_{ij} = \bar{R}_t(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{P}(f^n), \quad i, j = 1, \cdots, N.$$

And assemble the matrices $\boldsymbol{L}$, $\boldsymbol{W}$ and $\bar{\boldsymbol{W}}$ as following:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}, \ \boldsymbol{W} = (w_{i,j}), \ \bar{\boldsymbol{W}} = (\bar{w}_{i,j}), \ i, j = 1, \cdots, N.$$

3: Solve following linear systems

$$(\boldsymbol{L} + \bar{\mu}\bar{\boldsymbol{W}})\boldsymbol{U} = \bar{\mu}\bar{\boldsymbol{W}}\boldsymbol{V}.$$

where $\boldsymbol{V} = \mathcal{P}(f^n) - d^n$.

4: Update f by solving a least square problem.

$$f^{n+1} = \arg\min_{f \in \mathbb{R}^{m \times n}} \quad \lambda \|y - \Phi f\|_2^2 + \bar{\mu} \|\boldsymbol{U} - \mathcal{P}(f) + d^n\|_{\mathrm{F}}^2$$

5: Update d^n,

$$d^{n+1} = d^n + \boldsymbol{U} - \mathcal{P}(f^{n+1})$$

6: **end while**

The nonlocal methods provide powerful tools in image processing and were widely used in many problems. However, from the continuous point of view, the graph based approach has an intrinsic drawback.

Figure 3.2 (d) shows one example of subsample image recovery computed with an L^2 energy norm. The image of Barbara (Figure 3.2 (a)) is subsampled. Only 10% of the pixels are retained at random (Figure 3.2 (b)). It is clear that in the recovered image, some pixels are not consistent with their neighbors. From the zoomed in image, it is easy to see that these

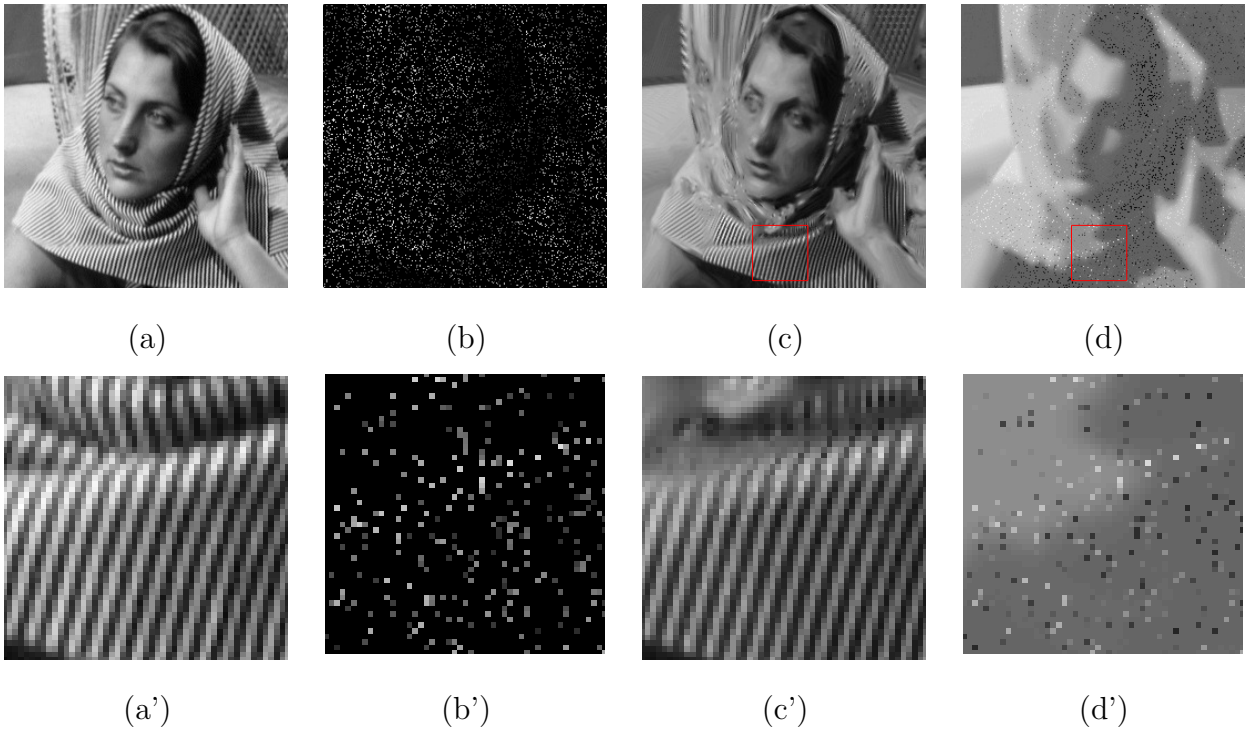


Figure 3.2: Subsampled image recovery of Barbara based on low dimensional manifold model (LDMM) and nonlocal method. (a): original image; (b): subsampled image (10% pixels are retained at random); (c): recovered image by LDMM; (d): recovered image by Graph Laplacian. The bottom row shows the zoom in image of the red box enclosed area.

pixels are just the retained pixels. This phenomenon shows that in the graph based approach, the value at the retained pixels do not spread to their neighbours properly. Compared with the graph based approach, the result given by LDMM method is much better (Figure 3.2 (c)).

This phenomenon can be explained by using a simple model problem, Laplace-Beltrami equation with the Dirichlet boundary condition,

$$\begin{cases} \Delta_{\mathcal{M}} u(x) = 0, & x \in \mathcal{M}, \\ u(x) = g(x), & x \in \partial\mathcal{M}. \end{cases} \tag{3.55}$$

where $\mathcal{M}$ is a smooth manifold embedded in $\mathbb{R}^d$ and $\partial\mathcal{M}$ is its boundary. Suppose the point cloud $X = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$ samples the manifold $\mathcal{M}$, and $B \subset X$ samples the boundary $\partial\mathcal{M}$.

Using the graph based method, the solution of the Dirichlet problem (3.55) is approxi-

mately obtained by solving the following linear system

$$\begin{cases} \sum_{j=1}^{N} w(\boldsymbol{x}_i, \boldsymbol{x}_j)(u(\boldsymbol{x}_i) - u(\boldsymbol{x}_j)) = 0, & \boldsymbol{x}_i \in X \backslash B, \\ u(\boldsymbol{x}_i) = g(\boldsymbol{x}_i), & \boldsymbol{x}_i \in B. \end{cases} \tag{3.56}$$

In the point integral method, we know that the Dirichlet problem can be approximated by an integral equation

$$\begin{cases} \frac{1}{t}\int_{\mathcal{M}}(u(\boldsymbol{x}) - u(\boldsymbol{y}))R_t(\boldsymbol{x}, \boldsymbol{y})d\boldsymbol{y} - 2\int_{\partial\mathcal{M}} \frac{\partial u(\boldsymbol{y})}{\partial \mathbf{n}} \bar{R}_t(\boldsymbol{x}, \boldsymbol{y})d\tau_{\boldsymbol{y}} = 0, & \boldsymbol{x} \in \mathcal{M}, \\ u(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \partial\mathcal{M}. \end{cases} \tag{3.57}$$

Comparing these two approximations, (3.56) and (3.57), we can see that in the graph based method, the boundary term, $-2\int_{\partial\mathcal{M}} \frac{\partial u(\boldsymbol{y})}{\partial \mathbf{n}} \bar{R}_t(\boldsymbol{x}, \boldsymbol{y})d\tau_{\boldsymbol{y}}$, is dropped. However, it is easy to check that this term is not small. Since the boundary term is dropped, the boundary condition is not enforced correctly in the graph based method.

Another difference between LDMM and the nonlocal methods is that the choice of patch is more flexible in LDMM. In nonlocal methods, for each pixel there is a patch and the patch has to be centered around this pixel. In LDMM, we only require that patches have the same size and cover the whole image. This feature gives us more freedom to choose the patches.

3.6 Numerical results

In the numerical simulations, the weight function we used is the Gaussian weight

$$R_t(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{\sigma(\boldsymbol{x})^2}\right) \tag{3.58}$$

$\sigma(\boldsymbol{x})$ is chosen to be the distance between $\boldsymbol{x}$ and its 20th nearest neighbour, To make the weight matrix sparse, the weight is truncated to the 50 nearest neighbors.

The patch size is 10×10 in the denoising and inpainting examples and is 20×20 in the super-resolution examples.

For each point in X, the nearest neighbors are obtained by using an approximate nearest neighbor (ANN) search algorithm. We use a k-d tree approach as well as an ANN search algorithm to reduce the computational cost. The linear system in Algorithm 7 is solved by GMRES.

PSNR defined as following is used to measure the accuracy of the results

$$\mathrm{PSNR}(f, f^*) = -20\log_{10}(\|f - f^*\|/255) \tag{3.59}$$

where f^* is the ground truth.

3.6.1 Inpainting

In the inpainting problems, the pixels are removed from the original image and we want to recover the original image from the remaining pixels. The corresponding operator, Φ, is

$$(\Phi f)(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ 0, & \boldsymbol{x} \notin \Omega, \end{cases} \tag{3.60}$$

where $\Omega \subset \{0, \cdots, m\} \times \{0, \cdots, n\}$ is the region where the image is retained. The pixels outside of Ω are removed. In our simulations, Ω is selected at random.

We assume that the original images do not have noise. In this noise free case, the parameter λ in the least-squares problem in Algorithm 7 is set to be ∞. Then, the least-squares problem can be solved as

$$f^{n+1}(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ (\mathcal{P}^*\mathcal{P})^{-1}(\mathcal{P}^*(\boldsymbol{U} + d^n)), & \boldsymbol{x} \notin \Omega \end{cases} \tag{3.61}$$

where $\mathcal{P}^*$ is the adjoint operator of $\mathcal{P}$. Notice that $\mathcal{P}^*\mathcal{P}$ is a diagonal operator. So f^{n+1} can be solved explicitly without inverting a matrix.

The initial guess of f is obtained by filling the missing pixels with random numbers satisfy Gauss distribution, $N(\mu_0, \sigma_0)$, where μ_0 is the mean of Φf and σ_0 is the standard deviation of Φf. The parameter $\bar{\mu} = 0.5$.

In our simulations, the original images are subsampled. Only 10% of the pixels are retained. The retained pixels are selected at random. From these 10% subsampled images, the LDMM method is employed to recover the original images. The recovery of several different images are shown in Figure 3.3. In this example, we compare the performance of LDMM with NLTV (3.6) and BPFA [ZCP12]. As we can see in Figure 3.3, the LDMM

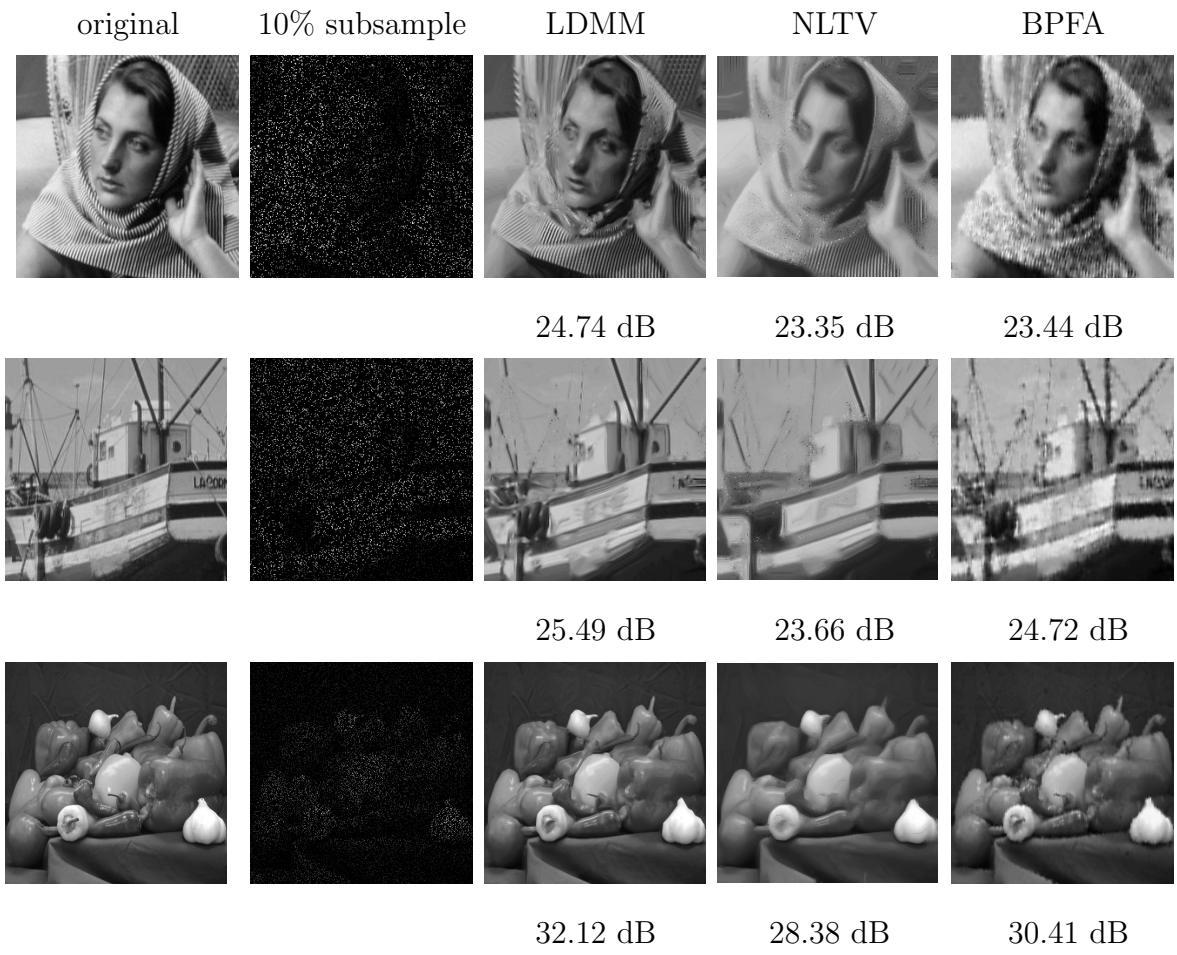


Figure 3.3: Examples of subsampled image recovery.

achieves the best results. LDMM recovers the image very well both in the cartoon part and the texture part. NLTV works well in the cartoon part, but the reconstruction of the texture is not satisfactory. BPFA has problems to recover sharp edges which makes the results visually less comfortable, although PSNR given by BPFA is better than NLTV.

In NLTV, we use the nonlocal gradient (3.3) to discretize the gradient operator, which may introduce inconsistency as we have pointed out in Section 5. In Figure 3.4, we show the zoomed in results given by NLTV. It can be clearly seen that there are many inconsistent pixels in the reconstructed images. If we zoom in the recovered images of Lena and pepper, we can also see similar phenomena.

Figure 3.4: Restored image from 10% subsample image by NLTV.

3.6.2 Super-resolution

Super-resolution is the recovery of a high-definition image from a low resolution image. Usually, the low resolution image is obtained by filtering and sub-sampling from a high resolution image.

$$\Phi f = (f * h) \downarrow^k \tag{3.62}$$

where h is a low-pass filter, $\downarrow^k$ is the down-sampling operator by a factor k along each axis. Here, we consider a simple case in which the filter h is the identity, i.e.,

$$\Phi f = (f) \downarrow^k \tag{3.63}$$

This downsample problem can be seen as a special case of subsample. However, in this problem, the pixels are retained over regular grid points which makes the recovery much more difficult than that in the random subsample problem considered in the previous example.

$$(\Phi f)(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ 0, & \boldsymbol{x} \notin \Omega, \end{cases} \tag{3.64}$$

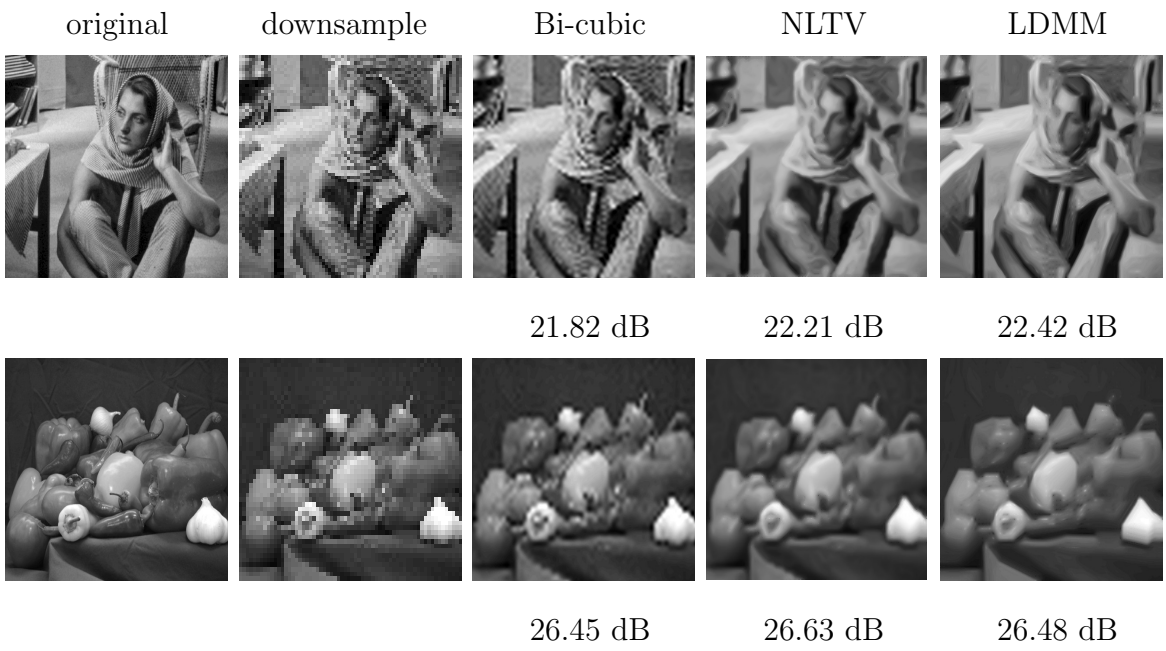


Figure 3.5: Example of image super-resolution with downsample rate $k = 8$.

where $\Omega = \{1, k+1, 2k+1, \cdots\} \times \{1, k+1, 2k+1, \cdots\}$.

Here, we also assume the measure is noise free and use the same formula (3.61) to update the image f. The parameter $\bar{\mu} = 0.5$ in the simulation. The initial guess is obtained by Bi-cubic interpolation.

Figure 3.5 shows the results for two images. The downsample rate is set to be 8. In terms of PSNR, compared with Bi-cubic interpolation, the improvement in LDMM is not substantial. However, the images given by LDMM are visually more pleasing than those given by Bi-cubic since the edges are reconstructed much better. The results of LDMM and NLTV are very close, except the edges in LDMM are a little sharper than those in NLTV.

3.6.3 Denoising

In the denoising problem, the operator Φ is the identity operator. In this case, the least-squares problem in Algorithm 7 has a closed form solution.

$$f^{n+1} = \left(\lambda \,\mathrm{Id} + \bar{\mu}\mathcal{P}^*\mathcal{P}\right)^{-1} \left(\lambda y + \bar{\mu}\mathcal{P}^*(\boldsymbol{U} + d^n)\right) \tag{3.65}$$

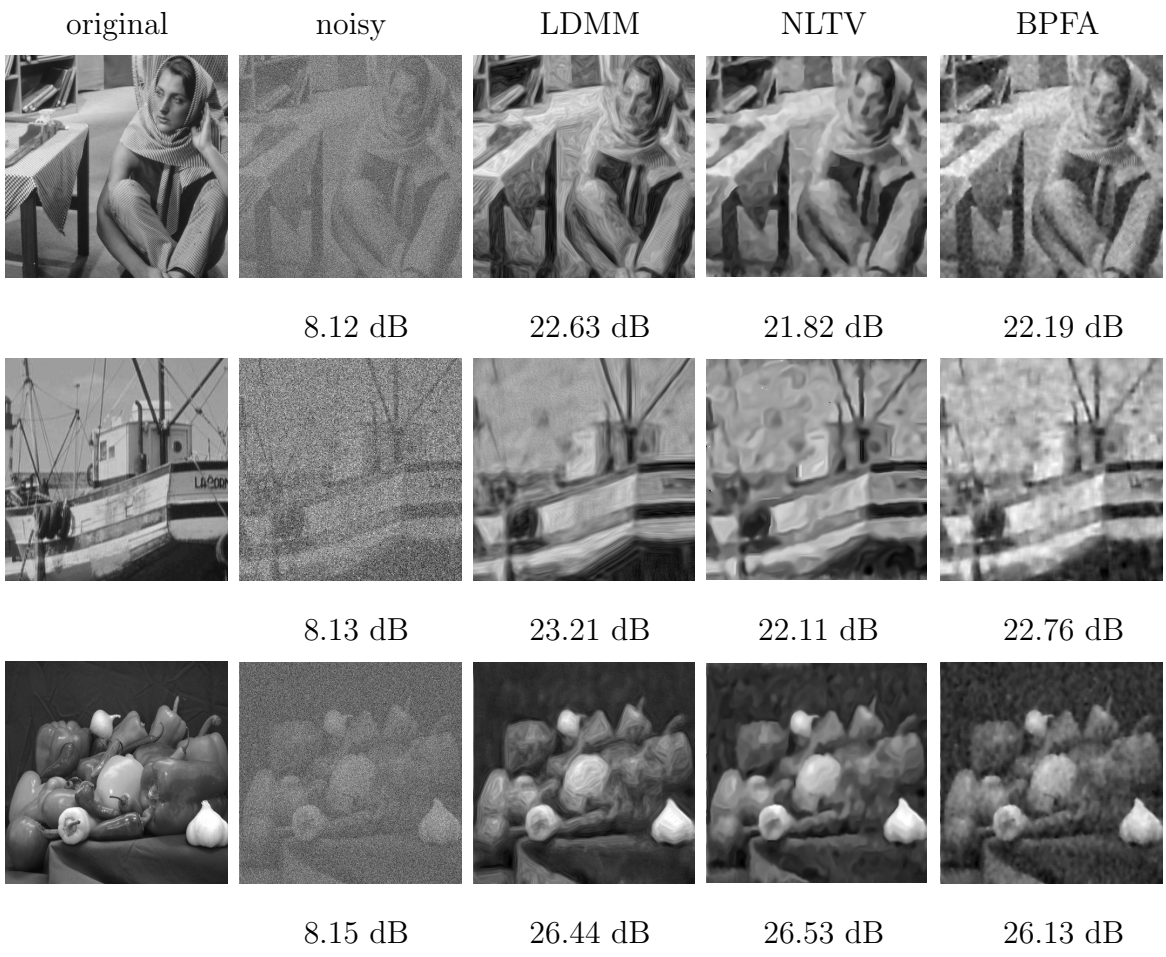


Figure 3.6: Example of image denoising with standard deviation $\sigma = 100$.

where $\mathcal{P}^*$ is the adjoint operator of $\mathcal{P}$.

In the denoising tests, Gaussian noise is added to the original images. The standard deviation of the noise is 100. Parameter $\bar{\mu} = 0.5$ and $\lambda = 0.2$ in the simulation.

Figure 4.9 shows the results obtained by LDMM, NLTV, and BPFA. The results given by LDMM are a little better in terms of PSNR. However, LDMM is much better visually since edges are reconstructed better.

Another very powerful image denoising method is BM3D [DFK07], which is also based on patches of images. In image denoising, we have to admit that the results of the LDMM model presented in this chapter are not as good as those obtained in BM3D. Some better results of LDMM using other numerical procedures will be reported in Chapter 4.

At the end of this section, we want to make some remarks on the computational speed of LDMM. As shown in this section, LDMM achieves good results for inpainting, super-resolution and denoising problems. On the other hand, the computational cost of LDMM is relatively high. For the example of Barbara (256×256) in inpainting problem, LDMM needs about 18 mins while BPFA needs about 15 mins. For Barbara (512×512) in the denoising problem, LDMM spends about 9 mins (about 25 mins in BPFA). Both of the tests are run with matlab code in a laptop equiped with CPU intel i7-4900 2.8GHz.

3.7 Conclusion

In this chapter, we presented a novel low dimensional manifold model (LDMM) for image processing. In the LDMM, instead of the image itself, we study the patch manifold of the image. Many studies reveal that the patch manifold has low dimensional structure for various classes of images. In LDMM, we just use the dimension of the patch manifold as the regularization to recover the original image from the partial information. The point integral method (PIM) also plays a very important role. It gives a correct way to solve the Laplace-Beltrami equation over the point cloud. We show the performance of the LDMM in subsample, downsample and denoising problems. LDMM achieves better results compared to the competing algorithms, especially in image inpainting problems. On the other hand, the dimension of the manifold can be used as a general regularization not only in image processing problems. The application of LDMM on hyperspectral image processing and large-scale scientific data will be presented in the next chapter.

CHAPTER 4

Low Dimensional Manifold Model with Weighted Graph Laplacian and Semi-local Patches

4.1 Introduction

The low dimensional manifold model presented in Chapter 3 performs very well in many image processing problems. However, the algorithm is fairly computationally expensive in two aspects: the computation of the weight, and solving over 100 linear systems every iteration on the patch domain. These two problems cause LDMM to be computationally infeasible when dealing with large images and high dimensional data. In this chapter, we will present two numerical techniques, weighted graph Laplacian and semi-local patches to solve the problems mentioned above. The rest of the chapter is organized as follows:

Section 4.2 explains the two numerical techniques, weighted graph Laplacian and semi-local patches, to speed up the computation. Section 4.3 illustrates the detailed implementation for the inpainting and denoising problems. Section 4.4 presents the numerical results. Finally, we draw our conclusion in Section 4.5.

4.2 Weighted Graph Laplacian and Semi-local Patches

4.2.1 Weighted Graph Laplacian

The weighted graph Laplacian (WGL) was proposed in [SOZ17] to smoothly interpolate functions on a point cloud. Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of points in $\mathbb{R}^d$, and a function g is given on a subset $S = \{s_1, s_2, \ldots, s_n\}$ of P. The goal is to find a function u on P that

extends g smoothly.

The widely used graph Laplacian model seeks to solve the interpolation problem via minimizing the following energy:

$$\mathcal{J}(u) = \sum_{x,y\in P} w(x,y)\left(u(x)-u(y)\right)^2, \quad \text{subject to:} \quad u(x) = g(x) \quad \text{on } S, \tag{4.1}$$

where $w(x,y)$ is some weight function, e.g. $w(x,y) = \exp\left(-\frac{\|x-y\|^2}{\sigma^2}\right)$. It is easy to check that $\mathcal{J}(u) = \|\nabla_{\mathcal{M}} u\|^2$ when the manifold gradient is discretized by the nonlocal gradient:

$$\nabla_{\mathcal{M}} u(x)(y) = \sqrt{w(x,y)}\left(u(x)-u(y)\right).$$

As mentioned in Chapter 3, minimizing the energy $\mathcal{J}$ in (4.1) would fail to achieve satisfactory results when the sample rate $|S|/|P|$ is very low. The reason is that the minimizer u of (4.1) does not ensure continuity of u on the labeled set S, or the information of g on S is not properly propagated to its vicinity. More specifically, after rewriting (4.1) in the following form:

$$\mathcal{J}(u) = \sum_{x\in S}\sum_{y\in P} w(x,y)\left(u(x)-u(y)\right)^2 + \sum_{x\in P\setminus S}\sum_{y\in P} w(x,y)\left(u(x)-u(y)\right)^2, \tag{4.2}$$

one can see that the first term in (4.2) is much smaller than the second term when $|S| \ll |P|$. As a result, the minimizing procedure would prioritize the second term, and therefore sacrifice the continuity of u on the sampled set S. An easy remedy for this scenario is to add a large weight $\mu = |P|/|S|$ in front of the first term in (4.2) to balance the two terms:

$$\mathcal{J}_{\mathrm{WGL}}(u) = \mu\sum_{x\in S}\sum_{y\in P} w(x,y)\left(u(x)-u(y)\right)^2 + \sum_{x\in P\setminus S}\sum_{y\in P} w(x,y)\left(u(x)-u(y)\right)^2. \tag{4.3}$$

It is readily checked that $\mathcal{J}_{\mathrm{WGL}}$ generalizes the graph Laplacian $\mathcal{J}$ in the sense that $\mathcal{J}_{\mathrm{WGL}} = \mathcal{J}$ when $|S| = |P|$. The generalized energy functional $\mathcal{J}_{\mathrm{WGL}}$ is called the weighted graph Laplacian.

The corresponding Euler-Lagrange equation of the variational model with the weighted graph Laplacian $\mathcal{J}_{\mathrm{WGL}}$ is:

$$\begin{cases} \displaystyle\sum_{y\in P} 2w(x,y)(u(x)-u(y)) + (\mu-1)\sum_{y\in S} w(y,x)(u(x)-g(y)) = 0, & x\in P\setminus S, \\ u(x) = g(x), & x\in S. \end{cases} \tag{4.4}$$

On the other hand, if we use the point integral method to solve the interpolation problem, the resulting linear system would be:

$$\sum_{y\in P} R_t(x,y)(u(x)-u(y)) + \frac{2}{\lambda}\sum_{y\in S} R_t(x,y)(u(y)-g(y)) = 0, \tag{4.5}$$

where $\lambda \ll 1$ is a positive parameter, $R_t(x,y) = \exp(-\frac{|x-y|^2}{4t})$ is the Gaussian weight function.

By comparing (4.4) and (4.5), it is clear that WGL can also be derived by replacing $u(y)$ in the second term of (4.5) by $u(x)$. This simplification is also quite intuitive: since u is supposed to be a smooth function on $\mathcal{M}$, $u(x)$ should be similar to $u(y)$ if the distance between x and y is small, i.e. $w(x,y)$ is large.

4.2.2 Semi-local Patches

The semi-local patches are obtained by adding local coordinates to the nonlocal patches in Chapter 3 with a weight λ, i.e.

$$(\bar{\mathcal{P}}f)(x) = [(\mathcal{P}f)(x), \lambda x] \in \mathbb{R}^{d+2}, \tag{4.6}$$

and the semi-local patch set is

$$\bar{\mathcal{P}}f = \{(\bar{\mathcal{P}}f(x) : x \in \bar{\Omega} = \{1,2,\ldots,m\}\times\{1,2,\ldots,n\}\}. \tag{4.7}$$

where $f \in \mathbb{R}^{m\times n}$ is the image, $x \in \bar{\Omega} = \{1,2,\ldots,m\}\times\{1,2,\ldots,n\}$ is the index of the pixel. The weight λ is used to get different locality in the semi-local patch. If $\lambda = 0$, the semi-local patch is tbe same as the nonlocal patch. If $\lambda \to \infty$, the patches are completely determined by local coordinates.

The semi-local patches have been successfully applied to many image processing problems [Pey08, TM98, SB97, SKS07]. In the geometrical point of view, if $\lambda \to \infty$, the patch set $\bar{\mathcal{P}}(f)$ is parametrized by a local 2D coordinate, $x \mapsto \bar{\mathcal{P}}(\mathbf{u})(x)$. However, as mentioned in Chapter 3, this parameterization is globally not injective and typically leads to high curvature variations and self-intersections. With this parametrization, the dimension of the manifold is very low, while the regularity is poor. If $\lambda = 0$, the underlying patch manifold may have

higher dimension, however the manifold becomes smoother. The idea in this chapter is to find a good compromise between the dimension and the regularity of the patch manifold by choosing a proper weight λ. With a proper λ, our experiments show that LDMM is accelerated significantly.

4.3 Numerical Implementation

4.3.1 Inpainting

We provide a detailed explaination of the numerical implementation of LDMM with WGL in inpainting. Recall from Chapter 3 that the variational problem is:

$$\min_{\substack{f\in\mathbb{R}^{m\times n},\\ \mathcal{M}\subset\mathbb{R}^d}} \quad \sum_{i=1}^{d} \|\nabla_{\mathcal{M}}\alpha_i\|^2_{L^2(\mathcal{M})}, \quad \text{subject to:} \quad b=\Phi_\Omega f, \quad \mathcal{P}(f)\subset\mathcal{M}, \tag{4.8}$$

where Φ_Ω is the projection onto the sampled set $\Omega\subset\bar{\Omega}=\{1,2,\ldots,m\}\times\{1,2,\ldots,n\}$, and b is the observed data.

The variational problem (4.8) is solved by alternating minimization with respect to $\mathcal{M}$ and f. More specifically, given $\mathcal{M}^k$ and f^k at step k satisfying $\mathcal{P}(f^k)\subset\mathcal{M}^k$:

- With fixed $\mathcal{M}^k$, update the data f^{k+1} by solving:

$$\begin{aligned}\min_{f\in\mathbb{R}^{m\times n}} \quad & \sum_{i=1}^{d} \|\nabla_{\mathcal{M}^k}\alpha_i\|^2_{L^2(\mathcal{M}^k)}, \\ \text{subject to:} \quad & \alpha_i(\mathcal{P}(f^k)(\boldsymbol{x}))=\mathcal{P}_i f(\boldsymbol{x}), \quad \boldsymbol{x}\in\overline{\Omega}, \quad i=1,\cdots,d, \\ & f(\boldsymbol{x})=b(\boldsymbol{x}), \quad \boldsymbol{x}\in\Omega\subset\overline{\Omega},\end{aligned} \tag{4.9}$$

  where $\mathcal{P}_i f(\boldsymbol{x})$ is the i-th element in the patch at the voxel $\boldsymbol{x}$.

- Update the manifold $\mathcal{M}^{k+1}$ by setting:

$$\mathcal{M}^{k+1}=\alpha(\mathcal{M}^k)$$

Using the terminology introduced in Section 4.2.1, the functions to be interpolated in (4.9) are α_i, the point cloud P is $\mathcal{P}(f^k)$, and the sampled set for α_i is

$$S_i=\left\{\mathcal{P}f^k(\boldsymbol{x}):\mathcal{P}_i f^k(\boldsymbol{x}) \text{ is sampled}\right\}.$$

Based on the discussion in Section 4.2.1, (4.9) can be discretized into the following problem:

$$\min_{f\in\mathbb{R}^{m\times n}} \quad \sum_{i=1}^{d}\left(\sum_{\boldsymbol{x}\in\overline{\Omega}\setminus\Omega_i}\sum_{\boldsymbol{y}\in\overline{\Omega}}\overline{w}(\boldsymbol{x},\boldsymbol{y})((\mathcal{P}_i f)(\boldsymbol{x})-(\mathcal{P}_i f)(\boldsymbol{y}))^2\right. \tag{4.10}$$
$$\left.+\frac{|\bar{\Omega}|}{|\Omega|}\sum_{\boldsymbol{x}\in\Omega_i}\sum_{\boldsymbol{y}\in\overline{\Omega}}\overline{w}(\boldsymbol{x},\boldsymbol{y})((\mathcal{P}_i f)(\boldsymbol{x})-(\mathcal{P}_i f)(\boldsymbol{y}))^2\right),$$

$$\text{Subject to:} \qquad f(\boldsymbol{x})=b(\boldsymbol{x}), \quad \boldsymbol{x}\in\Omega\subset\overline{\Omega},$$

where $\Omega_i = \left\{x\in\bar{\Omega}:\mathcal{P}_i f^k(\boldsymbol{x}) \text{ is sampled}\right\}$, $\bar{w}(\boldsymbol{x},\boldsymbol{y}) = w(\bar{\mathcal{P}}f(\boldsymbol{x}),\bar{\mathcal{P}}f(\boldsymbol{y}))$, and w is a symmetric sparse weight function computed from the point cloud $\bar{\mathcal{P}}f^k$ with semi-local patches. More specifically,

$$w(\boldsymbol{x},\boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{\sigma(\boldsymbol{x})\sigma(\boldsymbol{y})}\right), \tag{4.11}$$

where $\sigma(\boldsymbol{x})$ is the normalizing factor. In the numerical experiments, the weight w has been truncated to 20 nearest neighbors, and the normalizing factor is chosen as the distance between $\boldsymbol{x}$ and its 10th nearest neighbor.

It is readily checked by standard variational techniques that the Euler-Lagrange equation of (4.10) is:

$$\begin{cases}\left[\sum_{i=1}^{d}\mathcal{P}_i^*(h_i)+\mu\sum_{i=1}^{d}\mathcal{P}_i^*(g_i)\right](\boldsymbol{x})=0, & \boldsymbol{x}\in\bar{\Omega}\setminus\Omega\\ f(\boldsymbol{x})=b(\boldsymbol{x}), & \boldsymbol{x}\in\Omega\end{cases} \tag{4.12}$$

where $\mu=\frac{|\bar{\Omega}|}{|\Omega|}-1$, $\mathcal{P}_i^*$ is the adjoint operator of $\mathcal{P}_i$, and

$$\begin{cases}h_i(\boldsymbol{x})=\sum_{\boldsymbol{y}\in\bar{\Omega}}2\bar{w}(\boldsymbol{x},\boldsymbol{y})(\mathcal{P}_i f(\boldsymbol{x})-\mathcal{P}_i f(\boldsymbol{y}))\\ g_i(\boldsymbol{x})=\sum_{\boldsymbol{y}\in\Omega_i}\bar{w}(\boldsymbol{x},\boldsymbol{y})(\mathcal{P}_i f(\boldsymbol{x})-\mathcal{P}_i f(\boldsymbol{y}))\end{cases} \tag{4.13}$$

Recall that $\mathcal{P}_i f(\boldsymbol{x})$ is the i-th element of the patch $\mathcal{P}f(x)$. We use the notation $\boldsymbol{x}_{\widehat{i-1}}$ to denote the $(i-1)$-th element after $\boldsymbol{x}$ in the a patch, and a periodic padding is used when

patches exceed the domain of the image. It is easy to verify that $\mathcal{P}_i f(\boldsymbol{x}) = f(\boldsymbol{x}_{\widehat{i-1}})$, and $\mathcal{P}_i^* = \mathcal{P}_i^{-1}$.

Using such notation, one can deduce that:

$$\begin{aligned}\mathcal{P}_i^* h_i(\boldsymbol{x}) = h_i(\boldsymbol{x}_{\widehat{1-i}}) &= \sum_{\boldsymbol{y}\in\bar{\Omega}} 2\bar{w}(\boldsymbol{x}_{\widehat{1-i}}, \boldsymbol{y}) \left(\mathcal{P}_i f(\boldsymbol{x}_{\widehat{1-i}}) - \mathcal{P}_i f(\boldsymbol{y})\right) \\ &= \sum_{\boldsymbol{y}\in\bar{\Omega}} 2\bar{w}(\boldsymbol{x}_{\widehat{1-i}}, \boldsymbol{y}) \left(f(\boldsymbol{x}) - f(\boldsymbol{y}_{\widehat{i-1}})\right) \\ &= \sum_{\boldsymbol{y}\in\bar{\Omega}} 2\bar{w}(\boldsymbol{x}_{\widehat{1-i}}, \boldsymbol{y}_{\widehat{1-i}}) \left(f(\boldsymbol{x}) - f(\boldsymbol{y})\right)\end{aligned}$$

Therefore

$$\sum_{i=1}^{d} \mathcal{P}_i^*(h_i)(\boldsymbol{x}) = \sum_{i=1}^{d} \sum_{\boldsymbol{y}\in\bar{\Omega}} 2\bar{w}(\boldsymbol{x}_{\widehat{1-i}}, \boldsymbol{y}_{\widehat{1-i}}) \left(f(\boldsymbol{x}) - f(\boldsymbol{y})\right) \tag{4.14}$$

Similarly,

$$\sum_{i=1}^{d} \mathcal{P}_i^*(g_i)(\boldsymbol{x}) = \sum_{i=1}^{d} \sum_{\boldsymbol{y}\in\Omega} \bar{w}(\boldsymbol{x}_{\widehat{1-i}}, \boldsymbol{y}_{\widehat{1-i}}) \left(f(\boldsymbol{x}) - f(\boldsymbol{y})\right) \tag{4.15}$$

Combining (4.12)(4.14)(4.15), the Euler-Lagrange equation becomes:

$$\begin{cases} \displaystyle\sum_{\boldsymbol{y}\in\bar{\Omega}} \left(\sum_{i=1}^{d} 2\bar{w}(\boldsymbol{x}_{\widehat{1-i}}, \boldsymbol{y}_{\widehat{1-i}})\right) \left(f(\boldsymbol{x}) - f(\boldsymbol{y})\right) \\ \displaystyle + \mu \sum_{\boldsymbol{y}\in\Omega} \left(\sum_{i=1}^{d} \bar{w}(\boldsymbol{x}_{\widehat{1-i}}, \boldsymbol{y}_{\widehat{1-i}})\right) \left(f(\boldsymbol{x}) - f(\boldsymbol{y})\right) = 0, \quad \boldsymbol{x} \in \bar{\Omega} \setminus \Omega \\ \qquad\qquad\qquad\qquad f(\boldsymbol{x}) = b(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega \end{cases}$$

Let $\tilde{w}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{d} \bar{w}(\boldsymbol{x}_{\widehat{1-i}}, \boldsymbol{y}_{\widehat{1-i}})$, i.e. $\tilde{\boldsymbol{W}}$ is assembled from translated versions of the original matrix $\bar{\boldsymbol{W}}$, then

$$\begin{cases} \displaystyle 2\sum_{\boldsymbol{y}\in\bar{\Omega}} \tilde{w}(\boldsymbol{x}, \boldsymbol{y}) \left(f(\boldsymbol{x}) - f(\boldsymbol{y})\right) + \mu \sum_{\boldsymbol{y}\in\Omega} \tilde{w}(\boldsymbol{x}, \boldsymbol{y}) \left(f(\boldsymbol{x}) - f(\boldsymbol{y})\right) = 0, \quad \boldsymbol{x} \in \bar{\Omega} \setminus \Omega \\ \qquad\qquad\qquad\qquad f(\boldsymbol{x}) = b(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega \end{cases} \tag{4.16}$$

Define the graph Laplacian matrix $\tilde{\boldsymbol{L}}$ associated with the new weight matrix $\tilde{\boldsymbol{W}}$ as $\tilde{\boldsymbol{L}} = \tilde{\boldsymbol{D}} - \tilde{\boldsymbol{W}}$, where $\tilde{\boldsymbol{D}}$ is the diagonal matrix with diagonal entries $\tilde{\boldsymbol{D}}(\boldsymbol{x}, \boldsymbol{x}) = \sum_{\boldsymbol{y}\in\bar{\Omega}} \tilde{w}(\boldsymbol{x}, \boldsymbol{y})$.

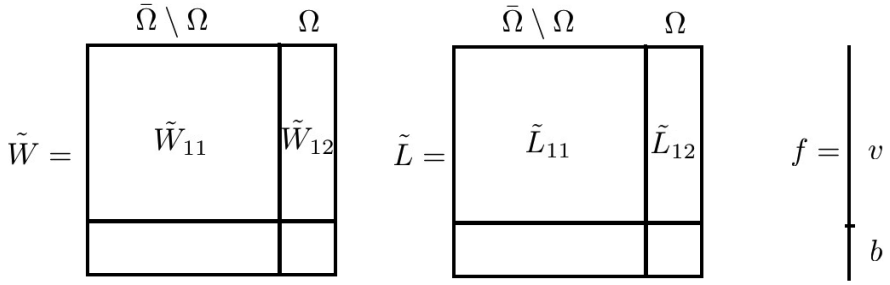


Figure 4.1: A visual illustration of the matrix/vector definitions. The matrices $\tilde{\boldsymbol{W}}$, $\tilde{\boldsymbol{L}}$ and $\boldsymbol{f}$ are partitioned into sampled (Ω) and unsampled ($\bar{\Omega}\setminus\Omega$) blocks. For example, $\tilde{\boldsymbol{W}}_{12}$ is the matrix corresponding to the weights between unsampled and sampled points.

It is easy to check that (4.16) can be written in the matrix form:

$$\begin{aligned} &2\tilde{\boldsymbol{L}}_{11}\boldsymbol{v} + 2\tilde{\boldsymbol{L}}_{12}\boldsymbol{b} + \mu(\boldsymbol{\Delta v} - \tilde{\boldsymbol{W}}_{12}\boldsymbol{b}) = \boldsymbol{0} \\ \Longleftrightarrow\ &(2\tilde{\boldsymbol{L}}_{11} + \mu\boldsymbol{\Delta})\boldsymbol{v} = \mu\tilde{\boldsymbol{W}}_{12}\boldsymbol{b} - 2\tilde{\boldsymbol{L}}_{12}\boldsymbol{b} \end{aligned} \tag{4.17}$$

where $\tilde{\boldsymbol{W}}_{ij}$ and $\tilde{\boldsymbol{L}}_{ij}$ are submatrices corresponding to unsampled ($i,j=1$) or sampled ($i,j=2$) parts of $\tilde{\boldsymbol{W}}$ and $\tilde{\boldsymbol{L}}$, $\boldsymbol{v}$ and $\boldsymbol{b}$ correspond to unsampled and sampled parts of $\boldsymbol{f}$, and $\boldsymbol{\Delta}$ is the diagonal matrix with its diagonal entries equaling the sums of the rows of $\tilde{\boldsymbol{W}}_{12}$. See Figure 4.1 for a visual illustration of the definitions of the matricies. The final LDMM algorithm with WGL for image inpainting is shown in Algorithm 8.

4.3.2 Denoising

LDMM with WGL can also be applied to image denoising. The observed data b in denoising is:

$$b(x) = f(x) + \varepsilon(x), \quad x \in \overline{\Omega} = \{1,2,\ldots,m\} \times \{1,2,\ldots,n\}, \tag{4.18}$$

where $f \in \mathbb{R}^{m\times n}$ is the original image, $b \in \mathbb{R}^{m\times n}$ is the noisy image, ε is the Gaussian noise.

Following the algorithm of LDMM in Chapter 3, we need to solve the following optimization problem:

$$\begin{aligned} &\min_{\substack{f\in\mathbb{R}^{m\times n},\\ \alpha_1,\cdots,\alpha_d}} \quad \sum_{i=1}^{d} \|\nabla_{\mathcal{M}}\alpha_i\|^2_{L^2(\mathcal{M})} + \gamma \sum_{x\in\overline{\Omega}} |b(x) - f(x)|^2, \\ &\text{subject to:} \quad \alpha_i(\bar{\mathcal{P}}(f^k)(x)) = (\mathcal{P}_i f)(x), \quad x \in \overline{\Omega}, \quad i = 1,\cdots,d. \end{aligned} \tag{4.19}$$

Algorithm 8 LDMM with WGL for image inpainting

Require: A subsampled data $f|_\Omega = b$.

Ensure: Reconstructed data f.

Initial guess f^0.

while not converge **do**

1. Compute the patch set $\mathcal{P}(f^k)$ from the current iterate f^k.
2. Compute the weight function

$$\overline{w}(\boldsymbol{x}, \boldsymbol{y}) = w(\bar{\mathcal{P}} f^k(\boldsymbol{x}), \bar{\mathcal{P}} f^k(\boldsymbol{y})), \quad \boldsymbol{x}, \boldsymbol{y} \in \overline{\Omega}.$$

3. Assemble the new weight function

$$\tilde{w}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{d} \bar{w}(\boldsymbol{x}_{\widehat{1-i}}, \boldsymbol{y}_{\widehat{1-i}})$$

4. Update the data f^{k+1} by solving equation (4.17).
5. $k \leftarrow k + 1$.

end while

$f = f^k$.

where γ is a penalty weight related to the variance of the noise.

Or equivalently, we can impose the penalty on the patches,

$$\begin{aligned} \min_{\substack{f \in \mathbb{R}^{m \times n}, \\ \alpha_1, \cdots, \alpha_d}} \quad & \sum_{i=1}^{d} \|\nabla_{\mathcal{M}} \alpha_i\|^2_{L^2(\mathcal{M})} + \gamma \sum_{i=1}^{d} \sum_{x \in \overline{\Omega}} |\alpha_i(\bar{\mathcal{P}}(f^k)(x)) - (\mathcal{P}_i b)(x)|^2, \\ \text{subject to:} \quad & \alpha_i(\bar{\mathcal{P}}(f^k)(x)) = (\mathcal{P}_i f)(x), \quad x \in \overline{\Omega}, \quad i = 1, \cdots, d. \end{aligned} \tag{4.20}$$

Next, we split f and $\alpha_1, \cdots, \alpha_d$, and solve them separately.

- Compute $\alpha_i^{k+1},\ i = 1, \cdots, d$ by solving

$$\min_{\alpha_1, \cdots, \alpha_d} \quad \sum_{i=1}^{d} \|\nabla_{\mathcal{M}} \alpha_i\|^2_{L^2(\mathcal{M})} + \gamma \sum_{i=1}^{d} \sum_{x \in \overline{\Omega}} |\alpha_i(\bar{\mathcal{P}}(f^k)(x)) - (\mathcal{P}_i b)(x)|^2 \tag{4.21}$$

- Update the image f^{k+1} by solving the least-squares problem

$$(\mathcal{P}_i f)(x) = \alpha_i(\bar{\mathcal{P}}(f^k)(x)), \quad i = 1, \cdots, d, \quad x \in \overline{\Omega}. \tag{4.22}$$

$\alpha_1, \cdots, \alpha_d$ are decoupled in (4.21). For each α_i, we need to solve the following type of problem:

$$\min_{u \in H^1(\mathcal{M})} \quad \|\nabla_{\mathcal{M}} u\|^2_{L^2(\mathcal{M})} + \gamma \sum_{\boldsymbol{y} \in \bar{P}} |u(\boldsymbol{y}) - v(\boldsymbol{y})|^2, \tag{4.23}$$

where u can be any α_i, $\bar{P} = \bar{\mathcal{P}}(f^k)$ is the semi-local patch set of f^k and $v(\boldsymbol{y})$ is a given function on $\bar{P}(f^k)$ corresponding to $(\mathcal{P}_i b)(x)$.

By a standard variational approach, we know that the solution of (4.23) can be obtained by solving the following PDE

$$\begin{cases} -\Delta_{\mathcal{M}} u(\boldsymbol{x}) + \gamma \sum_{\boldsymbol{y} \in \bar{P}} \delta(\boldsymbol{x} - \boldsymbol{y})(u(\boldsymbol{y}) - v(\boldsymbol{y})) = 0, & \boldsymbol{x} \in \mathcal{M} \\ \frac{\partial u}{\partial \mathbf{n}}(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \partial\mathcal{M}. \end{cases} \tag{4.24}$$

where $\partial\mathcal{M}$ is the boundary of $\mathcal{M}$ and $\mathbf{n}$ is the out normal of $\partial\mathcal{M}$. If $\mathcal{M}$ has no boundary, $\partial\mathcal{M} = \emptyset$.

Using the point integral method, the above Laplace-Beltrami equation is discretized as follows:

$$\sum_{\boldsymbol{y} \in \bar{P}} w(\boldsymbol{x}, \boldsymbol{y})(u(\boldsymbol{x}) - u(\boldsymbol{y})) + \bar{\gamma} \sum_{\boldsymbol{y} \in \bar{P}} w(\boldsymbol{x}, \boldsymbol{y})(u(\boldsymbol{y}) - v(\boldsymbol{y})) = 0, \quad \boldsymbol{x} \in \bar{P}.$$

$\bar{\gamma}$ is a parameter related to γ. We only need to set $\bar{\gamma}$ in the computation.

Based on the discussion in Section 4.2.1, the above equation is modified to get the equation in WGL approach,

$$\sum_{\boldsymbol{y} \in \bar{P}} w(\boldsymbol{x}, \boldsymbol{y})(u(\boldsymbol{x}) - u(\boldsymbol{y})) + \bar{\gamma} \sum_{\boldsymbol{y} \in \bar{P}} w(\boldsymbol{x}, \boldsymbol{y})(u(\boldsymbol{x}) - v(\boldsymbol{y})) = 0, \quad \boldsymbol{x} \in \bar{P}, \tag{4.25}$$

Based on the above derivation, we get an iterative algorithm (Algorithm 9) for image denoising.

Algorithm 9 Image Denoising

Require: A noisy image b.

Ensure: A denoised image f.

Let $f^0 = b$.

while not converge **do**

1. Generate semi-local patch set $\bar{\mathcal{P}}(f^k)$ from current image f^k with the weight λ in the semi-local patch.
2. Compute the weight function $w(\boldsymbol{x}, \boldsymbol{y}), \quad \boldsymbol{x}, \boldsymbol{y} \in \bar{P}(f^k)$.
3. Compute $\alpha_i,\ i = 1, \cdots, d$ by solving (4.25) with $P = \bar{\mathcal{P}}(f^k)$, $v = (\mathcal{P}_i b)(x)$.
3. Update the image f^{k+1} by

$$f^{k+1} = \left(\sum_{i=1}^{d} \mathcal{P}_i^* \mathcal{P}_i \right)^{-1} \left(\sum_{i=1}^{d} \mathcal{P}_i^* \alpha_i^{k+1} \right)$$

where $\mathcal{P}_i^*$ is the adjoint operator of $\mathcal{P}_i$.

4. $k \leftarrow k + 1$.

end while

$f = f^k$.

4.4 Numerical Results

4.4.1 Image and HSI Inpainting

4.4.1.1 Normal Image Inpainting

The numerical performance of LDMM with WGL in image inpainting is compared to that of the exemplar-based interpolation (EBI) [FAC09] and the piecewise linear estimator (PLE) [YSM12] in the case of random sampling interpolation. In all of the tests, the weight matricies built from semi-local patches are truncated to the 20 nearest neighbors, and the patch size is 10×10. PSNR defined in Chapter 3 is used to evaluate the results. The inpainting results from 5%, 10%, and 20% random subsamples are shown in Figure 4.2, Figure 4.3, Figure 4.4, and Table 4.1. It is easy to see both visually and numerically that LDMM consistently

Figure 4.2: Comparison of image inpainting with 95% missing pixels.

outperforms the competing algorithms, and the superiority of LDMM is more predominant when the sample rate is low. Moreover, Figure 4.5 shows the convergence of the algorithm in PSNR. It can be seen that the algorithm typically converges in less than 10 iterations, which is much less than the LDMM algorithm introduced in the previous chapter (100 iterations).

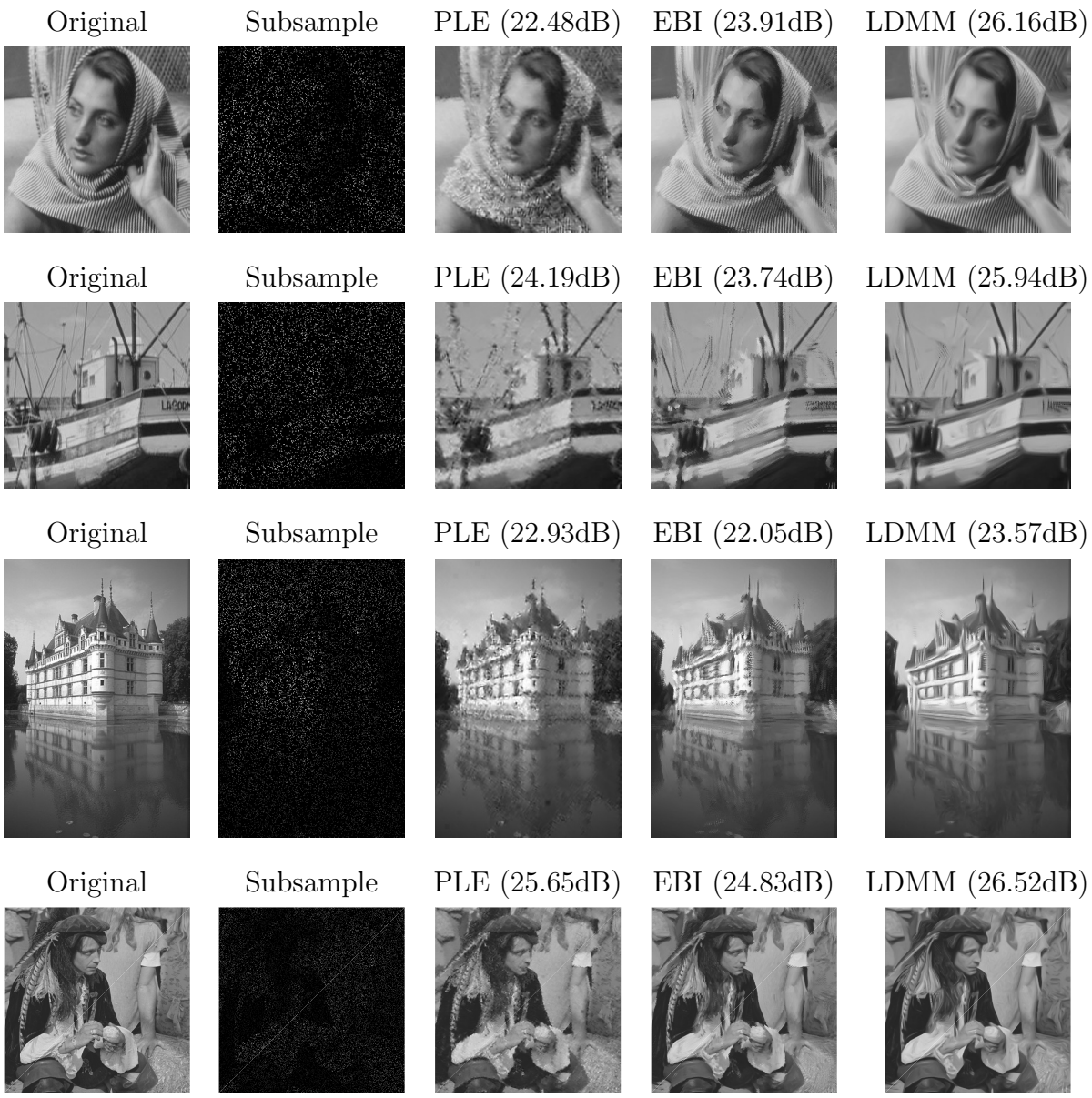


Figure 4.3: Comparison of image inpainting with 90% missing pixels.

4.4.1.2 Hyperspectral Image Inpainting

The original LDMM introduced in the previous chapter is too computationally expensive for high dimensional data such as hyperspectral images. The algorithm explained in this chapter is considerably faster and more suitable for hyperspectral image inpainting. Three datasets are used for testing:

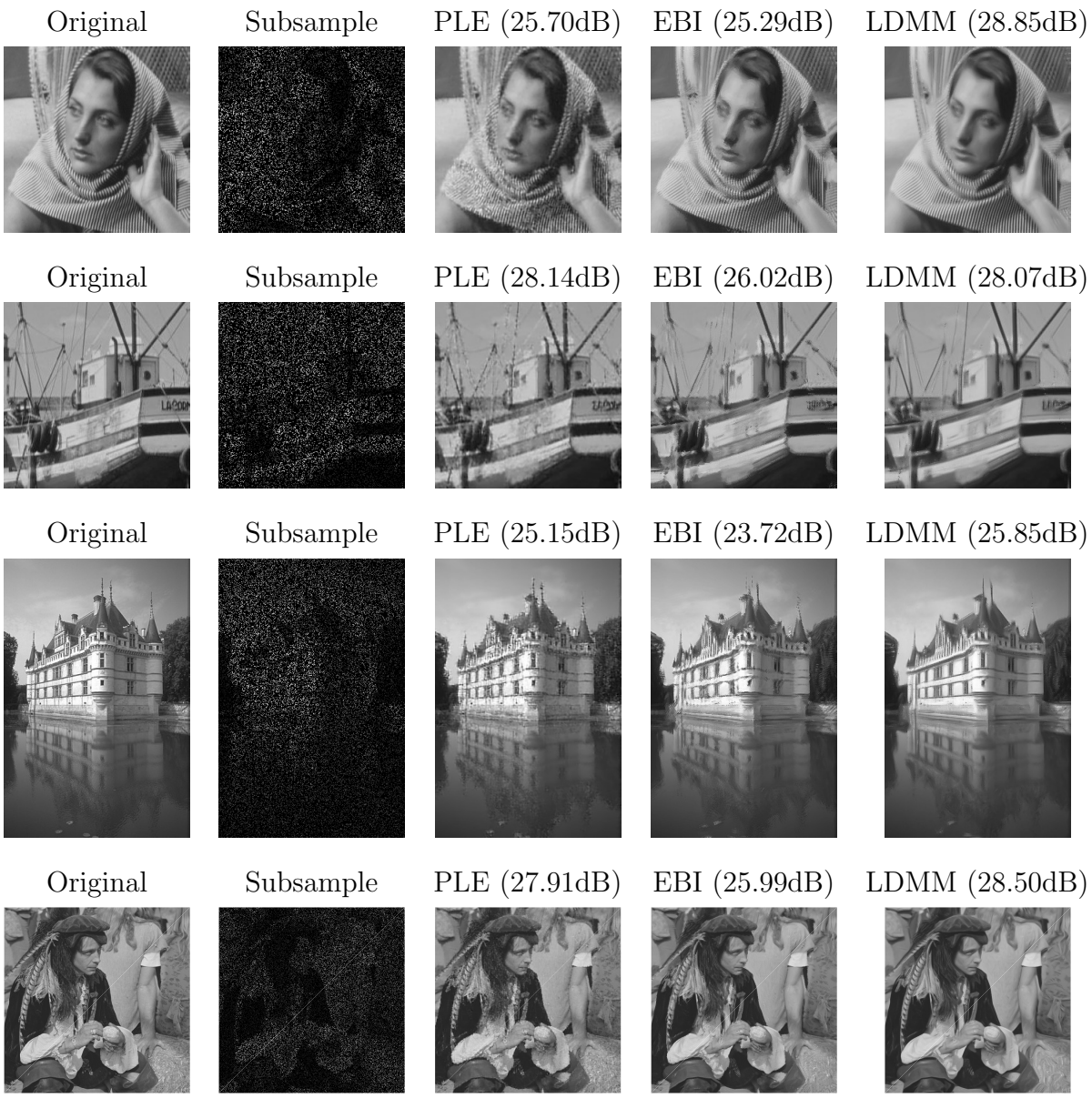


Figure 4.4: Comparison of image inpainting with 80% missing pixels.

- **Urban dataset:** almost linear mixed HSI with moderate noise.
- **Synthetic dataset:** generated by bilinear mixing model with moderate noise.
- **Kiwi dataset:** hyperspectral image with significant noise.

The inpainting results for these three datasets are displayed in Figure 4.6 and Figure 4.7. It is clear to see that the recovering results from only 5% subsample are almost indis-

	5%			10%			20%		
	EBI	PLE	LDMM	EBI	PLE	LDMM	EBI	PLE	LDMM
Barbara	20.66	19.72	**23.37**	23.91	22.48	**26.16**	25.29	25.70	**28.85**
Boat	20.55	20.54	**23.17**	23.74	24.19	**25.94**	26.02	**28.14**	28.07
Castle	20.54	20.78	**21.94**	22.05	22.93	**23.57**	23.72	25.15	**25.85**
Man	22.47	21.98	**24.32**	24.83	25.64	**26.52**	25.99	27.91	**28.50**

Table 4.1: PSNR of EBI, PLE, LDMM with 5%, 10% and 20% sample rate.

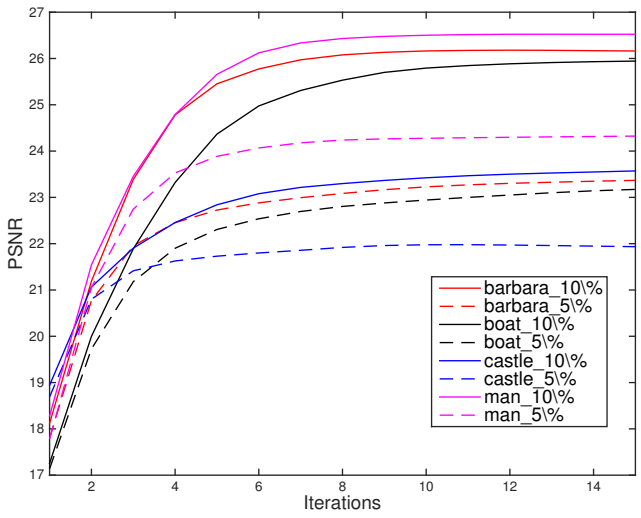


Figure 4.5: Convergence of LDMM for image inpainting in PSNR

tinguishable from the original images with little noise, and Figure 4.7 shows that LDMM with a penalty instead of hard constraint can reconstruct the image from significant noise and missing voxels.

4.4.2 Image Denoising

We use the same weight given in Section 4.4.1. The patch size is 10×10. The number of iterations is 3. The weight λ in the semi-local patches is 5. In our tests, the noise ε is Gaussian noise with the standard deviation $\sigma = 100$. $\bar{\gamma} = 0.5$ in (4.25).

The denoising results are shown in Figure 4.9. The performance of LDMM is compared

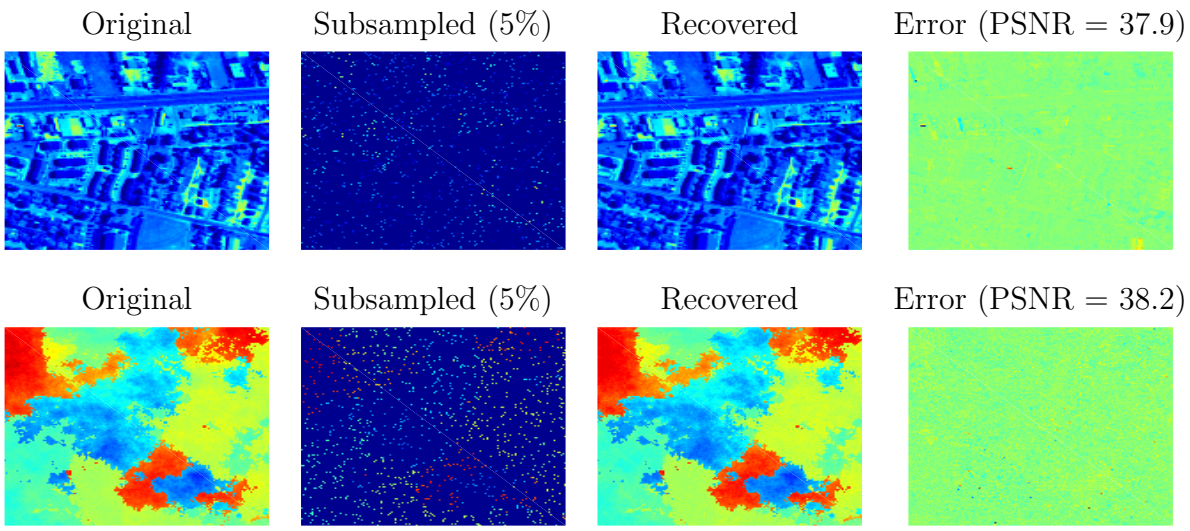


Figure 4.6: HSI reconstruction from 95% missing voxels.

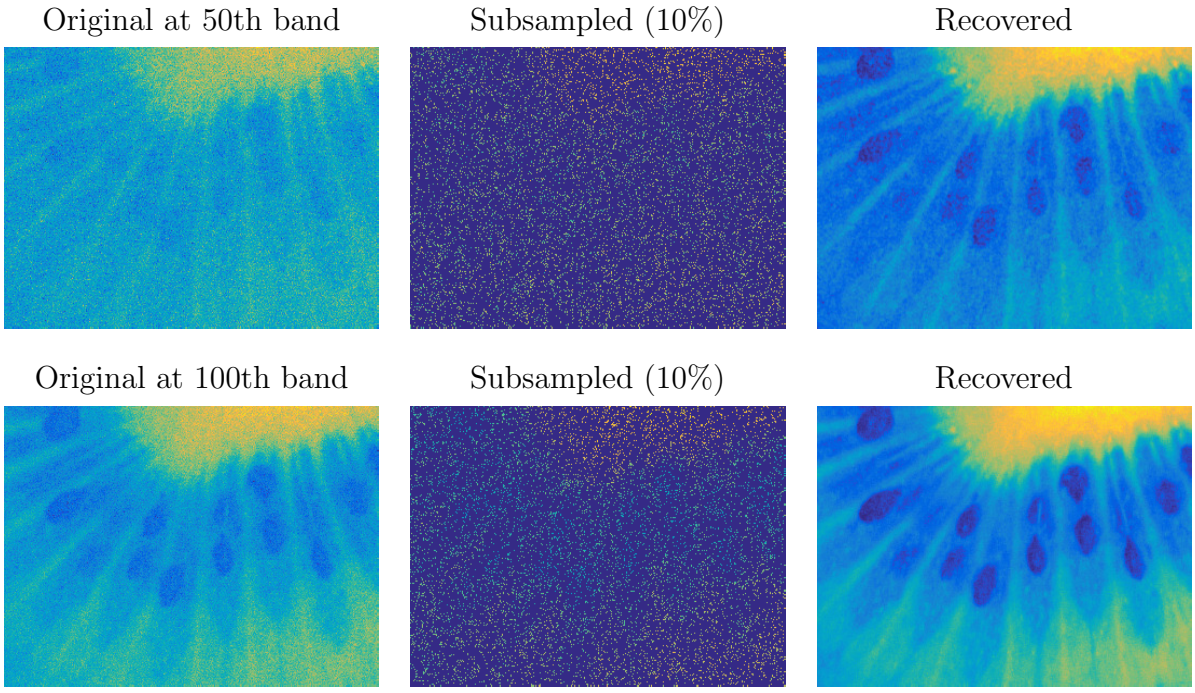


Figure 4.7: HSI reconstruction from 95% missing voxels and significant noise.

to the method of block-matching with 3D collaborative filtering (BM3D) [DFK07], which is known to achieve state-of-the-art results in image denoising. As we can see in Figure 4.9, the results of BM3D is better in PSNR except for the image of Barbara. LDMM seems

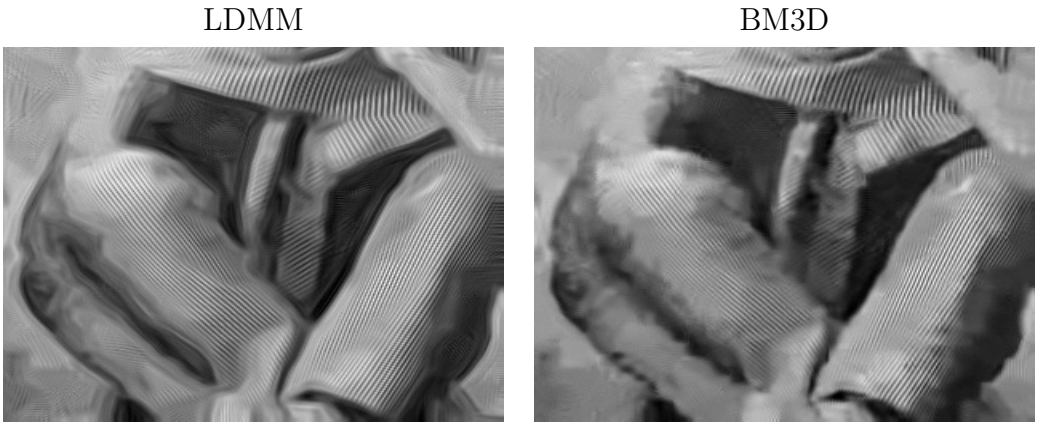


Figure 4.8: Fragment of the denoised image of Barbara using LDMM and BM3D.

to reconstruct the edges and textures sharper than BM3D as shown in Figure 4.8. This difference can also be found in the result of the image of the house. LDMM seems to recover the edges of the house smoother and sharper, although the PSNR is lower than that of BM3D.

4.4.3 Interpolation of Scientific Data

In this section, we present the numerical results of LDMM on various 2D and 3D scientific data interpolation from either regular or irregular samplings. The performance of LDMM is compared to that of the exemplar-based interpolation (EBI) [FAC09] and the piecewise linear estimator (PLE) [YSM12] in the case of random sampling interpolation. As pointed out in [YSM12], PLE fails to work on regular sampling interpolation without a proper initialization (bicubic interpolation in their case). We also noticed in our experiment that the result of EBI on regular sampling interpolation is inferior to that of the simple cubic spline interpolation. Therefore, in the case of regular sampling interpolation, we instead compare the results of LDMM to the standard methods including cubic spline interpolation, discrete Fourier transform (DFT), discrete cosine transform (DCT), and wavelet transform. Moreover, we also examine the effectiveness of LDMM as a data compression technique and compare it to other standard compression methods including DFT, DCT, wavelet transform, and tensor decomposition. As for the tensor decomposition methods, we use the singular

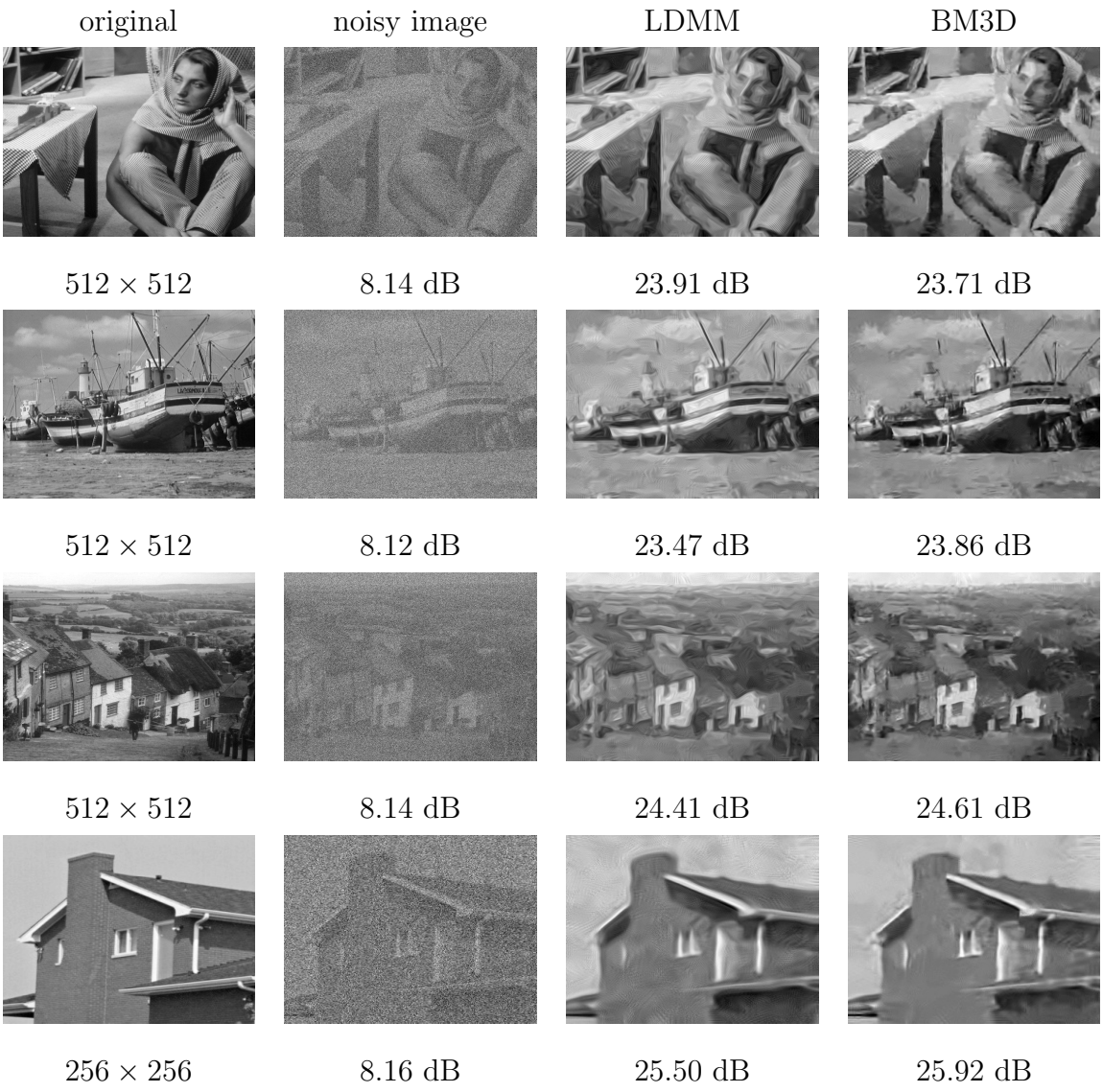


Figure 4.9: Results of image denoising.

value decomposition (SVD) for 2D data, and the Tucker decomposition [BK15, BK06] for 3D data. The Tucker decomposition is a form of higher-order SVD, which decomposes a tensor into a core tensor multiplied by a matrix along each mode.

4.4.3.1 Description of the Testing Data and Parameter Setup

The algorithms are tested on six scientific datasets, three of which are three-dimensional. See Figure 4.10 and Figure 4.11 for visual illustrations of the data.

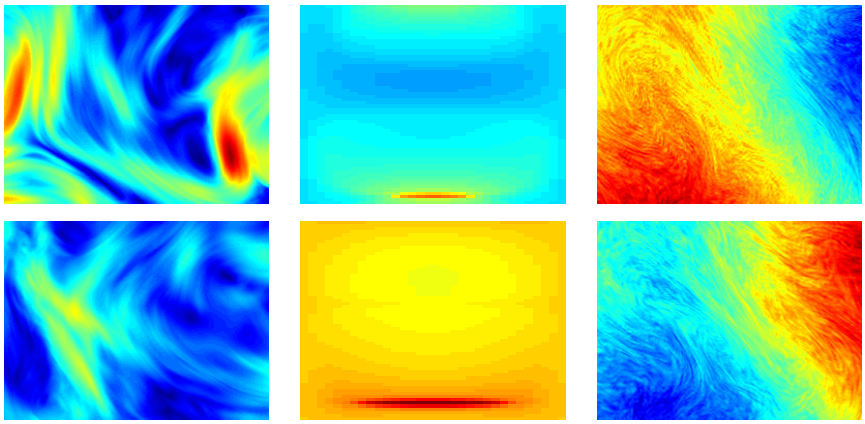
- **3D plasma (magnetic field)**: The data set is taken from a gyrokinetic simulation of Alfvénic turbulence in 5D phase space (3D real space plus 2D velocity space, with the fast gyroangle dependence removed) [TJT15], carried out with the GENE code [JDK00]. It represents a snapshot of the magnitude of magnetic field fluctuations in real space during the statistically quasi-stationary state of fully developed turbulence. In this simulation, the focus is on the dissipation range of this weakly collisional turbulent plasma which cannot be described adequately by magnetohydrodynamics (MHD). Gyrokinetics offers an efficient description of the very tail of the MHD cascade. The size of this data is $256 \times 256 \times 32$.

- **3D/2D lattice**: The lattice benchmark problem, originally due to Brunner [Bru02, BH05], is a two-dimensional cartoon of a nuclear reactor assembly that has become a common test problem of angular discretization methods for kinetic equations of radiation transport [HM13, MH10a, MH10b, SFL09].

  The simulated quantity is a distribution function that depends on five independent variables: two spatial, two angular, plus time. The data used here was generated using the algorithm described in [CCHed] which combines a third-order space-time discretization (discontinuous Galerkin in space and integral deferred correction in time) and an angular discretization based on a tensor product collocation scheme.

  We consider for this problem two quantities of interest. The first (**2D lattice**) is the angular average of the distribution function at a fixed time; this is a two-dimensional data set of size 896×896. The second is the distribution function at a fixed time and fixed vertical location along the line $y = 4.5$. This is a three dimensional data set of size $188 \times 64 \times 32$. Both sets of data are given in log scale.

- **3D/2D plasma (distribution function)**: This data set is again taken from a gyrokinetic simulation of Alfvénic turbulence in 5D phase space (3D real space plus 2D velocity space, with the fast gyroangle dependence removed) described in [TJT15]. The 3D data set describes the distribution function for the ion species as a function of the two spatial coordinates perpendicular to the background magnetic field and of the



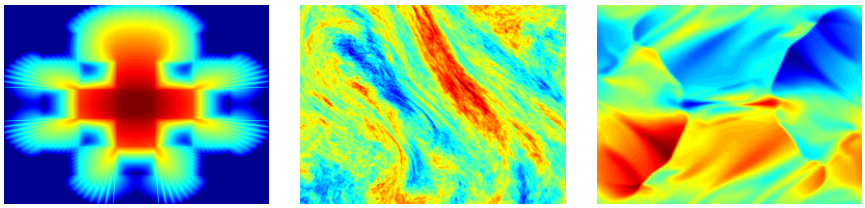
(a) (b) (c)

Figure 4.10: Visual illustrations of the 3D data sets. The two figures in column (a) are 2D spatial cross sections of the 3D plasma (magnetic field) data set at different z coordinates. The figures in column (b) are 2D cross sections of the 3D lattice data set correspongding to angular flux at $x = 0.24$ and $x = 1.18$. The figures in column (c) are 2D spatial cross sections of the 3D plasma (distribution function) data set.

velocity parallel to this guide field at a given value of perpendicular velocity and time. Meanwhile, the 2D data set describes a snapshot of the same distribution function for the ion species as a function of the two perpendicular spatial coordinates integrated over velocity space. The sizes of the 3D and 2D data sets are $256 \times 256 \times 32$ and 256×256 respectively.

- **2D vortex**: This data set comes from a numerical solution of the Orszag-Tang vortex system [OT79], which provides a model of complex flow with many features of magnetohydrodynamics systems. Starting from a smooth state, the system evolves into turbulance, generating complex interactions between different shock waves. The data used in this paper is the numerical solution at time $t = 2$ of the density component obtained with the third order Chebyshev polynomial approximate Osher-Solomon scheme [CGM16] on a 256×256 uniform mesh.

For irregular sampling interpolation, the algorithms are tested to reconstruct the original data from 5% and 10% random subsamples. For the regular aliased sampling, the original 2D data are decimated by a factor of 4 in both directions; for 3D data, we consider two types



(a) (b) (c)

Figure 4.11: Visual illustrations of the 2D data sets. (a) 2D lattice. (b) 2D plasma (distribution function). (c) 2D vortex.

of sampling procedures: downsampling by a factor of 2 in all directions, or by a factor of 4 in only the first two dimensions.

For all the datasets listed above, the weight matrices in LDMM are truncated to 20 nearest neighbors, and the normalizing factor $\sigma(\boldsymbol{x})$ in (4.11) is chosen as the distance between $\boldsymbol{x}$ and its 10th nearest neighbor. The patch sizes chosen for different datasets are listed in Table 4.2. The reason why the 2D plasma (distribution function) dataset uses a much larger patch size, 16×16 instead of 6×6, is that the structures in this dataset are much more complicated than the other datasets. This complexity implies a much higher intrinsic dimension of the patch manifold. Therefore a larger patch size is chosen so that the manifold dimension can be still smaller than that of the embedding space. Notice also that $6 \times 6 \times 1$ patch size is chosen for the 3D plasma (magnetic field) dataset. This is because of the low resolution of the data in the third dimension. However, $6 \times 6 \times 4$ patches are chosen in the $2 \times 2 \times 2$ regular down sampling. This is because we want to avoid patches that do not contain any sampled voxels.

The quality of the reconstruction $\hat{f}$ of the original data $f \in \mathbb{R}^{m \times n \times B}$ is evaluated in the following three norms:

$$\|e\|_1 = \frac{1}{mnB} \sum_{i,j,k} |e_{i,j,k}/R|, \tag{4.26}$$

$$\|e\|_2 = \left(\frac{1}{mnB} \sum_{i,j,k} |e_{i,j,k}/R|^2 \right)^{\frac{1}{2}}, \tag{4.27}$$

$$\|e\|_\infty = \max_{i,j,k} |e_{i,j,k}/R|, \tag{4.28}$$

	5%	10%	4×4	$4 \times 4 \times 1$	$2 \times 2 \times 2$
2D lattice	6×6	6×6	6×6	N/A	N/A
2D plasma (D)	16×16	16×16	16×16	N/A	N/A
2D vortex	6×6	6×6	6×6	N/A	N/A
3D plasma (M)	$6 \times 6 \times 1$	$6 \times 6 \times 1$	N/A	$6 \times 6 \times 1$	$6 \times 6 \times 4$
3D lattice	$4 \times 4 \times 4$	$4 \times 4 \times 4$	N/A	$4 \times 4 \times 4$	$4 \times 4 \times 4$
3D plasma (D)	$6 \times 6 \times 4$	$6 \times 6 \times 4$	N/A	$6 \times 6 \times 4$	$6 \times 6 \times 4$

Table 4.2: Patch sizes for different datasets. The first row of the table indicates the different types of downsampling procedures. 3D/2D plasma (D) stands for 3D/2D plasma (distribution function), and 3D plasma (M) stands for 3D plasma (magetic field).

where $e = f - \hat{f}$ is the error of the reconstruction, $R = \max_{i,j,k} \hat{f}_{i,j,k} - \min_{i,j,k} \hat{f}_{i,j,k}$ is the numerical range of the data. Moreover, PSNR is also given to measure the performance of the algorithms.

4.4.3.2 Interpolation with Random Sampling

The visual results of the interpolation with 10% and 5% are shown in Figure 4.12 $\sim$ Figure 4.19. The errors of the reconstruction in different norms are displayed in Table 4.3 $\sim$ Talbe 4.8. It can be observed that LDMM consistently performs at a higher accuracy than EBI and PLE either visually or numerically. The superiority of LDMM is more dramatic when the sample rate is very low (5%), in which case PLE fails to achieve reasonable results. LDMM also manages to yield smoother results, whereas EBI tends to create artificial patchy patterns. We point out that the reconstruction of the 3D data with PLE and EBI are obtained by applying the algorithms to 2D cross sections because of a lack of 3D implementations of both algorithms. Therefore it is not entirely fair to compare LDMM to PLE and EBI on the 3D data. This is especially clear on the 3D lattice dataset, where values change smoothly on each direction. Nonethelss, the vast superiorty of LDMM on 2D examples illustrates its advantage over the competing algorithms.

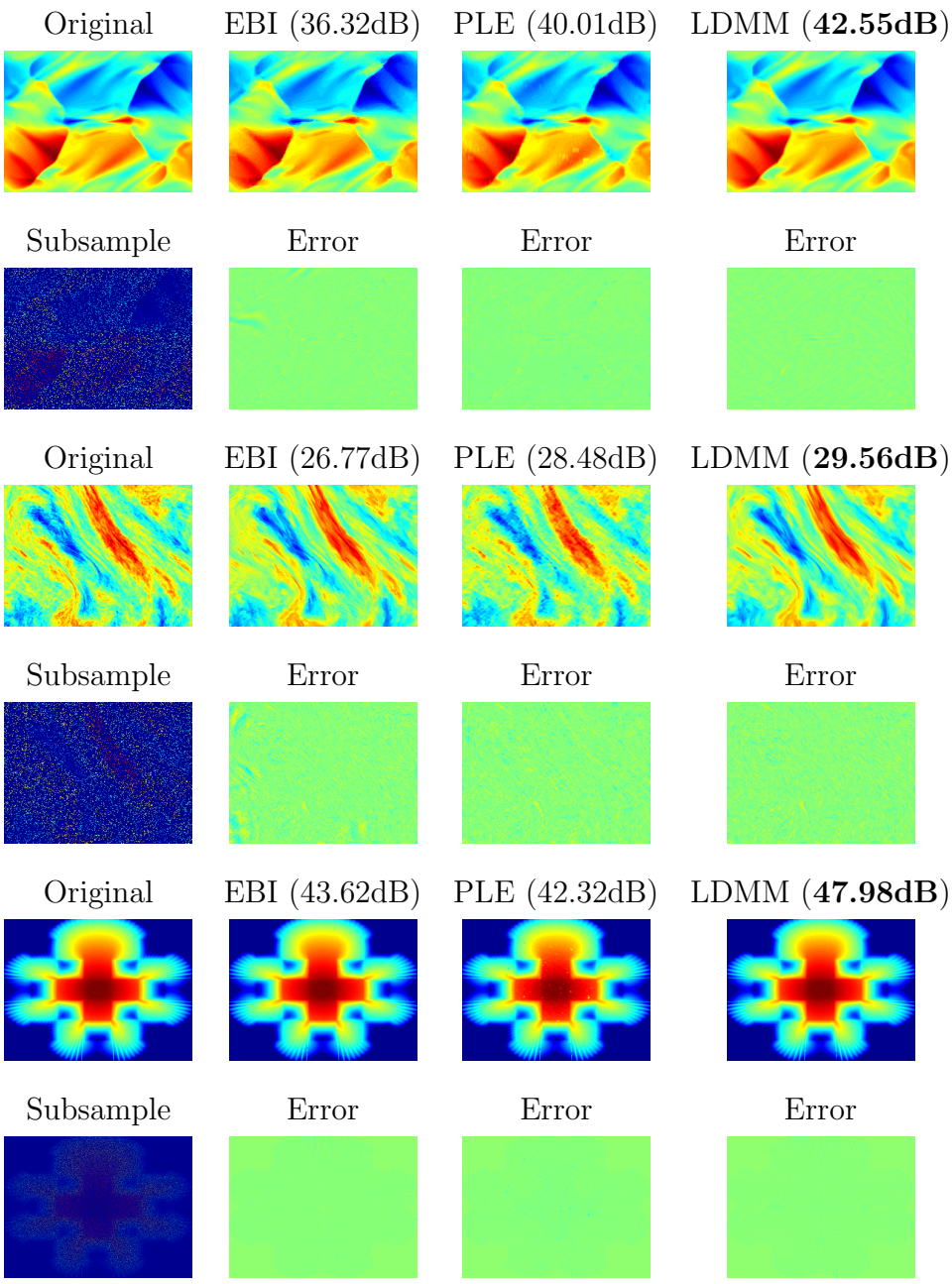


Figure 4.12: Interpolation of 2D scientific data from 10% random sampling. The figures in the first column are the original and subsampled data. The figures in the other three columns are the results and errors of the competing algorithms.

4.4.3.3 Interpolation with Regular Sampling

Unlike the random sampling interpolation in the previous section, reasonable initializations of LDMM can be obtained from other standard algorithms for regular sampling interpolation.

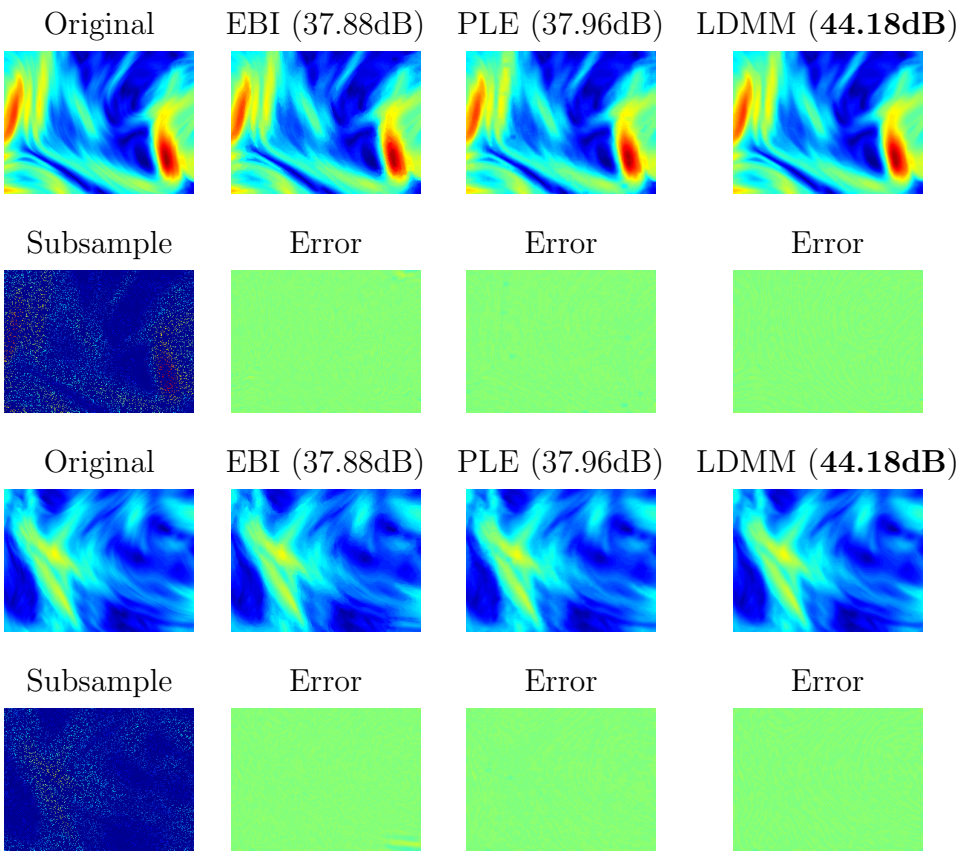


Figure 4.13: Interpolation of the 3D plasma (magnetic field) data from 10% random sampling. The figures in the first column are two spatial cross sections of the original and subsampled data. The figures in the other three columns are the results and errors of the competing algorithms.

10%	EBI	PLE	LDMM	5%	EBI	PLE	LDMM
L_1	0.0082	0.0053	**0.0034**	L_1	0.0148	0.0296	**0.0056**
L_2	0.0153	0.0100	**0.0075**	L_2	0.0270	0.0573	**0.0111**
L_∞	0.2280	**0.1232**	0.1376	L_∞	0.3327	0.7872	**0.1102**
PSNR	36.32	40.01	**42.55**	PSNR	31.36	24.84	**39.09**

Table 4.3: Errors of the interpolation of the 2D vortex dataset from 10% and 5% random sampling.

In the numerical experiments on all the datasets, the results of DCT and cubic spline have been used as the initial iterates for LDMM, and the final results of LDMM initialized with DCT (LDMM (D)) and cubic spline (LDMM (C)) are obtained after three iterations of manifold updates.

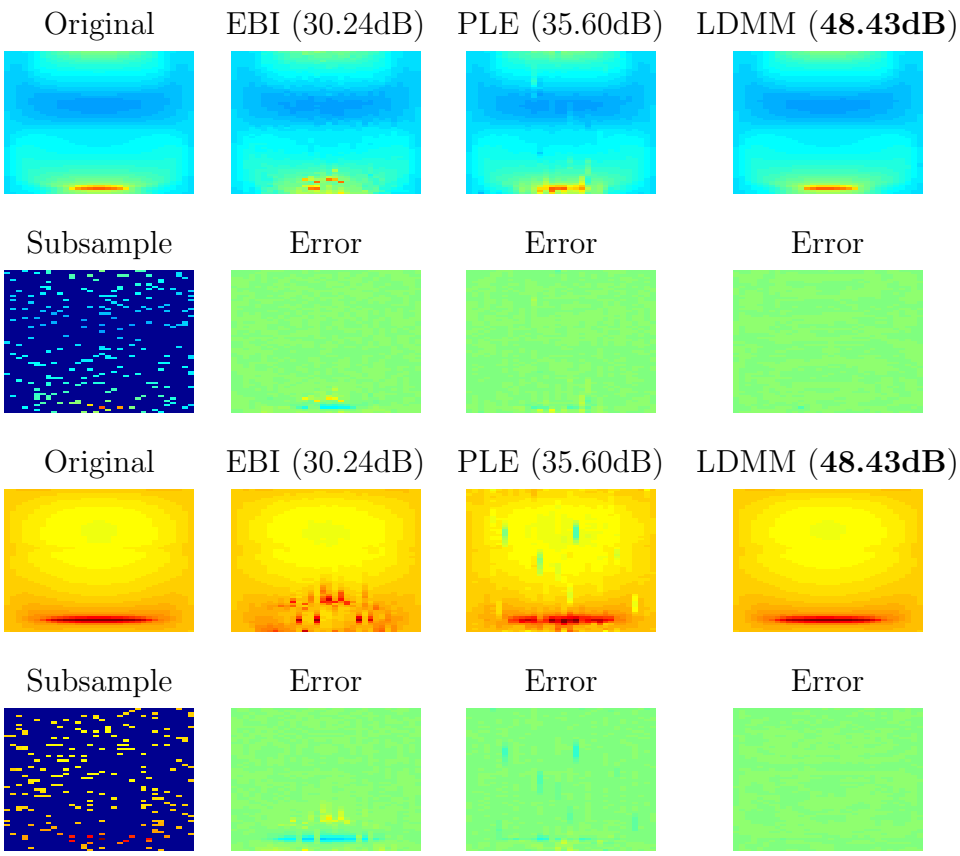


Figure 4.14: Interpolation of the 3D lattice data from 10% random sampling. The figures in the first column are the original and subsampled angular flux at $x = 0.24$ and $x = 1.18$. The figures in the other three columns are the results and errors of the competing algorithms.

10%	EBI	PLE	LDMM	5%	EBI	PLE	LDMM
L_1	0.0335	0.0272	**0.0243**	L_1	0.0393	0.0535	**0.0303**
L_2	0.0459	0.0377	**0.0333**	L_2	0.0522	0.0805	**0.0401**
L_∞	0.3782	0.2158	**0.1882**	L_∞	0.2588	0.7148	**0.2063**
PSNR	26.77	28.48	**29.56**	PSNR	25.65	21.88	**27.93**

Table 4.4: Errors of the interpolation of the 2D plasma (distribution function) dataset from 10% and 5% random sampling.

The visual rsults of the interpolation with regular sampling (4×4 for 2D data, $4 \times 4 \times 1$ and $2 \times 2 \times 2$ for 3D data) are shown in Figure 4.20 $\sim$ Figure 4.26. The errors in different norms are displayed in Table 4.9 $\sim$ Table 4.14. It can be observed that the results of

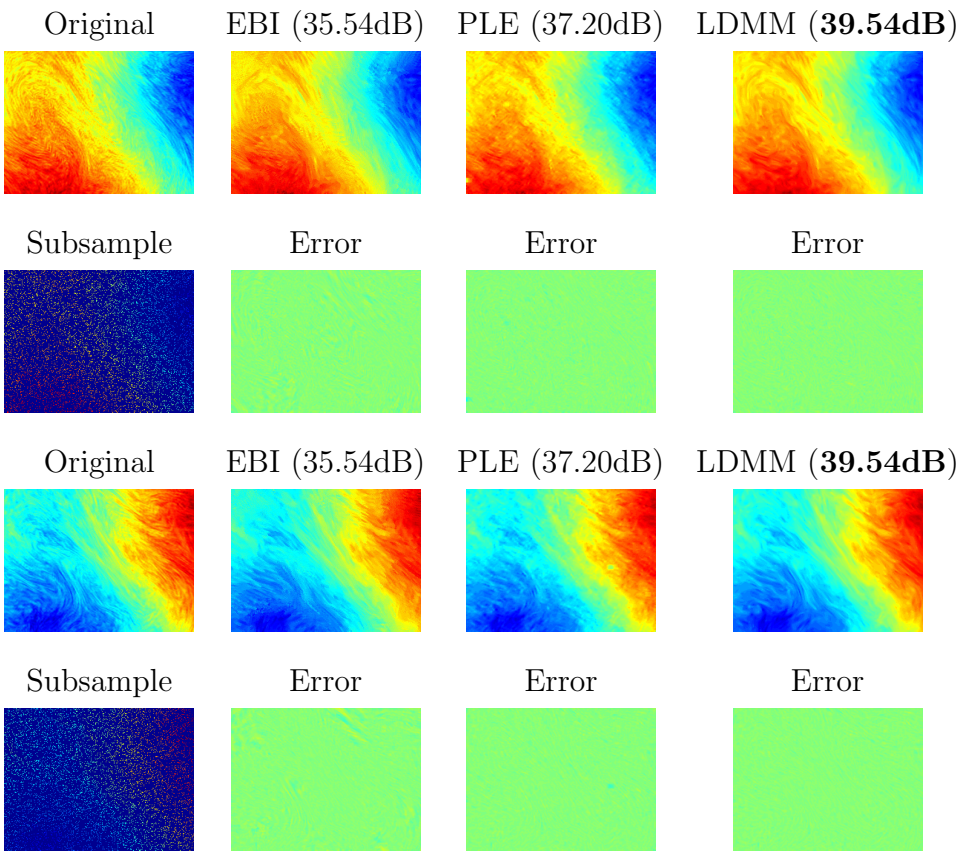


Figure 4.15: Interpolation of the 3D plasma (distribution function) data from 10% random sampling. The figures in the first column are two spatial cross sections of the original and subsampled data . The figures in the other three columns are the results and errors of the competing algorithms.

10%	EBI	PLE	LDMM	5%	EBI	PLE	LDMM
L_1	0.0033	0.0030	**0.0013**	L_1	0.0048	0.0187	**0.0022**
L_2	0.0066	0.0077	**0.0040**	L_2	0.0124	0.0442	**0.0062**
L_∞	0.2172	0.2889	**0.1979**	L_∞	0.8758	0.6156	**0.2097**
PSNR	43.62	42.32	**47.98**	PSNR	38.16	27.08	**44.15**

Table 4.5: Errors of the interpolation of the 2D lattice dataset from 10% and 5% random sampling.

LDMM are significantly more accurate than the DCT and cubic spline initializations, and the accuracy of the result does not depend on the choice of the initialization. Moreover, LDMM consistently outperforms all the other competing algorithms on every dataset.

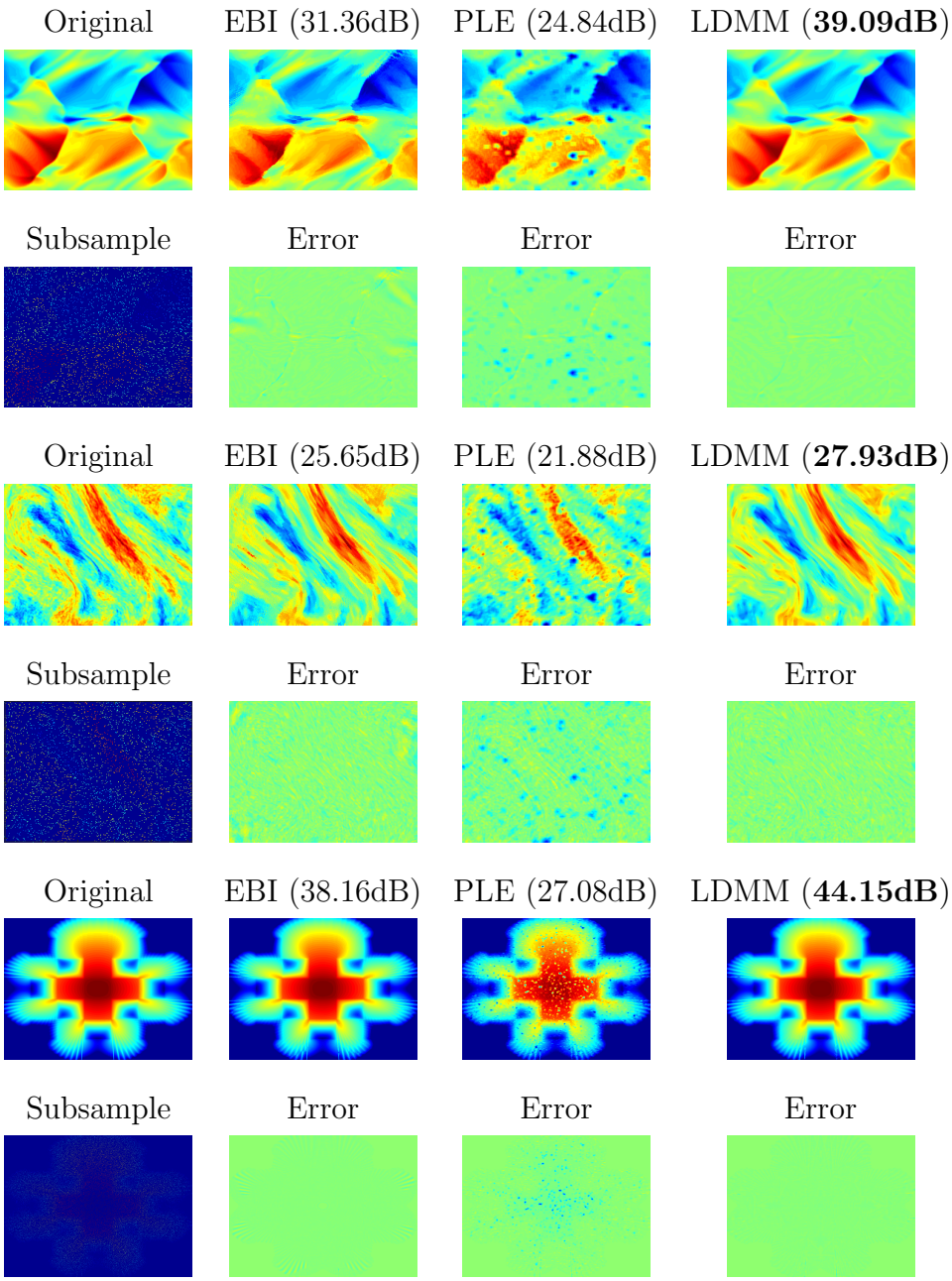


Figure 4.16: Interpolation of 2D scientific data from 5% random sampling. The figures in the first column are the original and subsampled data. The figures in the other three columns are the results and errors of the competing algorithms.

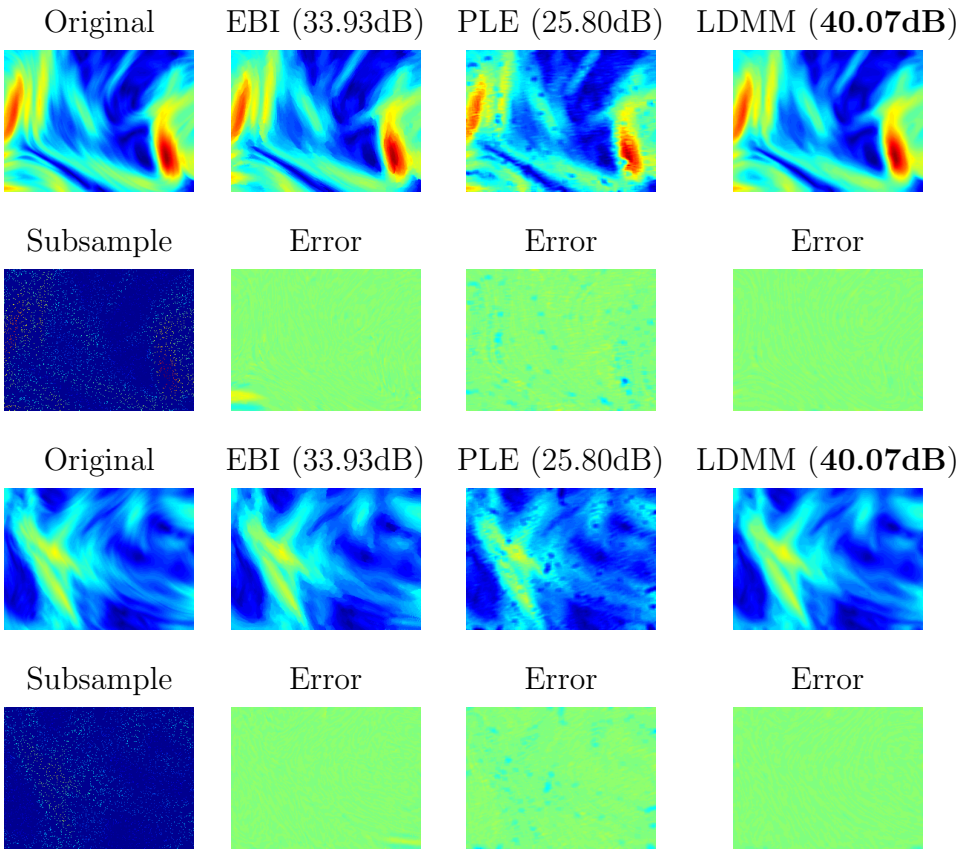


Figure 4.17: Interpolation of the 3D plasma (magnetic field) data from 5% random sampling. The figures in the first column are two spatial cross sections of the original and subsampled data . The figures in the other three columns are the results and errors of the competing algorithms.

10%	EBI	PLE	LDMM	5%	EBI	PLE	LDMM
L_1	0.0075	0.0053	**0.0038**	L_1	0.0115	0.0285	**0.0062**
L_2	0.0128	0.0126	**0.0062**	L_2	0.0201	0.0513	**0.0099**
L_∞	0.3510	0.9432	**0.1330**	L_∞	0.3740	0.7531	**0.2012**
PSNR	37.88	37.96	**44.18**	PSNR	33.93	25.80	**40.07**

Table 4.6: Errors of the interpolation of the 3D plasma (magnetic field) dataset from 10% and 5% random sampling.

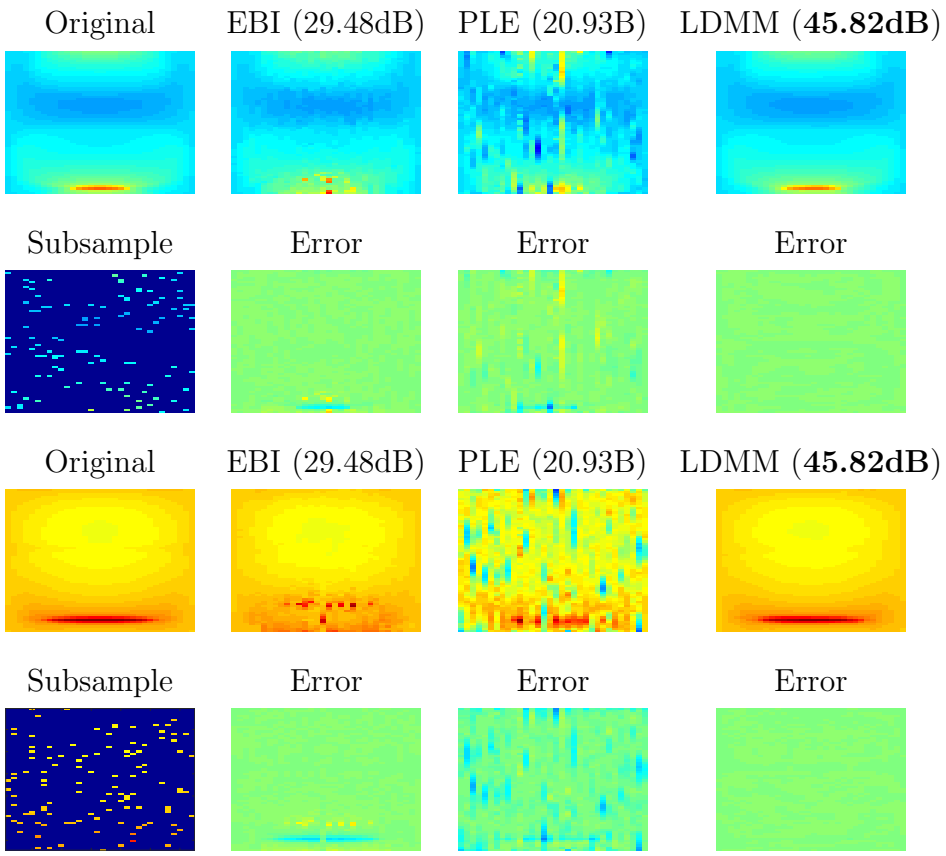


Figure 4.18: Interpolation of the 3D lattice data from 5% random sampling. The figures in the first column are the original and subsampled angular flux at $x = 0.24$ and $x = 1.18$. The figures in the other three columns are the results and errors of the competing algorithms.

10%	EBI	PLE	LDMM	5%	EBI	PLE	LDMM
L_1	0.0094	0.0062	**0.0008**	L_1	0.0112	0.0545	**0.0013**
L_2	0.0308	0.0166	**0.0038**	L_2	0.0336	0.0899	**0.0051**
L_∞	0.5291	0.6635	**0.4262**	L_∞	0.4768	0.0.8595	**0.4530**
PSNR	30.24	35.60	**48.43**	PSNR	29.48	20.93	**45.82**

Table 4.7: Errors of the interpolation of the 3D lattice dataset from 10% and 5% random sampling.

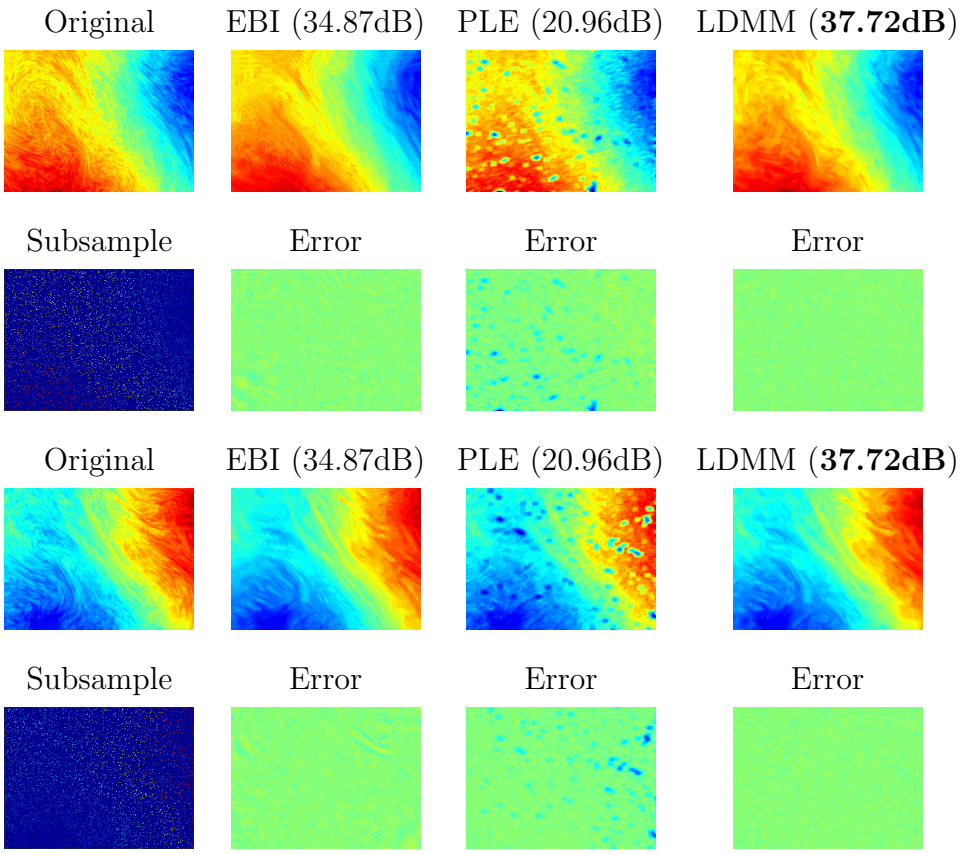


Figure 4.19: Interpolation of the 3D plasma (distribution function) data from 5% random sampling. The figures in the first column are two spatial cross sections of the original and subsampled data . The figures in the other three columns are the results and errors of the competing algorithms.

10%	EBI	PLE	LDMM	5%	EBI	PLE	LDMM
L_1	0.0098	0.0085	**0.0060**	L_1	0.0108	0.0593	**0.0075**
L_2	0.0167	0.0138	**0.0105**	L_2	0.0181	0.0895	**0.0130**
L_∞	0.2005	0.2912	**0.1181**	L_∞	0.1865	0.9093	**0.1793**
PSNR	35.54	37.20	**39.54**	PSNR	34.87	20.96	**37.72**

Table 4.8: Errors of the interpolation of the 3D plasma (distribution function) dataset from 10% and 5% random sampling.

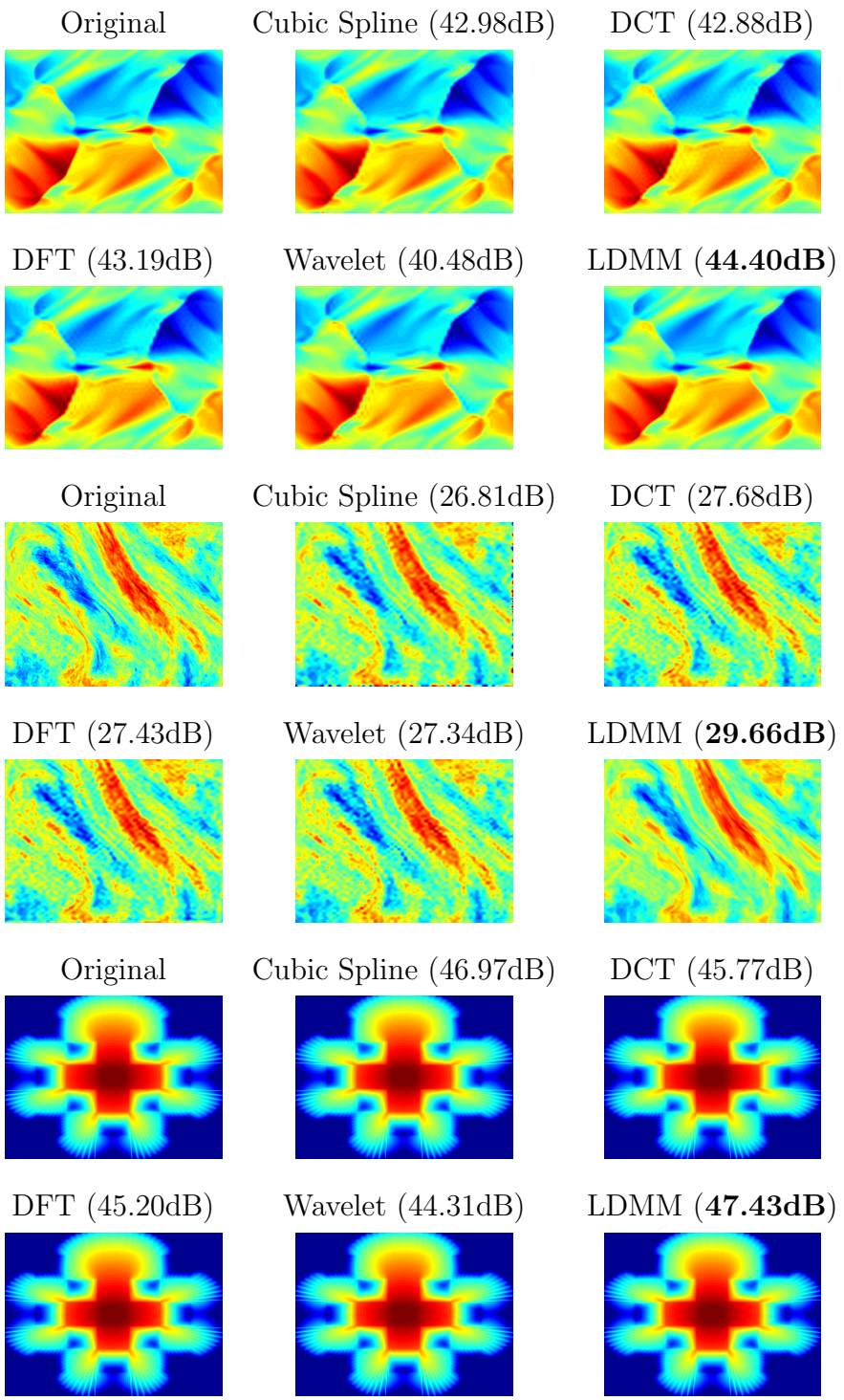


Figure 4.20: Interpolation of 2D scientific data from regular sampling with spacing 4×4. The original data are shown on the upper left corners for each dataset. The results of cubic spline, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

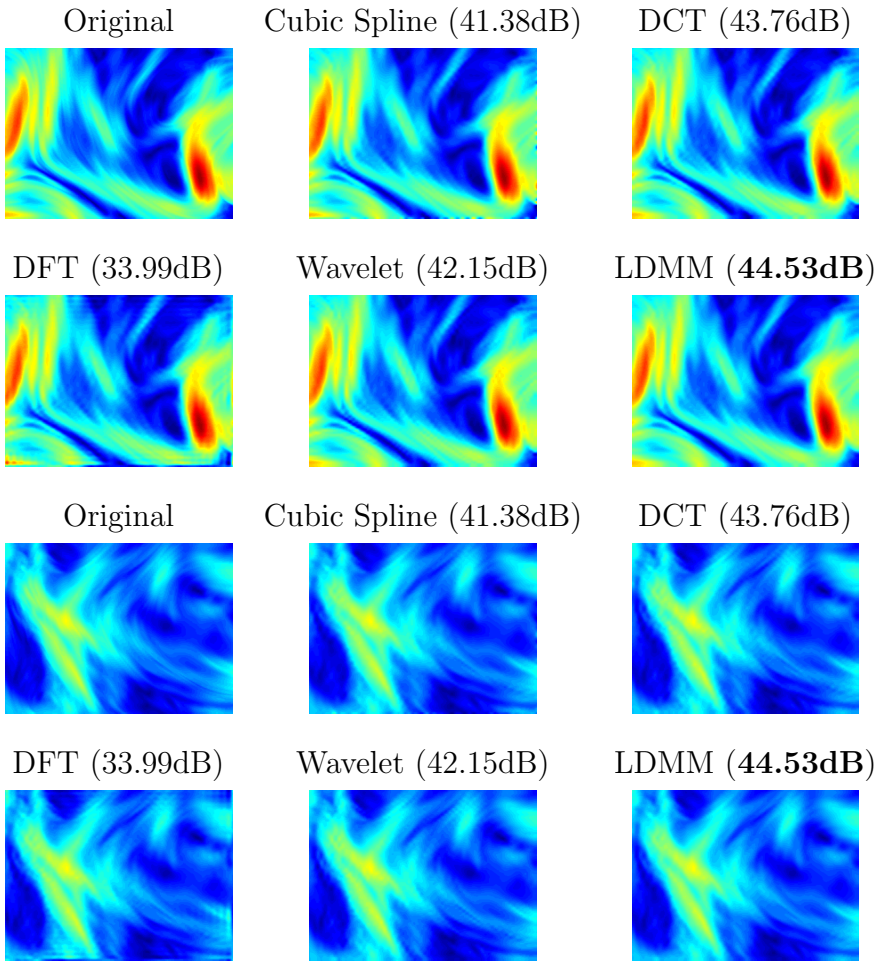


Figure 4.21: Interpolation of the 3D plasma (magnetic field) data from regular sampling with spacing $4 \times 4 \times 1$. Two spatial cross sections of the original data are shown in the first figures on the first and third row. The results of cubic spline, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

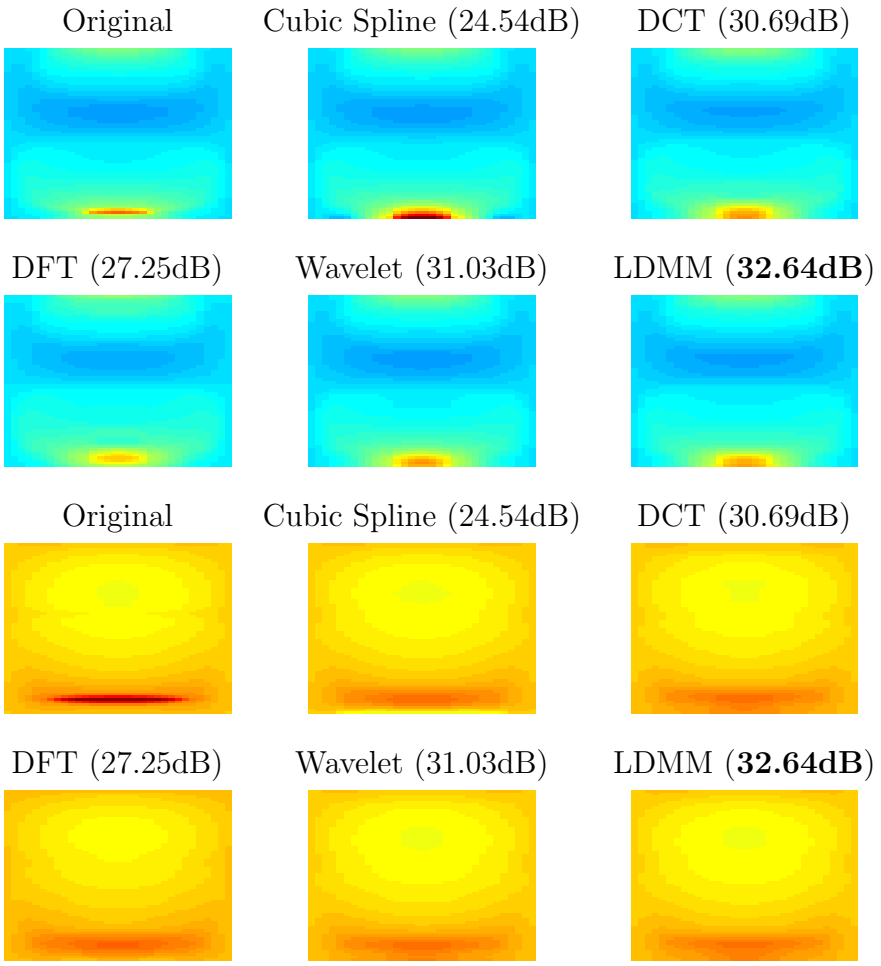


Figure 4.22: Interpolation of the 3D lattice data from regular sampling with spacing $4 \times 4 \times 1$. The original angular flux at $x = 0.24$ and $x = 1.18$ are shown in the first figures on the first and third row. The results of cubic spline, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

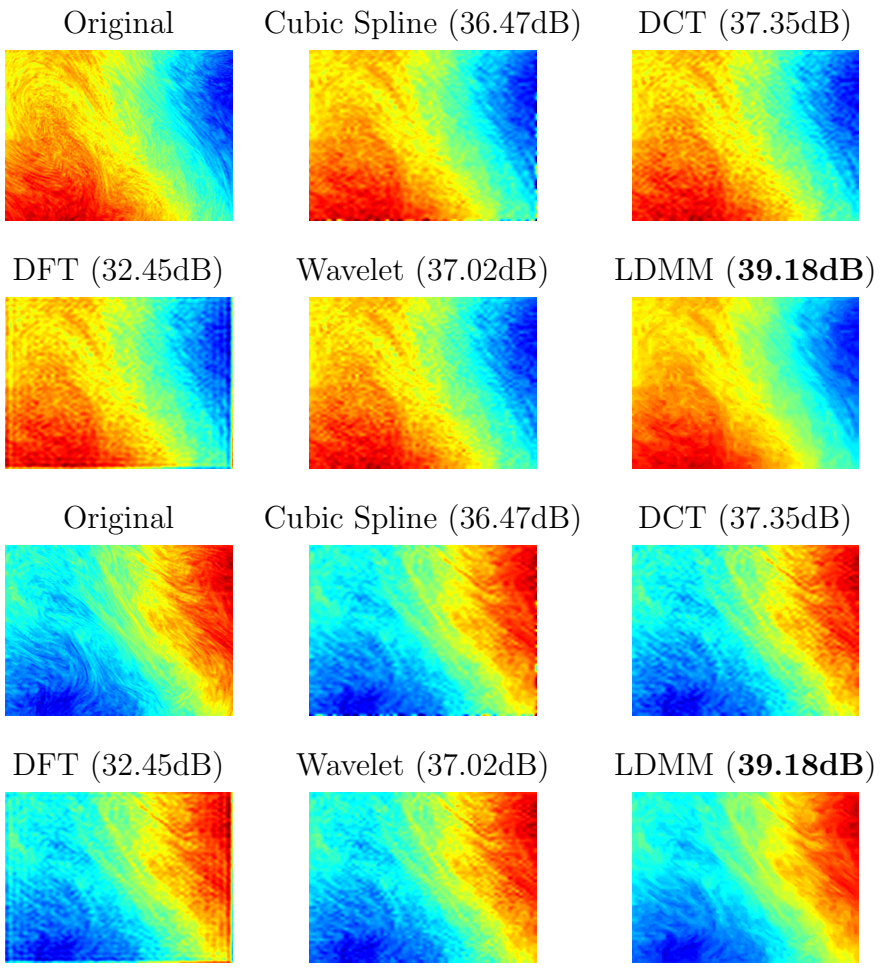


Figure 4.23: Interpolation of the 3D plasma (distribution function) data from regular sampling with spacing $4 \times 4 \times 1$. Two spatial cross sections of the original data are shown in the first figures on the first and third row. The results of cubic spline, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

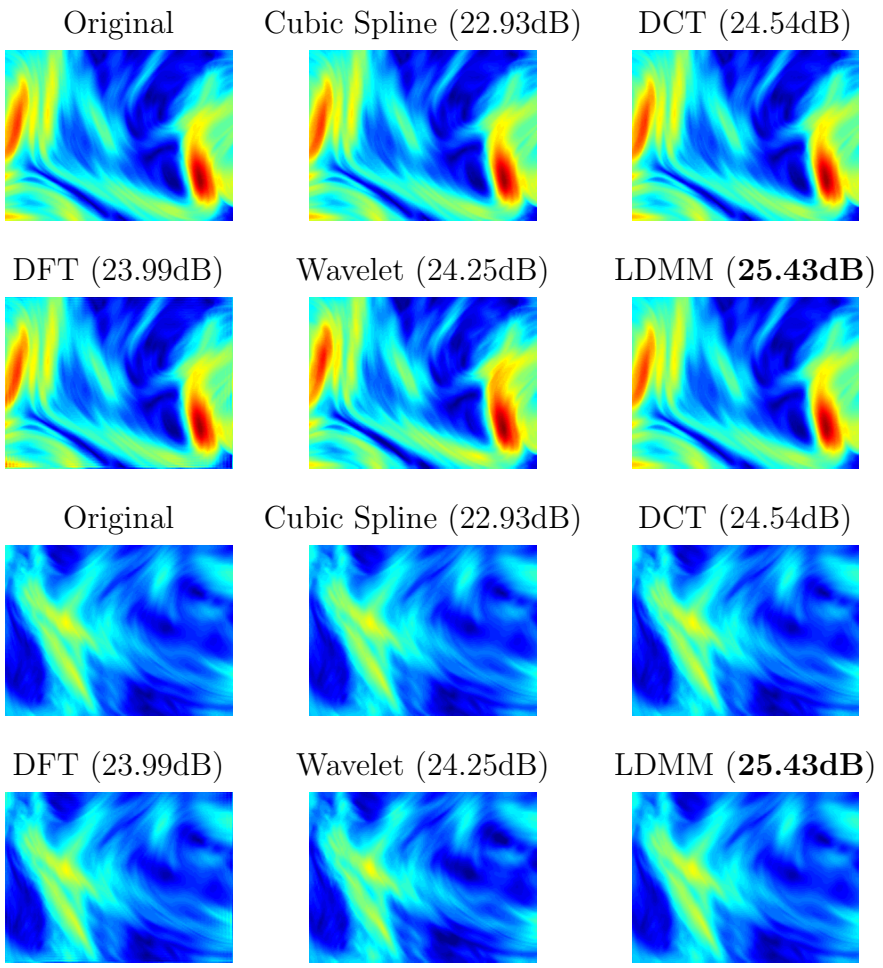


Figure 4.24: Interpolation of the 3D plasma (magnetic field) data from regular sampling with spacing $2 \times 2 \times 2$. Two spatial cross sections of the original data are shown in the first figures on the first and third row. The results of cubic spline, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

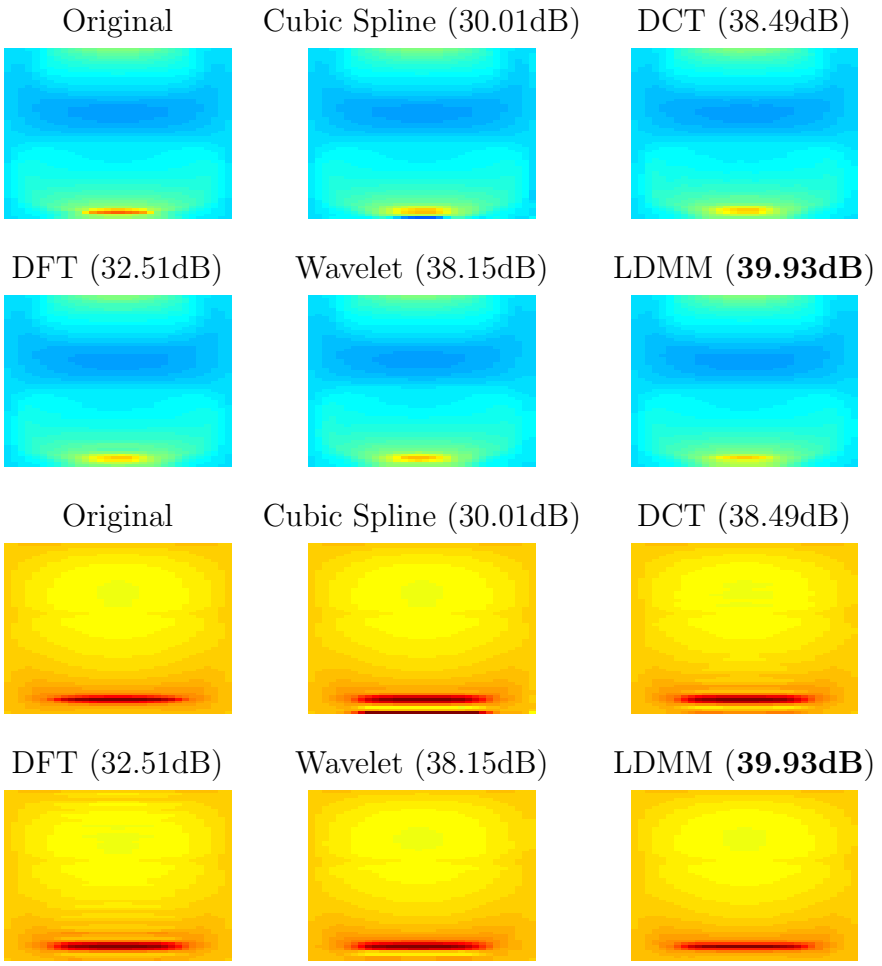


Figure 4.25: Interpolation of the 3D lattice data from regular sampling with spacing $2 \times 2 \times 2$. The original angular flux at $x = 0.24$ and $x = 1.18$ are shown in the first figures on the first and third row. The results of cubic spline, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

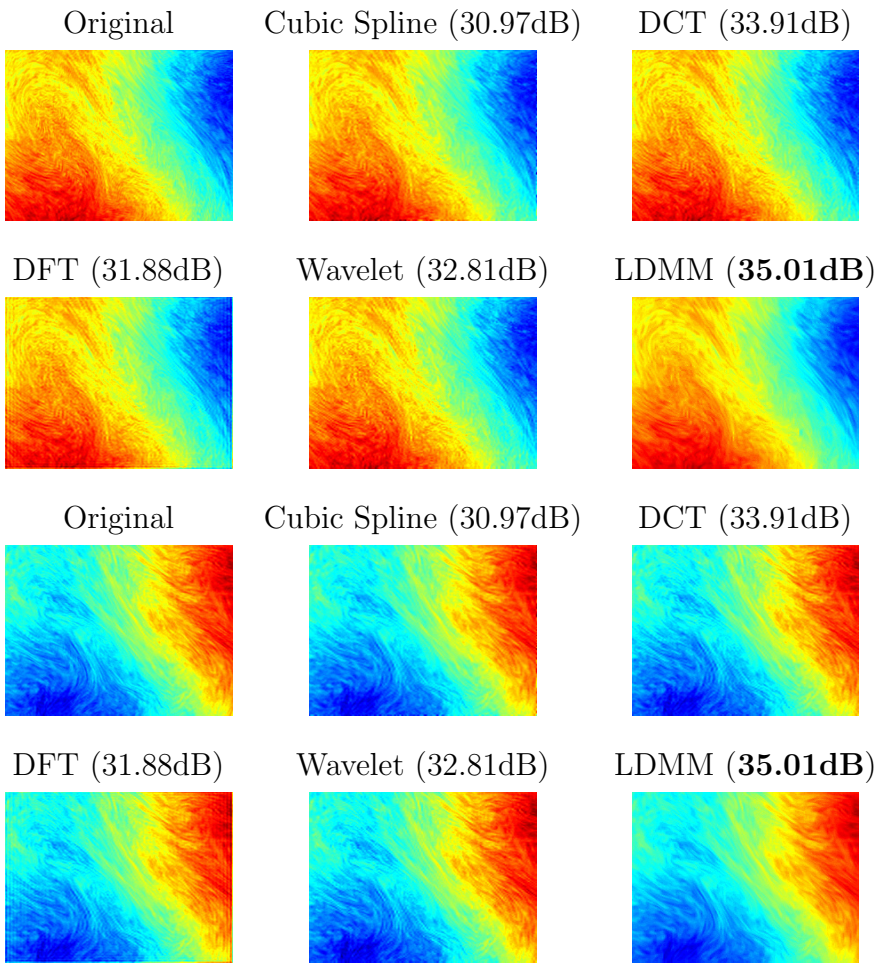


Figure 4.26: Interpolation of the 3D plasma (distribution function) data from regular sampling with spacing $2 \times 2 \times 2$. Two spatial cross sections of the original data are shown in the first figures on the first and third row. The results of cubic spline, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

4×4	Cubic	DCT	DFT	Wavelet	LDMM (D)	LDMM (C)
L_1	**0.0025**	0.0038	0.0035	0.0049	0.0029	0.0028
L_2	0.0071	0.0072	0.0069	0.0095	**0.0060**	0.0061
L_∞	0.1789	**0.0937**	0.0940	0.1122	0.0961	0.1005
PSNR	42.98	42.88	43.19	40.48	**44.40**	44.33

Table 4.9: Errors of the interpolation of the 2D vortex dataset from regular sampling with spacing 4×4.

4×4	Cubic	DCT	DFT	Wavelet	LDMM (D)	LDMM (C)
L_1	0.0302	0.0310	0.0314	0.0326	0.0249	**0.0248**
L_2	0.0456	0.0413	0.0425	0.0430	0.0329	**0.0329**
L_∞	0.7629	0.2411	0.3776	0.2514	0.1779	**0.1741**
PSNR	26.81	27.68	27.43	27.34	29.64	**29.66**

Table 4.10: Errors of the interpolation of the 2D plasma (distribution function) dataset from regular sampling with spacing 4×4.

4×4	Cubic	DCT	DFT	Wavelet	LDMM (D)	LDMM (C)
L_1	**0.0009**	0.0015	0.0016	0.0020	0.0013	0.0012
L_2	0.0045	0.0051	0.0055	0.0061	0.0044	**0.0041**
L_∞	0.1461	0.1547	0.2202	0.1892	0.1393	**0.1278**
PSNR	46.97	45.77	45.20	44.31	47.18	**47.43**

Table 4.11: Errors of the interpolation of the 2D lattice dataset from regular sampling with spacing 4×4.

4.4.3.4 Data Compression

Finally, we compare the performance of LDMM as a data compression technique to other standard compression methods including singular value/ Tucker Decomposition, DFT, DCT, and the wavelet transformations. We point out that, unlike the other testing methods which usually involve hard thresholding the transformed data under certain bases, LDMM does not require access to the original full data. Therefore we do not expect LDMM to perform equally well compared to other data compression methods. However, using sampling-based method

$4 \times 4 \times 1$	Cubic	DCT	DFT	Wavelet	LDMM (D)	LDMM (C)
L_1	0.0038	0.0040	0.0071	0.0052	0.0037	**0.0036**
L_2	0.0085	0.0065	0.0200	0.0078	**0.0059**	0.0065
L_∞	0.9649	0.1366	0.6449	0.1357	**0.1259**	0.1911
PSNR	41.38	43.76	33.99	42.15	**44.53**	43.73
$2 \times 2 \times 2$	Cubic	DCT	DFT	Wavelet	LDMM (D)	LDMM (C)
L_1	0.0356	0.0334	0.0352	0.0439	**0.0305**	0.0313
L_2	0.0714	0.0593	0.0632	0.0613	**0.0535**	0.0559
L_∞	0.8770	0.4073	0.5203	0.4283	**0.3711**	0.4060
PSNR	22.93	24.54	23.99	24.25	**25.43**	25.05

Table 4.12: Errors of the interpolation of the 3D plasma (magnetic field) dataset from regular sampling with spacing $4 \times 4 \times 1$ and $2 \times 2 \times 2$.

$4 \times 4 \times 1$	Cubic	DCT	DFT	Wavelet	LDMM (D)	LDMM (C)
L_1	0.0094	0.0072	0.0168	0.0066	0.0058	**0.0056**
L_2	0.0593	0.0292	0.0434	0.0281	**0.0233**	0.0254
L_∞	1.1890	0.4223	0.5405	0.4245	**0.4164**	0.4362
PSNR	24.54	30.69	27.25	31.03	**32.64**	31.90
$2 \times 2 \times 2$	Cubic	DCT	DFT	Wavelet	LDMM (D)	LDMM (C)
L_1	0.0039	0.0027	0.0069	0.0045	0.0017	**0.0015**
L_2	0.0316	0.0119	0.0237	0.0124	**0.0101**	0.0101
L_∞	0.7459	0.4109	0.4282	0.4233	**0.4078**	0.4096
PSNR	30.01	38.49	32.51	38.15	**39.93**	39.92

Table 4.13: Errors of the interpolation of the 3D lattice dataset from regular sampling with spacing $4 \times 4 \times 1$ and $2 \times 2 \times 2$.

as a data compression technique has the advantage of easy implementation during the data compressing step. Moreover, it is also faster for sampling-based method to reconstruct a small portion of the data if only that part of the data is required. LDMM with random

$4 \times 4 \times 1$	Cubic	DCT	DFT	Wavelet	LDMM (D)	LDMM (C)
L_1	0.0076	0.0078	0.0103	0.0083	**0.0064**	0.0064
L_2	0.0150	0.0136	0.0238	0.0141	**0.0110**	0.0111
L_∞	0.8851	0.1551	0.4805	0.1469	**0.1093**	0.1417
PSNR	36.47	37.35	32.45	37.02	**39.18**	39.13
$2 \times 2 \times 2$	Cubic	DCT	DFT	Wavelet	LDMM (D)	LDMM (C)
L_1	0.0109	0.0098	0.0127	0.0139	**0.0089**	0.0092
L_2	0.0283	0.0202	0.0255	0.0229	**0.0178**	0.0181
L_∞	0.7388	0.2976	0.3438	0.2993	**0.2088**	0.2097
PSNR	30.97	33.91	31.88	32.81	**35.01**	34.85

Table 4.14: Errors of the interpolation of the 3D plasma (distribution function) dataset from regular sampling with spacing $4 \times 4 \times 1$ and $2 \times 2 \times 2$.

sampling has been used in all the numerical experiments.

The visual and numerical results of the competing methods are reported in Figure 4.27 $\sim$ Figure 4.34 and Table 4.15 $\sim$ Table 4.20. As expected, the performance of LDMM in data compression is usually inferior compared to the other competing methods. However, it does outperform SVD in two of the more complicated 2D datasets (2D vortex and 2D plasma (distribution)) and the wavelet transform in the 3D plasma (magnetic field) dataset. DCT almost consistently yields the best result among all the methods, and it can also be observed that tensor decomposition methods tend to achieve better results when the dimension of the data becomes larger.

We point out that although LDMM does not perform equally well in data compression when compared to other methods that assume full access to the entire data, there is still much room for improvement for LDMM. For instance, instead of randomly sampling the data in the physical domain, we may strategically choosing pixels to sample if certain prior information is available. Moreover, if the original data is known to the user, we can also modify the LDMM algorithm by sampling gradient values or certain entries in the weight matrices. Modifing LDMM for it to work as a data compression method will be the focus of

10%	SVD	DCT	DFT	Wavelet	LDMM
L_1	0.0152	**0.0003**	0.0010	0.0005	0.0034
L_2	0.0208	**0.0005**	0.0014	0.0007	0.0075
L_∞	0.1357	**0.0056**	0.0132	0.0067	0.1376
PSNR	33.65	**66.87**	57.03	63.01	42.55
5%	SVD	DCT	DFT	Wavelet	LDMM
L_1	0.0295	**0.0011**	0.0024	0.0016	0.0056
L_2	0.0385	**0.0015**	0.0035	0.0021	0.0111
L_∞	0.1964	0.0154	0.0314	**0.0149**	0.1102
PSNR	28.29	**56.36**	49.12	53.36	39.09

Table 4.15: Errors of the compression of the 2D vortex dataset.

10%	SVD	DCT	DFT	Wavelet	LDMM
L_1	0.0345	**0.0131**	0.0147	0.0145	0.0243
L_2	0.0437	**0.0165**	0.0190	0.0182	0.0333
L_∞	0.2597	**0.0844**	0.1499	0.0861	0.1882
PSNR	27.19	**35.63**	34.44	34.78	29.56
5%	SVD	DCT	DFT	Wavelet	LDMM
L_1	0.0494	**0.0189**	0.0206	0.0202	0.0303
L_2	0.0624	**0.0238**	0.0263	0.0253	0.0401
L_∞	0.2794	0.1121	0.1920	**0.1057**	0.2063
PSNR	24.10	**32.47**	31.59	31.94	27.93

Table 4.16: Errors of the compression of the 2D plasma (distribution) dataset.

our future work.

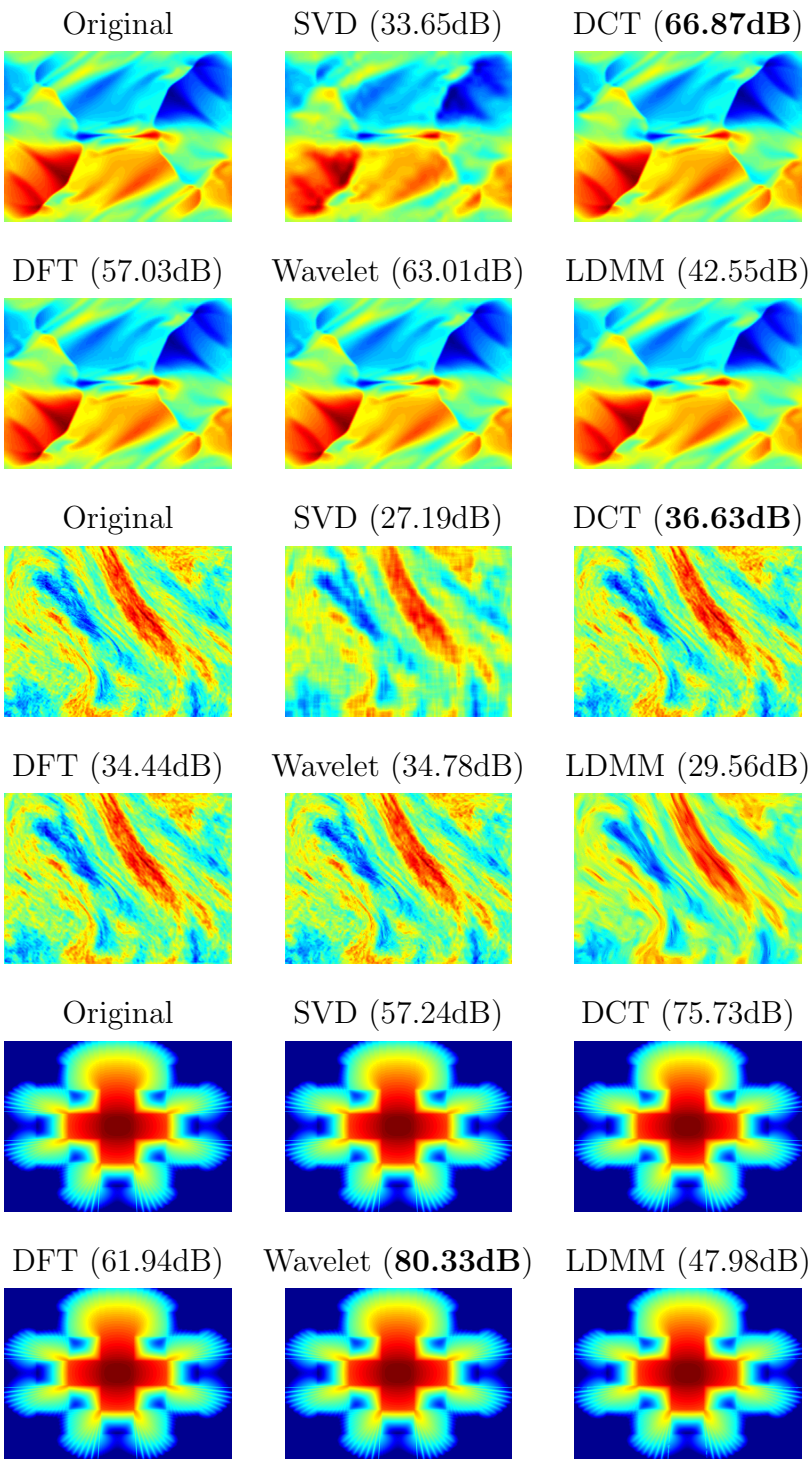


Figure 4.27: Compression of 2D scientific data with a 10% data compression rate. The original data are shown on the upper left corners for each dataset. The results of SVD, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

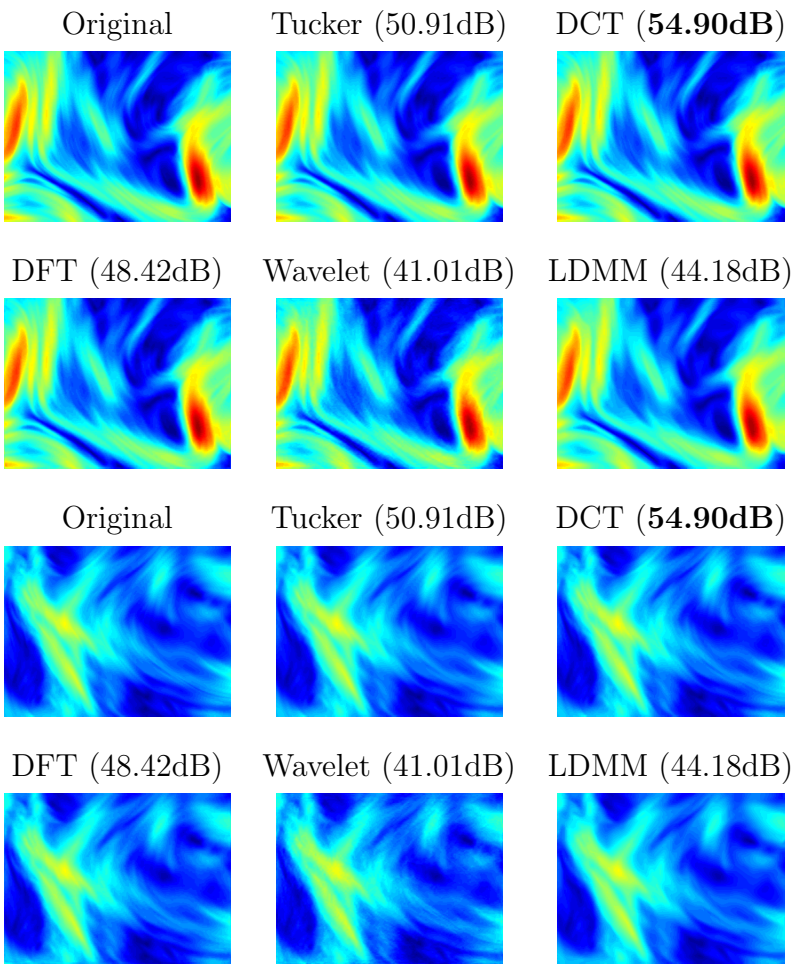


Figure 4.28: Compression of the 3D plasma (magnetic field) data with a 10% data compression rate. Two spatial cross sections of the original data are shown in the first figures on the first and third row. The results of Tucker decomposition, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

4.5 Conclusion

This chapter presents two numerical procedures to speed up the implementation of LDMM, making it feasible to deal with large images and high dimensional data. Solving the Laplace-Beltrami equation using the weighted graph Laplacian instead of the point integral method avoids solving over 100 linear systems in the patch domain every iteration, and the semi-local patches make the algorithm converge much faster in the first few iterations of the weight updates. Numerical experiments on (hyperspectral) image inpainting, denoising, and high dimensional data interpolation demonstate that LDMM is a powerful tool with much

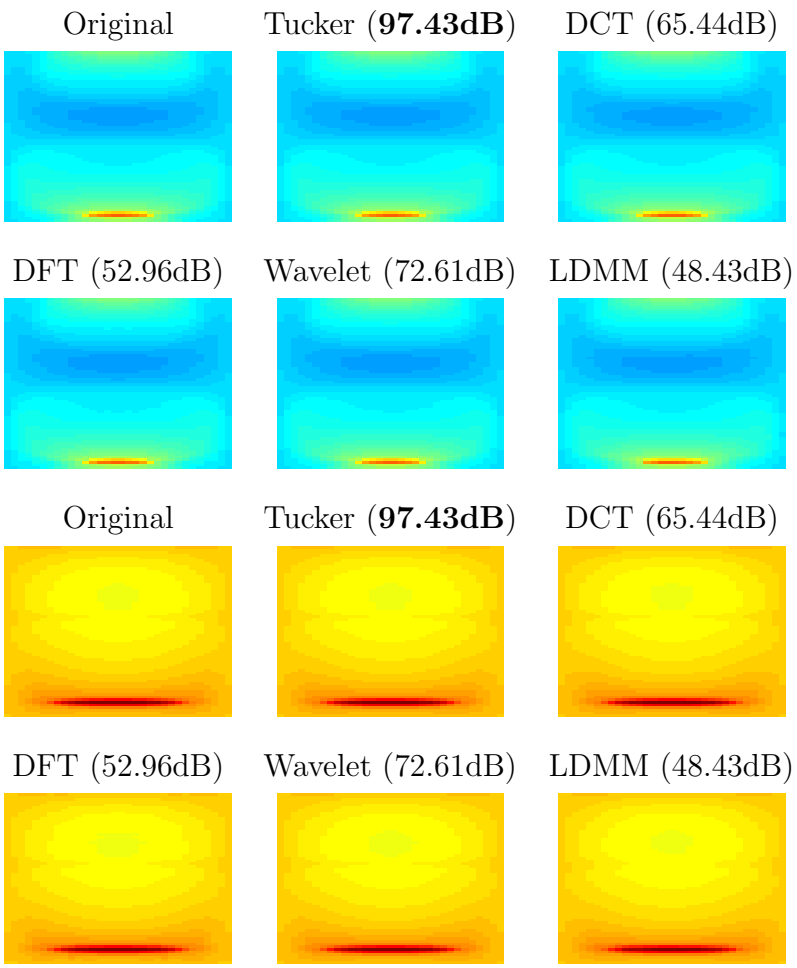


Figure 4.29: Compression of the 3D lattice data with a 10% data compression rate. The original angular flux at $x = 0.24$ and $x = 1.18$ are shown in the first figures on the first and third row. The results of Tucker decomposition, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

potential in image and data processing.

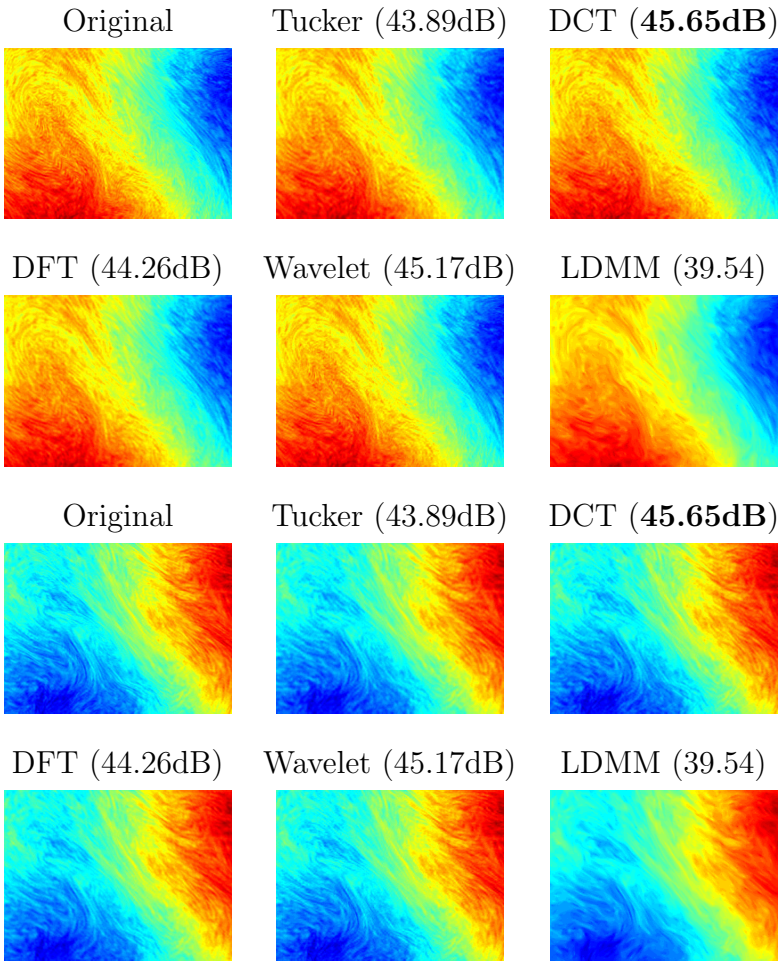


Figure 4.30: Compression of the 3D plasma (distribution function) data with a 10% data compression rate. Two spatial cross sections of the original data are shown in the first figures on the first and third row. The results of Tucker decomposition, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

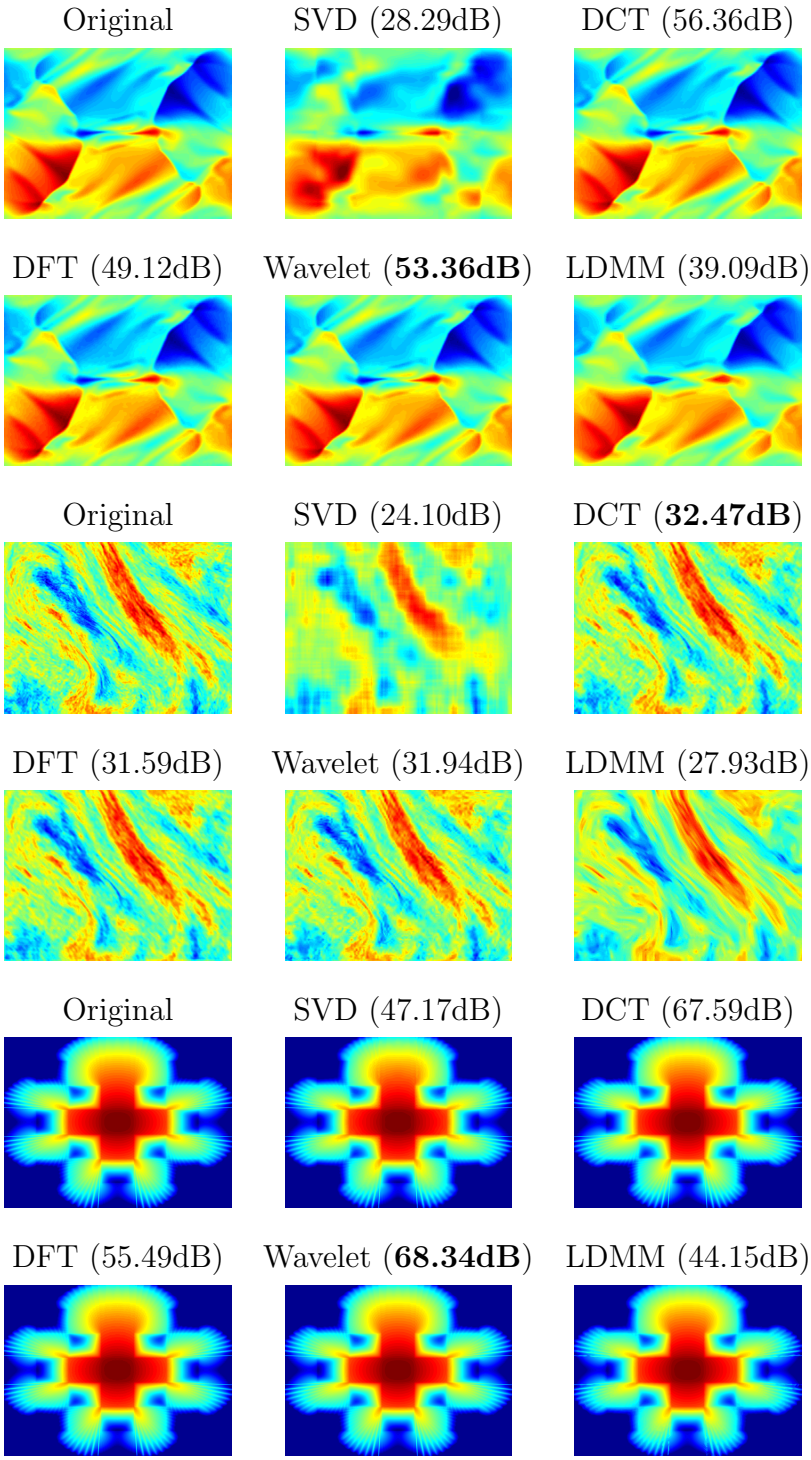


Figure 4.31: Compression of 2D scientific data with a 5% data compression rate. The original data are shown on the upper left corners for each dataset. The results of SVD, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

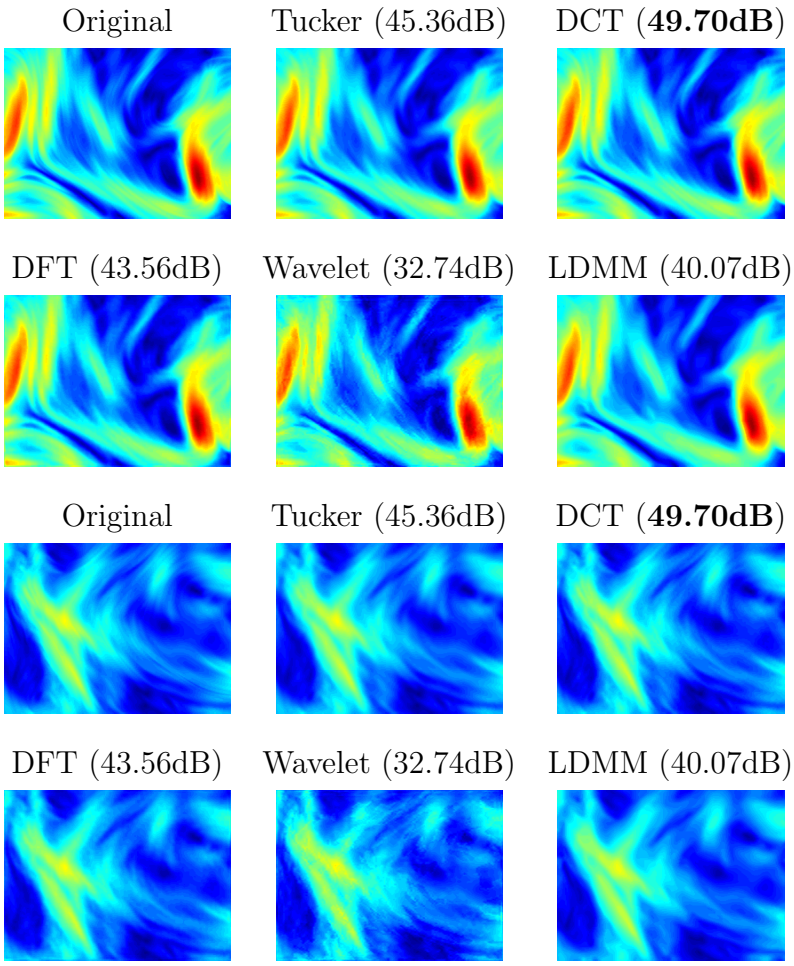


Figure 4.32: Compression of the 3D plasma (magnetic field) data with a 5% data compression rate. Two spatial cross sections of the original data are shown in the first figures on the first and third row. The results of Tucker decomposition, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

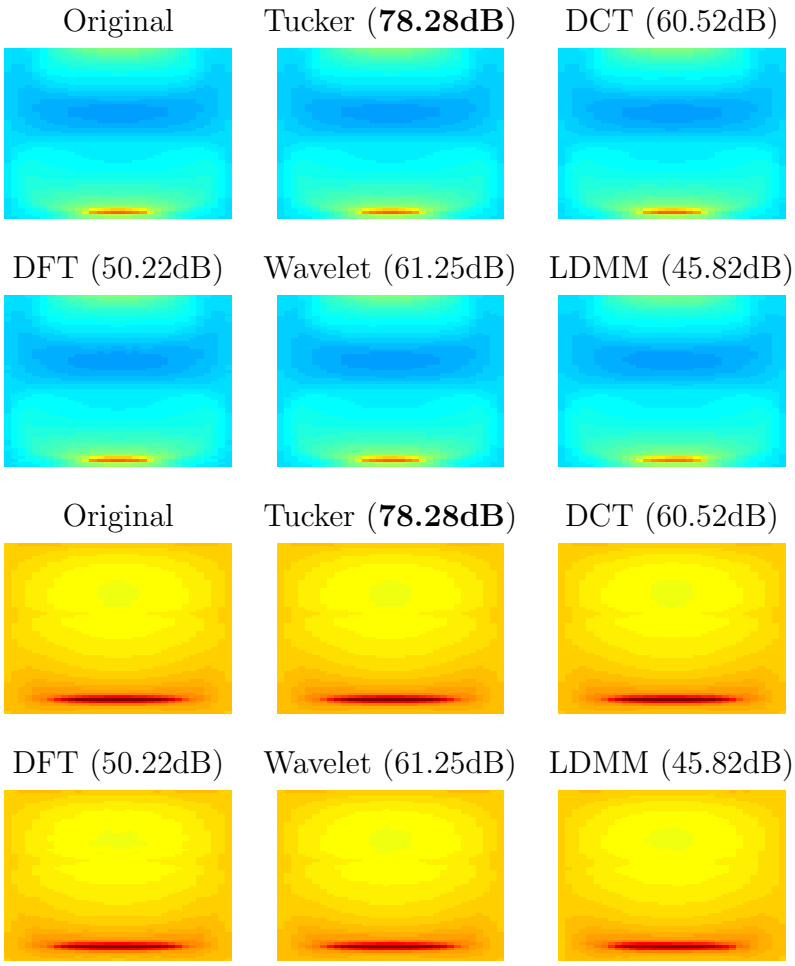


Figure 4.33: Compression of the 3D lattice data with a 5% data compression rate. The original angular flux at $x = 0.24$ and $x = 1.18$ are shown in the first figures on the first and third row. The results of Tucker decomposition, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

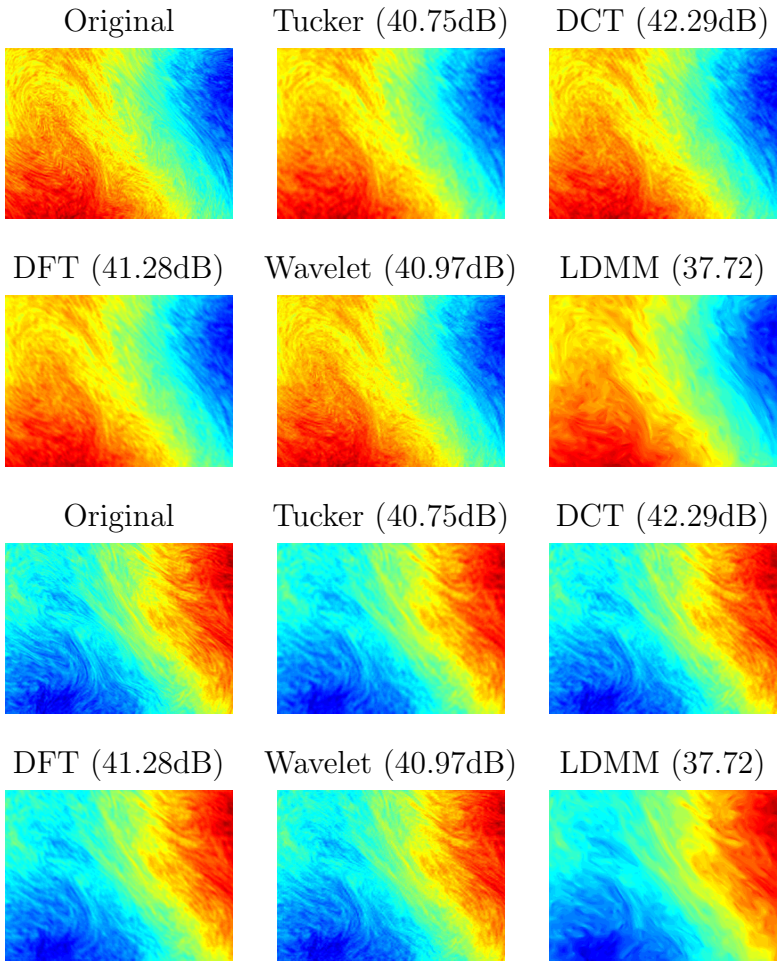


Figure 4.34: Compression of the 3D plasma (distribution function) data with a 5% data compression rate. Two spatial cross sections of the original data are shown in the first figures on the first and third row. The results of Tucker decomposition, DCT, DFT, wavelet, and LDMM are shown in the remaining five figures.

10%	SVD	DCT	FFT	Wavelet	LDMM
L_1	0.0009	**0.0001**	0.0004	0.0006	0.0013
L_2	0.0014	0.0002	0.0008	**0.0001**	0.0040
L_∞	0.0186	0.0101	0.0603	**0.0011**	0.1979
PSNR	57.24	75.73	61.94	**80.33**	47.98
5%	SVD	DCT	DFT	Wavelet	LDMM
L_1	0.0029	0.0003	0.0010	**0.0002**	0.0022
L_2	0.0044	0.0004	0.0017	**0.0004**	0.0062
L_∞	0.0539	0.0244	0.0743	**0.0049**	0.2097
PSNR	47.17	67.59	55.49	**68.34**	44.15

Table 4.17: Errors of the compression of the 2D lattice dataset.

10%	Tucker	DCT	DFT	Wavelet	LDMM
L_1	0.0021	**0.0014**	0.0024	0.0068	0.0038
L_2	0.0028	**0.0018**	0.0038	0.0089	0.0062
L_∞	0.0613	**0.0433**	0.1757	0.0739	0.1330
PSNR	50.91	**54.90**	48.42	41.01	44.18
5%	Tucker	DCT	DFT	Wavelet	LDMM
L_1	0.0040	**0.0025**	0.0043	0.0183	0.0062
L_2	0.0054	**0.0033**	0.0066	0.0231	0.0099
L_∞	0.0911	**0.0698**	0.2141	0.1558	0.2012
PSNR	45.36	**49.70**	43.56	32.74	40.07

Table 4.18: Errors of the compression of the 3D plasma (magnetic field) dataset.

10%	Tucker	DCT	DFT	Wavelet	LDMM
L_1	9×10^{-6}	0.0002	0.0007	0.0002	0.0008
L_2	1×10^{-5}	0.0005	0.0022	0.0002	0.0038
L_∞	**0.0002**	0.1338	0.2843	0.0020	0.4262
PSNR	**97.43**	65.44	52.96	72.61	48.43
5%	Tucker	DCT	DFT	Wavelet	LDMM
L_1	**0.0001**	0.0004	0.0010	0.0006	0.0013
L_2	**0.0001**	0.0009	0.0031	0.0008	0.0051
L_∞	**0.0042**	0.2053	0.4266	0.0095	0.4530
PSNR	**78.28**	60.52	50.22	61.25	45.82

Table 4.19: Errors of the compression of the 3D lattice dataset.

10%	Tucker	DCT	DFT	Wavelet	LDMM
L_1	0.0042	**0.0039**	0.0045	0.0042	0.0060
L_2	0.0064	**0.0052**	0.0061	0.0055	0.0105
L_∞	0.0637	0.0644	0.0837	**0.0373**	0.1181
PSNR	43.89	**45.65**	44.26	45.17	39.54
5%	Tucker	DCT	DFT	Wavelet	LDMM
L_1	0.0060	**0.0057**	0.0063	0.0067	0.0075
L_2	0.0092	**0.0077**	0.0086	0.0089	0.0130
L_∞	0.0890	0.0766	0.1018	**0.0660**	0.1793
PSNR	40.75	**42.29**	41.28	40.97	37.72

Table 4.20: Errors of the compression of the 3D plasma (distribution function) dataset.

REFERENCES

[AV07] David Arthur and Sergei Vassilvitskii. "K-means++: The Advantages of Careful Seeding." In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pp. 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[BCM05] 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.

[BCM06] A. Buades, B. Coll, and J.-M. Morel. "Neighborhood filters and PDE's." *Numer. Math.*, **105**:1–34, 2006.

[BF12] Andrea L. Bertozzi and Arjuna Flenner. "Diffuse Interface Models on Graphs for Classification of High Dimensional Data." *Multiscale Modeling & Simulation*, **10**(3):1090–1118, 2012.

[BH05] T. A. Brunner and J. P. Holloway. "Two-dimensional time dependent Riemann solvers for neutron transport." *Journal of Computational Physics*, **210**:386–399, 2005.

[BK06] Brett W. Bader and Tamara G. Kolda. "Algorithm 862: MATLAB tensor classes for fast algorithm prototyping." *ACM Transactions on Mathematical Software*, **32**(4):635–653, December 2006.

[BK15] Brett W. Bader, Tamara G. Kolda, et al. "MATLAB Tensor Toolbox Version 2.6." Available online, February 2015.

[BPD12] J.M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Qian Du, P. Gader, and J. Chanussot. "Hyperspectral Unmixing Overview: Geometrical, Statistical, and Sparse Regression-Based Approaches." *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, **5**(2):354–379, April 2012.

[Bro14] Russell A Brown. "Building a Balanced kd Tree in O (kn log n) Time." *arXiv preprint arXiv:1410.5420*, 2014.

[Bru02] Thomas A. Brunner. "Forms of approximate radiation transport." Technical Report SAND2002-1778, Sandia National Laboratories, 2002.

[BS10] Xavier Bresson and Arthur D Szlam. "Total variation, cheeger cuts." In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 1039–1046, 2010.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[CB05] G. Camps-Valls and L. Bruzzone. "Kernel-based methods for hyperspectral image classification." *Geoscience and Remote Sensing, IEEE Transactions on*, **43**(6):1351–1362, June 2005.

[CCHed] Michael M. Crockatt, Andrew J. Christlieb, Cory D. Hauck, and C. Kristopher Garrett. "An Arbitrary-Order, Fully Implicit, Hybrid Kinetic Solver for Linear Radiative Transport Using Integral Deferred Correction." *Journal of Computational Physics*, submitted.

[CEP05] Tony Chan, Selim Esedoglu, Frederick Park, and A Yip. "Recent developments in total variation image restoration." *Mathematical Models of Computer Vision*, **17**, 2005.

[CGM16] Manuel J. Castro, Jos M. Gallardo, and Antonio Marquina. "Approximate OsherSolomon schemes for hyperbolic systems." *Applied Mathematics and Computation*, **272, Part 2**:347 – 368, 2016. Recent Advances in Numerical Methods for Hyperbolic Partial Differential Equations.

[Cha03] Chein-I Chang. *Hyperspectral imaging: techniques for spectral detection and classification*, volume 1. Springer Science & Business Media, 2003.

[CIS08] G. Carlsson, T. Ishkhanov, V. de Silva, and A. Zomorodian. "On the local behavior of spaces of natural images." *International Journal of Computer Vision*, **76**:1–12, 2008.

[CP10] Antonin Chambolle and Thomas Pock. "A First-Order Primal-Dual Algorithm for Convex Problems withApplications to Imaging." *Journal of Mathematical Imaging and Vision*, **40**(1):120–145, 2010.

[DE07] B. Demir and S. Erturk. "Hyperspectral Image Classification Using Relevance Vector Machines." *Geoscience and Remote Sensing Letters, IEEE*, **4**(4):586–590, Oct 2007.

[DFK07] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. "Image denoising by sparse 3D transform-domain collaborative filtering." *IEEE Trans. Image Processing*, **16**:2080–2095, 2007.

[DTR14] N. Dobigeon, J.-Y. Tourneret, C. Richard, J.C.M. Bermudez, S. McLaughlin, and A.O. Hero. "Nonlinear Unmixing of Hyperspectral Images: Models and Algorithms." *Signal Processing Magazine, IEEE*, **31**(1):82–94, Jan 2014.

[EZC10] Ernie Esser, Xiaoqun Zhang, and Tony F. Chan. "A General Framework for a Class of First Order Primal-Dual Algorithms for Convex Optimization in Imaging Science." *SIAM Journal on Imaging Sciences*, **3**(4):1015–1046, 2010.

[FAC09] Gabriele Facciolo, Pablo Arias, Vicent Caselles, and Guillermo Sapiro. *Exemplar-Based Interpolation of Sparsely Sampled Images*, pp. 331–344. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[FBC08] M. Fauvel, J.A. Benediktsson, J. Chanussot, and J.R. Sveinsson. "Spectral and Spatial Classification of Hyperspectral Data Using SVMs and Morphological Profiles." *Geoscience and Remote Sensing, IEEE Transactions on*, **46**(11):3804–3814, Nov 2008.

[FBF77] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. "An Algorithm for Finding Best Matches in Logarithmic Expected Time." *ACM Trans. Math. Softw.*, **3**(3):209–226, September 1977.

[Foo08] G. M. Foody. "RVMbased multiclass classification of remotely sensed data." *International Journal of Remote Sensing*, **29**(6):1817–1823, 2008.

[GKP15] N. Gillis, Da Kuang, and Haesun Park. "Hierarchical Clustering of Hyperspectral Images Using Rank-Two Nonnegative Matrix Factorization." *Geoscience and Remote Sensing, IEEE Transactions on*, **53**(4):2066–2078, April 2015.

[GMB14] C. Garcia-Cardona, E. Merkurjev, A.L. Bertozzi, A. Flenner, and A.G. Percus. "Multiclass Data Segmentation Using Diffuse Interface Methods on Graphs." *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **36**(8):1600–1613, 2014.

[GO07] G. Gilboa and S. Osher. "Nonlocal linear image regularization and supervised segmentation." *Multiscale Model. Simul.*, **6**:595–630, 2007.

[GO09a] Guy Gilboa and Stanley Osher. "Nonlocal Operators with Applications to Image Processing." *Multiscale Modeling & Simulation*, **7**(3):1005–1028, 2009.

[GO09b] T. Goldstein and S. Osher. "The split Bregman method for L1-regularized problems." *SIAM J. Imageing Sci.*, **2**:323–343, 2009.

[GV14] N. Gillis and S.A. Vavasis. "Fast and Robust Recursive Algorithmsfor Separable Nonnegative Matrix Factorization." *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **36**(4):698–714, April 2014.

[HM13] Cory D Hauck and Ryan G McClarren. "A collision-based hybrid method for time-dependent, linear, kinetic transport equations." *Multiscale Modeling & Simulation*, **11**(4):1197–1227, 2013.

[HPG14] R. Heylen, M. Parente, and P. Gader. "A Review of Nonlinear Hyperspectral Unmixing Methods." *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, **7**(6):1844–1868, June 2014.

[HSB15] Huiyi Hu, Justin Sunu, and Andrea L. Bertozzi. *Energy Minimization Methods in Computer Vision and Pattern Recognition: 10th International Conference, EMMCVPR 2015, Hong Kong, China, January 13-16, 2015. Proceedings*, chapter Multi-class Graph Mumford-Shah Model for Plume Detection Using the MBO scheme, pp. 209–222. Springer International Publishing, Cham, 2015.

[JDK00] Frank Jenko, W Dorland, M Kotschenreuther, and BN Rogers. "Electron temperature gradient driven turbulence." *Physics of Plasmas*, **7**(5):1904–1910, 2000.

[JQ09] Sen Jia and Yuntao Qian. "Constrained Nonnegative Matrix Factorization for Hyperspectral Unmixing." *Geoscience and Remote Sensing, IEEE Transactions on*, **47**(1):161–173, Jan 2009.

[KP11] Jingu Kim and Haesun Park. "Fast Nonnegative Matrix Factorization: An Active-Set-Like Method and Comparisons." *SIAM Journal on Scientific Computing*, **33**(6):3261–3281, 2011.

[LPM03] Ann B. Lee, Kim S. Pedersen, and David Mumford. "The Nonlinear Statistics of High-Contrast Patches in Natural Images." *International Journal of Computer Vision*, **54**(1):83–103, 2003.

[LSS] Zhen Li, Zuoqiang Shi, and Jian Sun. "Point Integral Method for Solving Poisson-type Equations on Manifolds from Point Clouds with Convergence Guarantees." *arXiv:1409.2623*.

[Mac03] David MacKay. "An Example Inference Task: Clustering." In *Information Theory, Inference and Learning Algorithms*, chapter 20, pp. 284–292. Cambridge University Press, 2003.

[MB04] F. Melgani and L. Bruzzone. "Classification of hyperspectral remote sensing images with support vector machines." *Geoscience and Remote Sensing, IEEE Transactions on*, **42**(8):1778–1790, Aug 2004.

[MBO94] Barry Merriman, James K. Bence, and Stanley J. Osher. "Motion of Multiple Junctions: A Level Set Approach." *Journal of Computational Physics*, **112**(2):334 – 363, 1994.

[MH10a] Ryan G. McClarren and C. D. Hauck. "Robust and Accurate Filtered Spherical Harmonics Expansions for Radiative Transfer." *Journal of Computational Physics*, **229**:5597–5614, 2010.

[MH10b] Ryan G. McClarren and C. D. Hauck. "Simulating radiative transfer with filtered spherical harmonics." *Physics Letters A*, **374**:2290–2296, Jan 2010.

[ML09] Marius Muja and David G Lowe. "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration." *VISAPP (1)*, **2**:331–340, 2009.

[MS89] David Mumford and Jayant Shah. "Optimal approximations by piecewise smooth functions and associated variational problems." *Communications on pure and applied mathematics*, **42**(5):577–685, 1989.

[MSB14] E. Merkurjev, J. Sunu, and A.L. Bertozzi. "Graph MBO method for multiclass segmentation of hyperspectral stand-off detection video." In *Image Processing (ICIP), 2014 IEEE International Conference on*, pp. 689–693, 2014.

[mul14] "Diffuse interface methods for multiclass segmentation of high-dimensional data." *Applied Mathematics Letters*, **33**:29 – 34, 2014.

[MZ11] F.A. Mianji and Ye Zhang. "Robust Hyperspectral Classification Using Relevance Vector Machine." *Geoscience and Remote Sensing, IEEE Transactions on*, **49**(6):2100–2112, June 2011.

[OBG05] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. "An iterative regularization method for total variation-based image restoration." *Multiscale Model. Simul.*, **4**:460–489, 2005.

[OSZ16] Stanley Osher, Zuoqiang Shi, and Wei Zhu. "Low dimensional manifold model for image processing." Technical report, Technical Report, CAM report 16-04, UCLA, 2016.

[OT79] Steven A Orszag and Cha-Mei Tang. "Small-scale structure of two-dimensional magnetohydrodynamic turbulence." *Journal of Fluid Mechanics*, **90**(01):129–143, 1979.

[PBC08] Gabriel Peyré, Sébastien Bougleux, and Laurent Cohen. *Non-local Regularization of Inverse Problems*, pp. 57–68. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[Pey08] Gabriel Peyré. "Image Processing with Nonlocal Spectral Bases." *Multiscale Modeling & Simulation*, **7**(2):703–730, 2008.

[Pey09] Gabriel Peyré. "Manifold models for signals and images." *Computer Vision and Image Understanding*, **113**(2):249 – 260, 2009.

[ROF92] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. "Nonlinear total variation based noise removal algorithms." *Physica D: Nonlinear Phenomena*, **60**(14):259 – 268, 1992.

[SB97] Stephen M. Smith and J. Michael Brady. "SUSAN—A New Approach to Low Level Image Processing." *International Journal of Computer Vision*, **23**(1):45–78, 1997.

[SB09] Arthur Szlam and Xavier Bresson. "A Total Variation-based Graph Clustering Algorithm for Cheeger Ratio Cuts." Technical Report UCLA CAM Report 09-68, 2009.

[SBB07] Marine Soret, Stephen L Bacharach, and Irene Buvat. "Partial-volume effect in PET tumor imaging." *Journal of Nuclear Medicine*, **48**(6):932–945, 2007.

[SFL09] M. Schaefer, M. Frank, and C. D. Levermore. "Diffusive corrections to P_N approximations." *Multiscale Model. Simul.*, **9**:1–28, 2009.

[Shi] Zuoqiang Shi. "Point Integral Method for elliptic equation with isotropic coefficients with convergence guarantee." *arXiv:1506.03606*.

[SKS07] A. Spira, R. Kimmel, and N. Sochen. "A Short- Time Beltrami Kernel for Smoothing Images and Manifolds." *IEEE Transactions on Image Processing*, **16**(6):1628–1636, June 2007.

[SM00] Jianbo Shi and J. Malik. "Normalized cuts and image segmentation." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(8):888–905, Aug 2000.

[SOZ16] Zuoqiang Shi, Stanley Osher, and Wei Zhu. "Low dimensional manifold model with semi-local patches." Technical report, Technical Report, CAM report 16-63, UCLA, 2016.

[SOZ17] Zuoqiang Shi, Stanley Osher, and Wei Zhu. "Weighted Nonlocal Laplacian on Interpolation from Sparse Data." *Journal of Scientific Computing*, pp. 1–14, 2017.

[SSa] Zuoqiang Shi and Jian Sun. "Convergence of the Point Integral method for the Poisson equation on manifolds I: the Neumann Boundary." *arXiv:1403.2141*.

[SSb] Zuoqiang Shi and Jian Sun. "Convergence of the Point Integral Method for the Poisson Equation on Manifolds II: the Dirichlet Boundary." *arXiv:1312.4424*.

[SW97] Mechthild Stoer and Frank Wagner. "A Simple Min-cut Algorithm." *J. ACM*, **44**(4):585–591, July 1997.

[SZO16] Zuoqiang Shi, Wei Zhu, and Stanley Osher. "Low dimensional manifold model in hyperspectral image reconstruction." *arXiv preprint arXiv:1605.05652*, 2016.

[TJT15] D. Told, F. Jenko, J. M. TenBarge, G. G. Howes, and G. W. Hammett. "Multiscale Nature of the Dissipation Range in Gyrokinetic Simulations of Alfvénic Turbulence." *Phys. Rev. Lett.*, **115**:025003, Jul 2015.

[TM98] C. Tomasi and R. Manduchi. "Bilateral filtering for gray and color images." In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pp. 839–846, Jan 1998.

[YSM12] 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, May 2012.

[ZC08] Mingqiang Zhu and Tony Chan. "An efficient primal-dual hybrid gradient algorithm for total variation image restoration." Technical Report UCLA CAM Report 08-34, 2008.

[ZCP12] M. Zhou, H. Chen, J. Paisley, L. Ren, L. Li, Z. Xing, D. Dunson, G. Sapiro, and L. Carin. "Nonparametric Bayesian Dictionary Learning for Analysis of Noisy and Incomplete Images." *IEEE Transactions on Image Processing*, **21**(1):130–144, Jan 2012.

[ZCT17] W. Zhu, V. Chayes, A. Tiard, S. Sanchez, D. Dahlberg, A. L. Bertozzi, S. Osher, D. Zosso, and D. Kuang. "Unsupervised Classification in Hyperspectral Imagery With Nonlocal Total Variation and Primal-Dual Hybrid Gradient Algorithm." ©IEEE. Reprinted, with permission, from *IEEE Transactions on Geoscience and Remote Sensing*, **55**(5):2786–2798, May 2017.

[ZTO15] Dominique Zosso, Giang Tran, and Stanley J. Osher. "Non-Local Retinex—A Unifying Framework and Beyond." *SIAM Journal on Imaging Sciences*, **8**(2):787–826, 2015.

[ZWB] Wei Zhu, Bao Wang, Richard Barnard, Cory Hauck, Frank Jenko, and Stanley Osher. "Scientific Data Interpolation with Low Dimensional Manifold Model." *In preparation.*

[ZWC08] Mingqiang Zhu, Stephen J. Wright, and Tony F. Chan. "Duality-based algorithms fortotal-variation-regularized image restoration." *Computational Optimization and Applications*, **47**(3):377–400, 2008.

[ZWX14] Feiyun Zhu, Ying Wang, Shiming Xiang, Bin Fan, and Chunhong Pan. "Structured Sparse Method for Hyperspectral Unmixing." {*ISPRS*} *Journal of Photogrammetry and Remote Sensing*, **88**:101 – 118, 2014.

[ZZW12] Junzhe Zhang, Wenquan Zhu, Lingli Wang, and Nan Jiang. "Evaluation of similarity measure methods for hyperspectral remote sensing data." In *Geoscience and Remote Sensing Symposium (IGARSS), 2012 IEEE International*, pp. 4138–4141, July 2012.