Wavelets for Graphs and their Deployment to Image Processing

Idan Ram

Wavelets for Graphs and their Deployment to Image Processing

Research Thesis

As Partial Fulfillment of the Requirements for
the Degree Doctor of Philosophy

Idan Ram

Submitted to the Senate of the Technion—Israel Institute of Technology

AV 5774 Haifa August 2014

The Research Thesis Was Done Under The Supervision of
Professor Israel Cohen (Department of Electrical Engineering) and
Professor Michael Elad (Department of Computer Science)
in the Department of Electrical Engineering.

Acknowledgement

I wish to express my deep gratitude and appreciation to Prof. Israel Cohen and Prof. Michael Elad for their guidance and dedicated supervision. I greatly appreciate their professional support, encouragement to perfection and contribution to my personal development.

I would like to thank the staff of the SIPL laboratory for their care and support that made many of the experiments reported in this thesis possible.

Finally, I would like to express my deep gratitude to my wife Sigal, my children Hila and Alon, and my parents Ruthy and Zeev, for their constant love, support and encouragement.

The Generous Financial Help of the Technion, and the Israel Science Foundation (grant no. 1130/11) is Gratefully Acknowledged.

List of Publications

[1] I. Ram, I. Cohen, and M. Elad. Facial Image Compression using Patch-Ordering-Based Adaptive Wavelet Transform. *IEEE Signal Processing Letters*, 21(10):1270–1274, 2014.

[2] I. Ram, I. Cohen, and M. Elad. Patch-ordering-based wavelet frame and its use in inverse problems. *IEEE Trans. Image Processing*, 33(7):2779–2792, 2014.

[3] I. Ram, M. Elad, and I. Cohen. Generalized Tree-Based Wavelet Transform . *IEEE Trans. Signal Processing*, 59(9):4199–4209, 2011.

[4] I. Ram, M. Elad, and I. Cohen. Redundant Wavelets on Graphs and High Dimensional Data Clouds. *IEEE Signal Processing Letters*, 19(5):291–294, 2012.

[5] I. Ram, M. Elad, and I. Cohen. Image denoising using nl-means via smooth patch ordering. In *Proc. 38th IEEE Internat. Conf. Acoust. Speech Signal Process., ICASSP-2013*, pages 1350–1354. IEEE, 2013.

[6] I. Ram, M. Elad, and I. Cohen. Image processing using smooth ordering of its patches. *IEEE Trans. Image Processing*, 22(7):2764 – 2774, 2013.

Contents

1 Introduction **9**

1.1 Graph Wavelet Transforms 10

1.2 Image Processing with Graph Wavelet Transforms 11

1.3 Facial Image Compression 12

1.4 Image Processing using Local Patches 12

1.5 Thesis Structure 13

1.6 List of Publications 15

2 Research Methods **17**

2.1 Orthogonal Wavelet Transform 17

2.2 Redundant Wavelet Transform 19

2.2.1 The implementation of Shensa and Beylkin 19

2.2.2 The à trous algorithm 20

2.2.3 Frame Properties 22

2.3 Sparse Representations using Redundant Dictionaries 22

3 Generalized Tree-Based Wavelet Transform **25**

3.1 Introduction 25

3.2 Problem Formulation 26

3.3 Generalized Tree-Based Wavelets 27

3.4 Building the operators $\mathbf{P}_\ell$ 28

3.5 Conclusion 33

4 Redundant Tree-Based Wavelet Transform **35**

4.1 Introduction 35

4.2 Problem Formulation 36
4.3 Redundant Tree-Based Wavelet Transform 37
4.3.1 Decomposition and Reconstruction schemes 37
4.3.2 Building the Operators $\mathbf{P}_\ell^s$ 39
4.4 À Trous Implementation and Frame Properties 42
4.4.1 À Trous Implementation 42
4.4.2 Frame Properties 45
4.4.3 Computational complexity 46
4.5 Label recovery using RTBWT 46
4.6 Conclusion 47
4.A Proof of Proposition 1 48

5 Image Processing using Tree-Based Wavelets 51
5.1 Introduction 51
5.2 Image-Derived Tree-Based Wavelets 52
5.3 Image Representation using the GTBWT 56
5.4 Image Denoising using Tree-Based Wavelets 59
5.4.1 The Basics 59
5.4.2 Subimage averaging 64
5.4.3 Robustness to noise 66
5.4.4 Vector Thresholding 66
5.5 Conclusion 68

6 RTBWT Subimage Averaging Frames 69
6.1 Introduction 69
6.2 Problem Formulation 70
6.3 RTBWT Subimage Averaging Frames 71
6.4 Image Reconstruction using RTBWT SA Frames 73
6.4.1 Image Reconstruction Scheme 73
6.4.2 Image Denoising 75
6.4.3 Image Deblurring 80
6.4.4 Computational Complexity 83

6.5 Conclusions . . . 86
6.A Proof of Proposition 2 . . . 86

7 Facial Image Compression using RTBWT 89
7.1 Introduction . . . 89
7.2 The Sparsifying Transform . . . 90
7.2.1 Sparse Representation of Facial Images . . . 90
7.2.2 RTBWT-Based Dictionary for a Set of Images . . . 91
7.3 Facial Image Compression Scheme . . . 94
7.4 Experimental Results . . . 96
7.5 Conclusion . . . 99

8 Image Processing using Patch Ordering 101
8.1 Introduction . . . 101
8.2 Image Processing using Patch Ordering . . . 103
8.2.1 The Basic Scheme . . . 103
8.2.2 Building the Permutation Matrix **P** . . . 104
8.2.3 Subimage Averaging . . . 108
8.2.4 Connection to BM3D and Clustering-Based Methods . . . 110
8.2.5 Applications and Results . . . 111
8.3 Image denoising using NL-Means via Smooth Patch Ordering . . . 123
8.3.1 The Basic Denoising Scheme . . . 123
8.3.2 Improving the Denoising results . . . 124
8.3.3 Experimental Results . . . 125
8.4 Conclusions . . . 127

9 Research Summary and Future Directions 129
9.1 Research summary . . . 129
9.2 Future research directions . . . 131

Bibliography 133

List of Figures

2.1 Discrete orthonormal wavelet decomposition scheme. 18
2.2 Discrete orthonormal wavelet reconstruction scheme. 18
2.3 Redundant wavelet decomposition scheme. 20
2.4 Redundant wavelet reconstruction scheme. 21
2.5 à trous algorithm decomposition scheme. 21
2.6 à trous algorithm reconstruction scheme. 22
2.7 The orthogonal matching pursuit algorithm. 23

3.1 Proposed orthonormal wavelet decomposition scheme. 27
3.3 Proposed orthonormal wavelet reconstruction scheme. 28
3.2 Proposed orthonormal wavelet decomposition algorithm. 28
3.4 Proposed orthonormal wavelet reconstruction algorithm. 29
3.5 Illustrations of a binary and "generalized" trees. 31
3.6 Algorithm for the calculation of the GTBWT reordering operators. 32

4.1 Proposed redundant wavelet decomposition scheme. 38
4.2 Proposed redundant wavelet decomposition algorithm. 39
4.3 Proposed redundant wavelet reconstruction scheme. 40
4.4 Proposed redundant wavelet reconstruction algorithm. 40
4.5 An illustration of a "generalized" tree. 42
4.6 Algorithm for the calculation of the RTBWT reordering operators. 43
4.7 RTBWT à trous based decomposition scheme. 44
4.8 RTBWT à trous based reconstruction scheme. 45
4.9 Label recovery results. 47

5.1 Calculation of the image-derived GTBWT reordering operators. 54
5.2 Calculation of the image-derived RTBWT reordering operators. 55
5.3 Calculation of the reordering operators for a 2-level decomposition of an image with 8 pixels. 56
5.4 m-term approximation results. 57
5.5 GTBWT basis elements derived from a synthetic image. 59
5.6 GTBWT basis elements derived from an image. 60
5.7 Denoising results of noisy versions of the images Barbara and Lena. 63
5.8 Improved denoising results for the image Barbara 68

6.1 Image reconstruction scheme. 75
6.2 Denoising Performance vs. permutation number and B size. 78
6.3 Denoising Results using RTBWT SA Frames. 81
6.4 Deblurring Results using RTBWT SA Frames. 85

7.1 RTBWT decomposition scheme 92
7.2 Facial image encoding and decoding schemes. 94
7.3 Mean face image and 2 atoms from the RTBWT dictionary. 96
7.4 Rate distortion curves. 97
7.5 Facial image compression results. 98

8.1 The basic image processing scheme. 104
8.2 The patch reordering algorithm we use. 106
8.3 Noisy and Reordered Barbara and histograms 107
8.4 Two subimages and classification results 109
8.5 Average of the PSNR values 115
8.6 Average of the PSNR values 116
8.7 Trained filters 117
8.8 Denoising results 119
8.9 Inpainting Results 122
8.10 NL-means Denoising Results 127

List of Tables

5.1 Denoising results of a noisy version of the image in Figure 5.6(a) 62
5.2 Denoising results of noisy versions of the images Barbara and Lena. 62
5.3 Denoising results using clean and noisy patches. 67
5.4 Improved Denoising Results for the Images Barbara and Lena 67

6.1 Parameters Used in the Denoising Experiments. 79
6.2 Denoising Results using RTBWT SA Frames. 80
6.3 Blur PSF and Noise Variances used in the Different Deblurring Scenarios. . 81
6.4 Parameters Used in the Deblurring Experiments. 82
6.5 Deblurring Results using RTBWT SA Frames. 84

8.1 Parameters used in the denoising experiments. 114
8.2 Denoising results 115
8.3 Denoising results 118
8.4 Parameters used in the inpainting experiments. 120
8.5 Inpainting results 121
8.6 Parameters used in the NL-means denoising experiments. 125
8.7 NL-means Denoising Results 126

Abstract

In this dissertation we introduce novel methods for processing signals defined on graphs and high dimensional data clouds. Finding efficient methods to represent and process such signals is quite a challenging problem since they are non-uniformly sampled. As such, traditional signal processing methods are usually not helpful, since they are designed for data defined on regular Euclidean grids. Our solution consists of changing the way we look at the data by "putting an order in the disorder". We develop several methods which enable us to return to the comfort zone of uniformly sampled one dimensional (1D) signal processing, by reordering the given topologically complicated data. This makes the processing of the resulting signals much simpler and intuitive, and allows us to develop new algorithms which are based on common 1D methods. Throughout this work we explore the use of these algorithms mainly for image processing, and demonstrate high quality results.

Our work starts by introducing a generalized tree-based wavelet transform (GTBWT), which is an extension of the 1D discrete wavelet transform to functions defined on graphs and high dimensional data points. This transform is calculated by adding data-dependent permutation operators into the classical 1D wavelet filter-bank implementation. We show how this transform is applicable to images, by converting them to graph-structures. This is achieved by breaking the image into overlapping patches and referring to the resulting set of points as a point-cloud. This leads to a multi-scale and highly robust decomposition that is adaptive to the image content and is highly effective for sparsifying it. We use the above construction to further develop a redundant tree-based wavelet transform (RTBWT) frame, which is a redundant image representation that can be used as a powerful sparsity-promoting regularizer in general inverse problems in image processing. We demonstrate the application of the RTBWT to image denoising and deblurring, and even

harness it for face image compression.

Finally, we strip-down the proposed scheme to its core, and introduce an image processing scheme that is based only on permutations obtained by reordering of its patches, without the multi-scale treatment used by the aforementioned transforms. We reorder the corrupted image pixels using such permutations, and apply on the resulting 1D signal either relatively simple one-dimensional smoothing operations (such as filtering or interpolation) or the non-local means (NL-means) algorithm. We demonstrate the application of this scheme to image denoising and inpainting.

Notation

$*$	convolution
$\star$	cyclic convolution
$(\cdot)^T$	transpose operation
$(\cdot)^H$	Hermitian
$\lvert \cdot \rvert$	absolute value
$\lVert \cdot \rVert$	Euclidean norm
$\lVert \cdot \rVert_0$	the number of nonzero entries in a vector
$\lVert \cdot \rVert_{TV}$	total variation of a vector
$\downarrow 2$	$2:1$ decimator
$\downarrow 2^o$ and $\downarrow 2^e$	$2:1$ decimators that keep the odd and even samples of their input, respectively
$\uparrow 2$	$2:1$ interpolator
$\uparrow 2^o$ and $\uparrow 2^e$	interpolators that place the samples of their input vector in the odd and even locations of their output vector, respectively
$\mathbf{0}$	vector of all zeros
α	coefficient vector
σ	noise standard deviation
$\mathbf{\Phi}$	redundant wavelet transform matrix
$\mathbf{\Phi}^{SA}$	analysis operator used to obtain $\mathbf{c}^{SA}$
ϕ_n	the nth row in $\mathbf{\Phi}$
$\mathbf{\Psi}$	redundant wavelet inverse transform matrix
$\mathbf{\Psi}^{SA}$	synthesis operator used to reconstruct an image from $\mathbf{c}^{SA}$
ψ_n	the nth column in $\mathbf{\Psi}$

$\mathbf{a}_\ell$	ℓth scale wavelet approximation coefficient vector
$\mathbf{a}_\ell^s$	signal that contain subset of the samples in $\mathbf{a}_\ell$
$\mathbf{B}$	matrix
$\mathbf{B}^{-1}$	matrix inverse
$B(\omega)$	Fourier transform of $b[k]$
$\mathbf{b}$	column vector
$\hat{\mathbf{b}}$	estimate of $\mathbf{b}$
$\mathbf{b}^p$	permuted version of the vector $\mathbf{b}$
b	scalar
$b[k]$	kth sample in the vector $\mathbf{b}$
$\mathbf{c}^{SA}$	extended RTBWT representation composed of subimage coefficients
$\mathbf{D}$	diagonal weight matrix
$\mathbf{d}_\ell$	ℓth scale wavelet detail coefficient vector
$\mathbf{d}_\ell^s$	signal that contain subset of the samples in $\mathbf{d}_\ell$
$\mathbf{e}_k$	a vector used to extract the kth pixel from $(\hat{\mathbf{y}}^g)^T$
f_i	the value of a scalar function on $\mathbf{x}_i$, i.e. $f(\mathbf{x}_i)$
$\mathbf{g}$	wavelet reconstruction filter of the discrete wavelet transform
$\bar{\mathbf{g}}$	wavelet decomposition filter of the discrete wavelet transform
$\tilde{\mathbf{g}}$	filter satisfying $\tilde{\mathbf{g}} = \frac{1}{2}\mathbf{g}$
$\mathbf{g}_\ell$	upsampled version of $\mathbf{g}$
$\bar{\mathbf{g}}_\ell$	upsampled version of $\bar{\mathbf{g}}$
$H\{\cdot\}$	one dimensional smoothing operator
$\mathbf{h}$	scaling reconstruction filter of the discrete wavelet transform
$\bar{\mathbf{h}}$	scaling decomposition filter of the discrete wavelet transform
$\tilde{\mathbf{h}}$	filter satisfying $\tilde{\mathbf{h}} = \frac{1}{2}\mathbf{h}$
$\bar{\mathbf{h}}_a$	normalized version of the filter $\bar{\mathbf{h}}$
$\mathbf{h}_\ell$	upsampled version of $\mathbf{h}$
$\bar{\mathbf{h}}_\ell$	upsampled version of $\bar{\mathbf{h}}$
$\mathbf{h}_s, \mathbf{h}_e, \mathbf{h}_t, \mathbf{h}_k$	vectors containing learned filters
$\mathbf{I}$	identity matrix

N_h	length of the filters $\mathbf{h}_s$ and $\mathbf{h}_e$
N_p	number of overlapped $\sqrt{n} \times \sqrt{n}$ patches in an image
$\mathbf{P}, \mathbf{P}_\ell, \mathbf{P}_\ell^s$	operators that apply a permutation to their input vectors
$\tilde{\mathbf{P}}_\ell^s$	operator that cancels the ordering done by $\mathbf{P}_\ell^s$
$\mathbf{P}^{-1}, \mathbf{P}_\ell^{-1}$	operators that cancel the ordering done by $\mathbf{P}, \mathbf{P}_\ell$, respectively
$\mathbf{q}_i, \mathbf{q}_{\ell,j}, \mathbf{q}_{\ell,j}^s$	2D spatial coordinates of image patches
$\mathbb{R}^N$	N-dimensional real-valued vector
$\mathbf{R}_j$	matrix which extracts the subimage $\tilde{\mathbf{y}}_j$ from the image $\mathbf{y}$
$\mathbf{R}_\ell^s$	matrix which extracts the signal $\mathbf{a}_\ell^s$ from the signal $\mathbf{a}_\ell$
$\bar{\mathbf{R}}_k$	matrix used to extract the 5×5 patch surrounding the kth pixel in $(\hat{\mathbf{y}}^g)^T$
S^{hard}, S_τ	thresholding operators
S_i	set of indices of non-missing pixels in a patch $\mathbf{x}_i$
S_i^{NL}	a square neighborhood surrounding a noisy pixel
S_s and S_e	sets containing smooth and non-smooth patches, respectively
std($\mathbf{b}$)	standard deviation of a vector $\mathbf{b}$
V	the space of all functions on the dataset $\mathbf{X}_{data}$
$w(\cdot, \cdot)$	a distance measure in $\mathbb{R}^n$
$w_{i,j}$	weights used by the NL-means algorithm
$\mathbf{X}$	matrix containing the feature points $\mathbf{x}_i$ in its columns
$\mathbf{X}_{data}$	dataset containing feature points $\mathbf{x}_i$
$\mathbf{x}_i, \mathbf{x}_{\ell,j}, \mathbf{x}_{\ell,j}^s$	n dimensional feature points
$\mathbf{y}^g$	training image
$\tilde{\mathbf{Y}}_j$	subimage of an image $\mathbf{Y}$
$\tilde{\mathbf{y}}_j$	column stacked version of the subimage $\tilde{\mathbf{Y}}_j$
$\tilde{\mathbf{z}}_j$	noisy version of $\tilde{\mathbf{y}}_j$
$\tilde{\mathbf{z}}_{j,s}, \tilde{\mathbf{z}}_{j,e}$	vectors containing the smooth and non-smooth pixels in $\tilde{\mathbf{z}}_j$

Abbreviations

1D	One dimensional
1-NN	1-nearest neighbor
2D	Two dimensional
GNE	Generalized Nash equilibrium
GTBWT	Generalized tree-based wavelet transform
MAE	Mean absolute error
NP-hard	Non-deterministicPolynomial-timehard
NL-means	Non-local means
OMP	Orthogonal matching pursuit
PSNR	Peak signal-to-noise ratio
RTBWT	Redundant tree-based wavelet transform
TSP	Traveling salesman problem

Chapter 1

Introduction

Most traditional signal processing methods are designed for data defined on regular Euclidean grids. Development of comparable methods capable of handling non-uniformly sampled signals, data defined on graphs or "point clouds", is important and very much needed. In this thesis we develop novel wavelet transforms designed to efficiently (sparsely) represent functions defined on graphs and high dimensional data points. We have chosen to extend the wavelet transform as it has been proven to be a very useful tool for signal processing [43]. A major advantage of wavelet methods is their ability to simultaneously localize signal content in both space and frequency. This property allows them to compactly represent signals such as 1D steps or piece-wise smooth content, whose primary information content lies in localized singularities. In order to take advantage of the above property, we propose a new signal processing approach, which is in fact the key concept shared by all the methods proposed in this work. We move from the realm of signals defined on such topologically complicated grids to the comfort zone of one dimensional (1D) signal processing, by reordering these signals into regular 1D signals. Incorporating this method into the 1D orthogonal and redundant wavelet transforms allows us to extend them to irregular, non-Euclidean spaces, and obtain transforms that efficiently represent functions defined on such data.

We demonstrate the effectiveness of the obtained transforms mainly as tools for image processing. We show that applying either of the transforms to an image by converting it to a graph-structure, produces an image adaptive multi-scale decomposition that is highly effective for sparsifying its content. We develop image processing algorithms that

utilize this property and produce high quality image denoising, deblurring and compression results. Moreover, we show that even when the multiscale treatment is discarded, our concept of reordering images into a piecewise regular 1D signals can still serve as a base for high quality image processing algorithms. Taking advantage of the prior knowledge that the reordered image should be regular simplifies its treatment, and this allows image processing problems such as inpainting and denoising to be solved using simple 1D operators such as filtering and interpolation.

In this chapter we briefly describe the scientific background for the main topics of this research and outline the structure of the thesis.

1.1 Graph Wavelet Transforms

Several extensions of the wavelet transform, operating on graphs and high dimensional data, have already been proposed prior to our work. The wavelet transforms proposed by Chen and Maggioni [8], Lee et al. [38] , and Murtagh [45], were applied to the data points (coordinates) themselves, rather than on functions defined on the data points, as we do in our work. Other methods took different approaches to construct wavelets applied to functions on the data. Maggioni and Coifman [13] and Hammond et al. [32] proposed wavelets based on diffusion operators and the graph Laplacian [10], [63], respectively. Jansen et al. [36] proposed three methods which were based on a variation of the lifting scheme [57, 58]. Gavish et al. [30] assumed that the geometry and structure of the input data are captured in a hierarchical tree. Then, given such a tree, they built a data adaptive Haar-like orthonormal basis for the space of functions over the data set. This basis can be seen as a generalization of the one proposed in [24] for binary trees.

In this thesis we introduce the generalized tree-based wavelet transform (GTBWT), that extends the transform proposed by Gavish et al. [30], and can also construct data adaptive orthonormal wavelets beyond Haar. We calculate the transform by adding data-dependent permutation operators into the classical 1D discrete wavelet filter-bank implementation. These permutations are applied to the approximation coefficient signals so as to increase their regularity and cause their representation with the GTBWT to be sparser. The reordering operators are obtained by organizing the high dimensional coordinates or

feature vectors associated with the graph vertices, such that they are chained in the shortest possible path, essentially obtaining an approximation to the solution of the traveling salesman problem (TSP) [14]. We also introduce the redundant tree-based wavelet transform (RTBWT), which extends the redundant wavelet transform to functions defined on graphs and data clouds in a similar manner.

1.2 Image Processing with Graph Wavelet Transforms

Although the wavelet transforms introduced in this thesis were originally designed to represent scalar functions defined on high-dimensional data clouds and graphs, we focus in our research on their application for image processing. The transforms are applied to an image by converting it to a graph-structure, where the image pixels act as a sampled function defined of the graph vertices, and their surrounding patches act as features associated with these samples. The resultant multi-scale transforms are image adaptive and highly effective for sparsifying the image content. We demonstrate this fact for the GTBWT by showing that it is more efficient than both the one-dimensional (1D) and two-dimensional (2D) separable wavelet transforms in representing images. We also show that the adaptiveness of this transform is reflected in its basis functions, whose content varies from smooth versions of the image to varying edges in the image which are not necessarily aligned with the vertical and horizontal axes.

We next explore the application of the image-derived transforms to image denoising, and show that combined with a subimage averaging scheme they achieve denoising results which are close to the state-of-the-art. We also show that the image representation obtained with the RTBWT combined with a subimage averaging scheme is a frame expansion, and derive the analysis and synthesis operators associated with it. We explore the use of this frame operators to image deblurring, and demonstrate state-of-the-art results.

1.3 Facial Image Compression

In recent years, facial images are being extensively used and stored in large databases by social networks, web services, or various organizations such as states, law-enforcement, schools, universities, and private companies. Facial images are also expected to be stored in digital passports and ID cards. Thus, efficient storage of such images is beneficial, and their compression is an appealing application. The limitation to a specific and narrow family of images increases their combined spatial redundancy, and this allows algorithms that are specially tailored for the task of facial image compression, to surpass general purpose compression methods. More specifically, recent work by Elad et al. [27], Bryt and Elad [4, 5] , and Zepeda et al. [69] has shown that this kind of algorithms lead to performance far exceeding JPEG2000 [60].

In this thesis, we propose a novel such compression algorithm that exploits the redundant tree-based wavelet transform. Given a training set of frontal facial images, the transform is designed to best sparsify the whole set using a common feature-ordering. Our compression scheme consists of sparse coding using the transform, followed by entropy coding of the obtained coefficients. The inverse transform and a post-processing stage are used to decode the compressed image.

1.4 Image Processing using Local Patches

Image processing using local patches has become very popular in recent years, and was shown to be highly effective – see [6, 7, 16, 19, 20, 26, 39–41, 67, 68, 70, 71] for representative work. The core idea behind these and many other contributions is the same: given the image to be processed, extract all possible patches with overlaps; these patches are typically very small compared to the original image size (a typical patch size would be 8×8 pixels). The processing itself proceeds by operating on these patches and exploiting interrelations between them. The manipulated patches (or sometimes only their center pixels) are then put back into the image canvas to form the resulting image.

There are various ways in which the relations between patches can be taken into account: weighted averaging of pixels with similar surrounding patches, as the NL-Means

algorithm does [6], clustering the patches into disjoint sets and treating each set differently, as performed in [7, 19, 20, 67, 68, 71], seeking a representative dictionary for the patches and using it to sparsely represent them, as practiced in [26, 40, 41, 70] gathering groups of similar patches and applying a sparsifying transform on them [16,39,40]. A common theme to many of these methods is the expectation that every patch taken from the image may find similar ones extracted elsewhere in the image. Put more broadly, the image patches are believed to exhibit a highly-structured geometrical form in the embedding space they reside in. A joint treatment of these patches supports the reconstruction process by introducing a non-local force, thus enabling better recovery.

In this thesis, we introduce a novel approach towards image processing scheme, that is based on permutations calculated by reordering of its patches. We propose to reconstruct corrupted images by reordering them using such permutations, and applying on the reordered set of pixels either relatively simple one-dimensional smoothing operations (such as filtering or interpolation) or the NL-means algorithm.

1.5 Thesis Structure

This thesis is organized as follows. Chapter 2 briefly outlines the basic theory and methods which were used during this research. The original contribution of this research starts in Chapter 3.

In Chapter 3, we introduce the generalized tree-based wavelet transform, applicable to functions defined on high dimensional data, weighted graphs and networks. This transform generalizes the Haar-like transform proposed by Gavish et al. [30], and can also construct data adaptive orthonormal wavelets beyond Haar. It is defined via a hierarchical tree-like structure, which is assumed to capture the geometry and structure of the input data. The transform is applied to the data using a modified version of the common 1D wavelet filtering and decimation scheme. The adaptivity of this wavelet scheme is obtained by permutations derived from the tree and applied to the approximation coefficients in each decomposition level, before they are filtered.

In Chapter 4, we propose the redundant tree-based wavelet transform, that is also applicable to scalar functions defined on high dimensional coordinates, weighted graphs

and networks. The proposed transform utilizes the distances between the given data points to construct tree-like structures. We modify the filter-bank decomposition scheme of the redundant wavelet transform by adding in each decomposition level operators that reorder the approximation coefficients. These reordering operators are derived by organizing the tree-node features so as to shorten the path that passes through these points. We also introduce an alternative implementation for the transform, which is based on the à trous algorithm, and use it to show that the RTBWT is a valid frame expansion. We explore the use of the proposed transform for the recovery of labels defined on point clouds, and show that the results are promising.

Chapter 5 shows that the schemes proposed in Chapters 3 and 4 can be used to construct image-adaptive transforms that are highly effective for sparsifying image content. We show that the GTBWT transform is more efficient than both the 1D and two-dimensional (2D) separable wavelet transforms in representing images. We also explore the application of the proposed transforms to image denoising, and show that combined with a subimage averaging scheme, they achieves denoising results which are close to the state-of-the-art.

In Chapter 6, we utilize the redundant tree-based wavelet transform proposed in Chapter 4 as a powerful sparsity-promoting regularizer in inverse problems in image processing. We show that the image representation obtained with this transform combined with a subimage averaging scheme is a frame expansion, and derive the analysis and synthesis operators associated with it. We explore the use of this frame operators to image denoising and deblurring, and demonstrate in both these cases state-of-the-art results.

In Chapter 7, we propose a novel facial image compression algorithm, exploiting the redundant tree-based wavelet transform introduced in Chapter 4. Given a training set of aligned frontal facial images, the transform is designed to best sparsify the whole set using a common feature-ordering. Our compression scheme consists of sparse coding using the transform, followed by entropy coding of the obtained coefficients. The inverse transform and a post-processing stage are used to decode the compressed image. We demonstrate the performance of the proposed scheme and compare it to other competing algorithms.

In Chapter 8, we propose an image processing scheme based on reordering of its patches. For a given corrupted image, we extract all patches with overlaps, refer to these

as coordinates in high-dimensional space, and order them such that they are chained in the "shortest possible path", essentially solving the traveling salesman problem. The obtained ordering applied to the corrupted image implies a permutation of the image pixels to what should be a regular signal. This enables us to obtain good recovery of the clean image by applying either relatively simple one-dimensional smoothing operations (such as filtering or interpolation) or the NL-means algorithm to the reordered set of pixels. We explore the use of the proposed approach to image denoising and inpainting, and show promising results in both cases.

Finally, Chapter 9 summarizes the main contributions of this dissertation and presents some future research directions.

1.6 List of Publications

The chapters of this thesis are based on the following publications:
Chapter 3 and parts of Chapter 5 are based on:

1. I. Ram, M. Elad and I. Cohen, *Generalized Tree-Based Wavelet Transform*, IEEE Trans. Signal Processing, Vol. 59, Number 9, Sep. 2011, pp. 4199–4209.

Chapter 4 and parts of Chapter 5 are based on:

2. I. Ram, M. Elad and I. Cohen, *Redundant Wavelets on Graphs and High Dimensional Data Clouds*, IEEE Signal Processing Letters, Vol. 19, Number 5, May 2012, pp. 291-294.

Chapter 6 and parts of Chapter 4 are based on:

3. I. Ram, I. Cohen and I. Elad, *Patch-Ordering-Based Wavelet Frame and Its Use in Inverse Problems*, IEEE Trans. Image Processing, Vol. 33, Number 7, July 2014, pp. 2779–2792.

Chapter 7 is based on:

4. I. Ram, I. Cohen and I. Elad, *Facial Image Compression using Patch-Ordering-Based Adaptive Wavelet Transform*, IEEE Signal Processing Letters, Vol. 21, Number 10, October 2014, pp. 1270-1274.

Chapter 8 is based on:

5. I. Ram, M. Elad and I. Cohen, *Image Processing using Smooth Ordering of its Patches*, IEEE Trans. Image Processing, Vol. 22, Number 7, July 2013, pp. 2764-2774.

6. I. Ram, M. Elad and I. Cohen, *Image Denoising Using NL-Means Via Smooth Patch Ordering*, Proc. 38th IEEE Internat. Conf. Acoust. Speech Signal Process., ICASSP-2013, Vancouver, Canada, May 26-31, 2013, pp. 1350-1354.

Chapter 2

Research Methods

In this chapter, we briefly review research methods which were useful during this research. We start by describing the orthogonal wavelet transform decomposition and reconstruction schemes. We then describe two different implementations of the redundant wavelet transform, and detail its frame properties. Finally, we briefly review the topic of sparse representation of a signal using a redundant dictionary, and describe the orthogonal matching pursuit (OMP) algorithm.

2.1 Orthogonal Wavelet Transform

One of the most successful types of transforms, which has been proven to be a very useful tool for signal and image processing, is wavelet analysis [43]. A major advantage of wavelet methods is their ability to simultaneously localize signal content in both space and frequency. This property allows them to compactly represent signals such as one dimensional (1D) steps or images with edges, whose primary information content lies in localized singularities. Moreover, wavelet methods represent such signals much more compactly than either the original domain or transforms with global basis elements such as the Fourier transform.

The discrete wavelet transform is a unitary transform, whose basis elements are translated and dilated versions of two function called a mother wavelet and a scaling function. This transform is known to sparsely represent piecewise regular signals, i.e. it produces a small number of large coefficients when it is applied to such signals [43]. Let $\mathbf{f}$ be a 1D

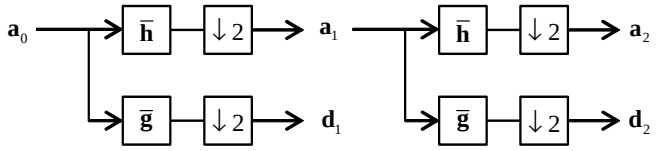


Figure 2.1: Discrete orthonormal wavelet decomposition scheme.

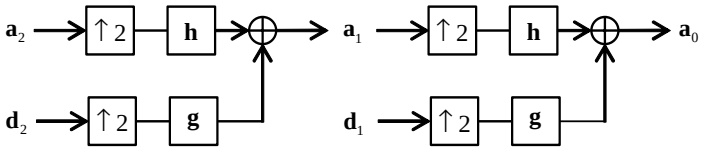


Figure 2.2: Discrete orthonormal wavelet reconstruction scheme.

signal of length N, and let $\mathbf{a}_\ell$ and $\mathbf{d}_\ell$ denote the approximation and detail coefficient vectors in the ℓth scale, respectively, where $\mathbf{a}_0 = \mathbf{f}$. Then an L-level wavelet decomposition is applied on $\mathbf{f}$ by repeating for $\ell = 0 \ldots L-1$ the following filter bank operations [43]:

$$a_{\ell+1}[k] = a_\ell * \bar{h}[2k] \tag{2.1}$$

$$d_{\ell+1}[k] = a_\ell * \bar{g}[2k]. \tag{2.2}$$

where $\bar{\mathbf{h}}$ and $\bar{\mathbf{g}}$ denote the wavelet decomposition filters of an orthonormal discrete wavelet transform. The decomposition scheme is also described in Figure 2.1, where $\downarrow 2$ denotes a $2:1$ decimator.

Similarly, let

$$\check{a}_\ell[k] = \begin{cases} a_\ell[p], & \text{if } k = 2p \\ 0, & \text{if } k = 2p+1. \end{cases} \tag{2.3}$$

Then the wavelet reconstruction scheme is applied by repeating for $\ell = L \ldots 1$ the following filter bank operations [43]:

$$a_\ell[n] = \check{a}_{\ell+1} * \bar{h}[n] + \check{d}_{\ell+1} * \bar{g}[n]. \tag{2.4}$$

where $\mathbf{h}$ and $\mathbf{g}$ denote the wavelet reconstruction filters. The wavelet reconstruction scheme is also described in Figure 2.2, where $\uparrow 2$ denotes an upsampler.

The orthogonal wavelet and scaling filters satisfy [43]

$$\bar{h}[n] = h[-n] \text{ and } \bar{g}[n] = g[-n] \tag{2.5}$$

$$\sum_n h[n] = \sqrt{2} \text{ and } \sum_n g[n] = 0 \tag{2.6}$$

and that the filters $\mathbf{h}$ and $\mathbf{g}$ are power complementary, i.e. their Fourier transforms satisfy

$$|H(2^\ell \omega)|^2 + |G(2^\ell \omega)|^2 = 2. \tag{2.7}$$

In Chapter 3 we introduce the generalized tree-based wavelet transform, which extends the discrete wavelet transform to scalar functions defined on high-dimensional data clouds, graphs and networks.

2.2 Redundant Wavelet Transform

The main limitation of the orthogonal wavelet transform is sensitivity to translation. This problem may be alleviated by employing the redundant version of the wavelet transform, that also sparsely represents piecewise regular signals. We next describe two different implementations for the application of this transform. In Chapter 4 we introduce the redundant tree-based wavelet transform, which extends these two implementations to scalar functions defined on high-dimensional data clouds, graphs and networks.

2.2.1 The implementation of Shensa and Beylkin

The first implementation was proposed by Shensa [54] and Beylkin [2]. This implementation employs a filter-bank decomposition scheme, similarly to the orthonormal discrete wavelet transform. However, in each level of this scheme all the coefficients are retained since the highpass bands do not contain decimators, and the decimation in the lowpass bands is replaced by a split into even and odd sequences, which are further decomposed in the next decomposition level.

Figure 2.3 describes the first implementation of the redundant wavelet transform decomposition scheme. The filters $\bar{\mathbf{h}}$ and $\bar{\mathbf{g}}$ are the scaling and wavelet decomposition filters of an orthonormal discrete wavelet transform, and they are applied using cyclic convolution. $\downarrow 2^o$ and $\downarrow 2^e$ denote $2:1$ decimators that keep the odd and even samples of their

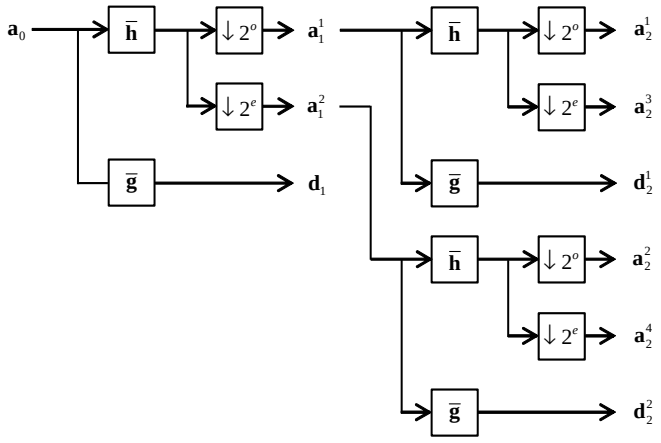


Figure 2.3: Redundant wavelet decomposition scheme.

input, respectively. $\mathbf{a}_\ell$ and $\mathbf{d}_\ell$ denote the approximation and detail coefficient vectors in the ℓth scale, respectively, where $\mathbf{a}_0 = \mathbf{f}$. We note that for $\ell > 1$ these signals do not appear explicitly in the decomposition scheme, and instead it employs the signals denoted by $\mathbf{a}_\ell^s$ and $\mathbf{d}_\ell^s$, which contain subsets of the samples in $\mathbf{a}_\ell$ and $\mathbf{d}_\ell$, respectively. $\mathbf{a}_\ell^s$ is obtained from $\mathbf{a}_\ell$ by starting from the sth sample, and keeping every 2^ℓth sample. $\mathbf{d}_\ell^s$ is obtained from $\mathbf{d}_\ell$ by starting from the sth sample, and keeping every $2^{\ell-1}$th sample. We notice that $\mathbf{a}_0^1 = \mathbf{a}_0$ and $\mathbf{d}_1^1 = \mathbf{d}_1$.

In a similar manner, Figure 2.4 describes the first implementation of the redundant wavelet transform reconstruction scheme. $\mathbf{h}$ and $\mathbf{g}$ denote the wavelet and scaling reconstruction filters, and they are also applied using cyclic convolution. The upsamplers denoted by $\uparrow 2^o$ and $\uparrow 2^e$ place the samples of their input vector in the odd and even locations of their output vector, respectively.

2.2.2 The à trous algorithm

An alternative implementation for the redundant wavelet transform is the common and widely known "à trous" algorithm [43] proposed by Holschneider et al. [35]. Instead of explicitly applying the filters $\bar{\mathbf{h}}$ and $\bar{\mathbf{g}}$ to subsets of the signals $\mathbf{a}_\ell$, the à trous algorithm applies upsampled versions of these filters directly to $\mathbf{a}_\ell$. The upsampled filters are obtained by inserting "holes" (trous in French) between nonzero filter taps. Let $\bar{\mathbf{h}}_\ell$ be a filter obtained from $\bar{\mathbf{h}}$ by inserting $2^\ell - 1$ zeros between each sample, where $\bar{\mathbf{h}}_0 = \bar{\mathbf{h}}$. Also,

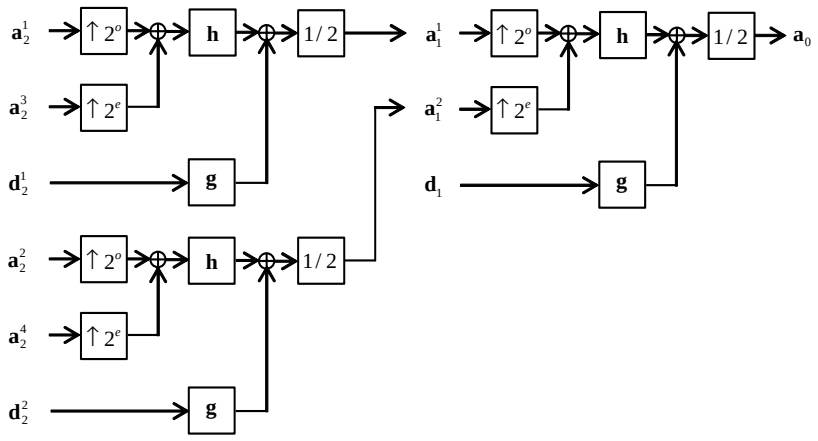


Figure 2.4: Redundant wavelet reconstruction scheme.

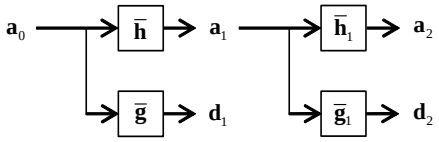


Figure 2.5: à trous algorithm decomposition scheme.

let $\bar{\mathbf{g}}_\ell$ be a filter obtained from $\bar{\mathbf{g}}$ in a similar manner. Both filters are applied using cyclic convolution, which we denote by $\star$. Then the redundant wavelet transform can be applied by repeating for $\ell = 0 \dots L-1$ the following filter bank operations, described in Figure 2.5,

$$a_{\ell+1}[n] = a_\ell \star \bar{h}_\ell[n] \tag{2.8}$$

$$d_{\ell+1}[n] = a_\ell \star \bar{g}_\ell[n]. \tag{2.9}$$

In a similar manner, let $\mathbf{h}_\ell$ be a filter obtained from $\mathbf{h}$ by inserting $2^\ell - 1$ zeros between each sample, where $\mathbf{h}_0 = \mathbf{h}$. Also, let $\mathbf{g}_\ell$ be a filter obtained from $\mathbf{g}$ in a similar manner. Both filters are applied using cyclic convolution. Then the RTBWT reconstruction scheme can be applied by repeating for $\ell = L \dots 1$ the following filter bank operations, described in Figure 2.6,

$$a_\ell[n] = \frac{1}{2}\left(a_{\ell+1} \star h_\ell[n] + d_{\ell+1} \star g_\ell[n]\right). \tag{2.10}$$

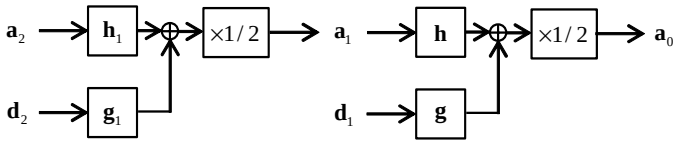


Figure 2.6: à trous algorithm reconstruction scheme.

2.2.3 Frame Properties

Let $\mathbf{\Phi}$ denote an $(L+1)N \times N$ transform matrix, which applies an L-level redundant wavelet decomposition. We recall that a sequence $\{\phi_n\}$ is a frame if there exist two constants $\beta \geq \alpha > 0$ such that for all $\mathbf{y}$

$$\alpha\|\mathbf{y}\|^2 \leq \sum_n |\langle \phi_n, \mathbf{y} \rangle|^2 \leq \beta\|\mathbf{y}\|^2. \tag{2.11}$$

It is shown by Fowler [29] that the rows of $\mathbf{\Phi}$ constitute a frame $\{\phi_n\}$ in $\mathbb{R}^N$, with frame bounds $\alpha = 2$ and $\beta = 2^L$, and these bounds are the tightest possible.

Also, let $\mathbf{\Psi}$ denote an $N \times (L+1)N$ matrix that applies the L-level redundant wavelet reconstruction. $\mathbf{\Phi}$ and $\mathbf{\Psi}$ apply the redundant wavelet analysis and synthesis operations, respectively, and $\mathbf{\Psi\Phi} = \mathbf{I}$ where $\mathbf{I}$ is the identity matrix. Thus $\mathbf{\Psi}$ is the pseudo inverse of $\mathbf{\Phi}$, and the columns of $\mathbf{\Psi}$ constitute a frame $\{\psi_n\}$ which is dual to $\{\phi_n\}$, i.e. it satisfies

$$\frac{1}{2^L}\|\mathbf{y}\|^2 \leq \sum_n |\langle \psi_n, \mathbf{y} \rangle|^2 \leq \frac{1}{2}\|\mathbf{y}\|^2. \tag{2.12}$$

2.3 Sparse Representations using Redundant Dictionaries

Let $\mathbf{f}$ be a signal which contains N pixels. We assume that this signal follows the sparseland model [25], and therefore we can compress it by obtaining an efficient (sparse) representation for it. The sparseland model suggests that the signal $\mathbf{f}$ can be sparsely represented using a redundant matrix $\mathbf{A}$ of size $N \times J$ $(J > N)$, which we term a dictionary. More specifically, let $\|\alpha\|_0$ denote the number of nonzero entries in a coefficient vector α. Then we expect that the solution of

$$\hat{\alpha} = \underset{\alpha}{\operatorname{argmin}} \ \|\alpha\|_0 \text{ subject to } \|\mathbf{A}\alpha - \mathbf{f}\|_2^2 \leq \epsilon_0^2 \tag{2.13}$$

Task: Approximate the solution of $\underset{\alpha}{\operatorname{argmin}} \|\alpha\|_0$ subject to$\|\mathbf{A}\alpha - \mathbf{f}\|_2^2 \leq \epsilon_0^2$.

Parameters: The matrix $\mathbf{A}$, the vector $\mathbf{f}$, and the error threshold ϵ_0.

Initialization: Initialize $k = 0$, and set

- The initial solution $\alpha^0 = 0$.
- The initial residual $\mathbf{r}^0 = \mathbf{f}$
- The initial solution support $S_0 = Support\{\alpha^0\} = \emptyset$.

Main Iteration: Increment k by 1 and perform the following steps:

- **Sweep:** Compute the errors $\epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{r}^{k-1}\|_2^2$ for all j using the optimal choice $z_j^* = \mathbf{a}_j^T \mathbf{r}^{k-1} / \|\mathbf{a}_j\|_2^2$.
- **Update Support:** Find a minimizer, j_0 of $\epsilon(j) : \forall j \notin S^{k-1}, \epsilon(j_0) \leq \epsilon(j)$, and update $S^k = S^{k-1} \cup \{j_0\}$.
- **Update Provisional Solution:** Compute, α^k, the minimizer of $\|\mathbf{A}\alpha - \mathbf{f}\|_2^2$ subject to $Support\{\alpha\} = S^k$.
- **Stopping Rule:** If $\|\mathbf{r}_k\|_2 < \epsilon_0$, stop. Otherwise apply another iteration.

Output: The proposed solution is α^k.

Figure 2.7: The orthogonal matching pursuit algorithm.

should be sparse, i.e. $\|\alpha\|_0 \ll N$. This means that the image $\mathbf{f}$ can be represented using a small number of columns (termed atoms) from $\mathbf{A}$, with an error measured by the distance $\|\mathbf{A}\alpha - \mathbf{f}\|_2^2$. Naturally, there exists a tradeoff between the number of atoms used to represent $\mathbf{f}$ and the size of the representation error, i.e., the more atoms we use the smaller the error gets, and vice versa.

Problem (2.13) is a non-deterministic polynomial-time hard (NP-hard) problem, since it requires an examination of $O(J^N)$ possible non-zero supports for α. However, the matching and the basis pursuit algorithms [9, 44, 46] can be used quite effectively to obtain an approximation for the solution. In fact, these approximation techniques can be quite accurate if the solution is sparse enough to begin with [21–23, 61, 62]. In this work we make use of the Orthogonal Matching Pursuit (OMP) because of its simplicity [46] and efficiency. Let $\mathbf{a}_j$ denote the jth column in the dictionary $\mathbf{A}$. Then figure 2.7 describes the OMP algorithm.

In Chapters 4 and 7 we employ the OMP algorithm and the redundant tree-based wavelet transform introduced in Chapter 4 to obtain sparse representations of 1D signals.

Chapter 3

Generalized Tree-Based Wavelet Transform[1]

3.1 Introduction

In this chapter we introduce a generalized tree-based wavelet transform (GTBWT), that extends the orthogonal wavelet transform to irregular, non-Euclidean spaces, and efficiently represents functions defined on such data. The proposed transform is an extension of the Haar-like transform introduced by Gavish et al. [30], which can also construct data adaptive orthonormal wavelets beyond Haar. We calculate the transform by adding data-dependent operators in each level of the common one-dimensional (1D) wavelet decomposition scheme. These operators apply permutations to the approximation coefficient signals before they are filtered, thus increasing their regularity and causing their representation with the transform to be more efficient (sparse). We derive the operators from a tree-like structure, which we term a "generalized tree". This structure is constructed from the data points in a recursive manner, where the calculation of each coarse level involves organizing the set of data points in the finer level, such that they are chained in the shortest possible path, essentially obtaining an approximation to the solution of the traveling salesman problem (TSP) [14]. The points order in this path defines the permutation applied to the approximation coefficients of the finer level in the wavelet decomposition scheme. We propose a path constructed by starting from a random point,

[1] This chapter is based on [50].

and then continuing from each point to its nearest neighbor according to some distance measure, visiting each point only once.

We note that the wavelet transforms proposed by Heinen and Plonka [34], Mallat [42], and Plonka [47], share some similarities with our proposed algorithm. The first two methods are defined on images, and employ pairing or reordering of wavelet coefficients in the decomposition schemes in order to adapt to their input images. In fact, the easy path wavelet transform proposed by Plonka [47] employs a decomposition scheme which is very similar to ours. Constraining our algorithm to a regular data grid and using the same starting point and search neighborhood as the ones employed by Plonka [47], both algorithms essentially coincide. Nevertheless, our approach is more general, as it tackles the more abstract problem of devising a transform for point clouds or high-dimensional graph-data, whereas Plonka [47] concentrates on images. The third method, uses a scheme which is very similar to ours, but differs in the way the permutation operators are calculated.

The chapter is organized as follows: In Section 3.2 we formulate the problem. In Section 3.3, we introduce the generalized tree-based wavelet transform decomposition and reconstructions schemes. In Section 3.4, we explain how the reordering operators used by the proposed transform are obtained. We summarize the chapter in Section 3.5.

3.2 Problem Formulation

Let $\mathbf{X}_{data} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ be the dataset we wish to analyze, where the samples $\mathbf{x}_i \in \mathbb{R}^n$ may be points in high dimension, or feature points associated with the nodes of a weighted graph or network. Also, let $f : \mathbf{X}_{data} \to \mathbb{R}$ be a scalar function defined on the dataset, and let $\mathbf{f} = [f_1, \ldots, f_N]^T$, where $f_i = f(\mathbf{x}_i)$. A key assumption in this work is that under a distance measure $w(\cdot, \cdot)$ in $\mathbb{R}^n$, proximity between the two coordinates $\mathbf{x}_i$ and $\mathbf{x}_j$ implies proximity between their corresponding values f_i and f_j. The distance function describes the first order interaction between data-points, and therefore it should be chosen so as to capture some notion of similarity between them, which would be meaningful to the application at hand. We note that when the dataset is a weighted graph, but no feature points are available, the diffusion map framework [12, 37] can be employed in order to obtain such points from the graph. The Euclidean distance between the

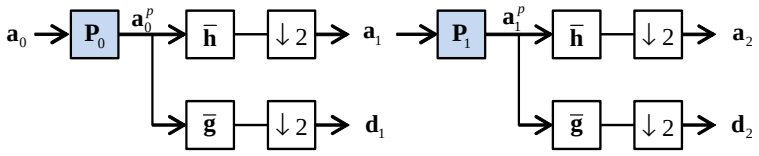


Figure 3.1: Proposed orthonormal wavelet decomposition scheme.

points produced by this framework approximates the "diffusion distance" [12, 37], which describes the relationship between the graph nodes in terms of their connectivity. Now, let $V = \{f \,|\, f : \mathbf{X}_{data} \to \mathbb{R}\}$ be the space of all functions on the dataset $\mathbf{X}_{data}$. Then our goal in this chapter is to develop a multiscale wavelet-like orthonormal basis for the space V, that can efficiently (sparsely) represent the high-dimensional function $\mathbf{f}$.

3.3 Generalized Tree-Based Wavelets

The wavelet transform introduced in this section is an extension of the Haar-like transform proposed by Gavish et al. [30]. Gavish et al. assume in their work that the geometry and structure of the data $\mathbf{X}_{data}$ are captured by one or several hierarchical trees. They do not insist on any specific construction method for these trees, but only that they will be balanced. Given such a tree, they construct a multiscale Haar wavelet-like orthonormal basis for the space V. Here we take a similar approach but construct wavelets with general wavelet filters by modifying the filter bank implementation of the discrete wavelet transform [43].

Figure 3.1 describes the decomposition scheme of the proposed wavelet transform. $\bar{\mathbf{h}}$ and $\bar{\mathbf{g}}$ denote the wavelet decomposition filters of an orthonormal discrete wavelet transform, and $\downarrow 2$ denotes a $2:1$ decimator. $\mathbf{a}_\ell$ and $\mathbf{d}_\ell$ denote the approximation and detail coefficient vectors in the ℓth scale, respectively, where $\mathbf{a}_0 = \mathbf{f}$. The operators $\mathbf{P}_\ell$ make the difference between our proposed wavelet decomposition scheme and the common orthogonal wavelet transform. Each such operator produces a permuted "smoother" version $\mathbf{a}_\ell^p$ of its input vector $\mathbf{a}_\ell$, and these permutations are what make the transform adaptive to its input signal. Also, since each permutation operator may be interpreted as a linear and unitary operator given its input vector, the described transform remains unitary. We

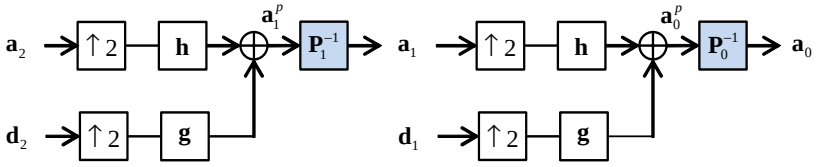


Figure 3.3: Proposed orthonormal wavelet reconstruction scheme.

will explain in the following Section how to obtain the permutation operators from the feature points $\mathbf{x}_i$. Assuming that the operators $\mathbf{P}_\ell$ are known, the algorithm in Figure 3.2 is used to apply an L-level wavelet decomposition.

Task: Apply on $\mathbf{f}$ an L-level tree-based wavelet decomposition.

Parameters: The signal $\mathbf{f}$, the operators $\{\mathbf{P}_\ell\}_{\ell=0}^{L-1}$, and the filters $\bar{\mathbf{h}}$ and $\bar{\mathbf{g}}$.

Initialization: Set $\mathbf{a}_0 = \mathbf{f}$.

Main Iteration: Perform the following steps for $\ell = 0, \ldots, L-1$:

- Apply $\mathbf{P}_\ell$ on $\mathbf{a}_\ell$ and receive $\mathbf{a}_\ell^p$.
- Filter $\mathbf{a}_\ell^p$ with $\bar{\mathbf{h}}$ and decimate the result by 2 to receive $\mathbf{a}_{\ell+1}$.
- Filter $\mathbf{a}_\ell^p$ with $\bar{\mathbf{g}}$ and decimate the result by 2 to receive $\mathbf{d}_{\ell+1}$.

Output: The approximation coefficient vector $\mathbf{a}_L$ and detail coefficient vectors $\{\mathbf{d}_\ell\}_{\ell=1}^{L}$.

Figure 3.2: Proposed orthonormal wavelet decomposition algorithm.

In a similar manner, Figure 3.3 describes the reconstruction scheme of our proposed wavelet transform. $\mathbf{h}$ and $\mathbf{g}$ denote the wavelet reconstruction filters, and $\uparrow 2$ denotes an interpolator. The operator $\mathbf{P}_\ell^{-1}$ reorders a vector so as to cancel the ordering done by $\mathbf{P}_\ell$. Assuming that the operators $\mathbf{P}_\ell^{-1}$ are known, the algorithm in Figure 3.4 reconstructs $\mathbf{f}$ from an L-level decomposition of our proposed transform. We next explain how the operators $\mathbf{P}_\ell^{-1}$ are determined in each level of the proposed transform.

3.4 Building the operators $\mathbf{P}_\ell$

We wish to design the operators $\mathbf{P}_\ell$ in a manner which results in an efficient (sparse) representation of the input signal $\mathbf{f}$ by the proposed transform. The wavelet transform

Task: Reconstruct $\mathbf{f}$ from an L-level tree-based wavelet decomposition.

Parameters: The approximation coefficient vector $\mathbf{a}_L$, the detail coefficient vectors $\{\mathbf{d}_\ell\}_{\ell=1}^{L}$, the operators $\{\mathbf{P}_\ell^{-1}\}_{\ell=0}^{L-1}$, and the filters $\mathbf{h}$ and $\mathbf{g}$.

Main Iteration: Perform the following steps for $\ell = L, \ldots, 1$:

- Interpolate $\mathbf{a}_\ell$ by a factor of 2 and filter the result with $\mathbf{h}$.
- Interpolate $\mathbf{d}_\ell$ by a factor of 2 and filter the result with $\mathbf{g}$.
- Sum the results of the two previous steps to receive $\mathbf{a}_{\ell-1}^p$.
- Apply $\mathbf{P}_{\ell-1}^{-1}$ to $\mathbf{a}_{\ell-1}^p$ and receive $\mathbf{a}_{\ell-1}$.

Output: The reconstructed signal $\mathbf{a}_0 = \mathbf{f}$.

Figure 3.4: Proposed orthonormal wavelet reconstruction algorithm.

is known to produce a small number of large coefficients when it is applied to piecewise regular signals [43]. Thus, we would like the operator $\mathbf{P}_\ell$, applied to $\mathbf{a}_\ell$, to produce a signal which is as regular as possible. We start with the finest level, and try to find the permutation that the operator $\mathbf{P}_0$ applies to $\mathbf{a}_0 = \mathbf{f}$. When the signal $\mathbf{f}$ is known, the optimal solution would be to apply it a simple sort operation. However, we are interested in the case where $\mathbf{f}$ may be corrupted (noisy, contain missing samples, etc.), and any permutation defined by simply reordering the corrupted samples into a regular signal does not necessarily smooth $\mathbf{f}$. Therefore, as the samples in the corrupted signal are not helpful to us, we settle for a suboptimal ordering operation, using the point $\mathbf{x}_i$.

As we mentioned above, we assume that under a distance measure $w(\mathbf{x}_i, \mathbf{x}_j)$, proximity between the two points $\mathbf{x}_i$ and $\mathbf{x}_j$ suggests proximity between the scalar values f_i and f_i. Thus, we shall try to reorder the points $\mathbf{x}_i$ so that they form a smooth path, hoping that the corresponding reordered 1D signal $\mathbf{a}_0^p$ will also be smooth. The "smoothness" of the reordered signal $\mathbf{a}_0^p$ can be measured using its total variation measure

$$\|\mathbf{a}_0^p\|_{TV} = \sum_{j=2}^{N} |a_0^p[j] - a_0^p[j-1]|. \tag{3.1}$$

Let $\{\mathbf{x}_j^p\}_{j=1}^N$ denote the points $\{\mathbf{x}_i\}_{i=1}^N$ in their new order. Then by analogy, we evaluate the "smoothness" of the path through the points $\mathbf{x}_j^p$ by the measure

$$TV(\mathbf{x}_j^p) = \sum_{j=2}^{N} w(\mathbf{x}_j^p, \mathbf{x}_{j-1}^p). \tag{3.2}$$

We note that in case that w is the Euclidean distance, and assuming that f is Lipschitz continuous, i.e., there exists a real constant $K \geq 0$ such that

$$|f_i - f_j| \leq K\|\mathbf{x}_i - \mathbf{x}_j\|_2 \tag{3.3}$$

for every two patches $\mathbf{x}_i$ and $\mathbf{x}_j$, we obtain

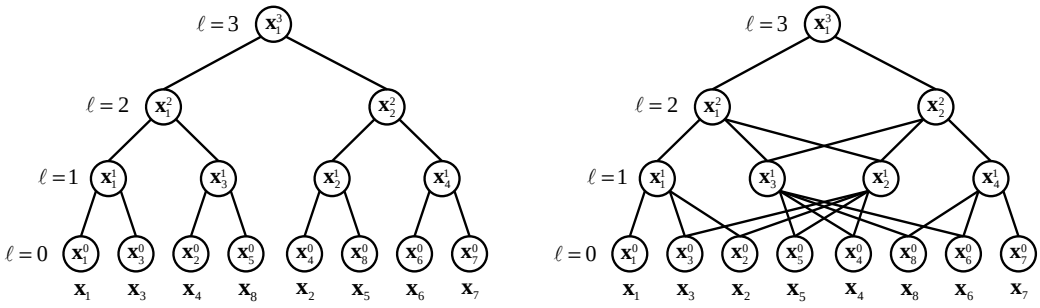
$$\sum_{j=2}^{N} |a_0^p[j] - a_0^p[j-1]| \leq K \sum_{j=2}^{N} \|\mathbf{x}_j^p - \mathbf{x}_{j-1}^p\|_2 \tag{3.4}$$

which means that $K \cdot TV(\mathbf{x}_j^p)$ is an upper bound for $\|\mathbf{a}_0^p\|_{TV}$. Thus, seeking the shortest path with respect to the set $\{\mathbf{x}_j^p\}_{j=1}^N$ does a good service also to the original TV measure in Equation 3.1.

Minimizing $TV(\mathbf{x}_j^p)$ comes down to finding the shortest path that passes through the set of points $\mathbf{x}_i$, visiting each point only once. This can be regarded as an instance of the traveling salesman problem (TSP) [14], which can become very computationally expensive for large sets of points. We choose a simple and crude approximation to the solution, which is to start from a random point and then continue from each point $\mathbf{x}_{j_0}$ to its nearest neighbor $\mathbf{x}_{j_1}$ with a probability $p_1 = \alpha \exp[-w(\mathbf{x}_{j_0}, \mathbf{x}_{j_1})/\epsilon]$, or to its second nearest neighbor $\mathbf{x}_{j_2}$ with a probability $p_2 = \alpha \exp[-w(\mathbf{x}_{j_0}, \mathbf{x}_{j_2})/\epsilon]$, where α is determined such that $p_1 + p_2 = 1$, ϵ is a design parameter, and $\mathbf{x}_{j_1}$ and $\mathbf{x}_{j_2}$ are taken from the set of unvisited points. The permutation applied by the operator $\mathbf{P}_0$ is defined as the order in the found path.

In order to use the aforementioned method to find the operators $\mathbf{P}_\ell$ which are applied to the signals $\mathbf{a}_\ell$ in a scale $\ell > 0$, we are required to associate a set of feature points with these signals. More specifically, we predict the proximity between the samples of the signal $\mathbf{a}_\ell$, by associating a feature point $\mathbf{x}_{\ell,j}$ with each sample $a_\ell[j]$. The calculation of the feature points is carried out in a recursive manner. We use the set of ℓth scale feature points $\{\mathbf{x}_{\ell,j}\}_{j=1}^{2^{-\ell}N}$ to calculate the feature points $\{\mathbf{x}_{\ell+1,j}\}_{j=1}^{2^{-\ell-1}N}$, used in the $\ell+1$th scale. This is done in analogy to the way we obtain the signal $\mathbf{a}_{\ell+1}$ from the signal $\mathbf{a}_\ell$. We first order the feature points $\{\mathbf{x}_{\ell,j}\}_{j=1}^{2^{-\ell}N}$ according to the permutation defined by $\mathbf{P}_\ell$, and place them in a matrix $\mathbf{X}_\ell^p$. We then filter the rows of the result with $\bar{\mathbf{h}}^T$, decimate the columns of the outcome by a factor of two, and obtain the set of feature points $\{\mathbf{x}_{\ell+1,j}\}_{j=1}^{2^{-\ell-1}N}$.

Having calculated the feature points $\mathbf{x}_{\ell+1,j}$, they can be employed by the approximate shortest path search method described above to obtain the operator $\mathbf{P}_{\ell+1}$ in the $\ell+1$th



(a) (b)

Figure 3.5: Illustrations of (a) Complete full binary tree. (b) "Generalized" tree.

scale. The calculation scheme of all the ordering operators $\mathbf{P}_\ell$ which are used in an L-level decomposition of the proposed scheme is summarized in the algorithm in Figure 3.6. We note that the relation between the feature points in a full decomposition can be described using a tree-like structure. When Haar filters are employed, the feature points form a complete full binary tree, such as the one showed in Figure 3.5 (a). However, when filters other than Haar are used in the feature point construction scheme described above, each point in the coarse level ℓ is calculated as a weighted mean of more than two points from the finer level $\ell - 1$, where the coefficients in $\bar{\mathbf{h}}$ serve as the weights. Therefore, the resultant structure is no longer a tree but rather a rooted L-partite graph, which is a graph that contains L disjoint sets of vertices so that no two vertices within the same set are adjacent. In order to avoid cumbersome distinction between trees and L-partite graphs, with a small abuse of terminology we will hereafter refer to the latter also as trees. Figure 3.5 (b) shows an example of a "generalized" tree, which may be obtained for a data set of length $N = 8$, using a filter $\bar{\mathbf{h}}$ of length 4 and disregarding boundary issues in the different levels. We term the filtering schemes described in the algorithms in Figures 3.2 and 3.4, combined with the tree construction algorithm described in Figure 3.6 generalized tree-based wavelet transform (GTBWT).

Before concluding this section, we evaluate the computational complexity of the proposed transform. We first recall that the complexity of applying the common 1D wavelet transform is $O(N)$. Let $\tilde{N} = 2^{-\ell}N$, then the number of distance calculations required to obtain the operator $\mathbf{P}_\ell$ in the ℓth scale is

$$(\tilde{N} - 1) + (\tilde{N} - 2) + \cdots + 2 = \frac{1}{2}[\tilde{N}^2 - \tilde{N} - 2]. \tag{3.5}$$

Task: Calculate all the operators $\{\mathbf{P}_\ell\}_{\ell=0}^{L-1}$, used in an L-level decomposition of the proposed transform.

Parameters: The points $\{\mathbf{x}_j\}_{j=1}^{N}$, the distance function w, and the filter $\bar{\mathbf{h}}$.

Initialization: Set $\mathbf{x}_j^0 = \mathbf{x}_j$.

Main Iteration: Perform the following steps for $\ell = 0, \ldots, L-1$:

- Set $\tilde{N} = 2^{-\ell}N$.

  Calculate $\mathbf{P}_\ell$:
- Choose a random index $1 \leq j \leq \tilde{N}$ and set $\Omega(1) = \{j\}$.
- Perform the following steps for $k = 1 \ldots \tilde{N} - 1$:
  - Find $\mathbf{x}_{j_1}$ – the nearest neighbor to $\mathbf{x}_{\Omega(k)}$ such that $j_1 \notin \Omega$.
  - Find $\mathbf{x}_{j_2}$ – the second nearest neighbor to $\mathbf{x}_{\Omega(k)}$ such that $j_2 \notin \Omega$.
  - Set $\Omega(k+1)$ to be:
    - $\{j_1\}$ with probability $p_1 \propto \exp\left[-\frac{w(\mathbf{x}_{\Omega(k)}, \mathbf{x}_{j_1})}{\epsilon}\right]$.
    - $\{j_2\}$ with probability $p_2 = 1 - p_1 \propto \exp\left[-\frac{w(\mathbf{x}_{\Omega(k)}, \mathbf{x}_{j_2})}{\epsilon}\right]$.
- The set Ω holds the ordering applied by $\mathbf{P}_\ell$.

  Calculate $\ell+1$th scale feature points:
- Order the feature points $\{\mathbf{x}_{\ell,j}\}_{j=1}^{\tilde{N}}$ according to the permutation defined by P_ℓ.
- Place the reordered points the columns of a matrix $\mathbf{X}_\ell^p$.
- Obtain the points $\{\mathbf{x}_{\ell+1,j}\}_{j=1}^{\tilde{N}}$ by:
  - applying the filter $\bar{\mathbf{h}}^T$ to the matrix $\mathbf{X}_\ell^p$.
  - decimating the columns of the outcome by a factor of 2.

Output: The operators $\{\mathbf{P}_\ell\}_{\ell=0}^{L-1}$.

Figure 3.6: Calculation of all the reordering operators used in an L-level decomposition of the proposed transform.

Thus, an L-level GTBWT decomposition requires

$$\frac{1}{2}\sum_{\ell=0}^{L-1}[\tilde{N}^2 - \tilde{N} - 2] = \frac{2N^2}{3}(1 - 4^{-L}) - N(1 - 2^{-L}) - L \tag{3.6}$$

distance calculations. Now, assuming that $O(n)$ operations are required to calculate the distance between two feature points, we obtain that the complexity of an L-level GTBWT decomposition is $O(N) + O(N^2 n) = O(N^2 n)$. Therefore, as might have been expected,

the computational complexity of the GTBWT decomposition scheme is dominated by the calculation of the GTBWT permutation operators. On the other hand, the reconstruction of a signal from the transform coefficients requires only $O(N)$ operations, as in the case of the common 1D wavelet transform.

3.5 Conclusion

We have proposed a new wavelet transform applicable to graphs and high dimensional data. This transform is an extension of the multiscale harmonic analysis approach proposed by Gavish et al. [30], which uses Haar-like wavelets. We have designed our transform to use general wavelet filters by adding data-driven operators into the common orthogonal wavelet transform decomposition scheme. In Chapter 5 we will demonstrate the ability of the proposed transform to represent images more efficiently than the common 1D and separable 2D wavelet transforms. We will also show that our proposed scheme can be used for image denoising, and that combined with a subimage averaging scheme it achieves denoising results which are close to the state-of-the-art.

Chapter 4

Redundant Tree-Based Wavelet Transform[1]

4.1 Introduction

In Chapter 3, we introduced the generalized tree-based wavelet transform (GTBWT), which is a critically sampled (in fact, unitary) wavelet transform applicable to functions defined on irregularly sampled grid of coordinates. The main limitation of this transform is sensitivity to translation. Indeed, in Chapter 5 we will demonstrate that in order to obtain a smooth image denoising result with an image-derived GTBWT, we need to utilize a redundant representation obtained by applying several random variants of the transform to the noisy image. This approach is effectively similar to applying a redundant transform to the image in a rather cumbersome and computationally intensive manner.

In this chapter, we introduce a redundant tree-based wavelet transform (RTBWT), which extends the redundant wavelet transform [2, 35, 43, 54] to scalar functions defined on high-dimensional data clouds, graphs and networks. We first obtain the transform by modifying an implementation of the redundant wavelet transform, which was proposed by Shensa [54] and Beylkin [2], similarly to the way we modified the decomposition scheme of the orthonormal transform in Chapter 3. This implementation employs a filter-bank decomposition scheme, similarly to the orthonormal discrete wavelet transform. However, in each level of this scheme none of the coefficients are discarded. We add in each de-

[1] This chapter is based on [49, 51].

composition level operators that reorder the approximation coefficients. These operators are data-dependent, and are obtained using tree-like structures constructed from the data points. Each reordering operator is derived by organizing the tree-node features in the corresponding level of the tree so as to shorten the path that passes through these points, essentially obtaining an approximation to the solution of the traveling salesman problem (TSP) [14]. The reordering operators increase the regularity of the permuted approximation coefficients signals, which cause their representation with the proposed wavelet transform to be more efficient (sparse).

We next introduce a simpler implementation of the RTBWT, which is based on the widely known "à trous" algorithm [43], proposed by Holschneider et al. [35]. This algorithm is a different implementation of the redundant wavelet transform that applies in each decomposition level upsampled versions of the scaling and wavelet filters to the whole approximation coefficient vectors. Thus, our scheme essentially adds to each decomposition level of this transform a single permutation operator that reorders the approximation coefficients before they are filtered, leading to the "à trous" implementation of our transform. We use this scheme and ideas from [29] to show that the RTBWT is a valid frame expansion, with the same bounds as the common redundant wavelet transform. We explore the use of the proposed transform for the recovery of labels defined on point clouds, and show that it outperforms the 1-nearest neighbor (1-NN) classifier.

The chapter is organized as follows: In Section 4.2, we formulate the problem. In Section 4.3, we introduce the proposed redundant tree-based wavelet transform. In Section 4.4 we introduce the transform implementation which is based on the à trous algorithm, and use it to analyze its frame properties. In Section 4.5 we explore the use of the proposed transform to the recovery of labels defined on point clouds, and present experimental results that demonstrate its advantages. We summarize the chapter in Section 4.6.

4.2 Problem Formulation

In this chapter we operate in the same setting as the one described in Chapter 3. Let $\mathbf{X}_{data} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ be the data set such that $\mathbf{x}_i \in \mathbb{R}^n$ are points in high-dimension, or feature points associated with the nodes of a weighted graph or network. Also,

let $f : \mathbf{X}_{data} \rightarrow \mathbb{R}$ be a scalar function defined on the above coordinates, and let $\mathbf{f} = [f_1, \ldots, f_N]^T$, where $f_i = f(\mathbf{x}_i)$. Here we also assume that under a distance measure $w(\cdot, \cdot)$ in $\mathbb{R}^n$, proximity between the two coordinates $\mathbf{x}_i$ and $\mathbf{x}_j$ implies proximity between their corresponding values f_i and f_j . Our goal in this chapter is to develop a *redundant* wavelet transform that can efficiently represent the high-dimensional function $\mathbf{f}$. Efficiency here implies sparsity, i.e. representing $\mathbf{f}$ accurately with as few as possible wavelet coefficients.

4.3 Redundant Tree-Based Wavelet Transform

4.3.1 Decomposition and Reconstruction schemes

We wish to develop a redundant wavelet transform that efficiently (sparsely) represents its input signal $\mathbf{f}$, defined on an irregularly sampled grid of coordinates. To this end, we extend the redundant wavelet transform, similarly to the way we extended the orthonormal transform in Chapter 3. We first construct our proposed transform by modifying an implementation of the redundant wavelet transform as proposed by Shensa [54] and Beylkin [2]. This implementation employs a filter-bank decomposition scheme, similarly to the orthonormal discrete wavelet transform. However, in each level of this scheme all the coefficients are retained since the highpass bands do not contain decimators, and the decimation in the lowpass bands is replaced by a split into even and odd sequences, which are further decomposed in the next decomposition level. In Section 4.4 we will introduce an alternative implementation for our proposed transform which is based on the à trous algorithm.

Figure 4.1 describes the decomposition scheme of our proposed redundant wavelet transform, which we term redundant tree-based wavelet transform (RTBWT). The filters $\bar{\mathbf{h}}$ and $\bar{\mathbf{g}}$ are the scaling and wavelet decomposition filters of an orthonormal discrete wavelet transform, and they are applied using cyclic convolution. $\downarrow 2^o$ and $\downarrow 2^e$ denote $2:1$ decimators that keep the odd and even samples of their input, respectively. $\mathbf{a}_\ell$ and $\mathbf{d}_\ell$ denote the approximation and detail coefficient vectors in the ℓth scale, respectively, where $\mathbf{a}_0 = \mathbf{f}$. We note that for $\ell > 1$ these signals do not appear explicitly in the

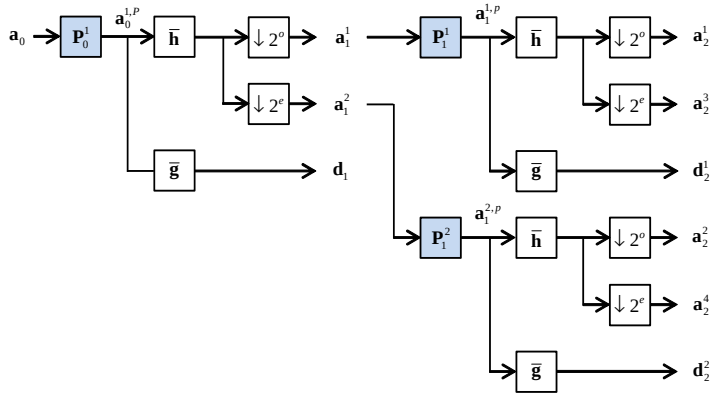


Figure 4.1: Proposed redundant wavelet decomposition scheme.

decomposition scheme, and instead it employs the signals denoted by $\mathbf{a}_\ell^s$ and $\mathbf{d}_\ell^s$, which contain subsets of the samples in $\mathbf{a}_\ell$ and $\mathbf{d}_\ell$, respectively. $\mathbf{a}_\ell^s$ is obtained from $\mathbf{a}_\ell$ by starting from the sth sample, and keeping every 2^ℓth sample. $\mathbf{d}_\ell^s$ is obtained from $\mathbf{d}_\ell$ by starting from the sth sample, and keeping every $2^{\ell-1}$th sample. For example, for signals $\mathbf{a}_2$ and $\mathbf{d}_2$ of length 16 we get that

$$\begin{aligned}
\mathbf{a}_2^1 &= [a_2[1], a_2[5], a_2[9], a_2[13]]^T \\
\mathbf{a}_2^2 &= [a_2[2], a_2[6], a_2[10], a_2[14]]^T \\
\mathbf{a}_2^3 &= [a_2[3], a_2[7], a_2[11], a_2[15]]^T \\
\mathbf{a}_2^4 &= [a_2[4], a_2[8], a_2[12], a_2[16]]^T \\
\mathbf{d}_2^1 &= [d_2[1], d_2[3], \ldots, d_2[15]]^T \\
\mathbf{d}_2^2 &= [d_2[2], d_2[4], \ldots, d_2[16]]^T .
\end{aligned} \tag{4.1}$$

We notice that $\mathbf{a}_0^1 = \mathbf{a}_0$ and $\mathbf{d}_1^1 = \mathbf{d}_1$.

The operators $\mathbf{P}_\ell^s$ make the difference between our proposed wavelet decomposition scheme and the common redundant wavelet transform [2,54]. Each such operator produces a permuted version $\mathbf{a}_\ell^{s,p}$ of its input vector $\mathbf{a}_\ell^s$, and it may be interpreted as a linear and unitary operator given that vector. These operators "smooth" the approximation coefficient signals in the different levels of the decomposition scheme. In Section 4.3.2 we explain how to obtain these operators from the data points $\mathbf{x}_i$. Assuming that the operators $\mathbf{P}_\ell^s$ are known, the algorithm in Figure 4.2 is used to apply an L-level RTBWT decomposition.

Task: Apply on $\mathbf{f}$ an L-level RTBWT decomposition.

Parameters: The signal $\mathbf{f}$, the operators $\{\mathbf{P}_\ell^s\}_{s=1}^{2^\ell}$, $\ell = 0, \ldots, L-1$, and the filters $\bar{\mathbf{h}}$ and $\bar{\mathbf{g}}$.

Initialization: Set $\mathbf{a}_0^1 = \mathbf{a}_0 = \mathbf{f}$.

Main Iteration: Perform the following steps for $\ell = 0, \ldots, L-1$:

- Perform the following steps for $s = 1, \ldots, 2^\ell$:
  - Apply $\mathbf{P}_\ell^s$ on $\mathbf{a}_\ell^s$ and obtain $\mathbf{a}_\ell^{s,p}$.
  - Filter $\mathbf{a}_\ell^{s,p}$ with $\bar{\mathbf{g}}$ and obtain $\mathbf{d}_{\ell+1}^s$.
  - Filter $\mathbf{a}_\ell^{s,p}$ with $\bar{\mathbf{h}}$ and decimate the result with
    - $\downarrow 2^o$ and obtain $\mathbf{a}_{\ell+1}^s$.
    - $\downarrow 2^e$ and obtain $\mathbf{a}_{\ell+1}^{s+2^\ell}$

Output: The approximation coefficient vectors $\mathbf{a}_L$ and detail coefficient vectors $\{\mathbf{d}_\ell\}_{\ell=1}^L$.

Figure 4.2: Proposed redundant wavelet decomposition algorithm.

In a similar manner, Figure 4.3 describes the reconstruction scheme of the proposed redundant wavelet transform. If $\mathbf{h}$ and $\mathbf{g}$ denote the scaling and wavelet reconstruction filters, then $\tilde{\mathbf{h}} = \frac{1}{2}\mathbf{h}$ and $\tilde{\mathbf{g}} = \frac{1}{2}\mathbf{g}$ are applied using cyclic convolution. The upsamplers denoted by $\uparrow 2^o$ and $\uparrow 2^e$ place the samples of their input vector in the odd and even locations of their output vector, respectively. The operator $\tilde{\mathbf{P}}_\ell^s$ reorders a vector so as to cancel the ordering done by $\mathbf{P}_\ell^s$, i.e. $\tilde{\mathbf{P}}_\ell^s = (\mathbf{P}_\ell^s)^{-1} = (\mathbf{P}_\ell^s)^T$. Assuming that the operators $\tilde{\mathbf{P}}_\ell^s$ are known, the algorithm in Figure 4.4 reconstructs $\mathbf{f}$ from an L-level RTBWT decomposition. We next explain how the operators $\mathbf{P}_{\ell,s}^t$ are determined in each level of the RTBWT.

4.3.2 Building the Operators $\mathbf{P}_\ell^s$

We wish to design the operators $\mathbf{P}_\ell^s$ in a manner which results in an efficient (sparse) representation of the input image by the proposed transform. The wavelet transform is known to produce a small number of large coefficients when it is applied to piecewise regular signals [43]. Thus, we would like the operator $\mathbf{P}_\ell^s$, applied to $\mathbf{a}_\ell^s$, to produce a signal $\mathbf{a}_\ell^{s,p}$ which is as regular as possible. We start with the finest level, and try to find

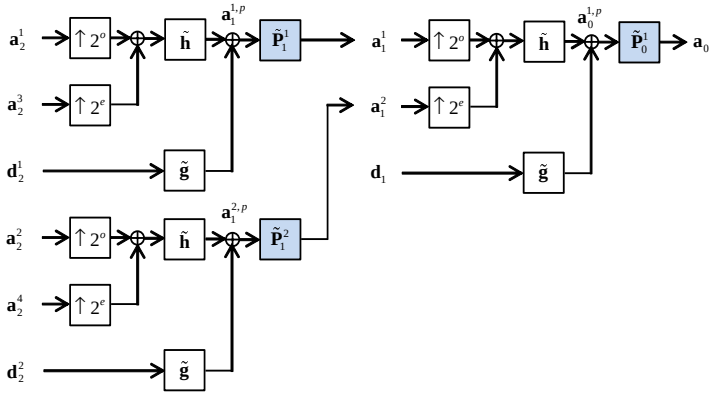


Figure 4.3: Proposed redundant wavelet reconstruction scheme.

Task: reconstruct $\mathbf{f}$ from an L-level RTBWT decomposition.

Parameters: The approximation coefficient vectors $\mathbf{a}_L$ and detail coefficient vectors $\{\mathbf{d}_\ell\}_{\ell=1}^L$, the operators $\{\tilde{\mathbf{P}}_\ell^s\}_{s=1}^{2^\ell}$, $\ell = 0, \ldots, L-1$, and the filters $\tilde{\mathbf{h}}$ and $\tilde{\mathbf{g}}$.

Main Iteration: Perform the following steps for $\ell = L-1, \ldots, 0$:

- Perform the following steps for $s = 1, \ldots, 2^\ell$:
  - Upsample $\mathbf{a}_{\ell+1}^s$ with $\uparrow 2^o$ and $\mathbf{a}_{\ell+1}^{s+2^\ell}$ with $\uparrow 2^e$, sum the results, and filter with $\tilde{\mathbf{h}}$.
  - Filter $\mathbf{d}_{\ell+1}^s$ with $\tilde{\mathbf{g}}$.
  - Sum the results of the two previous steps and obtain $\mathbf{a}_{\ell-1}^{s,p}$.
  - Apply $\tilde{\mathbf{P}}_\ell^s$ on $\mathbf{a}_\ell^{s,p}$ and obtain $\mathbf{a}_\ell^s$.

Output: The reconstructed signal $\mathbf{a}_0 = \mathbf{f}$.

Figure 4.4: Proposed redundant wavelet reconstruction algorithm.

the permutation that the operator $\mathbf{P}_0^1$ applies to $\mathbf{a}_0 = \mathbf{f}$. When the signal $\mathbf{f}$ is known, the optimal solution would be to apply a simple sort operation on it. However, since we are interested in the case where $\mathbf{f}$ may be corrupted (noisy, contain missing samples, etc.), we would try to find a near-optimal ordering operation using the data points $\mathbf{x}_i$, similarly to what we did in Chapter 3.

Let f_i denote the ith sample in the vector $\mathbf{f}$. We recall our assumption that under some distance measure $w(\mathbf{x}_i, \mathbf{x}_j)$, proximity between the two points $\mathbf{x}_i$ and $\mathbf{x}_j$ suggests proximity between the scalar values f_i and f_j. Thus, we shall try to reorder the points

$\mathbf{x}_i$ so that they form a smooth path, hoping that the corresponding reordered 1D signal $\mathbf{a}_0^{1,p} = \mathbf{f}^p$ will also be smooth. The "smoothness" of the reordered signal $\mathbf{a}_0^{1,p}$ can be measured using its total variation measure

$$\|\mathbf{a}_0^{1,p}\|_{TV} = \sum_{j=2}^{N} |a_0^{1,p}[j] - a_0^{1,p}[j-1]|. \tag{4.2}$$

Let $\{\mathbf{x}_j^p\}_{j=1}^N$ denote the patches $\{\mathbf{x}_i\}_{i=1}^N$ in their new order. Then by analogy, we evaluate the "smoothness" of the path through the patches $\mathbf{x}_j^p$ by the measure

$$TV(\mathbf{x}_j^p) = \sum_{j=2}^{N} w(\mathbf{x}_j^p, \mathbf{x}_{j-1}^p). \tag{4.3}$$

Minimizing $TV(\mathbf{x}_j^p)$ comes down to finding the shortest path that passes through the set of points $\mathbf{x}_i$, visiting each point only once. This can be regarded as an instance of the traveling salesman problem (TSP) [14], which can become very computationally exhaustive for large sets of points. We choose a simple and crude approximation to the solution, which is to start from an arbitrary point (random or pre-defined), and continue from each point to its nearest neighbor, not visiting any point twice. The permutation applied by the operator $\mathbf{P}_0^1$ is defined as the order in the found path.

In order to further increase the sparseness of the image representation with RTBWT, we use the aforementioned method to find operators $\{\mathbf{P}_\ell^s\}_{s=1}^{2^\ell}$ which are applied to the signals $\{\mathbf{a}_\ell^s\}_{s=1}^{2^\ell}$ in scales $\ell > 0$. To this end, we are required to associate a set of feature points with these signals. More specifically, we predict the proximity between the samples of the signal $\mathbf{a}_\ell^s$, by associating a feature point $\mathbf{x}_{\ell,j}^s$ with each sample $a_\ell^s[j]$. The calculation of the feature points is carried out in a recursive manner. We use the set of ℓ-th scale feature points $\{\mathbf{x}_{\ell,j}^s\}_{j=1}^{2^{-\ell}N}$ to calculate the two sets of feature points $\{\mathbf{x}_{\ell+1,j}^s\}_{j=1}^{2^{-\ell-1}N}$ and $\{\mathbf{x}_{\ell+1,j}^{s+2^\ell}\}_{j=1}^{2^{-\ell-1}N}$, used in the $\ell+1$th scale. This is done in analogy to the way we obtain the signals $\mathbf{a}_{\ell+1}^s$ and $\mathbf{a}_{\ell+1}^{s+2^\ell}$ from the signal $\mathbf{a}_\ell^s$. We first order the feature points $\{\mathbf{x}_{\ell,j}^s\}_{j=1}^{2^{-\ell}N}$ according to the permutation defined by $\mathbf{P}_\ell^s$, and place them in the columns of an $n \times 2^{-\ell}N$ matrix $\mathbf{X}_\ell^{s,p}$. We then filter the rows of the result with $\bar{\mathbf{h}}^T$, and obtain the sets of feature points $\{\mathbf{x}_{\ell+1,j}^s\}_{j=1}^{2^{-\ell-1}N}$ and $\{\mathbf{x}_{\ell+1,j}^{s+2^\ell}\}_{j=1}^{2^{-\ell-1}N}$ by keeping only the odd and even columns of the resulting matrix, respectively.

Having calculated the feature points $\mathbf{x}_{\ell+1,j}^s$, they can be fed to the approximate shortest path search method described above to obtain each of the operators $\mathbf{P}_{\ell+1}^s$ in the $\ell+$

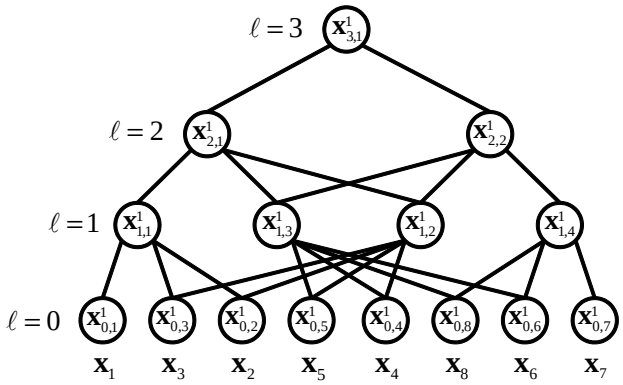


Figure 4.5: An illustration of a "generalized" tree.

1th scale. The calculation scheme of all the ordering operators $\mathbf{P}_\ell^s$ that are used in an L-level RTBWT decomposition is summarized in the algorithm in Figure 4.6. We note that similarly to the decomposition scheme of the generalized tree-based wavelet transform (GTBWT) described in Chapter 3, the relation between the feature points in a full RTBWT decomposition can be described using tree-like structures. Each such "generalized" tree contains all the feature points which have participated in the calculation of a single feature point $\mathbf{x}_{L,j}^s$ from the coarsest scale. Figure 4.5 shows an example of such a "generalized" tree, corresponding to length $N = 8$, using a filter $\bar{\mathbf{h}}$ of length 4 and disregarding boundary issues in the different levels. The construction of these tree-like structures play an integral part in our proposed transform, and this is the reason why we chose to we term it redundant tree-based wavelet transform (RTBWT). We next describe an alternative implementation of the RTBWT which is based on the à trous algorithm, and use it to show the frame properties of the transform.

4.4 À Trous Implementation and Frame Properties

4.4.1 À Trous Implementation

The common and widely known "à trous" algorithm [43], proposed by Holschneider et al. [35], is an alternative implementation for the redundant wavelet transform. Instead of explicitly applying the filters $\bar{\mathbf{h}}$ and $\bar{\mathbf{g}}$ to subsets of the signals $\mathbf{a}_\ell$, the à trous algorithm applies upsampled versions of these filters directly to $\mathbf{a}_\ell$. The upsampled filters

Task: calculate all the operators $\{\mathbf{P}_\ell^s\}_{s=1}^{2^\ell}$, $\ell = 0 \dots L-1$, used in an L-level RTBWT decomposition.

Parameters: The points $\{\mathbf{x}_j\}_{j=1}^N$, the distance function w, and the filter $\bar{\mathbf{h}}$.

Initialization: Set $\mathbf{x}_{0,j}^1 = \mathbf{x}_j$.

Main Iteration: Perform the following steps for $\ell = 0, \dots, L-1$:

- Set $\tilde{N} = 2^{-\ell}N$.
- Perform the following steps for $s = 1, \dots, 2^\ell$:

  Calculate $\mathbf{P}_\ell^s$:

  - Choose a random index $1 \le j \le \tilde{N}$ and set $\Omega(1) = \{j\}$.
  - Perform the following steps for $k = 1, \dots, \tilde{N}-1$:
    - Find $\mathbf{x}_{\ell,j_1}^s$ – the minimizer of $w(\mathbf{x}_{\ell,j}^s, \mathbf{x}_{\ell,\Omega(k)}^s)$ such that $j_1 \notin \Omega$.
    - Set $\Omega(k+1)$ to be j_1.
  - The set Ω holds the ordering applied by $\mathbf{P}_\ell^s$.

  Calculate $\ell+1$th scale feature points:

  - Order the feature points $\{\mathbf{x}_{\ell,j}^s\}_{j=1}^{\tilde{N}}$ according to the permutation defined by $\mathbf{P}_\ell^s$.
  - Place the reordered points in the columns of a matrix $\mathbf{X}_\ell^{s,p}$.
  - Apply the filter $\bar{\mathbf{h}}^T$ to the rows of $\mathbf{X}_\ell^{s,p}$ and
    - keep the odd columns of the result so as to obtain the points $\{\mathbf{x}_{\ell+1,j}^s\}_{j=1}^{\frac{\tilde{N}}{2}}$.
    - keep the even columns of the result so as to obtain the points $\{\mathbf{x}_{\ell+1,j}^{s+2^\ell}\}_{j=1}^{\frac{\tilde{N}}{2}}$.

Output: The operators $\{\mathbf{P}_\ell^s\}_{s=1}^{2^\ell}$, $\ell = 0, \dots, L-1$.

Figure 4.6: Calculation of all the reordering operators used in an L-level RTBWT decomposition.

are obtained by inserting "holes" (trous in French) between nonzero filter taps. We next show that the RTBWT can also be applied using a modification of the à trous algorithm. We then use this scheme to show that the RTBWT is a frame expansion, with the same bounds as the common redundant wavelet transform.

We first notice that in the aforementioned implementation of the RTBWT decomposition scheme, the operators $\mathbf{P}_\ell^s$ operate in the ℓth scale on disjoint subvectors $\mathbf{a}_\ell^s$ of $\mathbf{a}_\ell$. Therefore, these operators can be replaced by a single operator $\mathbf{P}_\ell$ which operates on $\mathbf{a}_\ell$, and produces the reordered signal $\mathbf{a}_\ell^p$. Let $\mathbf{R}_\ell^s$ be a matrix which extracts the signal $\mathbf{a}_\ell^s$

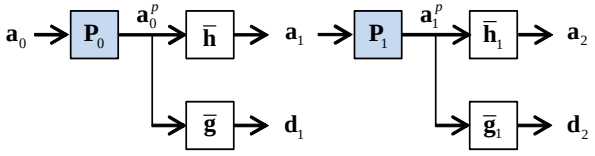


Figure 4.7: RTBWT à trous based decomposition scheme.

from the signal $\mathbf{a}_\ell$. Then we construct the permuted signal $\mathbf{a}_\ell^p$ by extracting all the signals $\{\mathbf{a}_\ell^s\}_{s=1}^{2^\ell}$ from $\mathbf{a}_\ell$, applying to each signal $\mathbf{a}_\ell^s$ the corresponding operator $\mathbf{P}_\ell^s$, and returning all the permuted signals back to their original location. More specifically,

$$\mathbf{a}_\ell^p = \sum_{s=1}^{2^{\ell-1}} (\mathbf{R}_\ell^s)^T \mathbf{P}_\ell^s \mathbf{R}_\ell^s \mathbf{a}_\ell = \mathbf{P}_\ell \mathbf{a}_\ell \tag{4.4}$$

where the matrices $(\mathbf{R}_\ell^s)^T$ return the permuted signals back to their locations. This way we have defined the permutation matrix

$$\mathbf{P}_\ell = \sum_{s=1}^{2^{\ell-1}} (\mathbf{R}_\ell^s)^T \mathbf{P}_\ell^s \mathbf{R}_\ell^s. \tag{4.5}$$

We can use the operators $\mathbf{P}_\ell$ to modify the à trous algorithm, and obtain the à trous implementation of the RTBWT decomposition scheme. Let $\bar{\mathbf{h}}_\ell$ be a filter obtained from $\bar{\mathbf{h}}$ by inserting $2^\ell - 1$ zeros between each sample, where $\bar{\mathbf{h}}_0 = \bar{\mathbf{h}}$. Also, let $\bar{\mathbf{g}}_\ell$ be a filter obtained from $\bar{\mathbf{g}}$ in a similar manner. Both filters are applied using cyclic convolution, which we denote by $\star$. Then assuming that the operators $\mathbf{P}_\ell$ are known, we can apply the RTBWT by repeating for $\ell = 0, \ldots, L-1$ the following filter bank operations, described in Figure 4.7,

$$\mathbf{a}_\ell^p = \mathbf{P}_\ell \mathbf{a}_\ell \tag{4.6}$$

$$a_{\ell+1}[n] = a_\ell^p \star \bar{h}_\ell[n] \tag{4.7}$$

$$d_{\ell+1}[n] = a_\ell^p \star \bar{g}_\ell[n]. \tag{4.8}$$

We note that when the operators $\mathbf{P}_\ell$ are removed, our scheme coincides with the common à trous algorithm.

In a similar manner, the operators $\tilde{\mathbf{P}}_\ell^s = (\mathbf{P}_\ell^s)^{-1}$, which operate on the signals $\mathbf{a}_\ell^{s,p}$ can also be replaced by a single operator $\mathbf{P}_\ell^{-1}$. We use the operators $\mathbf{P}_\ell^{-1}$ to construct

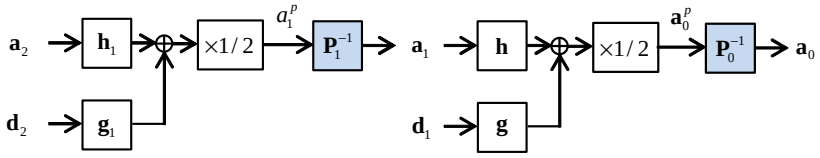


Figure 4.8: RTBWT à trous based reconstruction scheme.

the à trous implementation of the RTBWT reconstruction scheme. Let $\mathbf{h}_\ell$ be a filter obtained from $\mathbf{h}$ by inserting $2^\ell - 1$ zeros between each sample, where $\mathbf{h}_0 = \mathbf{h}$. Also, let $\mathbf{g}_\ell$ be a filter obtained from $\mathbf{g}$ in a similar manner. Both filters are applied using cyclic convolution. Then assuming that the operators $\mathbf{P}_\ell^{-1}$ are known, we can apply the RTBWT reconstruction scheme by repeating for $\ell = L \ldots 1$ the following filter bank operations, described in Figure 4.8,

$$a_\ell^p[n] = \frac{1}{2}\left(a_{\ell+1} \star h_\ell[n] + d_{\ell+1} \star g_\ell[n]\right) \tag{4.9}$$

$$\mathbf{a}_\ell = \mathbf{P}_\ell^{-1}\mathbf{a}_\ell^p. \tag{4.10}$$

4.4.2 Frame Properties

Let $\mathbf{\Phi}$ denote an $(L+1)N \times N$ transform matrix, which applies an L-level RTBWT decomposition. We recall that a sequence $\{\phi_n\}$ is a frame if there exist two constants $\beta \geq \alpha > 0$ such that for all $\mathbf{f}$

$$\alpha\|\mathbf{f}\|^2 \leq \sum_n |\langle \phi_n, \mathbf{f}\rangle|^2 \leq \beta\|\mathbf{f}\|^2. \tag{4.11}$$

The à trous implementation of the RTBWT will enable us to show that the rows of $\mathbf{\Phi}$ constitute a frame $\{\phi_n\}$ in $\mathbb{R}^N$, with the same frame bounds as the common redundant wavelet transform [29].

Proposition 1: Let $\mathbf{c} = \mathbf{\Phi f} = [\mathbf{a}_L^T, \mathbf{d}_L^T, \ldots, \mathbf{d}_1^T]^T$ denote an L-level RTBWT decomposition of a signal $\mathbf{f}$. Then $\mathbf{c}$ is a frame expansion with frame bounds $\alpha = 2$ and $\beta = 2^L$, and these bounds are the tightest possible.

The proof is presented in Appendix 4.A.

Also, let $\mathbf{\Psi}$ denote an $N \times (L+1)N$ matrix that applies the L-level RTBWT reconstruction. $\mathbf{\Phi}$ and $\mathbf{\Psi}$ apply the RTBWT analysis and synthesis operations, respectively, and $\mathbf{\Psi\Phi} = \mathbf{I}$ where $\mathbf{I}$ is the identity matrix. Thus $\mathbf{\Psi}$ is the pseudo inverse of $\mathbf{\Phi}$, and the columns of $\mathbf{\Psi}$ constitute a frame $\{\psi_n\}$ which is dual to $\{\phi_n\}$, i.e. it satisfies

$$\frac{1}{2^L}\|\mathbf{f}\|^2 \leq \sum_n |\langle \psi_n, \mathbf{f}\rangle|^2 \leq \frac{1}{2}\|\mathbf{f}\|^2. \tag{4.12}$$

4.4.3 Computational complexity

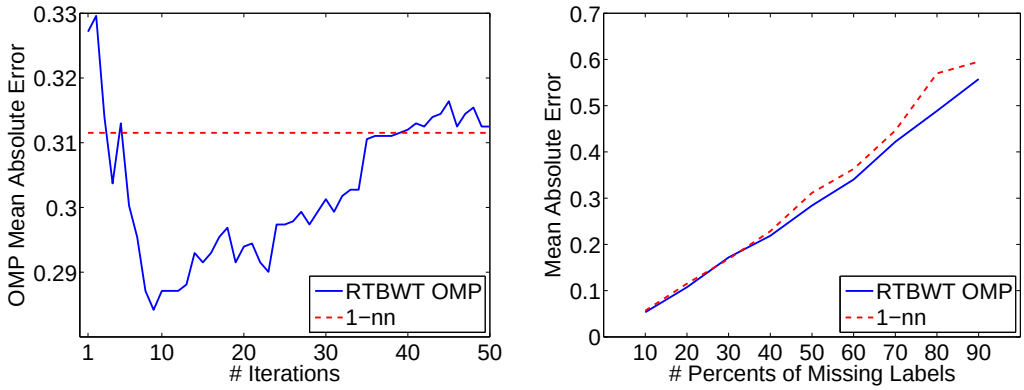
We next evaluate the computational complexity of the proposed transform, and show that it is dominated by the calculation of the RTBWT permutation operators. We first recall that the complexity of applying an L-level decomposition with the common 1D redundant wavelet transform is $O(LN)$. Now, let $\tilde{N} = 2^{-\ell}N$, then the number of distance calculations required to obtain all the 2^ℓ operators $\mathbf{P}_\ell$ in the ℓth scale is $\frac{2^\ell}{2}[\tilde{N}^2 - \tilde{N} - 2]$. Therefore, an L-level RTBWT decomposition requires

$$\frac{1}{2}\sum_{\ell=0}^{L-1} 2^\ell[\tilde{N}^2 - \tilde{N} - 2] = N^2(1-2^{-L}) - \frac{LN}{2} + 1 - 2^L. \tag{4.13}$$

distance calculations. Thus, assuming that $O(n)$ operations are required to calculate the distance between two feature points, we obtain that the complexity of an L-level RTBWT decomposition is $O(LN) + O(N^2 n) = O(N^2 n)$. We next demonstrate the application of our proposed transform to missing label recovery.

4.5 Label recovery using RTBWT

We next demonstrate the advantages of the RTBWT as a tool for representation and processing of functions on point clouds. We conduct several experiments on the white wine quality data set, described in [15]. The data set contains 4898 data points in $\mathbb{R}^{11}$ which describe physicochemical tests obtained for white variants of the Portuguese "Vinho Verde" wine. The target function is the wine quality, determined by experts, on a scale of 0 to 10. We first randomly select 2048 data points to be the points $\mathbf{x}_i$, and construct the vector $\mathbf{f}$ which contains the corresponding quality labels. We then calculate the RTBWT which corresponds to this data set, where we use the squared Euclidean distance to



(a) (b)

Figure 4.9: Label recovery results: (a) RTBWT OMP error vs. iteration number compared to 1-nn error (50% of the labels are missing). (b) RTBWT and 1-NN errors obtained for different percents of missing labels.

measure the dissimilarity between the feature points. Next, we carry out 9 experiments in which we randomly select between 10 to 90 percents of the quality labels, discard them, and employ the dictionary which corresponds to the RTBWT in order to recover them.

The signal $\mathbf{f}$ is recovered by finding the non zero labels a sparse representation in the RTBWT dictionary, using the orthogonal matching pursuit (OMP) algorithm [62]. We compare our results to the ones obtained with the 1-nearest neighbor (1-NN) classifier. Fig 4.9(a) depicts the mean absolute error (MAE) of the signal recovered by the OMP as a function of the iteration number, and the 1-NN classification error, in the case that 50 percents of the labels are missing. It can be seen that the RTBWT requires only 9 dictionary elements to represent the best recovered signal which achieves lower error than the 1-NN classifier. Fig 4.9(b) compares the MAE obtained with the two methods for different percents of missing labels. It can be seen that the RTBWT outperforms the 1-NN classifier in all the experiments but one.

4.6 Conclusion

We have proposed a new redundant wavelet transform applicable to scalar functions defined on graphs and high dimensional data clouds. This transform, which we termed redundant tree-based wavelet transform (RTBWT), is the redundant version of the gen-

eralized tree-based wavelet transform (GTBWT) introduced in Chapter 3. We have introduced an alternative implementation for this transform which is based on the à trous algorithm, and used it to show that the RTBWT is a frame. We have shown that our proposed transform can be used for the recovery of labels defined on point clouds. In Chapters 5 and 6 we will show that our proposed scheme can be used for image denoising and deblurring, and that combined with a subimage averaging scheme it achieves results which are close to the state-of-the-art.

4.A Proof of Proposition 1

Our proof is similar in spirit to the one given in [29] for the common redundant wavelet transform. By the definition of a frame, it is sufficient to show that the frame bounds exist in order to show that the RTBWT is a frame. We notice that $\|\mathbf{c}\|^2 = \sum_n |\langle \phi_n, \mathbf{f} \rangle|^2$, and we next show that the coefficient vector $\mathbf{c}$ satisfies

$$2\|\mathbf{f}\|^2 \leq \|\mathbf{c}\|^2 \leq 2^L \|\mathbf{f}\|^2. \tag{4.14}$$

We first recall that the orthogonal scaling and wavelet filters satisfy [43]

$$\bar{h}[n] = h[-n] \text{ and } \bar{g}[n] = g[-n] \tag{4.15}$$

$$\sum_n h[n] = \sqrt{2} \text{ and } \sum_n g[n] = 0 \tag{4.16}$$

and that the filters $\mathbf{h}$ and $\mathbf{g}$ are power complementary, i.e. their Fourier transforms satisfy

$$|H(2^\ell \omega)|^2 + |G(2^\ell \omega)|^2 = 2. \tag{4.17}$$

We apply the Fourier transform to (4.7) and (4.8), and using (4.15) we obtain that

$$A_{\ell+1}(\omega) = A_\ell^p(\omega) H^*(2^\ell \omega) \tag{4.18}$$

$$D_{\ell+1}(\omega) = A_\ell^p(\omega) G^*(2^\ell \omega). \tag{4.19}$$

Next we use (4.17) and the fact that $\|\mathbf{a}_\ell^p\|^2 = \|\mathbf{a}_\ell\|^2$ and obtain that

$$\begin{aligned} &\|\mathbf{d}_{\ell+1}\|^2 + \|\mathbf{a}_{\ell+1}\|^2 \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} |A_\ell^p(\omega)|^2 \left(|H(2^\ell \omega)|^2 + |G(2^\ell \omega)|^2 \right) d\omega \\ &= 2\|\mathbf{a}_\ell\|^2. \end{aligned} \tag{4.20}$$

We notice that

$$\begin{aligned}\|\mathbf{c}\|^2 &= \|\mathbf{a}_L\|^2 + \|\mathbf{d}_L\|^2 + \sum_{\ell=1}^{L-1} \|\mathbf{d}_\ell\|^2 \\ &= 2\|\mathbf{a}_{L-1}\|^2 + \|\mathbf{d}_{L-1}\|^2 + \sum_{\ell=1}^{L-2} \|\mathbf{d}_\ell\|^2 \end{aligned} \tag{4.21}$$

where we used (4.20) in the transition from the second to the third line. Finally from (4.20) and (4.21) and the fact that $\mathbf{a}_0 = \mathbf{f}$ we get that

$$\|\mathbf{c}\|^2 \leq 2\|\mathbf{a}_{L-1}\|^2 + 2\|\mathbf{d}_{L-1}\|^2 + \sum_{\ell=1}^{L-2} \|\mathbf{d}_\ell\|^2 \tag{4.22}$$

$$\leq 2^2\|\mathbf{a}_{L-2}\|^2 + 2^2\|\mathbf{d}_{L-2}\|^2 + \sum_{\ell=1}^{L-3} \|\mathbf{d}_\ell\|^2 \leq \ldots \leq 2^L\|\mathbf{f}\|^2 \tag{4.23}$$

and that

$$\|\mathbf{c}\|^2 \geq \|\mathbf{a}_{L-1}\|^2 + \|\mathbf{d}_{L-1}\|^2 + \sum_{\ell=1}^{L-2} \|\mathbf{d}_\ell\|^2 \tag{4.24}$$

$$\geq \|\mathbf{a}_{L-2}\|^2 + \|\mathbf{d}_{L-2}\|^2 + \sum_{\ell=1}^{L-3} \|\mathbf{d}_\ell\|^2 \geq \ldots \geq 2\|\mathbf{f}\|^2. \tag{4.25}$$

We now show that the bounds $\alpha = 2$ and $\beta = 2^L$ are the tightest possible frame bounds since we can find vectors that meet them. We first show that the vector $\mathbf{f}_N$, which satisfies $f_N[k] = \frac{1}{\sqrt{N}}$, $k = 1\ldots, N$ meets the upper bound. Since $\mathbf{a}_0 = \mathbf{f}_N$ is a constant signal, we get that $\mathbf{a}_0^{1,p} = \mathbf{a}_0 = \mathbf{f}_N$. Using (4.15) and (4.16) we get that

$$\begin{aligned}\bar{h}_j \star f_N[n] &= \frac{1}{\sqrt{N}} \sum_n \bar{h}[n] = \frac{1}{\sqrt{N}} \sum_n h[n] \\ &= \sqrt{2} f_N[n] \end{aligned} \tag{4.26}$$

and

$$\bar{g}_j \star f_N[n] = \frac{1}{\sqrt{N}} \sum_n \bar{g}[n] = \frac{1}{\sqrt{N}} \sum_n g[n] = 0. \tag{4.27}$$

From (4.26) and (4.27) we get that $\mathbf{a}_1 = \sqrt{2}\mathbf{a}_0 = \sqrt{2}\mathbf{f}_N$ and $\mathbf{d}_1 = \mathbf{0}$, where $\mathbf{0}$ denotes a vector of all zeros. Using similar calculations it can be shown that $\mathbf{a}_\ell = \left(\sqrt{2}\right)^\ell \mathbf{f}_N$ and $\mathbf{d}_\ell = \mathbf{0}$, therefore $\|\mathbf{c}\|^2 = 2^L\|\mathbf{f}_N\|^2$. Now, let $\tilde{\mathbf{f}}_N$ be a vector that satisfies $\tilde{f}_N[k] = \frac{1}{\sqrt{N}}(-1)^k$, $k = 1\ldots, N$. Then using a similar procedure it can be shown that the vector $\mathbf{P}_0^{-1}\tilde{\mathbf{f}}_N$ meets the lower bound.

Chapter 5

Image Processing using Tree-Based Wavelets[1]

5.1 Introduction

In Chapters 3 and 4 we introduced a generalized and a redundant tree-based wavelet transforms, designed to represent scalar functions defined on high-dimensional data clouds and graphs. In this chapter we show that the very same schemes can be used to construct image-adaptive transforms that are highly effective for sparsifying image content. This is obtained by converting the given image into a graph by considering all its overlapped patches as coordinates in high-dimensional space, and referring to them as features of the graph vertices. These are accompanied by their mutual Euclidean distance to define the graph edges, this way tying the vertices to each other. Due to the reliance on patches, both transforms are robust to corruptions in the image, such as additive noise, blur, or missing values, and are able to efficiently represent the unknown target image even when it is calculated from its corrupted version. We demonstrate in this chapter, and also in Chapter 6, that this last property allows image processing algorithms that are based on our proposed transforms to obtain state-of-the-art results.

We start by explaining how to construct from an image the generalized or the redundant tree-based wavelet transforms, in a manner such that they could be used for sparsifying its content. Next we show empirically that the generalized tree-based wavelet

[1] This chapter is based on [50, 51].

transform (GTBWT) is more efficient than both the common 1D and 2D separable wavelet transforms in representing images. Then we explore the application of the GTBWT to image denoising, and show that combined with a proposed subimage averaging scheme, it achieves denoising results similar to the ones obtained with the K-SVD algorithm proposed by Elad an Aharon [26]. We also explore the use of the redundant tree-based wavelet transform (RTBWT) to image denoising, and show that it outperforms the K-SVD based algorithm of Elad and Aharon [6], and achieves denoising results that are similar to those obtained with the BM3D algorithm [7]. We also show that the RTBWT and GTBWT achieve similar denoising results, while the former is computationally less-demanding

The chapter is organized as follows: In Section 5.2 we explain how to construct the tree-based wavelets from images, and also explore the efficiency with which the generalized tree-based wavelet transform represents an image. In Section 5.4 we explore the application of our proposed transforms to image denoising. We also describe the subimage averaging scheme and modifications made in the transforms schemes, and present some experimental results. We summarize the chapter in Section 5.5.

5.2 Image-Derived Tree-Based Wavelets

Let $\mathbf{Y}$ be a grayscale image of size $N_1 \times N_2$ where $N_1 N_2 = N$, and let $\mathbf{y}$ be its column stacked version, i.e. $\mathbf{y}$ is a vector of length N containing individual pixel intensities. We wish to construct from the image $\mathbf{Y}$ the generalized or the redundant tree-based wavelet transforms, in a manner such that they could be used for sparsifying its content. To this end, we first choose the feature points required by the transforms to calculate the permutation operators to be image patches with maximal overlaps. More specifically, let y_i be the i-th sample in $\mathbf{y}$, then we choose the point $\mathbf{x}_i$ associated with it as the column-stacked version of the $\sqrt{n} \times \sqrt{n}$ patch around the location of y_i in the image $\mathbf{Y}$. We also choose the distance function w to be the Euclidean distance, i.e. $w(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|$.

Next, in order to exploit the spatial redundancy inherent to images, we modify the orthonormal and redundant wavelet decomposition schemes, described in Chapters 3 and 4, respectively. When we calculate the permutation operators, we restrict the nearest neighbor searches performed for each patch to a surrounding square neighborhood. More

specifically , let $\mathbf{q}_j$ denote a vector containing the 2D spatial coordinate of the patch $\mathbf{x}_j$ in the image $\mathbf{Y}$. Then in the scale $\ell = 0$ we restrict the nearest neighbor search performed for each patch $\mathbf{x}_j$ to a square neighborhood of size $B \times B$ around $\mathbf{q}_j$. When no unvisited patch remains in that neighborhood, we search for nearest neighbors among all the unvisited patches in the whole image. This restriction decreases the overall computational complexity, and our experiments show that with a proper choice of B (found empirically so as to optimize the results) it also leads to improved image processing results, as it forces more relevant neighbors in the ordering. Further work is needed to set this parameter automatically and perhaps even consider a dynamic search area based on the patch-content. Similarly, in each scale $\ell > 0$ we associate a 2D spatial coordinate with each feature point, where we calculate of the coordinates in a recursive manner. The coordinates in a coarse level are obtained from the ones in the finer level in a similar manner to the way the feature points in a coarse level are obtained from these of a finer level. The only difference is that after we reorder the coordinates and place them in a matrix, we filter the rows of the result with a normalized filter $\bar{\mathbf{h}}_a^T$, which satisfies

$$\bar{h}_a[n] = |\bar{h}[n]| / \sum_k |\bar{h}[k]|, \tag{5.1}$$

so as to guarantee no drift in the coordinates.

We note that every time we advance from one scale to the next, the number of feature points used to calculate a single reordering operator decreases by a factor of two. Therefore, when we move to the coarser level we increase both dimensions of the search area by a factor of $\sqrt{2}$, so as to keep the number of candidate neighbors to consider. Thus, the size of the search area used in the ℓth scale is $2^{\frac{\ell}{2}}B \times 2^{\frac{\ell}{2}}B$. The algorithms in Figures 5.1 and 5.2 summarize the calculation schemes of all the ordering operators that are used in an image-derived L-level GTBWT and RTBWT decompositions, respectively. Figure 5.3 shows a scheme describing an example for the such calculation for a 2-level RTBWT decomposition of an image with 8 pixels.

Task: Calculate all the operators $\{\mathbf{P}_\ell\}_{\ell=0}^{L-1}$, used in an L-level GTBWT decomposition.

Parameters: The image patches $\{\mathbf{x}_j\}_{j=1}^{N}$, their 2D coordinates $\{\mathbf{q}_j\}_{j=1}^{N}$, the distance function w, the filters $\bar{\mathbf{h}}$ and $\bar{\mathbf{h}}_a$, and the search area size B.

Initialization: Set $\mathbf{x}_j^0 = \mathbf{x}_j$.

Main Iteration: Perform the following steps for $\ell = 0, \ldots, L-1$:

- Set $\tilde{B} = 2^{\frac{\ell}{2}} B$ and $\tilde{N} = 2^{-\ell} N$.

  Calculate $\mathbf{P}_\ell$:
- Choose a random index $1 \leq j \leq \tilde{N}$ and set $\Omega(1) = \{j\}$.
- Perform the following steps for $k = 1 \ldots \tilde{N} - 1$:
  - Set A_k to be the set of indices of the coordinates $\mathbf{q}_j^\ell$ which reside inside a $\tilde{B} \times \tilde{B}$ neighborhood around $\mathbf{q}_{\Omega(k)}^\ell$.
  - If $|A_k \setminus \Omega| = 1$, set $\Omega(k+1)$ to be $A_k \setminus \Omega$.
  - Else
    - If $|A_k \setminus \Omega| \geq 2$
      - Find $\mathbf{x}_{j_1}^\ell$ – the nearest neighbor to $\mathbf{x}_{\Omega(k)}^\ell$ such that $j_1 \in A_k \setminus \Omega$.
      - Find $\mathbf{x}_{j_2}^\ell$ – the second nearest neighbor to $\mathbf{x}_{\Omega(k)}^\ell$ such that $j_2 \in A_k \setminus \Omega$.
    - Elseif $|A_k \setminus \Omega| = 0$
      - Find $\mathbf{x}_{j_1}^\ell$ – the nearest neighbor to $\mathbf{x}_{\Omega(k)}^\ell$ such that $j_1 \notin \Omega$.
      - Find $\mathbf{x}_{j_2}^\ell$ – the second nearest neighbor to $\mathbf{x}_{\Omega(k)}^\ell$ such that $j_2 \notin \Omega$.
    - Set $\Omega(k+1)$ to be:
      - $\{j_1\}$ with probability $p_1 \propto \exp\left[-w(\mathbf{x}_{\Omega(k)}^\ell, \mathbf{x}_{j_1}^\ell)/\epsilon\right]$.
      - $\{j_2\}$ with probability $p_2 = 1 - p_1 \propto \exp\left[-w(\mathbf{x}_{\Omega(k)}^\ell, \mathbf{x}_{j_2}^\ell)/\epsilon\right]$.
- The set Ω holds the ordering applied by $\mathbf{P}_\ell$.

  Calculate $\ell+1$th scale feature points:
- Order the feature points $\{\mathbf{x}_j^\ell\}_{j=1}^{\tilde{N}}$ according to the permutation defined by P_ℓ.
- Place the reordered points the columns of a matrix $\mathbf{X}_\ell^p$.
- Obtain the points $\{\mathbf{x}_j^{\ell+1}\}_{j=1}^{\tilde{N}}$ by applying the filter $\bar{\mathbf{h}}^T$ to the matrix $\mathbf{X}_\ell^p$, and decimating the columns of the outcome by a factor of 2.

  Calculate $\ell+1$th scale coordinates:
- Order the coordinates $\{\mathbf{q}_j^\ell\}_{j=1}^{\tilde{N}}$ according to the permutation defined by P_ℓ.
- Place the reordered points the columns of a matrix $\mathbf{Q}_\ell^p$.
- Obtain the points $\{\mathbf{q}_j^{\ell+1}\}_{j=1}^{\tilde{N}}$ by applying the filter $\bar{\mathbf{h}}_a^T$ to the matrix $\mathbf{Q}_\ell^p$, and decimating the columns of the outcome by a factor of 2.

Output: The operators $\{\mathbf{P}_\ell\}_{\ell=0}^{L-1}$.

Figure 5.1: Calculation of all the reordering operators used in an image-derived L-level GTBWT decomposition.

Task: calculate all the operators $\{\mathbf{P}_\ell^s\}_{s=1}^{2^\ell}$, $\ell = 0 \dots L-1$, used in an L-level RTBWT decomposition.

Parameters: The image patches $\{\mathbf{x}_j\}_{j=1}^N$, their 2D coordinates $\{\mathbf{q}_j\}_{j=1}^N$, the distance function w, the filters $\bar{\mathbf{h}}$ and $\bar{\mathbf{h}}_a$, and the search area size B.

Initialization: Set $\mathbf{x}_{0,j}^1 = \mathbf{x}_j$ and $\mathbf{q}_{0,j}^1 = \mathbf{q}_j$.

Main Iteration: Perform the following steps for $\ell = 0, \dots, L-1$:

- Set $\tilde{B} = 2^{\frac{\ell}{2}} B$ and $\tilde{N} = 2^{-\ell} N$.
- Perform the following steps for $s = 1, \dots, 2^\ell$:

  Calculate $\mathbf{P}_\ell^s$:

  - Choose a random index $1 \le j \le \tilde{N}$ and set $\Omega(1) = \{j\}$.
  - Perform the following steps for $k = 1, \dots, \tilde{N}-1$:
    - Set A_k to be the set of indices of the coordinates $\mathbf{q}_{\ell,j}^s$ which reside inside a $\tilde{B} \times \tilde{B}$ neighborhood around $\mathbf{q}_{\ell,\Omega(k)}^s$.
    - If $|A_k \setminus \Omega| > 0$, find $\mathbf{x}_{\ell,j_1}^s$ – the minimizer of $w(\mathbf{x}_{\ell,j}^s, \mathbf{x}_{\ell,\Omega(k)}^s)$ such that $j_1 \in A_k$ and $j_1 \notin \Omega$.
    - Else, find $\mathbf{x}_{\ell,j_1}^s$ – the minimizer of $w(\mathbf{x}_{\ell,j}^s, \mathbf{x}_{\ell,\Omega(k)}^s)$ such that $j_1 \notin \Omega$.
    - Set $\Omega(k+1)$ to be j_1.
  - The set Ω holds the ordering applied by $\mathbf{P}_\ell^s$.

  Calculate $\ell+1$th scale feature points:

  - Order the feature points $\{\mathbf{x}_{\ell,j}^s\}_{j=1}^{\tilde{N}}$ according to the permutation defined by $\mathbf{P}_\ell^s$.
  - Place the reordered points in the columns of a matrix $\mathbf{X}_\ell^{s,p}$.
  - Apply the filter $\bar{\mathbf{h}}^T$ to the rows of $\mathbf{X}_\ell^{s,p}$ and
    - keep the odd columns of the result so as to obtain the points $\{\mathbf{x}_{\ell+1,j}^s\}_{j=1}^{\frac{\tilde{N}}{2}}$.
    - keep the even columns of the result so as to obtain the points $\{\mathbf{x}_{\ell+1,j}^{s+2^\ell}\}_{j=1}^{\frac{\tilde{N}}{2}}$.

  Calculate $\ell+1$th scale coordinates:

  - Order the coordinates $\{\mathbf{q}_{\ell,j}^s\}_{j=1}^{\tilde{N}}$ according to the permutation defined by $\mathbf{P}_\ell^s$.
  - Place the reordered points in the columns of a matrix $\mathbf{Q}_\ell^{s,p}$.
  - Apply the filter $\bar{\mathbf{h}}_a^T$ to the rows of $\mathbf{Q}_\ell^{s,p}$ and
    - keep the odd columns of the result so as to obtain the points $\{\mathbf{q}_{\ell+1,j}^s\}_{j=1}^{\frac{\tilde{N}}{2}}$.
    - keep the even columns of the result so as to obtain the points $\{\mathbf{q}_{\ell+1,j}^{s+2^\ell}\}_{j=1}^{\frac{\tilde{N}}{2}}$.

Output: The operators $\{\mathbf{P}_\ell^s\}_{s=1}^{2^\ell}$, $\ell = 0, \dots, L-1$.

Figure 5.2: Calculation of all the reordering operators used in an image-derived L-level RTBWT decomposition.

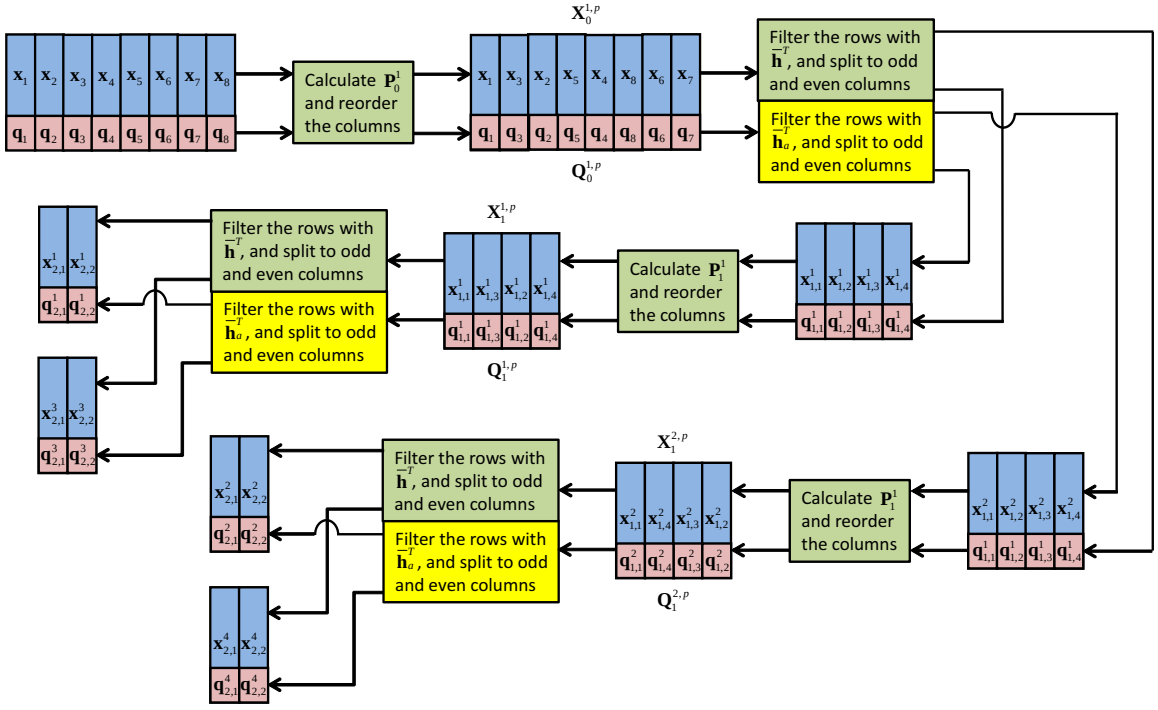


Figure 5.3: Calculation of the reordering operators for a 2-level decomposition of an image with 8 pixels.

5.3 Image Representation using the GTBWT

We next show that the generalized tree-based wavelet transform represents an image $\mathbf{y}$ more efficiently than the common 1D and 2D separable wavelet transforms [43]. Here, we measure efficiency by the m-term approximation error, i.e. the error obtained when representing a signal with m non-zero transform coefficients. We calculate three variants of the GTBWT, corresponding to three different wavelet filters, for the 128×128 image shown in Figure 5.6(a) (the center of the Lena image). In all the experiments carried out in this section, we use 9×9 patches to calculate the reordering operators, where in the nearest neighbor searches performed for each patch we do not restrict the search neighborhood, and always choose to the first nearest neighbor. The three obtained transforms are used to calculate m-term approximations of a column stacked version of this image. The approximations, shown in Figure 5.4, were carried out by keeping the highest coefficients (in absolute value) in the different transform domains. We compare these results between themselves, and to the m-term approximations obtained with the common 1D and the

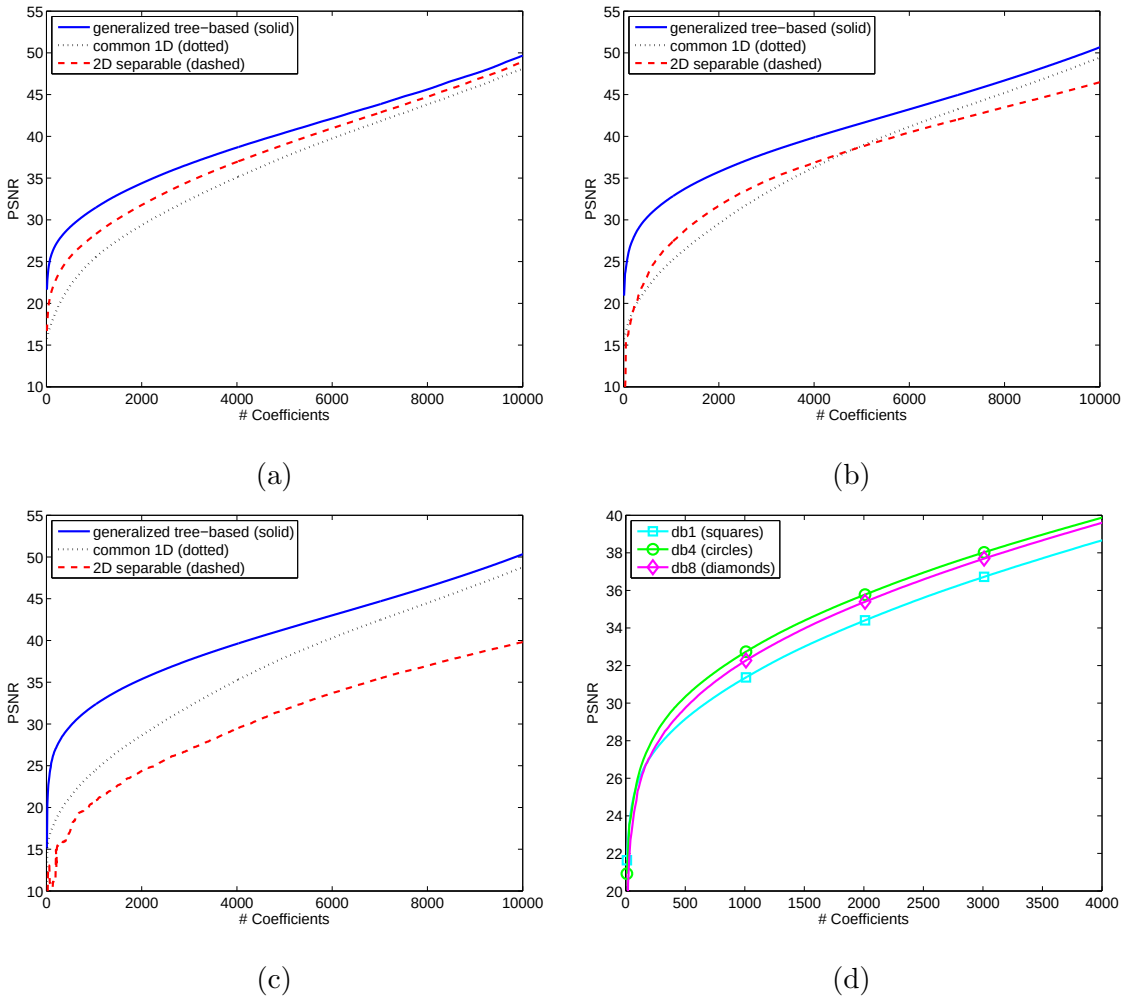


Figure 5.4: m-term approximation results (PSNR in dB) for the generalized tree-based, common 1D and 2D separable wavelet transforms, obtained with different wavelet filters: (a) Daubechies 1 (Haar). (b) Daubechies 4. (c) Daubechies 8. (d) Comparison between the generalized tree-based wavelet results obtained with the different filters.

2D separable wavelet transforms (the latter applied to the original image) corresponding to the same wavelet filters. The quality of the results is measured in peak signal-to-noise ratio (PSNR), defined by

$$PSNR = 20 \log \left[\frac{255}{\sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2}} \right] \tag{5.2}$$

where $\hat{y}_i$ is the estimate of y_i. It can be seen that the generalized tree-based wavelet transform outperforms the two other transforms for all the wavelet filters that have been used. It can also be seen that the PSNR gap in the first thousands of coefficients increases

with number of vanishing moments of the wavelet filters. Further, Figure 5.4(d) shows that the generalized tree-based wavelet results obtained with the db4 and db8 filters are close, and better than the ones obtained with the Haar filter.

Next, we are interested to see how the GTBWT basis elements of look like. Each basis element is obtained by calculating the transform from the input image, setting the corresponding transform coefficient to 1 and all the other coefficients to zero, and applying the inverse transform. Unlike the case of common 1D wavelet bases, the series of data derived permutations applied to the reconstructed coefficients in the different stages of the inverse transform, result in basis functions which adapt themselves to the input image $\mathbf{Y}$.

As a first example, we examine the basis elements obtained for the synthetic image of size 64×64, shown in Fig 5.5(a). The image contains in its middle a square rotated in an angle of 45 degrees. We visualize the basis elements as images by reshaping them to the size of the input image. Figs 5.5(b)-(n) show some of the basis elements corresponding to the synthetic image, and obtained with the Symmlet 8 filter. The figures show two scaling functions from scale $\ell = 12$, and two wavelet basis functions from each scale $\ell = 1, \ldots, 12$, all corresponding to the two largest coefficients in the same scale. We note that the reason we have more than one scaling function and one wavelet basis element in the coarsest scale $\ell = 12$ is related to our implementation of the transform. We use symmetric padding in the signal boundaries before applying the wavelet filters, which slightly increases the number of coefficients (and corresponding basis elements) obtained in each scale. It can be seen that the scaling functions and wavelets in the coarse scales ($\ell = 9, \ldots, 12$) represent the low frequency information in the image, i.e. smooth versions of the rectangle. It can also be seen that the wavelet functions represent finer edges as the scale decreases, and that they successfully manage to capture edges which are not aligned with the vertical and horizontal axes.

We next examine the wavelet basis elements corresponding to the image in Figure 5.6(a), obtained with the Symmlet 8 filter. Figs. 5.6(b)-(p) show two scaling functions from scale $\ell = 14$, and two wavelet basis functions from each scale $\ell = 1, \ldots, 14$, all corresponding to the two largest coefficients in the same scale. It can be seen that the basis functions adapt to the shapes in the images. Here again the scaling functions and

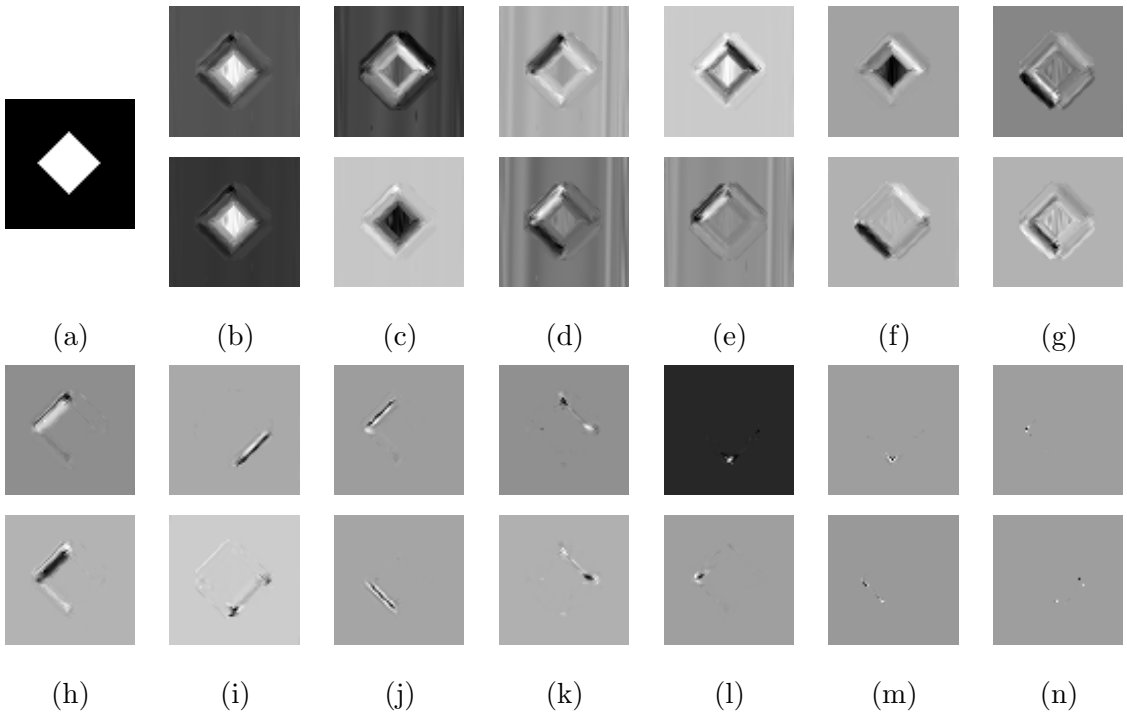


Figure 5.5: Generalized tree-based wavelet basis elements derived from a synthetic image: (a) The original image. (b) Scaling functions (ℓ=12). (c) Wavelets (ℓ=12). (d) Wavelets (ℓ=11). (e) Wavelets (ℓ=10). (f) Wavelets (ℓ=9). (g) Wavelets (ℓ=8). (h) Wavelet (ℓ=7). (i) Wavelet (ℓ=6). (j) Wavelet (ℓ=5). (k) Wavelets (ℓ=4). (l) Wavelets (ℓ=3). (m) Wavelets (ℓ=2). (n) Wavelets (ℓ=1).

wavelets in the coarse scales ($\ell = 11, \ldots, 14$) represent the low frequency information in the image, and the wavelet functions represent finer edges in the image as the scale decreases. We next demonstrate the application of the orthonormal and redundant tree-based wavelet transform to image denoising.

5.4 Image Denoising using Tree-Based Wavelets

5.4.1 The Basics

Let $\mathbf{Y}$ be a grayscale image of size $N_1 \times N_2$ where $N_1 N_2 = N$, and let $\mathbf{Z}$ be its noisy version. Also, let $\mathbf{z}$ and $\mathbf{y}$ be the column stacked versions of $\mathbf{Z}$ and $\mathbf{Y}$, respectively. We assume that the noisy image satisfies

$$\mathbf{Z} = \mathbf{Y} + \mathbf{V}, \tag{5.3}$$

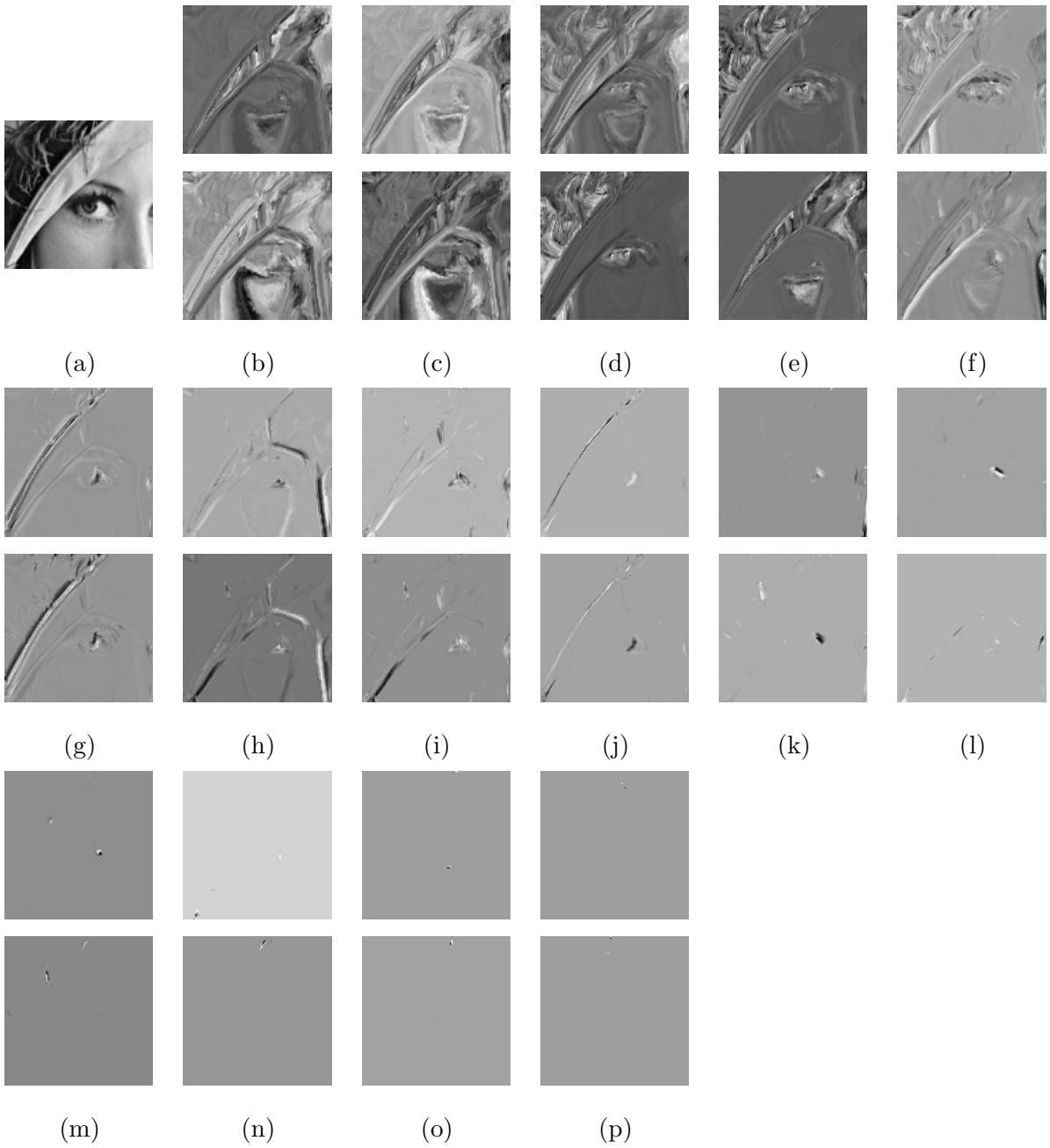


Figure 5.6: Generalized tree-based wavelet basis elements derived from an image: (a) The original image. (b) Scaling functions (ℓ=14). (c) Wavelets (ℓ=14). (d) Wavelets (ℓ=13). (e) Wavelets (ℓ=12). (f) Wavelets (ℓ=11). (g) Wavelets (ℓ=10). (h) Wavelets (ℓ=9). (i) Wavelets (ℓ=8). (j) Wavelets (ℓ=7). (k) Wavelets (ℓ=6). (l) Wavelets (ℓ=5). (m) Wavelets (ℓ=4). (n) Wavelets (ℓ=3). (o) Wavelets (ℓ=2). (p) Wavelets (ℓ=1).

where $\mathbf{V}$ is a matrix of size $N_1 \times N_2$ which denotes an additive white Gaussian noise independent of $\mathbf{Y}$ with zero mean and variance σ^2. Our goal is to reconstruct $\mathbf{y}$ from $\mathbf{z}$

using either the generalized or the redundant tree-based wavelet transforms. To this end, we first extract the feature points $\mathbf{x}_i$ from $\mathbf{Z}$ similarly to the way they were extracted from $\mathbf{Y}$ in the previous section. Let z_i be the i-th sample in $\mathbf{z}$, then here we choose the point $\mathbf{x}_i$ associated with it as the $\sqrt{n} \times \sqrt{n}$ patch around the location of z_i in the image $\mathbf{Z}$. We also choose the distance function w to be the Euclidean distance. We note that since different features are used for the clear and noisy images, the corresponding reordering operators derived from these images will also be different. Nevertheless, it was shown in [6, 55, 59] that the distance between the noisy patches $\mathbf{x}_i$ and $\mathbf{x}_j$ is a good predictor for the similarity between the clear versions of their middle pixels y_i and y_j. This means that the tree construction is roughly robust to noise, and that the obtained trees and corresponding transforms will capture the geometry and structure of the clear image quite well. We will further discuss the noise robustness of the algorithm in Section 5.4.3.

We perform image denoising using the GTBWT or the RTBWT by constructing the chosen transform from the image $\mathbf{z}$, applying it on that image, using hard thresholding on the transform coefficients, and computing the inverse transform. In the case where the GTBWT is used to denoise an image, we use an approach which resembles the "cycle spinning" method proposed by Coifman and Donoho [11] in order to smooth the image denoising outcome, similarly to Gavish et al. [30]. This means that we construct 10 random variants of the transform using the scheme described in Figure 5.1, utilize each transforms to denoise $\mathbf{z}$, and average the produced images.

We next assess the performance of the proposed denoising scheme, applied with the GTBWT. We apply the scheme with 6 different wavelet filters, to a noisy version of the image shown in Figure 5.6(a). We calculate the reordering operators using 9×9 patches, unrestricted search neighborhood for each patch, and $\epsilon = 0.1$. For each of the transforms corresponding to the different wavelet filters, we perform 5 experiments for different realizations of noise with standard deviation $\sigma = 25$, and average the results – these averages are given in Table 5.1. We note that the denoising thresholds were manually found to produce good denoising results, as the theoretical wavelet threshold $\sigma\sqrt{2\log_e N}$ led to poorer results, with PSNR values which were lower by about 0.6 dB. It can be seen that better results are obtained with the Symmlet wavelets, and that generally better

Table 5.1: Denoising results of a noisy version of the image in Figure 5.6(a) ($\sigma = 25$, input PSNR=20.18 dB), obtained using the GTBWT, with and without subimage averaging (SA), and with different wavelet filters. Also shown is the average number of coefficients (#coeffs.) used by each scheme to represent an image.

	GTBWT		with SA	
wavelet type	PSNR [dB]	#coeffs	PSNR [dB]	#coeffs
db1 (Haar)	28.19	303.28	29.05	349.02
db4	28.5	307.62	29.37	341.25
db8	28.49	384.08	29.33	403.2
sym2	28.45	293.84	29.25	326.48
sym4	28.51	308.6	29.38	349.81
sym8	28.55	342.76	29.39	392.77

Table 5.2: Denoising results (PSNR in dB) of noisy versions of the images Barbara and Lena ($\sigma = 25$, input PSNR=20.17 dB) obtained with: 1) GTBWT with a Symmlet 8 filter with subimage averaging (SA) and without it. 2) with the K-SVD algorithm.

Image	GTBWT	GTBWT + SA	K-SVD
Lena	30.3	31.21	31.32
Barbara	28.94	29.82	29.62

results are obtained with wavelets with a high number of vanishing moments. It can also be seen that all the transforms require about 320 coefficients to represent the image (for each of the 10 results produced by the different random realizations) which is about 2 percents of the number of coefficients required in the original space.

We next apply the proposed scheme with the Symmlet 8 wavelet to noisy versions of the images Lena and Barbara, with noise standard deviation $\sigma = 25$ and PSNR of 20.17 dB. We note that this time we use patches of size 13×13, and perform only one experiment for each image, for one realization of noise. The clear, noisy and recovered images can be seen in Figure 5.7. For comparison, we also apply to the two images the denoising scheme

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

(e) (f) (g) (h)

Figure 5.7: Denoising results of noisy versions of the images Barbara and Lena ($\sigma = 25$, input PSNR=20.17 dB) obtained with GTBWT with a Symmlet 8 filter with subimage averaging (SA) and without it: (a) Original Lena. (b) Noisy Lena (20.17 dB). (c) Lena denoised using GTBWT (30.3 dB). (d) Lena denoised using GTBWT with sub image averaging (31.21 dB) (e) Original Barbara. (f) Noisy Barbara (20.17 dB). (g) Barbara denoised using GTBWT (28.94 dB). (h) Barbara denoised using GTBWT with sub image averaging (29.82 dB).

of Elad and Aharon [26], which utilize the K-SVD algorithm proposed by Aharon et al. [1]. We chose to compare our results to the ones obtained by this scheme, as it also employs an efficient representation of the image content, and is one of many state-of-the-art image denoising algorithms [40,41], which are based on sparse and redundant representations [3]. However, we note that this scheme is based on efficient representations of individual fixed size image patches, while our tree-based approach attempts at an image-adaptive efficient representation of the whole image using multiple scales. The PSNR of the results obtained with the proposed scheme and the K-SVD algorithm are shown in Table 5.2. It can be seen that the results obtained by our algorithm are inferior compared to the ones obtained with the K-SVD. We next try to improve the results produced by our proposed scheme by adding an averaging element into it, which is also a variation on the "cycle spinning" method.

5.4.2 Subimage averaging

Let $N_p = (N_1 - \sqrt{n}+1)(N_2 - \sqrt{n}+1)$ denote the number of overlapped $\sqrt{n} \times \sqrt{n}$ patches $\mathbf{x}_i$ in the image $\mathbf{Y}$, and let $\mathbf{X}$ be an $n \times N_p$ matrix, containing column stacked versions of these patches. When we explained in Section 5.2 how to construct the generalized and redundant tree-based wavelet transforms from the image, we assumed that each patch is associated only with its middle pixel. Therefore the obtained transform was associated with the signal composed of the middle points in the patches, which is the middle row of $\mathbf{X}$, and was supposed to be applied to this signal. However, we can alternatively choose to associate all the patches with a pixel located in a different position, for example the top right pixel in each patch. Since the whole patches are used in the construction of the transform, effectively this means that the transform can be associated with any one of the n signals located in the rows of $\mathbf{X}$. These signals are the column stacked versions of the n subimages of size $(N_1 - \sqrt{n}+1) \times (N_2 - \sqrt{n}+1)$, whose top right pixel reside in the top right $\sqrt{n} \times \sqrt{n}$ patch in the image. Thus we can perform denoising using the following scheme which we hereafter refer to as Subimage Averaging (SA). We first calculate the GTBWT or the RTBWT using all the patches inside the image. We next apply the denoising scheme with the chosen wavelet transform to each of the aforementioned n subimages. We then plug each denoised subimage into its original place in the image, and average the different values obtained for each pixel, similarly to the way an image is reconstructed from its patches in the K-SVD image denoising scheme proposed by Elad and Aharon [26].

The subimage averaging scheme can also be viewed a little differently. The "generalized" trees used to construct the GTBWT and RTBWT also define wavelet transforms on the image patches, where the approximation coefficient vectors in each scale are the feature points in the corresponding level of the tree. The detail coefficients vectors are calculated in recursive manner, where the coarse scale vectors are obtained from the approximation coefficients vectors in the finer scale in analogy to the way each of the transforms calculates the coarse scale detail coefficients from the fine scale approximation coefficients. In practice, detail coefficients vectors can simply be obtained by applying the generalized or redundant tree-based wavelet transform to each row of the matrix $\mathbf{X}$.

Similarly, the inverse transform can be performed on the obtained coefficient matrix by applying each of its rows the chosen tree–based wavelet inverse transform. Therefore the subimage averaging scheme can be viewed as applying the transform derived from the tree directly on the image patches, performing hard thresholding on them, applying the inverse transform to the coefficient vectors, and reconstructing the image from the clean patches.

We next apply the proposed denoising scheme using the GTBWT and combined with subimage averaging, to the noisy version of the image in Figure 5.6(a). For the calculation of the GTBWT we again use 9×9 patches, unrestricted search neighborhood for each patch, and $\epsilon = 0.1$. The results are shown in Table 5.1. It can be seen that using subimage averaging increases the PSNR by about 0.85 dB, compared to the results of the basic scheme. We note that since we use 9×9 patches, each of the 10 images averaged in the cycle spinning procedure is reconstructed from 81 subimages, and it can be seen that on average about 360 transform coefficients are required to represent each one of these subimages. Thus the number of transform coefficients required to represent each subimage is slightly higher than the one required to represent an image when the subimage averaging is not used, but is still much lower than the number required in the original space.

We also applied the proposed denoising scheme to the noisy versions of the images Lena and Barbara, with noise standard deviation $\sigma = 25$, where here the GTBWT was calculated with 13×13 patches. The reconstructed images are shown in Figs. 5.7(d) and (h), and Table 5.2 shows the corresponding PSNR results. It can be seen that the results obtained with the proposed algorithm for the Lena image are now closer to the ones received by the K-SVD algorithm, and the results obtained for the Barbara image are better than the ones obtained with the K-SVD algorithm.

We note that the use of the proposed sub-image averaging scheme is not restricted to the our orthogonal and redundant tree-based wavelet transforms. As we will see for example in Chapter 8, this scheme may be used to improve the denoising results obtained with different methods which calculate mutual patch distances similarly to what is done here.

5.4.3 Robustness to noise

We wish to explore the robustness of our proposed denoising scheme to the noise in the points $\mathbf{x}_j$. More specifically we wish to check how using cleaner image patches will affect the denoising results. We obtain "clean" patches from a noisy image by applying our denoising scheme with the GTBWT once, and using the patches from the reconstructed image in a second iteration of the denoising algorithm for defining the permutations. For comparison we also use clean patches from the original clean image in our denoising scheme, and regard the obtained results as oracle estimates.

We start by applying our denoising scheme with the GTBWT to the noisy versions of the images Lena and Barbara, with noise standard deviation $\sigma = 25$, where we calculate the GTBWT using 13×13 patches, unrestricted search neighborhood for each patch, and $\epsilon = 0.1$. We then use the patches obtained from the clean and reconstructed images to calculate two versions of the GTBWT, and apply our denoising scheme with each of these versions to the noisy Lena and Barbara images. The results obtained with the Symmlet 8 filter and with subimage averaging are shown in Table 5.3. It can be seen that large improvements of about 1 dB for the Lena image and 0.8 dB for the Barbara image are obtained with clean patches. Applying 2 iterations of the denoising algorithm slightly improves the results of the Lena image by about 0.13 dB. However, applying 2 iterations of the denoising algorithm to the Barbara image degrades the obtained results by about 0.11 dB. This degradation probably results from oversmoothing of the patches of the reconstructed image, which leads to a loss of details. A choice of a different threshold in the denoising algorithm applied in first iteration, or a different method to clean the patches altogether, may lead to improved denoising results.

5.4.4 Vector Thresholding

We next present one more improvement to the denoising scheme described above which employs subimage averaging. As the subimage averaging scheme can be seen as a transform over that image patches, we propose to use a thresholding operator which operates on coefficient vectors. More specifically, instead of setting the transform coefficients of individual subimages to zero, this operator sets entire coefficient vectors to zero if their

Table 5.3: Denoising results (PSNR in dB) of noisy versions of the images Barbara and Lena ($\sigma = 25$, input PSNR=20.17 dB) obtained using different versions of the GTBWT with subimage averaging. The different transforms were constructed using patches obtained from the noisy (1 iter), original (clean) and reconstructed (2 iters) images.

Image	1 iter	2 iters	clean
Lena	31.21	31.34	32.25
Barbara	29.82	29.71	30.61

Table 5.4: Denoising Results for the Images Barbara and Lena Obtained with the K-SVD (top left), BM3D (top right), GTBWT (bottom left) and RTBWT (bottom right) Algorithms. For Each Image and Noise Level the Best Result is Highlighted.

σ/PSNR	Lena		Barbara	
10/20.18	35.51	**35.93**	34.44	**34.98**
	35.87	35.88	34.94	34.93
25/28.14	31.36	32.08	29.57	30.72
	32.16	**32.17**	30.75	**30.76**

norm is smaller than a threshold T.

In order to assess the performance of the proposed image denoising scheme we apply it both with the GTBWT and the RTBWT to noisy versions of the images Lena and Barbara, with noise standard deviations $\sigma = 10, 25$. We use the Symmlet 8 wavelet filter and patch sizes $\sqrt{n}$ that range from 7 to 15. Also, while in all the previous experiments we set the search neighborhood of each patch to be the entire image, in the following experiments we will finally demonstrate the performance gain obtained by restricting it to a smaller size. We use 10 random variants of a full GTBWT decomposition, and a single 9 level RTBWT decomposition, which corresponds to a redundancy factor of 10. The noisy and recovered Barbara images corresponding to $\sigma = 25$ can be seen in Figure 5.8. For comparison, we also apply to the two images the K-SVD based algorithm proposed by Elad and Aharon [26], and the BM3D algorithm proposed by Dabov et al. [16]. The

(a) (b)

Figure 5.8: Denoising results for the image Barbara ($\sigma = 25$): (a) Noisy Barbara (20.18 dB). (b) Barbara denoised using RTBWT (30.76 dB).

PSNR of the results obtained with all the four denoising schemes are shown in Table 5.4. It can be seen that the results obtained with the RTBWT and the GTBWT are almost identical, better than the ones obtained with the K-SVD algorithm, and close to the ones obtained with the BM3D algorithm. However, the RTBWT was about 6 times faster than the GTBWT since it required much less distance calculations.

5.5 Conclusion

We have explained how to construct the generalized and redundant tree-based wavelet transforms from images. We demonstrated the ability of the generalized tree-based wavelet transform (GTBWT) to represent images more efficiently than the common 1D and separable 2D wavelet transforms. We have also shown that our proposed transforms can be used for image denoising, and that combined with a subimage averaging and vector thresholding scheme they achieves denoising results which are close to the state-of-the-art.

Chapter 6

RTBWT Frames[1]

6.1 Introduction

In Chapter 5 we showed that the redundant tree-based wavelet transform (RTBWT), proposed in Chapter 4, can be used as an image-adaptive transform that is highly effective for sparsifying image content. This is obtained by converting the given image into a graph by considering all its overlapped patches as coordinates in high-dimensional space, and referring to them as features of the graph vertices. These are accompanied by their mutual Euclidean distance to define the graph edges, this way tying the vertices to each other. Due to the reliance on patches, the RTBWT is robust to corruptions in the image, such as additive noise, blur, or missing values, and is able to efficiently represent the unknown target image even when it is calculated from its corrupted version. As we shall see hereafter, in this work we will make use of this last property, and demonstrate the use of the RTBWT as a powerful sparsity-promoting regularizer in inverse problems in image processing.

In Chapter 5 we observed that the performance of the proposed wavelet-based denoising algorithms improve when a subimage averaging scheme is used. For algorithms using the RTBWT, this consists of applying the transform to different subimages of the treated image and then jointly operating on the transform coefficients of all the subimages. We refer to all the transform coefficients of the different subimages as a single extended representation of the treated image, and next take a path similar to the one described in [18]

[1]This chapter is based on [49].

for the BM3D algorithm. We construct matrices which act as analysis and synthesis operators [28], and are used to obtain this extended representation and reconstruct the image from it. We then show that these matrices constitute a frame and its dual. We explore the use of these operators in image denoising and deblurring algorithms. Despite the fact that the resulting transform is image dependent, we treat it as a fixed linear operator and use it within a sparsity-promoting regularizer, when handling image-processing tasks posed as inverse problems. We demonstrate state-of-the-art results in denoising and deblurring using this approach.

The chapter is organized as follows: In Section 6.2, we formulate the problem. In Section 6.3, we introduce the RTBWT subimage averaging frame, and analyzes its properties. In Section 6.4, we explore the use of this frame to image denoising and deblurring, and present experimental results that demonstrate their advantages. We summarize the chapter in Section 6.5.

6.2 Problem Formulation

Let $\mathbf{Y}$ be an image of size $N_1 \times N_2$ where $N_1 N_2 = N$, and let $\mathbf{Z}$ be its corrupted version. Also, let $\mathbf{z}$ and $\mathbf{y}$ be the column stacked versions of $\mathbf{Z}$ and $\mathbf{Y}$, respectively. Then we assume that the corrupted image is obtained via

$$\mathbf{z} = \mathbf{H}\mathbf{y} + \mathbf{v}, \tag{6.1}$$

where the $N \times N$ matrix $\mathbf{H}$ denotes a linear operator that corrupts the data, and $\mathbf{v}$ denotes an additive white Gaussian noise independent of $\mathbf{y}$ with zero mean and variance σ^2. This setting can be used to describe several of the classic image inverse problems such as image denoising, deblurring, and inpainting. Our goal is to reconstruct $\mathbf{y}$ from $\mathbf{z}$ by optimizing an objective function that uses sparsity promoting regularizers which are based on the RTBWT. We next introduce the RTBWT-based analysis and synthesis operators $\mathbf{\Phi}_{SA}$ and $\mathbf{\Psi}_{SA}$, which we will later use as such sparsity promoting regularizers.

6.3 RTBWT Subimage Averaging Frames

In order to recover a corrupted image, we can calculate for it the RTBWT, apply the obtained transform to that image, and then process it in the transform domain. However, we observed in Chapter 5 that improved results are obtained when a subimage averaging scheme is used instead. This scheme consists of applying the RTBWT to different subimages of the treated image and processing each subimage in the transform domain. Then the reconstructed image is obtained by plugging each processed subimage into its original place in the image canvas, and averaging the different values obtained for each pixel. As we show next, we can describe the transform coefficients of the different subimages as a single extended representation of the image. We construct matrices that act as analysis and synthesis operators, which are used to obtain this extended representation and reconstruct the image from it. We also show that the rows and columns of the analysis and synthesis matrices, respectively, constitute a frame and its dual.

Let $N_p = (N_1 - \sqrt{n} + 1)(N_2 - \sqrt{n} + 1)$ denote the number of overlapped patches in the image $\mathbf{Y}$, and let $\mathbf{X}$ be an $n \times N_p$ matrix, containing column stacked versions of these patches. We extract these patches column by column, starting from the top left-most patch. When we explained in the Chapter 5 how to construct the RTBWT from an image, it was assumed that each patch is associated only with its middle pixel. Therefore the transform was designed to efficiently represent the signal composed of the middle points in the patches, that reside in the middle row of $\mathbf{X}$. However, we can alternatively choose to associate all the patches with a pixel located in a different position, e.g., the top left pixel in each patch. This means that the transform can be used to represent any one of the signals located in the rows of $\mathbf{X}$. These signals are the column stacked versions of all the n subimages of size $(N_1 - \sqrt{n} + 1) \times (N_2 - \sqrt{n} + 1)$ contained in the image $\mathbf{Y}$. We denote these subimages by $\tilde{\mathbf{Y}}_j, j = 1, 2, \ldots, n$.

Let $\mathbf{\Phi}$ denote the RTBWT transform matrix, here it is of size $(L+1)N_p \times N_p$, applying an L-level decomposition. Also, let the vector $\tilde{\mathbf{y}}_j = \mathbf{R}_j\mathbf{y}$ of length N_p be the column stacked version of $\tilde{\mathbf{Y}}_j$, where the $N_p \times N$ matrix $\mathbf{R}_j$ extracts the jth subimage from the image $\mathbf{y}$. We first apply the RTBWT to each of the n subimages $\tilde{\mathbf{y}}_j$ and obtain the n

vectors

$$\mathbf{c}_j = \mathbf{\Phi}\tilde{\mathbf{y}}_j = \mathbf{\Phi}\mathbf{R}_j\mathbf{y}. \tag{6.2}$$

We obtain the extended representation vector $\mathbf{c}^{SA}$ by concatenating all these vectors into a single column

$$\mathbf{c}^{SA} = \left[\mathbf{c}_1^T, ..., \mathbf{c}_n^T\right]^T = \begin{bmatrix} \mathbf{\Phi}\mathbf{R}_1 \\ \vdots \\ \mathbf{\Phi}\mathbf{R}_n \end{bmatrix} \mathbf{y} = \mathbf{\Phi}^{SA}\mathbf{y} \tag{6.3}$$

where we defined the matrix

$$\mathbf{\Phi}^{SA} = \begin{bmatrix} \mathbf{\Phi}\mathbf{R}_1 \\ \vdots \\ \mathbf{\Phi}\mathbf{R}_n \end{bmatrix}. \tag{6.4}$$

This matrix applies the analysis operator used to obtain the representation $\mathbf{c}^{SA}$.

The image $\mathbf{y}$ is reconstructed from the representation $\mathbf{c}^{SA}$ in the following manner. Let $\mathbf{\Psi}$ denote the $(L+1)N_p \times N_p$ matrix that applies the L-level RTBWT reconstruction. We apply the matrix $\mathbf{\Psi}$ to each of the vectors $\mathbf{c}_j$, plug each of the obtained subimages into its original place in the image, and average the different values obtained for each pixel. More formally,

$$\begin{aligned} \mathbf{y} &= \mathbf{D}^{-1}\sum_{j=1}^{n}\mathbf{R}_j^T\mathbf{\Psi}\mathbf{c}_j \\ &= \mathbf{D}^{-1}\left[\mathbf{R}_1^T\mathbf{\Psi}, \ldots, \mathbf{R}_n^T\mathbf{\Psi}\right]\mathbf{c}^{SA} = \mathbf{\Psi}^{SA}\mathbf{c}^{SA}, \end{aligned} \tag{6.5}$$

where the matrix $\mathbf{R}_j^T$ returns the jth subimage into its original place in the image, and $\mathbf{D} = \sum_{j=1}^{n}\mathbf{R}_j^T\mathbf{R}_j$ is a diagonal weight matrix. The matrix

$$\mathbf{\Psi}^{SA} = \mathbf{D}^{-1}\left[\mathbf{R}_1^T\mathbf{\Psi}, \ldots, \mathbf{R}_n^T\mathbf{\Psi}\right] \tag{6.6}$$

applies the synthesis operator used to reconstruct $\mathbf{y}$ from the representation $\mathbf{c}^{SA}$. As we see next, the rows of $\mathbf{\Phi}^{SA}$ constitute a frame $\{\phi_n^{SA}\}$ and the columns of $\mathbf{\Psi}^{SA}$ constitute its dual $\{\psi_n^{SA}\}$.

Proposition 2: The extended representation $\mathbf{c}^{SA}$ is a frame expansion with frame bounds $\alpha = 2$ and $\beta = 2^L n$.

The proof is presented in Appendix 6.A.

Here we see that $\mathbf{\Phi}^{SA}$ and $\mathbf{\Psi}^{SA}$ apply analysis and synthesis operations, and $\mathbf{\Psi}^{SA}\mathbf{\Phi}^{SA} = \mathbf{I}$. Therefore, $\mathbf{\Psi}^{SA}$ is the pseudo-inverse of $\mathbf{\Phi}^{SA}$, and the columns of $\mathbf{\Psi}^{SA}$ constitute a frame $\{\psi_n^{SA}\}$ which is dual to $\{\phi_n^{SA}\}$, i.e. it satisfies

$$\frac{1}{2^L n}\|\mathbf{y}\|^2 \leq \sum_n |\langle \psi_n, \mathbf{y}\rangle|^2 \leq \frac{1}{2}\|\mathbf{y}\|^2. \tag{6.7}$$

Before concluding this section, we add a remark regarding the frame bounds calculated above. Let us assume that the RTBWT and corresponding frame have been calculated from an image $\mathbf{y}_0$. Then the transform is adaptive to the content in this image, since it is designed by ordering of its patches or their descendants (i.e. filtered patches), and so does the corresponding frame. Therefore, the frame bounds obtained above are in fact misleading, since inequalities such as (6.7) are true for a general image $\mathbf{y}$, but the frame was designed to handle the image $\mathbf{y}_0$. Thus, perhaps a better definition of the frame bounds would be one that has to hold true for all the images $\mathbf{y}$ in a small sphere $\|\mathbf{y} - \mathbf{y}_0\|_2 \leq \epsilon$, for which the transform (and the resulting frame) retain its topology. In such a case, it is quite likely that the actual frame obtained is nearly tight. We leave this matter open at this stage, as our next part of the Chapter does not rely directly on these frame properties.

6.4 Image Reconstruction using RTBWT SA Frames

6.4.1 Image Reconstruction Scheme

A common approach for image reconstruction is to solve an inverse problem which uses either an analysis-based or synthesis-based image priors [28]. Here we take a different approach, follow the footsteps of Danielyan eta al. [18], and consider the following combination of an analysis and a synthesis problems[2]

$$\begin{aligned}\{\hat{\mathbf{y}}, \hat{\mathbf{c}}\} = \underset{\mathbf{y},\mathbf{c}}{\operatorname{argmin}} \; & \|\mathbf{z} - \mathbf{H}\mathbf{y}\|_2^2 + \eta\|\mathbf{y} - \mathbf{\Psi}\mathbf{c}\|_2^2 \\ & + \lambda\|\mathbf{c}\|_0 + \mu\|\mathbf{c} - \mathbf{\Phi}\mathbf{y}\|_2^2.\end{aligned} \tag{6.8}$$

[2] As of this point in the chapter, we will simplify our notations and use $\mathbf{\Phi}$, $\mathbf{\Psi}$ and $\mathbf{c}$, referring to the full adaptive frames $\mathbf{\Phi}^{SA}$ and $\mathbf{\Psi}^{SA}$, and the representation $\mathbf{c}^{SA}$.

Indeed, it can be seen that the problem (6.8) reduces to a synthesis problem when $\eta \to \infty$ and $\mu = 0$, and to an analysis when $\eta = 0$ and $\mu \to \infty$. Danielyan et al. [18] brilliantly handle the problem (6.8) as a generalized Nash equilibrium (GNE) process, and split it into two subproblems, a denoising task:

$$\hat{\mathbf{c}} = \underset{\mathbf{c}}{\operatorname{argmin}} \frac{\lambda}{\mu}\|\mathbf{c}\|_0 + \|\mathbf{c} - \mathbf{\Phi}\mathbf{y}\|_2^2, \tag{6.9}$$

and an inversion task:

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmin}} \|\mathbf{z} - \mathbf{H}\mathbf{y}\|_2^2 + \eta\|\mathbf{y} - \mathbf{\Psi}\mathbf{c}\|_2^2 \tag{6.10}$$

which are solved sequentially. We note that $\mathbf{\Phi}$ and $\mathbf{\Psi}$ are calculated in the beginning of this iterative process, and one approach is to keep them fixed throughout the iterative process. Alternatively, one might update these operators based on the updated image. Unless said otherwise, our scheme takes the first path of fixing these frames during the iterations.

The solution of the inversion stage is simply

$$\hat{\mathbf{y}} = \left[\mathbf{H}^T\mathbf{H} + \eta\mathbf{I}\right]^{-1} \times \left[\mathbf{H}^T\mathbf{z} + \eta\mathbf{\Psi}\mathbf{c}\right]. \tag{6.11}$$

The solution of the denoising problem is also quite simple, given by

$$\hat{\mathbf{c}} = S^{hard}\left\{\mathbf{\Phi}\mathbf{y}\right\} \tag{6.12}$$

where the operator S^{hard} applies hard thresholding with the threshold $\sqrt{\lambda/\mu}$. We choose to replace S^{hard} with an operator S_τ that sets to zero coefficients in $\mathbf{c}$ in such a manner that the thresholded versions of the coefficient vectors $\mathbf{c}_j$ (see Equation (6.2)) share a common support. This way, the set of thresholded subimage coefficients form a joint sparsity pattern in the transform domain, and our experiments show that when such patterns are used, the quality of the reconstructed images improves. Let $c_j[k]$ be the kth sample in the coefficient vector $\mathbf{c}_j$. Then the operator S_τ goes over the indices $k = 1, \ldots, (L+1)N_p$, and for each index sets the coefficients $\{c_1[k], \ldots, c_n[k]\}$ to zero if

$$\sqrt{\frac{1}{n}\sum_{j=1}^{n} c_j^2[k]} < \tau\sigma \tag{6.13}$$

Task: recover the image $\mathbf{y}$ from the corrupted image $\mathbf{z}$.

Parameters: The corrupted image $\mathbf{z}$, an initial estimate $\mathbf{y}_{init}$, the matrix $\mathbf{H}$, the iteration number G, the threshold τ, and the design parameter η.

Initialization: Set $\mathbf{y}_0 = \mathbf{y}_{init}$ and use it to calculate $\mathbf{\Phi}$ and $\mathbf{\Psi}$.

Main Iteration: For $t = 1, \ldots, G$ perform:

- Denoising: $\mathbf{c}_t = S_\tau\{\mathbf{\Phi}\mathbf{y}_{t-1}\}$
- Inversion: $\mathbf{y}_t = \left[\mathbf{H}^T\mathbf{H} + \eta\mathbf{I}\right]^{-1}\left[\mathbf{H}^T\mathbf{z} + \eta\mathbf{\Psi}\mathbf{c}_t\right]$

Output: The estimate $\hat{\mathbf{y}} = \mathbf{y}_G$.

Figure 6.1: Image reconstruction scheme.

where τ is a design parameter. This corresponds to a simple modification in the original inverse problem formulation in Equation (6.8), replacing the term $\|\mathbf{c}\|_0$ by a mixed norm applied on the different pieces of this representation, namely, $\|\mathbf{c}\|_{2,0}$.

The obtained image reconstruction scheme is described in the algorithm in Figure 6.1. It can be seen that this algorithm is similar to the IDD-BM3D algorithm proposed by Danielyan et al. [18], to which we will later compare our image deblurring results. We next demonstrate the use of the scheme in Figure 6.1 for image denoising and deblurring. In all the experiments described next we choose the distance function w to be the Euclidean distance, and use a 9-level RTBWT decomposition with the Symmlet 8 wavelet filter.

6.4.2 Image Denoising

In the case of image denoising $\mathbf{z} = \mathbf{y} + \mathbf{v}$ and therefore $\mathbf{H} = \mathbf{I}$. We perform denoising using a simplified version of the scheme in Algorithm 6.1, which uses the initial estimate $\mathbf{y}_{init} = \mathbf{z}$, and applies only one iteration. Also, we notice that since $\mathbf{H} = \mathbf{I}$, the solution to the inversion problem (6.10) becomes:

$$\hat{\mathbf{y}} = \frac{1}{1+\eta}\mathbf{z} + \frac{\eta}{1+\eta}\mathbf{\Psi}\mathbf{c}. \tag{6.14}$$

which is a weighted average between a denoised image and the original noisy image. We further simplify our scheme and cancel the addition of the noisy data to the clean image

by setting $\eta \to \infty$. All these simplifications were done since we found experimentally that the resulting simplified algorithm is already producing near state-of-the-art results. Thus, the obtained denoising scheme consists of calculating $\mathbf{c} = \mathbf{\Phi z}$, applying to it the thresholding operator S_τ, and reconstructing the image using $\mathbf{\Psi}$, i.e.

$$\hat{\mathbf{y}} = \mathbf{\Psi} S_\tau\{\mathbf{c}\} = \mathbf{\Psi} S_\tau\{\mathbf{\Phi z}\}. \tag{6.15}$$

We note that aside from the use of the operator S , this scheme is similar to the common thresholding based denoising procedure for signal and image processing using wavelet frames.

We explore two different methods to further improve the results obtained with the denoising algorithm described above. Both these methods apply the above scheme, and then use the patches from the "cleaned" result $\hat{\mathbf{y}}^1$ to construct a "better version" of the RTBWT, and use this transform to calculate new analysis and synthesis operators $\mathbf{\Phi}^1$ and $\mathbf{\Psi}^1$. The first method obtains the denoised image $\hat{\mathbf{y}}^2$ by applying Equation (6.15) again with these modified operators, i.e.

$$\hat{\mathbf{y}}^2 = \mathbf{\Psi}^1 S_\tau\{\mathbf{\Phi}^1 \mathbf{z}\}. \tag{6.16}$$

The second method applies a different scheme in its second stage. Let $\mathbf{C}_{diag}$ be a diagonal matrix that contains the squares of the elements of the vector $\mathbf{\Phi}^1\hat{\mathbf{y}}^1$ in its main diagonal. Then we replace the ℓ_0-norm term $\|\mathbf{c}\|_0$ in the denoising problem (6.9) with a different sparsity promoting term $\mathbf{c}^T\mathbf{C}_{diag}^{-1}\mathbf{c}$, and obtain the new problem

$$\hat{\mathbf{c}} = \underset{\mathbf{c}}{\operatorname{argmin}} \ \frac{\lambda}{\mu}\mathbf{c}^T\mathbf{C}_{diag}^{-1}\mathbf{c} + \|\mathbf{c} - \mathbf{\Phi}^1\mathbf{y}\|_2^2. \tag{6.17}$$

We differentiate (6.17) with respect to $\mathbf{c}$, equate the derivative the zero, and obtain

$$\begin{aligned}
\hat{\mathbf{c}} &= \left[\frac{\lambda}{\mu}\mathbf{C}_{diag}^{-1} + \mathbf{I}\right]^{-1} \mathbf{\Phi}^1\mathbf{y} \\
&= \left[\frac{\lambda}{\mu}\mathbf{C}_{diag}^{-1} + \mathbf{C}_{diag}^{-1}\mathbf{C}_{diag}\right]^{-1} \mathbf{\Phi}^1\mathbf{y} \\
&= \left[\mathbf{C}_{diag} + \frac{\lambda}{\mu}\mathbf{I}\right]^{-1} \mathbf{C}_{diag}\mathbf{\Phi}^1\mathbf{y} = \mathbf{W}\mathbf{\Phi}^1\mathbf{y},
\end{aligned} \tag{6.18}$$

where we have defined the diagonal matrix

$$\mathbf{W} = \left[\mathbf{C}_{diag} + \frac{\lambda}{\mu}\mathbf{I}\right]^{-1} \mathbf{C}_{diag}. \tag{6.19}$$

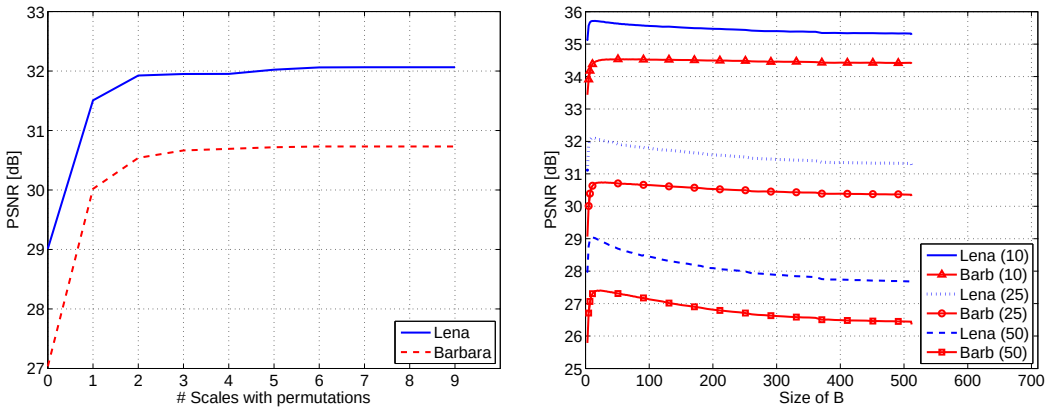
Thus, the second method obtains the denoised image $\hat{\mathbf{y}}_w^2$ by calculating in its second stage the coefficient vector $\mathbf{c} = \mathbf{\Phi}^1\mathbf{z}$, multiplying it with the matrix $\mathbf{W}$, and reconstructing the image using $\mathbf{\Psi}^1$, i.e.

$$\hat{\mathbf{y}}_w^2 = \mathbf{\Psi}^1\mathbf{W}\mathbf{\Phi}^1\mathbf{z}. \tag{6.20}$$

We note that by choosing $\lambda/\mu = (\tau\sigma)^2$, where τ is a design parameter, multiplication by the matrix $\mathbf{W}$ is equivalent to applying some sort of a Wiener filter to the noisy image, similarly to what is done in the second stage of the algorithms proposed by Dabov et al. in [16] and Ghael et al. [31].

We now proceed to assess the performance of our image denoising scheme. We note that in all our experiments the noise levels are assumed to be known. We start by demonstrating the performance gain obtained by applying the permutations in all the scales of the RTBWT. We apply the following procedure to the noisy versions of the images Lena and Barbara with noise standard deviation $\sigma = 25$, that are shown in the first column of Figure 8.8. For each image we calculate 10 different variants of the RTBWT with the parameters $\sqrt{n} = 12$ and $B = 21$: one with no permutations at all and the rest with all the permutations in the first F scales, $1 \leq F \leq 9$. We then apply to each image one stage of the aforementioned denoising scheme with each of the 10 different variants of the RTBWT and with the parameter $\tau = 1.5$. Figure 6.2(a) shows the PSNR values obtained for each image versus the number of scales in which permutations are applied. It can be seen that most of the performance gain is obtained by the permutation in the first scale ($\ell = 0$), but the PSNR increases as more permutations are used in coarser scales as well.

We next demonstrate the robustness of our denoising performance with respect to the size of the parameter B. We apply to noisy versions of the images Lena and Barbara, with noise standard deviations of 10, 25 and 50, one stage of our denoising scheme with $\sqrt{n}$ sizes of 9,12 and 14, respectively, and varying values of B. Figure 6.2(b) shows 6 curves, one for each combination of image and noise level, describing the PSNR values obtained for different sizes of the parameter B. First it can be seen that all the curves obtain their maximum values for sizes of B between 10 to 50, and that the change in the PSNR in that range is relatively moderate. Thus high quality denoising results may be obtained by choosing a single value of B in that range, and applying our denoising scheme with it to



(a) (b)

Figure 6.2: (a) PSNR gain versus the number of scales in which permutations are applied. (b) PSNR values obtained for different sizes of B for noisy versions of the images Lena and Barbara (Barb) with noise standard deviations of 10, 25 and 50 (in brackets).

all the noisy images with the different noise levels. It can also be seen that after the curves pass their maxima, their PSNR values decrease as the size of B increases. However, even in the case where B equals the size of the whole image our denoising scheme manages to obtain a high quality denoising result.

Now, in order to assess the performance of the proposed image denoising scheme we apply it to noisy versions of 6 images, with 8 different noise standard deviations. The parameters employed by the proposed denoising scheme for the different noise levels are shown in Table 6.1. Table 6.2 shows the PSNR values of the results obtained with the BM3D algorithm, two stages of the RTBWT denoising scheme, and the RTBWT followed by Wiener filtering. The noisy and recovered images obtained for $\sigma = 25$ with the BM3D algorithm, and our results are shown in Figure 6.3. First, it can be seen that both the second stage of the RTBWT scheme and the Wiener filter improve the first iteration results of the RTBWT scheme in many $(70\% - 85\%)$ of the cases. The Wiener filter results are generally better or comparable to the results of both the first and second stages of the RTBWT scheme. Finally it can be seen that for $\sigma \geq 25$, the RTBWT denoising scheme followed by Wiener filtering achieves in most of the cases either the best results or results that are nearly as good. As the noise decreases in strength, the performance of the RTBWT based denoising schemes deteriorates compared to the BM3D.

Table 6.1: Parameters Used in the Denoising Experiments.

σ	Stage	L	$\sqrt{n}$	B	τ
5	1	9	7	21	1.5
	2	9	4	211	1.45
	Wiener	9	5	191	1.05
10	1	9	9	21	1.5
	2	9	6	181	1.45
	Wiener	9	5	191	1.05
15	1	9	10	21	1.5
	2	9	7	211	1.45
	Wiener	9	5	191	1.05
20	1	9	11	21	1.5
	2	9	9	211	1.45
	Wiener	9	5	191	1.05
25	1	9	12	21	1.5
	2	9	10	211	1.45
	Wiener	9	6	191	1.05
50	1	9	14	21	1.5
	2	9	13	211	1.45
	Wiener	9	7	191	1.05
75	1	9	16	21	1.5
	2	9	14	211	1.45
	Wiener	9	7	191	1.05
100	1	9	16	21	1.5
	2	9	16	211	1.45
	Wiener	9	9	191	1.05

In terms of visual quality both the BM3D and the RTBWT-based algorithms produce high quality results, however the two approaches also suffer from different types of artifacts: the BM3D results suffer from smudges (for example in Lena's face) while the RTBWT-based algorithms suffer from false contours (for example around Lena's hat).

Table 6.2: Denoising Results (PSNR in dB) of Noisy Versions of 6 Images, Obtained with the BM3D Algorithm, Two Stages of the RTBWT Denoising Scheme, and the RTBWT Scheme Followed by Wiener Filtering. For Each Image and Noise Level the Best Result and Results within a Distance of 0.05 dB from It, are Highlighted.

Image	Method	σ/PSNR							
		5/34.16	10/28.14	15/24.61	20/22.11	25/20.18	50/14.16	75/10.63	100/8.14
Lena	BM3D	**38.72**	**35.93**	**34.27**	33.05	32.08	29.05	27.26	25.95
	RS1	38.44	35.70	34.12	32.98	32.06	28.97	27.19	25.90
	RS2	38.08	35.45	33.91	32.99	32.11	29.18	27.43	**26.31**
	RS1W	38.40	35.75	**34.22**	**33.11**	**32.26**	**29.30**	**27.50**	**26.36**
Barbara	BM3D	**38.31**	**34.98**	**33.11**	31.78	30.72	27.23	25.12	23.62
	RS1	37.54	34.50	32.89	31.70	30.73	27.39	25.38	23.97
	RS2	37.62	34.55	32.91	31.71	30.76	27.65	25.73	**24.44**
	RS1W	37.67	34.55	32.97	**31.85**	**30.90**	**27.78**	**25.82**	**24.46**
Boats	BM3D	**37.28**	**33.92**	**32.14**	**30.88**	**29.91**	26.78	**25.12**	23.97
	RS1	36.86	33.68	31.91	30.65	29.67	26.59	24.85	23.68
	RS2	36.86	33.64	31.92	30.67	29.71	26.80	**25.13**	**24.03**
	RS1W	36.98	33.79	32.05	30.82	**29.88**	**26.91**	**25.15**	**24.04**
Fingerprint	BM3D	**36.51**	**32.46**	**30.28**	**28.81**	**27.70**	**24.53**	**22.83**	**21.61**
	RS1	35.03	31.28	29.47	28.21	27.16	23.90	22.20	21.19
	RS2	35.86	31.88	29.80	28.17	27.01	23.74	22.22	21.37
	RS1W	35.38	31.60	29.73	28.45	27.32	24.06	22.47	21.53
House	BM3D	**39.83**	**36.71**	**34.94**	**33.77**	**32.86**	**29.69**	**27.51**	25.87
	RS1	39.32	36.41	34.75	33.61	32.74	29.55	27.43	25.75
	RS2	38.40	35.68	34.01	33.27	32.46	29.49	27.42	**25.96**
	RS1W	38.42	35.86	34.25	33.08	32.37	29.56	27.37	**25.98**
Peppers	BM3D	**38.12**	**34.68**	**32.70**	31.29	30.16	26.68	24.73	23.39
	RS1	37.65	34.27	32.36	30.99	29.90	26.48	24.50	23.08
	RS2	37.64	34.44	32.61	31.21	30.14	26.76	24.89	**23.53**
	RS1W	37.81	34.49	**32.66**	**31.35**	**30.33**	**26.93**	**24.98**	**23.56**

6.4.3 Image Deblurring

In the case of image deblurring, **H** is a blur matrix, and we perform deblurring by simply applying the algorithm in Figure 6.1 as is. We demonstrate the image deblurring performance obtained with this algorithm on the images Lena, Barbara, House, and Cameraman, for the 6 scenarios described in Table 6.3. We compare the results obtained with this algorithm to the ones obtained with the BM3DDEB [17] and the IDD-BM3D algorithms [18]. As the IDD-BM3D algorithm is initialized with the BM3DDEB results,

Figure 6.3: Denoising results (PSNR) for the images Lena and Barbara ($\sigma = 25$, input PSNR=20.18 dB): First column: noisy images, Second column - BM3D results, Third column - 2 stages results of the RTBWT denoising scheme, Fourth column - results of the RTBWT scheme followed by Wiener filtering.

Table 6.3: Blur Point Spread Functions (PSF) and Noise Variances used in the Different Deblurring Scenarios.

Scenario	PSF	σ^2
1	$1/(1+x_1^2+x_2^2)$, $x_1, x_2 = -7, \ldots, 7$	2
2	$1/(1+x_1^2+x_2^2)$, $x_1, x_2 = -7, \ldots, 7$	8
3	9×9 uniform	≈ 0.3
4	$[\ 1\ 4\ 6\ 4\ 1]^T[\ 1\ 4\ 6\ 4\ 1\ ]\ /\ 256$	49
5	Gaussian with $std = 1.6$	4
6	Gaussian with $std = 0.4$	64

for a fair comparison we examine the results obtained with our scheme using the same initialization. We also examine the results obtained with our scheme by initializing it with the blurry image, and try to improve them by applying two more stages of our scheme, where we initialize each stage with result of the previous ones. The parameters employed

Table 6.4: Parameters Used in the Deblurring Experiments for the Proposed Scheme Initialized with the BM3DDEB Results (BM3DDEB init.) and for 3 Stages of the Proposed Scheme Initialized with the Blurry Image (BI).

Scenario	Stage	L	$\sqrt{n}$	B	η	τ	G
1	1 (BM3DDEB init.)	9	3	51	0.07	20	50
	1 (BI)	9	6	31	0.03	60	30
	2(BI)	9	4	41	0.03	50	30
	3 (BI)	9	4	41	0.06	30	30
2	1 (BM3DDEB init.)	9	3	51	0.08	30	50
	1 (BI)	9	6	31	0.07	60	30
	2 (BI)	9	4	41	0.1	50	30
	3 (BI)	9	4	41	0.16	30	30
3	1 (BM3DDEB init.)	9	3	51	0.01	20	50
	1 (BI)	9	6	31	0.005	50	30
	2 (BI)	9	4	41	0.005	60	30
	3 (BI)	9	4	41	0.015	30	30
4	1 (BM3DDEB init.)	9	3	51	0.9	20	50
	1 (BI)	9	6	31	0.5	50	30
	2 (BI)	9	4	41	0.8	40	30
	3 (BI)	9	4	41	1.9	20	30
5	1 (BM3DDEB init.)	9	3	51	0.05	20	50
	1 (BI)	9	6	31	0.03	60	30
	2 (BI)	9	4	41	0.07	40	30
	3 (BI)	9	4	41	0.14	20	30
6	1 (BM3DDEB init.)	9	3	51	3.7	20	50
	1 (BI)	9	6	31	3.2	50	30
	2 (BI)	9	4	41	3.9	30	30
	3 (BI)	9	4	41	3.8	30	30

in the different stages of the proposed deblurring scheme, with the different initializations and for the different scenarios, are shown in Table 6.4. We note that similarly to what was done by Danielyan et al. [18], we optimized the parameters η and τ separately for each stage and each deblurring scenario to provide best reconstruction quality.

Table 6.5 shows the ISNR results obtained with the different algorithms. The blurred and recovered Lena and Cameraman images obtained with the BM3DDEB and IDD-BM3D algorithms, our proposed scheme initialized with the BM3DDEB results, and three stages of the proposed scheme initialized with the blurry images, are shown in Figure 6.4. It can be seen that for all the scenarios except for scenario 3, the proposed scheme initialized with BM3DDEB results achieves the best performance for every image. Our scheme obtains inferior results when the blurry image is used to initialize it, however its performance improve when two more rounds are applied. In fact, in some cases three rounds of our scheme initialized with the blurry image obtain the best or second best results.

In terms of visual quality, again both the BM3D-based and the RTBWT-based algorithms produce high quality results, accompanied by modest and different types of artifacts: here the results of the BM3D-based algorithms suffer from false contouring (for example near the Cameraman's chin) while those of the RTBWT-based algorithms suffer from a more significant noise residual.

6.4.4 Computational Complexity

We next evaluate the computational complexity of the image denoising and deblurring algorithms described above. We start by calculating the complexity of a single iteration of the denoising algorithm. First, extracting all the overlapped image patches requires $O(nN)$ operations. Next, we assume that for the calculation of each one of the 2^ℓ operators $\mathbf{P}_\ell^s$ in the ℓth scale of the RTBWT, each patch requires approximately B^2 distance calculations. As calculating the Euclidean distance between two patches requires $O(n)$ operations, the number of operations required to calculate a single reordering operator $\mathbf{P}_\ell^s$ from $\tilde{N}_\ell = N2^{-\ell}$ patches is approximately $O(\tilde{N}_\ell B^2 n)$. Reordering these patches and applying them the two wavelet filters requires $O(n\tilde{N}_\ell)$ operations. Therefore the number of operations required in order to calculate from an image all the operators $\mathbf{P}_\ell^s$ used in an

Table 6.5: Deblurring Results (ISNR in dB) of Blurry Versions of 4 Images, Obtained with the BM3DDEB and IDD-BM3D Algorithms, the Proposed (prop.) Scheme Initialized with the BM3DDEB Results, and 3 Stages of the Proposed Scheme Initialized with the Blurry Image (BI). For Each Image and Scenario (Scen.) the Best Result and Results within a Distance of 0.05 dB from It, are Highlighted.

Image	Method	Scen. 1	Scen. 2	Scen. 3	Scen. 4	Scen. 5	Scen. 6
Lena	BM3DDEB	7.96	6.55	7.99	4.80	4.34	6.42
	IDD-BM3D	7.97	6.61	8.91	4.97	4.85	6.34
	prop. (BM3DDEB init.)	**8.56**	**6.92**	8.86	**5.52**	**4.95**	**6.91**
	prop. (BI 1 iter.)	7.36	5.99	8.43	4.66	4.56	5.58
	prop. (BI 2 iter.)	7.43	6.25	8.69	4.76	4.83	5.89
	prop. (BI 3 iter.)	7.75	6.34	**8.98**	4.88	**4.91**	5.89
Barbara	BM3DDEB	7.88	4.13	5.91	2.05	1.29	5.85
	IDD-BM3D	7.64	3.96	**6.05**	1.88	1.16	5.45
	prop. (BM3DDEB init.)	**8.06**	**4.57**	**6.01**	**2.20**	**1.41**	**6.06**
	prop. (BI 1 iter.)	4.98	2.54	4.52	1.48	0.97	4.59
	prop. (BI 2 iter.)	5.98	2.68	4.76	1.48	0.99	4.88
	prop. (BI 3 iter.)	6.31	2.76	4.99	1.49	1.00	4.88
House	BM3DDEB	9.34	8.22	10.94	5.20	4.59	7.34
	IDD-BM3D	9.95	8.55	12.89	5.79	5.74	7.13
	prop. (BM3DDEB init.)	**10.44**	**8.79**	**13.11**	**6.38**	**5.95**	**7.56**
	prop. (BI 1 iter.)	9.09	7.58	12.03	5.27	5.30	6.16
	prop. (BI 2 iter.)	9.10	7.81	12.69	5.29	5.58	6.52
	prop. (BI 3 iter.)	9.58	8.01	**13.16**	5.58	5.74	6.56
Cameraman	BM3DDEB	8.20	6.46	8.37	3.35	3.72	4.70
	IDD-BM3D	8.85	7.12	**10.45**	3.98	4.31	4.89
	prop. (BM3DDEB init.)	**9.24**	**7.38**	10.21	**4.34**	**4.68**	**5.26**
	prop. (BI 1 iter.)	7.92	6.18	8.19	3.46	3.97	3.92
	prop. (BI 2 iter.)	8.27	6.65	9.57	3.80	4.39	4.27
	prop. (BI 3 iter.)	8.29	6.69	9.81	3.94	4.58	4.26

L-level RTBWT is

$$O(nN) + \sum_{\ell=0}^{L-1} 2^{\ell} \left[O(\tilde{N}_{\ell} B^2 n) + O(n\tilde{N}_{\ell}) \right] = O(LNB^2 n). \tag{6.21}$$

Figure 6.4: Deblurring results (ISNR) for the images Lena and Cameraman (Scenario 2): First column: blurry images, Second column: BM3DDEB results, Third column: IDD-BM3D results, Fourth column: results of the proposed scheme initialized with the BM3DDEB results, Fifth column: results of the three rounds of the proposed scheme initialized with the blurry images.

Now, reordering each signal $\mathbf{a}_\ell^s$ and applying on it the two wavelet filters requires $O(\tilde{N}_\ell)$ operations, therefore applying either of the analysis and synthesis operators $\mathbf{\Phi}$ and $\mathbf{\Psi}$, corresponding to the RTBWT calculated above, requires $\sum_{\ell=0}^{L-1} 2^\ell O(\tilde{N}_\ell) = O(LN)$ operations.

Next, each of the following actions require $O(nLN)$ operations: applying the operator $\mathbf{\Phi}$ to the n subimages $\tilde{\mathbf{z}}_j$, applying the threshold operator S_τ or the Wiener filter to the transform coefficients, and applying $\mathbf{\Psi}$ to the result, Constructing an estimate image by averaging the pixel values obtained with the different subimages requires $O(nN)$ operations, therefore the total complexity of a single denoising iteration is

$$O(LNB^2n) + O(nLN) + O(nN) = O(LNB^2n) \tag{6.22}$$

operations, which means that, as might be expected, the overall complexity is dominated by the calculation of the RTBWT permutation operators. For a typical case in our experiments, $N = 512^2$, $L = 9$, $n = 121$ and $B = 21$, the above amounts to $1.26 \cdot 10^{11}$ operations.

We proceed to calculate the complexity of the proposed image deblurring algorithm.

As we saw above, the calculation of all the RTBWT operators requires $O(LNB^2n)$ operations, and each denoising step requires $O(nLN) + O(nN) = O(nLN)$ operations. The deblurring step is performed in the Fourier domain and therefore requires $O(N \log N)$ operations. Thus, the overall complexity of the algorithm is

$$\begin{aligned} &O(LNB^2n) + G[O(nLN) + O(N \log N)] \\ &= O(LNB^2n + GnLN + GN \log N) \end{aligned} \tag{6.23}$$

and since the parameters we use satisfy $nLB^2 > G(nL + \log N)$ and $B^2 > G$, the complexity of the algorithm is again dominated by the calculation of the RTBWT permutation operators. For a typical case in our experiments, $N = 512^2$, $L = 9$, $n = 9$, $B = 51$ and $G = 50$, the above amounts to $5.64 \cdot 10^{10}$ operations.

In order to better illustrate these numbers, we also provide run-times: applying two iterations of our denoising scheme to a 512×512 image with noise level $\sigma = 25$ using a non optimized and non parallel matlab implementation, on an Intel(R) Core(TM) i7-2600 CPU @ 3.40 GHz, takes about 45 minutes. However, applying a single iteration followed by Wiener filtering takes only about 20 minutes. Applying a single round of our deblurring scheme, initialized with the BM3DDEB result, to a 512×512 image corrupted according to scenario 1 takes about 4 minutes. We should note that while in our experiments we employed exact exhaustive search, approximate nearest neighbor algorithms may be used to alleviate the computational burden.

6.5 Conclusions

We have shown that the image representation obtained with the RTBWT combined with the subimage averaging scheme constitutes a frame, and calculated the synthesis and analysis operators associated with it. We have proposed image denoising and deblurring algorithms which make use of these operators as sparsity promoting regularizers, and demonstrated state-of-the-art results.

6.A Proof of Proposition 2

.

We show that the representation $\mathbf{c}^{SA}$ is a frame by noticing that $\|\mathbf{c}^{SA}\|^2 = \sum_n |\langle \phi_n^{SA}, \mathbf{y} \rangle|^2$, and showing that

$$2\|\mathbf{y}\|^2 \leq \|\mathbf{c}^{SA}\|^2 \leq 2^L n \|\mathbf{y}\|^2. \tag{6.24}$$

We first use (6.3) and see that

$$\begin{aligned} \|\mathbf{c}^{SA}\|^2 = \mathbf{y}^T \left(\mathbf{\Phi}^{SA}\right)^T \mathbf{\Phi}^{SA} \mathbf{y} &= \sum_{j=1}^{n} \mathbf{y}^T \mathbf{R}_j^T \mathbf{\Phi}^T \mathbf{\Phi} \mathbf{R}_j \mathbf{y} \\ &= \sum_{j=1}^{n} \|\mathbf{\Phi} \mathbf{R}_j \mathbf{y}\|^2 \end{aligned} \tag{6.25}$$

From Proposition 1 we have that

$$2\|\mathbf{R}_j \mathbf{y}\|^2 \leq \|\mathbf{\Phi} \mathbf{R}_j \mathbf{y}\|^2 \leq 2^L \|\mathbf{R}_j \mathbf{y}\|^2. \tag{6.26}$$

We notice that

$$\sum_{j=1}^{n} \|\mathbf{R}_j \mathbf{y}\|^2 = \mathbf{y}^T \left(\sum_{j=1}^{n} \mathbf{R}_j^T \mathbf{R}_j \right) \mathbf{y} = \mathbf{y}^T \mathbf{D} \mathbf{y} \tag{6.27}$$

and therefore

$$2\mathbf{y}^T \mathbf{D} \mathbf{y} \leq \|\mathbf{c}^{SA}\|^2 \leq 2^L \mathbf{y}^T \mathbf{D} \mathbf{y}. \tag{6.28}$$

As the minimum value of the diagonal matrix $\mathbf{D}$ equals 1, and the maximum value equals n, we obtain that

$$2\|\mathbf{y}\|^2 \leq \|\mathbf{c}^{SA}\|^2 \leq 2^L n \|\mathbf{y}\|^2. \tag{6.29}$$

Chapter 7

Facial Image Compression using RTBWT[1]

7.1 Introduction

In recent years, facial images are being extensively used and stored in large databases by social networks, web services, or various organizations such as states, law-enforcement, schools, universities, and private companies. Facial images are also expected to be stored in digital passports and ID cards. Thus, efficient storage of such images is beneficial, and their compression is an appealing application. The limitation to a specific and narrow family of images increases their combined spatial redundancy, and this allows algorithms that are specially tailored for the task of facial image compression, to surpass general purpose compression algorithms. More specifically, recent work [4, 5, 27, 69] has shown that this kind of algorithms lead to performance far exceeding JPEG2000 [60].

In this chapter, we introduce a novel algorithm for compression of facial images, that exploits the redundant tree-based wavelet transform (RTBWT) introduced in Chapter 4. This transform was originally designed to represent scalar functions defined on high-dimensional data clouds and graphs. However, we have shown in Chapters 5 and 6 that when this transform is constructed from an image (by converting it to a graph-structure), it produces a redundant and multi-scale decomposition that is adaptive to that image, and is highly effective for sparsifying its content. In this chapter, we bring this signal-

[1]This chapter is based on [48].

adaptive transform to the realm of compression of aligned frontal facial images. The RTBWT is calculated by adding data-dependent permutation operators into the classical redundant wavelet filter-bank implementation [2, 54]. These operators apply reorderings on the approximation coefficient signals so as to increase their regularity and cause their representation with the RTBWT to be more efficient (sparse).

Given a training set of aligned face images, we propose to build these reordering operators by organizing feature vectors that are common to all these images, such that they are chained in the shortest possible path, essentially obtaining an approximation to the solution of the traveling salesman problem (TSP) [14]. We compress an image by applying on it sparse coding using the RTBWT decomposition, quantizing the coefficients in the obtained representation, and then applying entropy coding. We decompress the image by placing the entropy decoded coefficients in a sparse vector, applying the RTBWT reconstruction, and applying a post-processing stage to the result. We demonstrate the performance of the proposed scheme both qualitatively and visually, and compare it to other competing algorithms.

The chapter is organized as follows: In Section 7.2, we explain how to calculate the RTBWT which sparsely represents a set of face images, and how to use it to obtain a sparse representation for such images. Section 7.3 introduces our proposed image compression scheme, and in Section 7.4 we present experimental results that demonstrate its advantages. We summarize this chapter in Section 7.5.

7.2 The Sparsifying Transform

7.2.1 Sparse Representation of Facial Images

Let $\mathbf{y}$ be a column-stacked version of a face image, which contains N pixels. We assume that the image $\mathbf{y}$ follows the sparseland model described in Chapter 2, and therefore we can compress it by obtaining an efficient (sparse) representation for it. The sparseland model suggests that the image $\mathbf{y}$ can be sparsely represented using a redundant matrix $\mathbf{A}$ of size $N \times J$ ($J > N$), which we term a dictionary. More specifically, let $\|\alpha\|_0$ denote the number of nonzero entries in a coefficient vector α. Then we expect that the solution

of

$$\hat{\alpha} = \underset{\alpha}{\operatorname{argmin}} \|\alpha\|_0 \text{ subject to } \|\mathbf{A}\alpha - \mathbf{y}\|_2^2 \leq \epsilon^2 \tag{7.1}$$

should be sparse, i.e. $\|\alpha\|_0 \ll N$. This means that the image $\mathbf{y}$ can be represented using a small number of columns (termed atoms) from $\mathbf{A}$, with an error measured by the distance $\|\mathbf{A}\alpha - \mathbf{y}\|_2^2$. Naturally, there exists a tradeoff between the number of atoms used to represent $\mathbf{y}$ and the size of the representation error, i.e., the more atoms we use the smaller the error gets, and vice versa.

Problem (7.1) is an NP-hard problem, since it requires an examination of $O(J^N)$ possible non-zero supports for α. However, the matching and the basis pursuit algorithms [9, 44, 46] can be used quite effectively to obtain an approximation for the solution. In fact, these approximation techniques can be quite accurate if the solution is sparse enough to begin with [21–23, 61, 62]. In this chapter we make use of the Orthogonal Matching Pursuit (OMP), described in Section 2.3 of Chapter 2, because of its simplicity [46] and efficiency. We next describe how we obtain the dictionary $\mathbf{A}$, which is used for sparse representations of face images.

7.2.2 RTBWT-Based Dictionary for a Set of Images

Let $\mathbf{y}^g$, $g = 1 \ldots, G$ be a training set, which contains the column-stacked versions of G aligned[2] face images, each containing N pixels. We wish to construct a redundant dictionary using the images in this training set, which will enable to sparsely represent them and similar facial images in a different test set. We note that while other methods usually train dictionaries which sparsely represent image patches, here we wish to find a dictionary that will be used to represent the entire image. To this end we make use of the redundant tree-based wavelet transform (RTBWT), introduced in Chapter 4.

The RTBWT is a data-adaptive transform, providing a sparse and redundant representation for its input signal. In order to calculate the transform for an image $\mathbf{y}$, it requires that each pixel y_i will be associated with a feature vector $\mathbf{x}_i$, and it is assumed that under a distance measure $w(\mathbf{x}_i, \mathbf{x}_j)$ (e.g. Euclidean distance), proximity between two

[2]Geometrical pre-aligning of the facial images is crucial to any method that aims for effective compression. Indeed, previous work [27], [4], [5] used this extensively, and here we follow their footsteps and assume the availability of an aligned set.

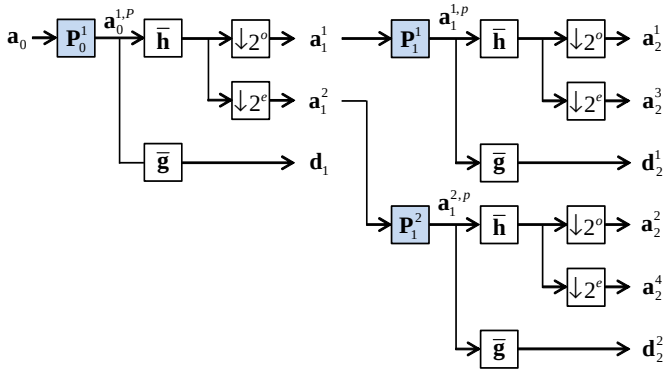


Figure 7.1: RTBWT decomposition scheme

such feature vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ implies proximity between their corresponding pixels y_i and y_j. When working with an image, the features may be chosen to be patches centered around the pixels of interest [49, 51]. The transform is constructed by modifying the classical redundant wavelet transform [2, 54]. Figure 7.1 describes the decomposition scheme of the RTBWT. The filters $\bar{\mathbf{h}}$ and $\bar{\mathbf{g}}$ are the scaling and wavelet decomposition filters of a regular discrete wavelet transform, and they are applied using cyclic convolution. The $2:1$ decimators denoted by $\downarrow 2^o$ and $\downarrow 2^e$ keep the odd and even samples of their input, respectively. The signals $\mathbf{a}_\ell^s$ and $\mathbf{d}_\ell^s$ contain subsets of the samples in the approximation and detail coefficient vectors $\mathbf{a}_\ell$ and $\mathbf{d}_\ell$ in the ℓth scale, respectively, where $\mathbf{a}_0 = \mathbf{y}$.

The operators $\mathbf{P}_\ell^s$ make the difference between our proposed wavelet decomposition scheme and the common redundant wavelet transform [54], [2]. *Each such operator produces a permuted version* $\mathbf{a}_\ell^{s,p}$ *of its input vector* $\mathbf{a}_\ell^s$. This may be interpreted as a linear and unitary operator given that vector. The reordering operators increase the regularity of the approximation coefficient signals in the different levels of the decomposition scheme and cause their representation with the RTBWT to be more sparse. These operators are obtained by organizing the feature vectors, calculated from the patches, such that they are chained in the shortest possible path [49–51, 53]. Thus, essentially an approximation to the solution of the traveling salesman problem (TSP) [14] is obtained. For example, let $\{\mathbf{x}_j^p\}_{j=1}^N$ denote the patches $\{\mathbf{x}_i\}_{i=1}^N$ in their new order, then $\mathbf{P}_0^1$ is obtained by minimizing the measure

$$TV(\mathbf{x}_j^p) = \sum_{j=2}^{N} w(\mathbf{x}_j^p, \mathbf{x}_{j-1}^p). \tag{7.2}$$

We note that the RTBWT reconstruction scheme is obtained in a similar manner by adding the operators $(\mathbf{P}_\ell^s)^{-1}$ into the redundant wavelet transform reconstruction scheme.

We now move to discuss the specifics of facial image compression. Before calculating the transform from the images in the training set, we first average them and obtain a mean face image. This image contains information shared by all the images in the training set, and therefore we subtract it from every training image in order to obtain a more efficient representation for it. Next, since we want the transform to sparsely represent all the images in the training set, we need to associate a single feature point with every set of G pixels that are located in the same index in each of the training images. Thus, we construct a $G \times N$ matrix $\mathbf{Y}^G$, which contains in its rows the images $\mathbf{y}^g$, and choose its kth column to be the feature vector $\mathbf{x}_k$ associated with kth pixel y_k^g in each of the training images. Note that this is different from the common practice mentioned above of using spatial patches. Also, we choose the distance function w to be the Euclidean distance. Having defined the feature points and the distance function, we use them to construct the RTBWT according to the scheme described in [51].

We next denote by $\mathbf{\Phi}$ and $\mathbf{\Psi}$ the matrices that apply the RTBWT decomposition and reconstruction, respectively, and choose our dictionary $\mathbf{A}$ to be a version of $\mathbf{\Psi}$, whose atoms have been normalized to have a unit norm. We note that because of its large size, the matrix $\mathbf{A}$ is not explicitly calculated nor stored. Instead, in order to multiply vectors by $\mathbf{A}$ and $\mathbf{A}^T$ within the OMP algorithm, it is required to apply the RTBWT reconstruction and decomposition schemes, respectively, on these vectors. More specifically, the product of $\mathbf{A}$ by a vector is calculated by replacing the filters used by the RTBWT reconstruction scheme [51] with the orthogonal wavelet scaling and reconstruction filters $\mathbf{h}$ and $\mathbf{g}$ [43], and applying the obtained scheme on this vector. On the other hand, the product of $\mathbf{A}^T$ by a vector is calculated simply by applying the RTBWT decomposition on this vector, since $\mathbf{\Phi} = \mathbf{A}^T$, as the matrix $\mathbf{\Phi}$ contains in its rows normalized versions of the atoms in the columns of $\mathbf{\Psi}$.

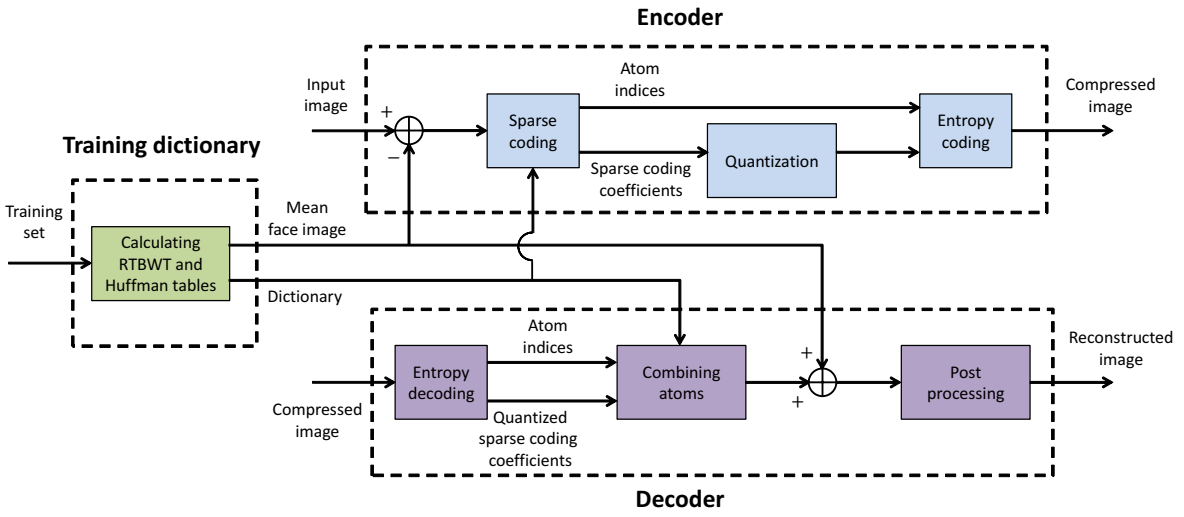


Figure 7.2: Facial image encoding and decoding schemes.

7.3 Facial Image Compression Scheme

Our proposed facial image encoding and decoding schemes are shown in Figure 7.2. We assume that we are given training and test sets containing aligned facial images. We average the images in the training set and subtract from them the mean image. We calculate a dictionary from the resulting images as described in the previous section, and then encode every one of these images using the following procedure: 1) We apply sparse coding to the image using the OMP algorithm to obtain a small set of coefficient values and their corresponding indices. 2) We replace the coefficient indices by the differences between the indices of consecutive coefficients, split the coefficient values into low and high ranges, and apply uniform quantization to the values in each range. 3) We calculate two different Huffman tables, one for the coefficient values and the other for their indices, relying on the statistics of their occurrences in the sparse representations of all the images in the training set. *All the aforementioned calculations are done offline, and we assume that the obtained RTBWT dictionary (along with its defining permutations) and Huffman tables are known both to the encoder and the decoder, along with the mean face image, and therefore they do not need to be transmitted as side information.*

Encoding an image from the test set starts with subtracting from it the mean face image, and applying the same encoding procedure that was applied above in order to find

the significant representation coefficients. We then perform entropy coding by applying the corresponding Huffman tables to the resulting sets of coefficient indices and values, and obtain the compressed image. We decode such an image by first applying entropy decoding, thus obtaining the quantized coefficient values and the index differences. We recover the coefficient indices from their differences, and use them and the corresponding quantized coefficient values to construct a sparse representation. The image is reconstructed by simply applying the RTBWT reconstruction to this sparse vector, and adding the mean face image to the result.

In order to further improve the quality of the obtained image, we use a post-processing scheme, which is a variation on the one proposed in [5]. This scheme consists of applying to the reconstructed image N different filters of size 5×5, each centered around a different pixel. We use the training images to learn several different sets of N filters, each corresponding to a different range of bit-rates. These filters are obtained as follows. We apply the encoding and decoding schemes to each of the training images $\mathbf{y}^g$, and arrange the obtained images $\hat{\mathbf{y}}^g$ as the rows of a $G \times N$ matrix $\hat{\mathbf{Y}}^G$. Now, let $\mathbf{h}_k$ be a column stacked version of the 5×5 filter applied to a reconstructed image in the location of its kth pixel. Also, we denote by $\mathbf{e}_k$ and $\bar{\mathbf{R}}_k$ a vector and a matrix, whose right multiplications with $(\hat{\mathbf{y}}^g)^T$ extract the kth pixel and the transposed column-stacked version of the surrounding 5×5 patch, respectively. Then the filter $\mathbf{h}_k$ is obtained by solving the following least squares problem

$$\begin{aligned} \hat{\mathbf{h}}_k &= \underset{\mathbf{h}_k}{\operatorname{argmin}} \; \|\hat{\mathbf{Y}}^g \bar{\mathbf{R}}_k \mathbf{h}_k - \mathbf{Y}^g \mathbf{e}_k\|_2^2 \\ &= \left[\bar{\mathbf{R}}_k^T (\hat{\mathbf{Y}}^g)^T \hat{\mathbf{Y}}^g \bar{\mathbf{R}}_k\right]^{-1} \bar{\mathbf{R}}_k^T (\hat{\mathbf{Y}}^g)^T \mathbf{Y}^g \mathbf{e}_k. \end{aligned} \tag{7.3}$$

This process is repeated for each pixel, and it is part of the off-line training process.

(a) (b) (c)

Figure 7.3: (a) Mean face image. (b) Two atoms from the RTBWT dictionary.

7.4 Experimental Results

We assess the performance of our compression scheme on a database[3] containing 4515 grayscale asian face images with 8 bits per pixel. These images are the same ones used in [4], and they undergo the same preprocessing stage as in [4] – alignment according to the scheme proposed in [27] followed by a scale-down by a factor of 2. We use a random subset of 4415 aligned images of size 221×179 as the training set, and the remaining 100 aligned images as the test set.

We start by calculating from the training images a mean face image, shown in Figure 7.3(a). We can see that this image contains a relatively sharp face, whose facial features are shared by all the training images. We subtract this image from all the images in the training set, and construct the RTBWT with the resulting images. We use a 13-level wavelet decomposition with the Symmlet 8 filter, and obtain a dictionary with redundancy factor of 14. Figures 7.3(b) and (c) contain examples of two atoms from this dictionary, which correspond to the coefficients with the second and fourth largest magnitudes, out of the ones used to represent the middle left-most image in Figure 7.5. It can be seen that the atoms are either images of complete faces, or images containing details around face edges. We then encode all the images in the training set and calculate one Huffman table for 128 coefficient values, and another for 1024 index difference values.

We next use our proposed compression scheme to encode and decode each image in the test set, and use the results (in PSNR) obtained with and without post-processing to

[3]We chose this database since it contains thousands of images of both men and women in varying ages, and it was used by previous papers by our group [4,5], that provide results which serve as reference to compare against.

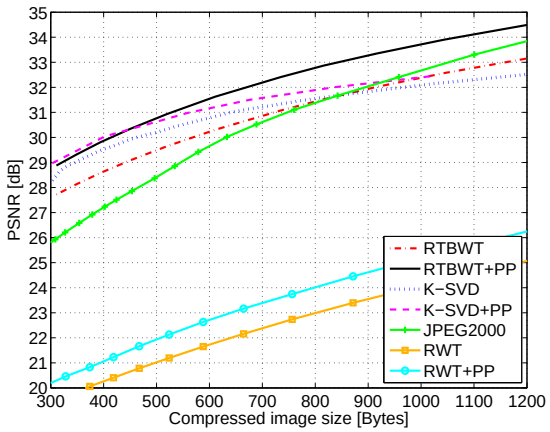


Figure 7.4: Rate distortion curves obtained with JPEG2000, the methods in [4] (K-SVD) and [5] (K-SVD+PP), and our proposed scheme using the RWT and the RTBWT with and without post-processing (PP).

calculate two rate-distortion curves. We compare these curves to the ones obtained by repeating this procedure with the common redundant wavelet transform (RWT) replacing the RTBWT in our scheme, and to rate-distortion curves obtained with JPEG2000[4], the algorithm described in [4] which is based on the K-SVD, and its improved version [5] which consists of a post-processing stage. We note that we used the matlab function "imwrite" to compress images in JPEG2000 format.

First it can be seen that the post-processing stage improves the performance of our scheme for all bit-rates both when the RTBWT and the RWT are used. Also, applying our scheme with the RTBWT improves its results by at least 8 dB compared to ones obtained with the RWT, both when post-processing is used and when it is not. Further, even without using post-processing our RTBWT-based algorithm outperforms JPEG2000, for all compressed-image sizes which are lower than 840 bytes. Our full RTBWT-based algorithm outperforms JPEG2000 for all bit rates: From a gain of 3 dB for low bit-rates to a gain of 0.6 dB for high bit-rates. Finally, without post-processing our RTBWT-based algorithm outperforms the algorithm in [4] for compressed-image sizes higher than 850 bytes, but obtains inferior results for smaller sizes. However, our full algorithm performs similarly to the algorithm in [5] for compressed-image sizes smaller than 450 bytes, but

[4]Since we test performance on very low bit rates, for a fair comparison we removed a fixed header size of 100 bytes from the JPEG2000 curve.

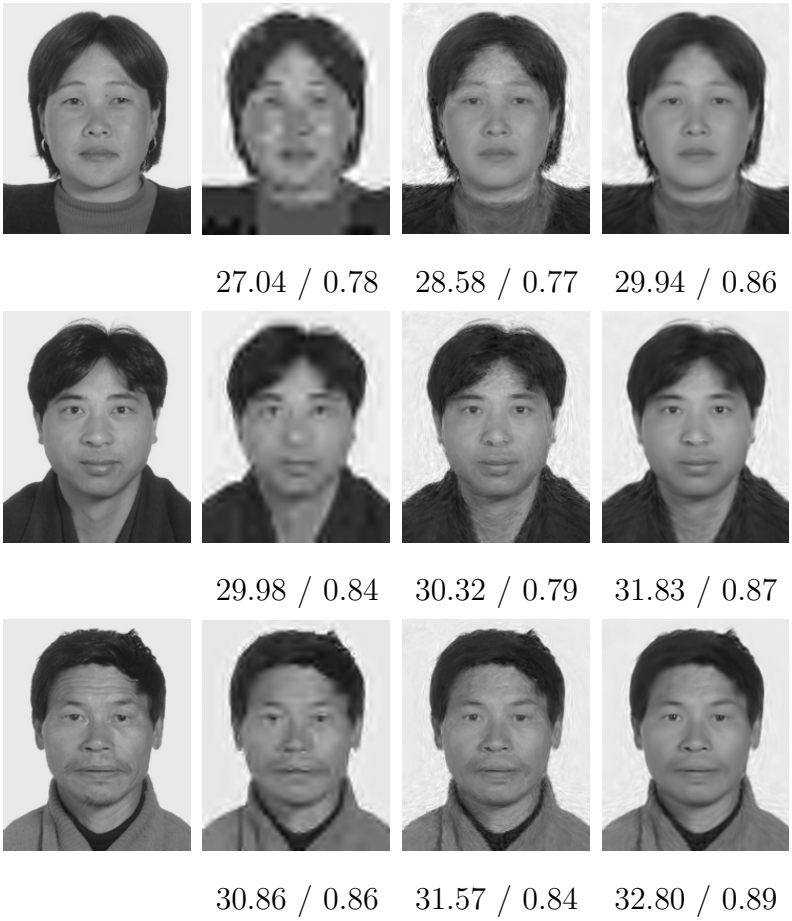


Figure 7.5: Facial image compression results (PSNR / SSIM) with compressed-image sizes of 400 bytes (first row), 600 bytes (center row), and 800 bytes (right row). The original images (first column) are compressed using JPEG2000 (second column), and our scheme without post-processing (third column) and with it (last column).

outperforms it for higher sizes even by more than 1 dB for compressed-image sizes higher than 1000 bytes.

We next demonstrate the visual quality of the results obtained with our RTBWT-based compression scheme when low bit-rates are used. Figure 7.5 compares both visually and in terms of PSNR and SSIM [64] the reconstructed images obtained for compressed-image sizes of 400, 600, and 800 bytes with our scheme, with and without post-processing, and with JPEG2000 with a header size of 100 bytes removed. It can be seen that our scheme obtains higher PSNR values than JPEG2000, and that post-processing further decreases these errors. In terms of SSIM, our scheme with post-processing outperforms JPEG2000. Without post-processing our results contain artifacts that look like paint-brush strokes,

causing a reduced SSIM. These artifacts are greatly reduced using the post-processing, and the obtained images are of relatively high quality despite the low bit-rates.

7.5 Conclusion

We have proposed a new face image compression scheme based on the redundant tree-based wavelet transform (RTBWT). We learn the transform from a training set containing aligned face images, and use it as a redundant dictionary when we encode images by applying sparse coding on them. Improved quality results are obtained by using a filtering-based post-processing scheme. We have demonstrated competitive performance compared to other methods, and managed to obtain results of high visual quality for low bit-rates.

Chapter 8

Image Processing using Patch Ordering[1]

8.1 Introduction

In Chapter 5 we proposed a patch-based image processing approach. We constructed image-adaptive wavelet transforms which are tailored to sparsely represent the given image. We used plain 1D wavelet transforms and adapted them to the image by operating on a permuted order of the image pixels[2]. The permutation we proposed is drawn from a shortest path ordering of the image patches. This way, the patches are leveraged to form multi-scale sparsifying global transforms for the image in question.

In this chapter we embark from our earlier work as reported in Chapter 5, adopting the core idea of ordering the patches. However, we discard the globality of the obtained transforms, the multi-scale treatment, and the sparsity-driven processing that follows. Thus, we propose a very simple image processing scheme that relies solely on patch reordering. We start by extracting all the patches of size $\sqrt{n} \times \sqrt{n}$ with maximal overlaps. Once these patches are extracted, we disregard their spatial relationships altogether, and seek a *new way* for organizing them. We propose to refer to these patches as a cloud of vectors/points in $\mathbb{R}^n$, and we order them such that they are chained in the "shortest

[1] This chapter is based on [52, 53].

[2] Note that the idea of adapting a wavelet transform to the image by reordering its pixels appears also in [47] and [34], but the schemes proposed there do not use image patches.

possible path", essentially solving the traveling salesman problem [14]. This reordering is the one we have used in Chapter 5, but as opposed to our past work, our treatment from this point varies substantially. A key assumption in this work is that proximity between two image patches implies proximity between their center pixels. Therefore if the image mentioned above is of high-quality, the new ordering of the patches is expected to induce a highly regular (smooth or at least piece-wise smooth) 1D ordering of the image pixels, being the center of these patches. When the image is deteriorated (noisy, containing missing pixels, etc.), the above ordering is expected to be robust to the distortions, thereby suggesting a reordering of the corrupted pixels to "what should be" a regular signal. Thus, applying relatively simple 1D smoothing operations (such as filtering or interpolation) to the reordered set of pixels should enable good recovery of the clean image.

This is the core process we propose in this chapter – for a given corrupted image, we reorder its pixels, operate on the new 1D signal using simplified algorithms, and reposition the resulting values to their original location. We explore two schemes which are based on two different approaches towards operating on the 1D reordered signals: 1) using simplified methods such as 1D filtering and interpolation; 2) applying a 1D version of the NL-means algorithm. We show that the two proposed schemes, applied with several randomly constructed orderings and combined with a proposed subimage averaging scheme, are able to lead to state-of-the-art results. We start by exploring the use of the first proposed scheme to image denoising, and show that it achieves better results than the ones obtained with the K-SVD based algorithm, proposed by Elad and Aharon [26], for medium and high noise levels, and generally performs better than the BM3D algorithm, propose by Dabov et al. [16], for high noise levels. We explore the use of this scheme to image inpainting, and show that it leads to better results compared to the ones obtained with a simple interpolation scheme and the method proposed in [25] which employs sparse representation modeling via the redundant DCT dictionary. We also show that our results are mostly better than the ones of the algorithm proposed by Li [39], and slightly inferior to the ones of the state-of-the-art PLE, proposed by Yu el tal. [68], for two of the three test images. Next, we draw some interesting ties between this scheme and BM3D rationale [16]. We then proceed to explore the use of the second proposed image processing scheme, which uses the NL-means algorithm, to image denoising. We show

that it outperforms both the NL-means algorithm and the first proposed image denoising scheme, and achieves results which are close to those of the BM3D algorithm proposed by Dabov et al. [16].

The chapter is organized as follows: In Section 8.2 we introduce the basic version of our first image processing scheme. We explain how the performance of the basic scheme can be improved using a subimage averaging scheme, and describe the connection between the improved scheme and the BM3D algorithm. We then explore the use of this scheme to image denoising and inpainting, and present experimental results that demonstrate the advantages of the proposed scheme. In Section 8.3 we introduce our second image processing scheme which uses the NL-means algorithm. We explore the use of the second scheme to image denoising, and present experimental results that demonstrate its advantages. We summarize the chapter in Section 8.4.

8.2 Image Processing using Patch Ordering

8.2.1 The Basic Scheme

Let $\mathbf{Y}$ be an image of size $N_1 \times N_2$ where $N_1 N_2 = N$, and let $\mathbf{Z}$ be a corrupted version of $\mathbf{Y}$, which may be noisy or contain missing pixels. Also, let $\mathbf{z}$ and $\mathbf{y}$ be the column stacked versions of $\mathbf{Z}$ and $\mathbf{Y}$, respectively. Then we assume that the corrupted image satisfies

$$\mathbf{z} = \mathbf{M}\mathbf{y} + \mathbf{v} \tag{8.1}$$

where the $N \times N$ matrix $\mathbf{M}$ denotes a linear operator which corrupts the data, and $\mathbf{v}$ denotes an additive white Gaussian noise independent of $\mathbf{y}$ with zero mean and variance σ^2. In this work the matrix $\mathbf{M}$ is restricted to represent a point-wise operator, covering applications such as denoising and inpainting. The reason for this restriction is the fact that we will be permuting the pixels in the image, and thus spatial operations become far more complex to handle.

Our goal is to reconstruct $\mathbf{y}$ from $\mathbf{z}$, and to this end we employ a permutation matrix $\mathbf{P}$ of size $N \times N$. We assume that when $\mathbf{P}$ is applied to the target signal $\mathbf{y}$, it produces a smooth signal $\mathbf{y}^p = \mathbf{P}\mathbf{y}$. We will explain how such a matrix may be obtained using the image patches in Section 8.2.2. We start by applying $\mathbf{P}$ to $\mathbf{z}$ and obtain $\mathbf{z}^p = \mathbf{P}\mathbf{z}$.

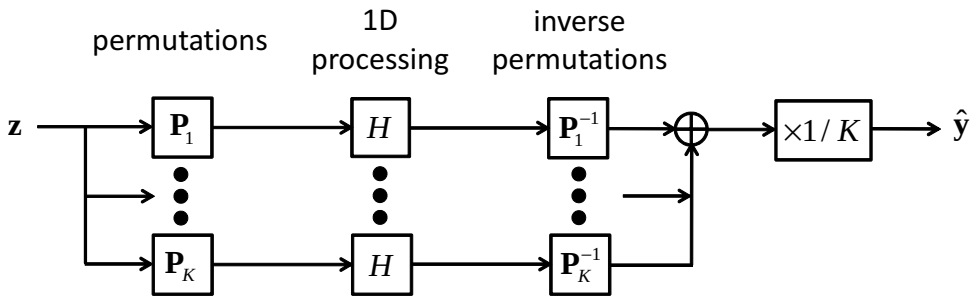


Figure 8.1: The basic image processing scheme.

Next, we take advantage of our prior knowledge that $\mathbf{y}^p$ should be smooth, and apply a "simple" 1D smoothing operator H on $\mathbf{z}^p$, such as 1D interpolation or filtering. Finally, we apply $\mathbf{P}^{-1}$ to the result, and obtain the reconstructed image

$$\hat{\mathbf{y}} = \mathbf{P}^{-1} H \left\{ \mathbf{P}\mathbf{z} \right\}. \tag{8.2}$$

In order to better smooth the recovered image, we use an approach which resembles the "cycle spinning" method proposed by Coifman and Donoho [11]. We randomly construct K different permutation matrices $\mathbf{P}_k$, utilize each to denoise the image $\mathbf{z}$ using the scheme described above, and average the results. This can be expressed by

$$\hat{\mathbf{y}} = \frac{1}{K} \sum_{k=1}^{K} \mathbf{P}_k^{-1} H \left\{ \mathbf{P}_k \mathbf{z} \right\}. \tag{8.3}$$

Figure 8.1 shows the proposed image processing scheme. We next describe how we construct the reordering matrix $\mathbf{P}$.

8.2.2 Building the Permutation Matrix P

We wish to design a matrix $\mathbf{P}$ which produces a smooth signal when it is applied to the target image $\mathbf{y}$. When the image $\mathbf{Y}$ is known, the optimal solution would be to reorder it as a vector, and then apply a simple *sort* operation on the obtained vector. However, we are interested in the case where we only have the corrupted image $\mathbf{Z}$ (noisy, containing missing pixels, etc.), and any permutation defined by simply reordering the corrupted pixels into a regular signal does not necessarily smooth $\mathbf{y}$. Therefore, as the pixels in the corrupted image are not helpful to us, we settle for a suboptimal ordering operation, using patches from the corrupted image.

Let y_i and z_i denote the ith samples in the vectors $\mathbf{y}$ and $\mathbf{z}$, respectively. We denote by $\mathbf{x}_i$ the column stacked version of the $\sqrt{n} \times \sqrt{n}$ patch around the location of z_i in $\mathbf{Z}$. We assume that under a distance measure[3] $w(\mathbf{x}_i, \mathbf{x}_j)$, proximity between the two patches $\mathbf{x}_i$ and $\mathbf{x}_j$ suggests proximity between the uncorrupted versions of their center pixels y_i and y_j. Thus, we shall try to reorder the points $\mathbf{x}_i$ so that they form a smooth path, hoping that the corresponding reordered 1D signal $\mathbf{y}^p$ will also become smooth. The "smoothness" of the reordered signal $\mathbf{y}^p$ can be measured using its total-variation measure

$$\|\mathbf{y}^p\|_{TV} = \sum_{j=2}^{N} |y^p(j) - y^p(j-1)|. \tag{8.4}$$

Let $\{\mathbf{x}_j^p\}_{j=1}^N$ denote the points $\{\mathbf{x}_i\}_{i=1}^N$ in their new order. Then by analogy, we measure the "smoothness" of the path through the points $\mathbf{x}_j^p$ by the measure

$$TV\left(\mathbf{x}_j^p\right) = \sum_{j=2}^{N} w(\mathbf{x}_j^p, \mathbf{x}_{j-1}^p). \tag{8.5}$$

Minimizing $TV\left(\mathbf{x}_j^p\right)$ comes down to finding the shortest path that passes through the set of points $\mathbf{x}_i$, visiting each point only once. This can be regarded as an instance of the traveling salesman problem [14], which can become very computationally expensive for large sets of points. We choose a simple and crude approximate solution, which is to start from a random point and then continue from each point $\mathbf{x}_{j_0}$ to its nearest neighbor $\mathbf{x}_{j_1}$ with a probability $p_1 = \alpha \exp\left[-w(\mathbf{x}_{j_0}, \mathbf{x}_{j_1})/\epsilon\right]$, or to its second nearest neighbor $\mathbf{x}_{j_2}$ with a probability $p_2 = \alpha \exp\left[-w(\mathbf{x}_{j_0}, \mathbf{x}_{j_2})/\epsilon\right]$, where α is determined such that $p_1 + p_2 = 1$, ϵ is a design parameter, and $\mathbf{x}_{j_1}$ and $\mathbf{x}_{j_2}$ are taken from the set of unvisited points.

We restrict the nearest neighbor search performed for each patch to a surrounding square neighborhood which contains $B \times B$ patches. When only one unvisited patch remains in that neighborhood, we simply continue to this patch, and in the case that no unvisited patches remain, we search for the first and second nearest neighbors among all the unvisited patches in the image. This restriction decreases the overall computational complexity, and our experiments show that with a proper choice of B it also leads to improved results. The permutation applied by the matrix $\mathbf{P}$ is defined as the order in the found path. The patch reordering scheme is summarized in the algorithm in Figure 8.2.

[3]Throughout this chapter we will be using variants of the squared Euclidean distance.

Task: Reorder the image patches $\mathbf{x}_j$.

Parameters: The image patches $\{\mathbf{x}_j\}_{j=1}^N$, the distance function w, and ϵ which is used in the probabilistic part of the algorithm.

Initialization: Choose a random index j and set $\Omega(1) = \{j\}$.

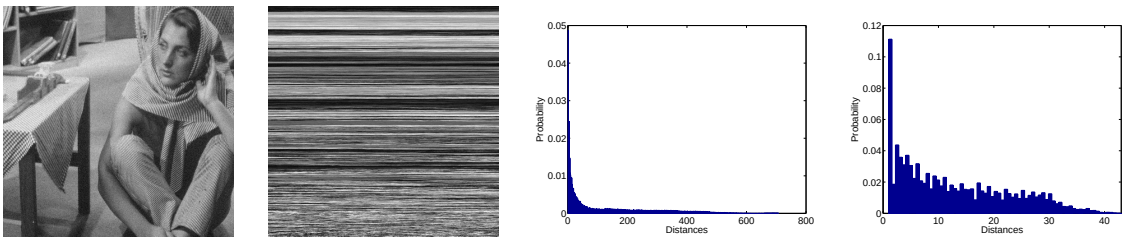
Main Iteration: Perform the following steps for $k = 1, \ldots, N-1$:

- Set A_k to be the set of indices of the $B \times B$ patches around $\mathbf{x}_{\Omega(k)}$.
- If $|A_k \setminus \Omega| = 1$
  - Set $\Omega(k+1)$ to be $A_k \setminus \Omega$.
- Else
  - If $|A_k \setminus \Omega| \geq 2$
    - Find $\mathbf{x}_{j_1}$ – the nearest neighbor to $\mathbf{x}_{\Omega(k)}$ such that $j_1 \in A_k$ and $j_1 \notin \Omega$.
    - Find $\mathbf{x}_{j_2}$ – the second nearest neighbor to $\mathbf{x}_{\Omega(k)}$ such that $j_2 \in A_k$ and $j_2 \notin \Omega$.
  - Elseif $|A_k \setminus \Omega| = 0$
    - Find $\mathbf{x}_{j_1}$ – the nearest neighbor to $\mathbf{x}_{\Omega(k)}$ such that $j_1 \notin \Omega$.
    - Find $\mathbf{x}_{j_2}$ – the second nearest neighbor to $\mathbf{x}_{\Omega(k)}$ such that $j_2 \notin \Omega$.
  - Set $\Omega(k+1)$ to be:
    - $\{j_1\}$ with probability $p_1 \propto \exp\left[-\frac{w(\mathbf{x}_{\Omega(k)}, \mathbf{x}_{j_1})}{\epsilon}\right]$
    - $\{j_2\}$ with probability $p_2 = 1 - p_1 \propto \exp\left[-\frac{w(\mathbf{x}_{\Omega(k)}, \mathbf{x}_{j_2})}{\epsilon}\right]$

Output: The set Ω holds the proposed ordering.

Figure 8.2: The patch reordering algorithm we use.

We next demonstrate the smoothing effect that a permutation obtained from a corrupted image has on its clean version. We apply the patch ordering scheme described above, with the parameters $\sqrt{n} = 6$, $B = 61$ and $\epsilon = 10^6$, to the patches of the noisy Barbara image with noise standard deviation $\sigma = 10$, shown in Figure 8.3(a). We apply the obtained permutation to the column stacked version $\mathbf{y}$ of the clean Barbara image, and obtain the reordered signal $\mathbf{y}^p$. We then calculate the total variations of $\mathbf{y}$ and $\mathbf{y}^p$, and obtain that $\|\mathbf{y}\|_{TV} = 2.49 \cdot 10^6$ and $\|\mathbf{y}^p\|_{TV} = 1.76 \cdot 10^6$, i.e. the latter is 29 percent smaller than the former. In order to visually demonstrate the regularity of the reordered signal $\mathbf{y}^p$, we fold it back by a zig-zag raster scan into the original image size. The obtained



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

Figure 8.3: (a) Noisy Barbara ($\sigma = 10$). (b) Reordered clean Barbara, folded back by s zig-zag raster scan. (c) and (d) – Normalized histograms of the spatial distances between adjacent patches after the reordering, obtained with unrestricted and restricted search areas, respectively The lengths of the pathes in the spatial domain obtained with the unrestricted and restricted searches were $3.63 \cdot 10^7$ and $3.42 \cdot 10^6$, respectively.

image is shown in Figure 8.3(b), and it can be seen that the reordered signal is indeed piecewise regular for most of its length, but becomes less and less regular towards its end. In fact, when the last 30% of the samples of $\mathbf{y}$ and $\mathbf{y}^p$ are discarded, the total variation of the obtained reordered signal becomes 37 percent smaller than the total variation of the obtained column stacked signal. This behavior of the reordered images might be expected due to the greedy nature of the patch ordering algorithm, which leaves for the patches near the end of the path a very small number of unvisited patches to choose from. The employed cycle spinning scheme prevents the non-regular parts of the reordered images from degrading the quality of the reconstructed images.

It is also interesting to examine the characteristics of the patch ordering in the spatial domain. To this end we also apply the patch ordering scheme to the patches of the noisy Barbara image with the same parameters, but with an unrestricted search neighborhood. We apply the permutations obtained with the two types of neighborhoods to the patches, and calculate two normalized histograms of the spatial distances between adjacent patches, shown in Figure 8.3. Figure 8.3(c) shows that when the search neighborhood is not restricted, only about 5% of neighboring patches in the path are also immediate spatial neighbors, and that far away patches are often assigned as neighbors in the reordering process. The histogram in Figure 8.3(d), obtained with restricted search neighborhood, is limited to show only distances which are smaller or equal to 43, the maximal possible distance within the search window. It can be seen that despite the re-

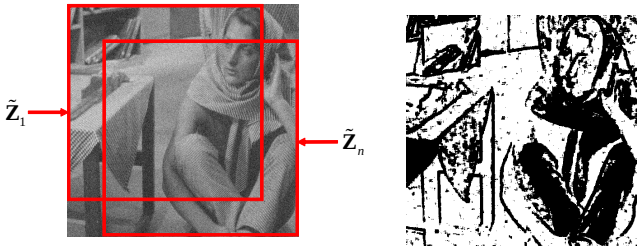
striction to a smaller search neighborhood, only about 11% of neighboring patches in the path are also immediate spatial neighbors, and patches all over the search neighborhood are assigned as neighbors in the reordering process.

In order to facilitate the cycle-spinning method mentioned above, we simply run the proposed ordering solver K times, and the randomness (both in the initialization and in assigning the neighbors) leads to different permutation results. We next describe how the quality of the produced images may be further improved using a subimage averaging scheme, which can be seen as another variation of cycle spinning.

8.2.3 Subimage Averaging

Let $N_p = (N_1 - \sqrt{n} + 1)(N_2 - \sqrt{n} + 1)$ denote the number of overlapped patches in the image $\mathbf{Z}$, and let $\mathbf{X}$ be an $n \times N_p$ matrix, containing column stacked versions of these patches. We extract these patches column by column, starting from the top left-most patch. When we calculated $\mathbf{P}$ as described in the previous section, we assumed that each patch is associated only with its middle pixel. Therefore $\mathbf{P}$ was designed to reorder the signal composed of the middle points in the patches, which reside in the middle row of $\mathbf{X}$. However, we can alternatively choose to associate all the patches with a pixel located in a different position, e.g., the top left pixel in each patch. This means that the matrix $\mathbf{P}$ can be used to reorder any one of the signals located in the rows of $\mathbf{X}$. These signals are the column stacked versions of all the n subimages of size $(N_1 - \sqrt{n} + 1) \times (N_2 - \sqrt{n} + 1)$ contained in the image $\mathbf{Z}$. We denote these subimages by $\tilde{\mathbf{Z}}_j$, $j = 1, 2, \ldots, n$. An example for two of them, $\tilde{\mathbf{Z}}_1$ and $\tilde{\mathbf{Z}}_n$, contained in a noisy version of the image Barbara with noise standard deviation $\sigma = 25$, is shown Figure 8.4(a).

We already observed in Chapters 5 and 6 that improved denoising results are obtained when all the n subimages of a noisy image are employed in its denoising process. Here we use a similar scheme in order to improve the quality of the recovered image. In order to avoid cumbersome notations we first describe a scheme which utilizes a single ordering matrix $\mathbf{P}$. Let the vector $\tilde{\mathbf{z}}_j = \mathbf{R}_j \mathbf{z}$ of length N_p be the column stacked version of $\tilde{\mathbf{Z}}_j$, where the $N_p \times N$ matrix $\mathbf{R}_j$ extracts the jth subimage from the image $\mathbf{z}$. We first calculate the $N_p \times N_p$ matrix $\mathbf{P}$ using the patches in $\mathbf{X}$ and apply it to each subimage $\tilde{\mathbf{z}}_j$. Then we apply the operator H to each of the reordered subimages $\tilde{\mathbf{z}}_j^p = \mathbf{P}\tilde{\mathbf{z}}_j$, apply the



(a) (b)

Figure 8.4: (a) Two subimages $\tilde{\mathbf{Z}}_1$ and $\tilde{\mathbf{Z}}_n$ contained in a noisy version of the image Barbara ($\sigma = 25$). (b) Classification of the pixels in a noisy version of the image Barbara to centers of smooth (white) and non-smooth (black) patches.

inverse permutation $\mathbf{P}^{-1}$ on the result, and obtain the reconstructed subimages

$$\hat{\tilde{\mathbf{y}}}_j = \mathbf{P}^{-1} H\{\mathbf{P}\tilde{\mathbf{z}}_j\} = \mathbf{P}^{-1} H\{\mathbf{P}\mathbf{R}_j\mathbf{z}\}. \tag{8.6}$$

We next reconstruct the image from all the reconstructed subimages $\hat{\tilde{\mathbf{y}}}_j$ by plugging each subimage into its original place in the image canvas and averaging the different values obtained for each pixel. More formally, we obtain the reconstructed image $\hat{\mathbf{y}}$ as follows:

$$\hat{\mathbf{y}} = \mathbf{D}^{-1} \sum_{j=1}^{n} \mathbf{R}_j^T \hat{\tilde{\mathbf{y}}}_j = \mathbf{D}^{-1} \sum_{j=1}^{n} \mathbf{R}_j^T \mathbf{P}^{-1} H\{\mathbf{P}\mathbf{R}_j\mathbf{z}\} \tag{8.7}$$

where the matrix $\mathbf{R}_j^T$ plugs the estimated jth subimage into its original place in the canvas, and

$$\mathbf{D} = \sum_{j=1}^{n} \mathbf{R}_j^T \mathbf{R}_j \tag{8.8}$$

is an $N \times N$ diagonal weight matrix that simply averages the overlapping contributions per each pixel. When K random matrices $\mathbf{P}_k$ are employed, we obtain the final estimate by averaging the images obtained with the different permutations

$$\hat{\mathbf{y}} = \frac{1}{K} \sum_{k=1}^{K} \left(\mathbf{D}^{-1} \sum_{j=1}^{n} \mathbf{R}_j^T \mathbf{P}_k^{-1} H\{\mathbf{P}_k\mathbf{R}_j\mathbf{z}\} \right). \tag{8.9}$$

This formula reveals two important properties of our scheme: (i) the two summations that correspond to the two cycle-spinning versions lead to an averaging of nK candidate solutions, a fact that boosts the overall performance of the recovery algorithm; and (ii) if H is chosen as linear, then the overall processing is linear as well, provided that we disregard the highly non-linear dependency of $\mathbf{P}$ on $\mathbf{z}$.

8.2.4 Connection to BM3D and Clustering-Based Methods

The above processing scheme can be described a little differently. We start by calculating the permutation matrix $\mathbf{P}$ from the image patches $\mathbf{x}_i$. We then gather the patches by arranging them as the columns of a matrix $\mathbf{X}^p$ in the order defined by $\mathbf{P}$. This matrix contains in its rows the reordered subimages $\tilde{\mathbf{z}}_j^p$, therefore we next apply the operator H to its rows, and shuffle the columns of the resulting matrix according to the permutation defined by $\mathbf{P}^{-1}$. We obtain a matrix $\bar{\mathbf{X}}$, which contains in its rows the reconstructed subimages $\hat{\tilde{\mathbf{y}}}_j$, and in its columns reconstructed versions $\bar{\mathbf{x}}_i$ of the image patches $\mathbf{x}_i$. We obtain the reconstructed image $\hat{\mathbf{y}}$ from the patches $\bar{\mathbf{x}}_i$ by plugging them into their original places in the image canvas, and averaging the different values obtained for each pixel. When K random matrices $\mathbf{P}_k$ are employed, we apply the aforementioned scheme with each of these matrices, and average the obtained images.

Now, looking at the image processing scheme described above, we can see some similarities to the first stage of the BM3D algorithm. Both algorithms stack the image patches into groups, apply 1D processing across the patches, return the patches into their place in the image, and average the results. We note that the BM3D algorithm also applies a 2D transform to the patches before performing 1D processing across them. This feature can be easily added to our scheme if needed, and we regard this as a preprocessing part of the operator H. On the other hand, there are some key differences between the two schemes. First, while the BM3D algorithm constructs a group of neighbors for each patch, here we order *all the patches* to one chain, which defines local neighbors. Furthermore, this process is repeated K times, implying that our approach consider K different neighbors assignments. Also, while in the BM3D the patch order in each group is not restricted, ours is carefully determined as it plays a major role in our scheme. Finally, the 1D processing applied by the BM3D consists of the use of a 1D transform, followed by thresholding and the inverse transform, implying a specific denoising. Here we do not restrict ourselves to any specific 1D processing scheme, and allow the operator H to be chosen based on the application at hand.

We conclude this discussion with the following comment. When observing clustering-based restoration methods in general, such as [16, 19, 20, 40], there is a clear similarity

to our approach, as near-by patches find themselves supporting each other's processing. However, as opposed to these methods, our technique also orders the near-by patches into a smooth path, and as such, our approach enables a better treatment of the patches even within the clustered groups. For example, if the corresponding center pixel forms a smooth linear slope of values, the clustering approaches would approximate this line by the average value, while our approach would regress to a smooth line. We next demonstrate the application of our proposed scheme to image denoising and inpainting.

8.2.5 Applications and Results

8.2.5.1 Image Denoising using Learned Filters

The problem of image denoising consists of the recovery of an image from its noisy version. In that case $\mathbf{M} = \mathbf{I}$ and the corrupted image satisfies $\mathbf{z} = \mathbf{y} + \mathbf{v}$. The patches $\mathbf{x}_i$ contain noise, and we choose the distance measure between $\mathbf{x}_i$ and $\mathbf{x}_j$ to be the squared Euclidean distance divided by n, i.e

$$w(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{n}\|\mathbf{x}_i - \mathbf{x}_j\|^2. \tag{8.10}$$

In Chapter 5 we applied a complex multi-scale processing on the ordered patches. Here we wish to employ a far simpler scheme; we choose a 1D linear shift-invariant filter, and as we show next, we learn this filter from training images. Furthermore, we suggest to switch between two such filters, based on the patch content.

We desire to treat smooth areas in the image differently than areas with edges or texture, as our experiments show that this approach leads to better results. More specifically, we employ different permutation matrices and filters in the smooth and non smooth areas of the image. We first divide the patches into two sets: S_s - which contains smooth patches, and S_e - which contains patches with edges or texture. Let $\mathrm{std}(\mathbf{x}_i)$ denote the standard deviation of the patch $\mathbf{x}_i$ and let C be a scalar design parameter. Then we use the following classification rule: if $\mathrm{std}(\mathbf{x}_i) < C\sigma$ then $\mathbf{x}_i \in S_s$, otherwise $\mathbf{x}_i \in S_e$. Figure 8.4(b) demonstrates the application of this classification rule to the noisy Barbara image shown in Figure 8.4(a), where we use the parameters $\sqrt{n} = 8$ and $C = 1.2$ which we will later use in the denoising process of this image. White pixels are the centers of smooth

patches and black pixels are the centers of patches containing texture or edges. It can be seen that the obtained image indeed contains a rough classification of the patches into smooth and non smooth sets.

We next divide each subimage $\tilde{\mathbf{z}}_j$ into two signals: $\tilde{\mathbf{z}}_{j,s}$ - a vector of length $|S_s|$ which contains the pixels corresponding to the smooth patches, and $\tilde{\mathbf{z}}_{j,e}$ - a vector of length $|S_e|$ which contains the pixels corresponding to the patches with edges and texture. We construct an $|S_s| \times |S_s|$ matrix $\mathbf{P}_s$, which extracts $\tilde{\mathbf{z}}_{j,s}$ from $\tilde{\mathbf{z}}_j$, and applies to it a permutation obtained from the patches in the set S_s using the nearest neighbors search method described above. We similarly construct an $|S_e| \times |S_e|$ matrix $\mathbf{P}_e$, which extracts $\tilde{\mathbf{z}}_{j,e}$ from $\tilde{\mathbf{z}}_j$, and applies to it a permutation obtained from the patches in the set S_e, using the same set of parameters. We apply $\mathbf{P}_s$ and $\mathbf{P}_e$ to $\tilde{\mathbf{z}}_j$ and obtain $\tilde{\mathbf{z}}^p_{j,s}$ and $\tilde{\mathbf{z}}^p_{j,e}$, respectively, which are the signals to which we apply the filters. We define the reordered subimage $\tilde{\mathbf{z}}^p_j$ and the $N_p \times N_p$ permutation matrix $\mathbf{P}$ which satisfy

$$\tilde{\mathbf{z}}^p_j = \begin{bmatrix} \tilde{\mathbf{z}}^p_{j,s} \\ \tilde{\mathbf{z}}^p_{j,e} \end{bmatrix}, \mathbf{P} = \begin{bmatrix} \mathbf{P}_s \\ \mathbf{P}_e \end{bmatrix} \tag{8.11}$$

and obtain that $\tilde{\mathbf{z}}^p_j = \mathbf{P}\tilde{\mathbf{z}}_j$.

We next wish to find the filters $\mathbf{h}_s$ and $\mathbf{h}_e$, each of length N_h, applied to $\tilde{\mathbf{z}}^p_{j,s}$ and $\tilde{\mathbf{z}}^p_{j,e}$, respectively. We denote by $\mathbf{M}^p_{j,s}$ the $|S_s| \times N_h$ convolution matrix corresponding to $\tilde{\mathbf{z}}^p_{j,s}$, and by $\mathbf{M}^p_{j,e}$ the $|S_e| \times N_h$ convolution matrix corresponding to $\tilde{\mathbf{z}}^p_{j,e}$, and obtain the filtered subimages

$$\begin{aligned} \hat{\tilde{\mathbf{y}}}_j &= \mathbf{P}^{-1} \begin{bmatrix} \mathbf{M}^p_{j,s}\mathbf{h}_s \\ \mathbf{M}^p_{j,e}\mathbf{h}_e \end{bmatrix} = \mathbf{P}^{-1} \begin{bmatrix} \mathbf{M}^p_{j,s} & 0 \\ 0 & \mathbf{M}^p_{j,e} \end{bmatrix} \begin{bmatrix} \mathbf{h}_s \\ \mathbf{h}_e \end{bmatrix} \\ &= \mathbf{P}^{-1}\mathbf{M}^p_j\mathbf{h}_t \end{aligned} \tag{8.12}$$

where we defined the $N_p \times N_p$ matrix $\mathbf{M}^p_j$, and the filters vector $\mathbf{h}_t$ of length $2N_h$, which satisfy

$$\mathbf{M}^p_j = \begin{bmatrix} \mathbf{M}^p_{j,s} & 0 \\ 0 & \mathbf{M}^p_{j,e} \end{bmatrix}, \mathbf{h}_t = \begin{bmatrix} \mathbf{h}_s \\ \mathbf{h}_e \end{bmatrix}. \tag{8.13}$$

The vector $\mathbf{h}_t$ stores the filter taps to be designed. We substitute (8.12) in (8.7), and

obtain the reconstructed image

$$\hat{\mathbf{y}} = \mathbf{D}^{-1} \sum_{j=1}^{n} \mathbf{R}_j^T \mathbf{P}^{-1} \mathbf{M}_j^p \mathbf{h}_t. \tag{8.14}$$

When K random matrices $\mathbf{P}_k$ are employed, we obtain the final estimate by averaging the images obtained with the different matrices

$$\begin{aligned} \hat{\mathbf{y}} &= \frac{1}{K} \sum_{k=1}^{K} \left(\mathbf{D}^{-1} \sum_{j=1}^{n} \mathbf{R}_j^T \mathbf{P}_k^{-1} \mathbf{M}_j^p \mathbf{h}_t \right) \\ &= \frac{1}{K} \sum_{k=1}^{K} \mathbf{D}^{-1} [\mathbf{R}_1^T, \ldots, \mathbf{R}_n^T] \begin{bmatrix} \mathbf{P}_k^{-1} \mathbf{M}_1^p \\ \vdots \\ \mathbf{P}_k^{-1} \mathbf{M}_n^p \end{bmatrix} \mathbf{h}_t = \mathbf{Q} \mathbf{h}_t \end{aligned} \tag{8.15}$$

where we defined the $N \times 2N_h$ matrix

$$\mathbf{Q} = \frac{1}{K} \sum_{k=1}^{K} \mathbf{D}^{-1} [\mathbf{R}_1^T, \ldots, \mathbf{R}_n^T] \begin{bmatrix} \mathbf{P}_k^{-1} \mathbf{M}_1^p \\ \vdots \\ \mathbf{P}_k^{-1} \mathbf{M}_n^p \end{bmatrix}. \tag{8.16}$$

Now let $\mathbf{y}^g$, $g = 1, \ldots, G$ be a training set which contains the column stack versions of G clean images. For each such image we create a noisy version $\mathbf{z}^g$ by adding noise with the same statistics as the noise in $\mathbf{z}$. Then we calculate for each image $\mathbf{z}^g$ a matrix $\mathbf{Q}^g$ using (8.16), and learn the filters vector $\mathbf{h}_t$ by minimizing

$$\begin{aligned} \hat{\mathbf{h}}_t &= \underset{\mathbf{h}_t}{\operatorname{argmin}} \sum_{g=1}^{G} \|\mathbf{y}^g - \mathbf{Q}^g \mathbf{h}_t\|^2 \\ &= \left[\sum_{g=1}^{G} (\mathbf{Q}^g)^T \mathbf{Q}^g \right]^{-1} \sum_{k=1}^{G} (\mathbf{Q}^g)^T \mathbf{y}^g. \end{aligned} \tag{8.17}$$

Once we have the filters vector $\hat{\mathbf{h}}_t$ we can employ it to denoise $\mathbf{z}$ by building $\mathbf{Q}$ using (8.16) and then calculating

$$\hat{\mathbf{y}} = \mathbf{Q} \hat{\mathbf{h}}_t. \tag{8.18}$$

We can further improve our results by applying a second iteration of our proposed scheme, in which all the processing stages remain the same, but the permutation matrices are built using patches extracted from the first iteration clean result.

In the following experiments we assess the performance of the proposed image denoising scheme by applying it to a test set containing noisy versions of 6 images, with

Table 8.1: Parameters used in the denoising experiments.

σ	Iteration	K	$\sqrt{n}$	C	B	ϵ	N_h
5	1	10	5	2.2	61	10^6	25
	2	10	4	1.2	361	10^3	25
10	1	10	6	1.6	61	10^6	25
	2	10	4	0.8	361	10^3	25
15	1	10	7	1.4	61	10^6	25
	2	10	4	0.6	361	10^3	25
20	1	10	8	1.3	61	10^6	25
	2	10	4	0.5	361	10^3	25
25	1	10	8	1.2	61	10^6	25
	2	10	4	0.4	361	10^3	25
50	1	10	14	1.1	61	10^6	25
	2	10	5	0.3	361	10^3	25
75	1	10	16	1.1	61	10^6	25
	2	10	6	0.2	361	10^3	25
100	1	10	16	1.1	61	10^6	25
	2	10	8	0.1	361	10^3	25

8 different noise standard deviations. We use a training set containing the images Man, Couple, and Hill to learn the filters vector $\mathbf{h}_t$. We start by demonstrating the performance gain obtained by combining the patch classification and filter learning into the proposed denoising scheme. We compare the results obtained with our scheme, for noisy images with noise standard deviation $\sigma = 25$, in three settings: 1) without patch classification and using a Gaussian smoothing filter; 2) without patch classification and using a learned filter; 3) with patch classification and using two learned filters. In all 3 cases we use the parameters shown in Table 8.1, which correspond to $\sigma = 25$ and 1 iteration. The results are shown in Table 8.2, and it can be seen that using a learned filter instead of a simple smoothing filter improves the results for all images, with an average increase in PSNR of 2.32 dB. Also, performing patch classification further improves the results for all images, with an average increase in PSNR of 0.54 dB.

Table 8.2: Denoising results (PSNR in dB) of noisy versions of 6 images ($\sigma = 25$), obtained with the proposed scheme in three settings 1) without patch classification and using Gaussian smoothing filter (Gaussian). 2) without patch classification and using a learned filter (1 learned). 3) with patch classification and using two learned filters (2 learned). For each image and setting the best result is highlighted.

Image	Gaussian	1 learned	2 learned
Lena	29.14	30.90	**31.54**
Barbara	27.42	30.14	**30.36**
Boats	26.66	28.91	**29.5**
Fingerprint	24.24	26.99	**27.24**
House	29.75	31.61	**32.34**
Peppers	26.38	28.99	**29.78**

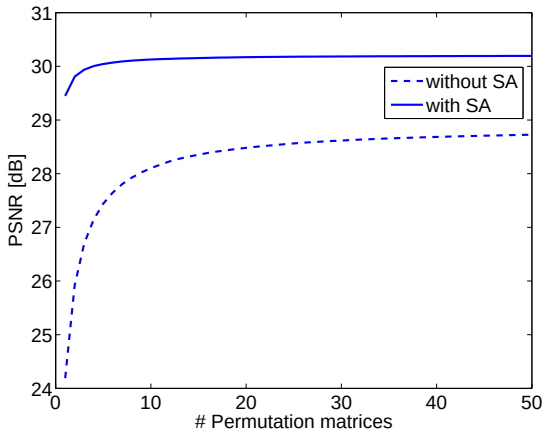


Figure 8.5: Average of the PSNR values obtained for the noisy test images ($\sigma = 25$), with and without subimage averaging, as a function of the number of permutations.

We also demonstrate the improvement in the performance obtained with cycle spinning and subimage averaging. Figure 8.5 shows the average of the PSNR values obtained for the noisy test images with our scheme, including patch classification and filter learning, with and without subimage averaging, as a function of the number of employed permutation matrices. First it can be seen that the results in both cases improve as the number of employed permutations K increases, and that the subimage averaging improves the performance by between 1.5 dB for large values of K to 5 dB when $K = 1$. It can also

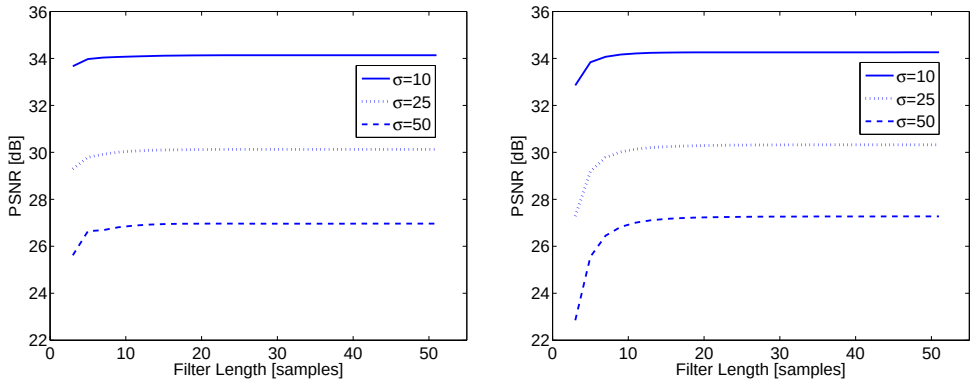


Figure 8.6: Average of the PSNR values obtained for the test images, as a function of the filter length, for three different noise levels: (a) First iteration. (b) Second Iteration.

be seen that when the subimage averaging scheme is used, most of the performance gain is obtained using the first 10 permutations. In fact, employing 40 more permutations increases the results by less than 0.07 dB. Thus, as a compromise between the quality of performance and computational cost, in the following experiments we utilize $K = 10$ permutation matrices when we apply our image denoising scheme.

We next apply our overall scheme, to all the noisy images in the test set. The parameters employed by the proposed denoising scheme for the different noise levels are shown in Table 8.1. We note that the reason we chose a uniform filter length of $N_h = 25$ samples for all noise levels can be justified using Figure 8.6. Figure 8.6. shows the average of the PSNR values obtained in the first and second iterations for all the test images, as a function of the filter length, for three different noise levels. It can be seen that in both iterations the performance gain obtained using filters longer than 25 samples is negligible. The trained filters obtained in each iteration for the three different noise levels are shown in Figure 8.7.

First, it can be seen that filters $\mathbf{h}_s$ and $\mathbf{h}_e$ indeed look different. It can also be seen that in the first iteration the shape of the filter $\mathbf{h}_s$ does not change much as the noise level increases, and in the second iteration the filters obtained for the higher noise levels are similar, but very different from the filter obtained for $\sigma = 10$. On the other hand, in both iterations the shape of the filter $\mathbf{h}_e$ changes greatly as the noise level increases.

For comparison, we also apply the K-SVD based algorithm proposed by Elad an Aharon [26] and BM3D algorithm proposed by Dabov et al. [16] . The PSNR values

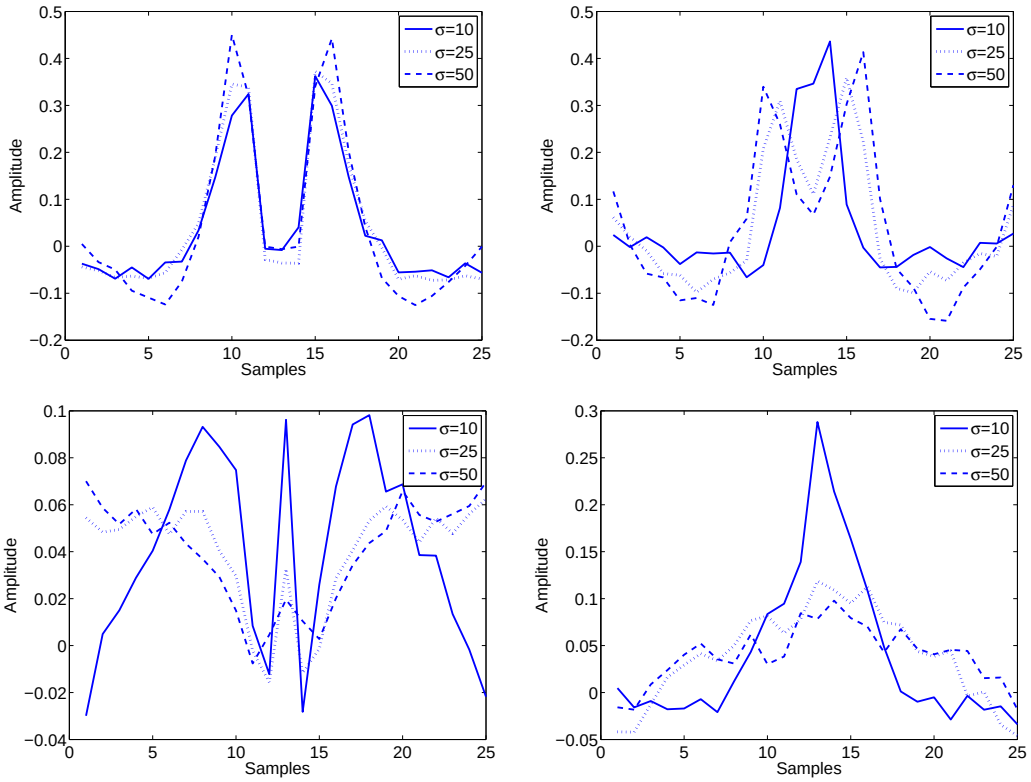


Figure 8.7: The trained filters learned from the noisy images for $\sigma = 10, 25, 50$: Left column - the filters $\mathbf{h}_s$ obtained in the first (top) and second (bottom) iterations. Right column - the filters $\mathbf{h}_e$ obtained in the first (top) and second (bottom) iterations.

of the results obtained with these algorithms, and two iterations of our denoising scheme are shown in Table 8.3. The noisy and recovered images obtained with our scheme for $\sigma = 25$ are shown in Figure 8.8. First, it can be seen that the second iteration improves the results of our proposed scheme in all the cases. It can also be seen that the results obtained with two iterations of our scheme are inferior to the those of the K-SVD for $\sigma < 15$, but are better almost everywhere for $\sigma \geq 15$. Further, our two iterations results are generally better than those of the state-of-the-art BM3D algorithm for $\sigma \geq 50$, but are inferior for higher SNRs.

8.2.5.2 Image Inpainting

The problem of image inpainting consists of the recovery of missing pixels in the given image. Here we handle the case where there is no additive noise, therefore $\mathbf{v} = 0$, and $\mathbf{M}$ is a diagonal matrix of size $N \times N$ which contains ones and zeroes in its main diagonal

Table 8.3: Denoising results (PSNR in dB) of noisy versions of 6 images, obtained with the K-SVD and BM3D algorithms and two iterations of the proposed scheme. For each image and noise level the best result is highlighted.

Image	Method	σ/PSNR							
		5/34.16	10/28.14	15/24.61	20/22.11	25/20.18	50/14.16	75/10.63	100/8.14
Lena	K-SVD	38.63	35.52	33.76	32.40	31.35	27.85	25.81	24.47
	BM3D	**38.72**	**35.93**	**34.28**	**33.04**	**32.05**	28.96	27.16	25.80
	prop1	38.22	35.26	33.64	32.48	31.54	28.66	26.8	25.45
	prop2	38.31	35.39	33.84	32.72	31.80	**28.96**	**27.22**	**26.01**
Barbara	K-SVD	38.08	34.40	32.33	30.79	29.54	25.43	23.02	21.89
	BM3D	**38.29**	**34.93**	**33.05**	**31.69**	**30.61**	27.16	25.11	23.61
	prop1	37.63	34.29	32.54	31.32	30.36	27.19	25.14	23.56
	prop2	37.74	34.39	32.65	31.43	30.47	**27.35**	**25.42**	**24.07**
Boats	K-SVD	37.25	33.65	31.74	30.35	29.30	25.94	24.04	22.85
	BM3D	**37.29**	**33.94**	**32.15**	**30.87**	**29.89**	**26.71**	**25.01**	23.88
	prop1	37.09	33.64	31.79	30.49	29.50	26.35	24.58	23.34
	prop2	37.10	33.70	31.91	30.67	29.70	26.69	24.99	**23.90**
Fingerprint	K-SVD	**36.66**	32.42	30.09	28.46	27.26	23.23	19.97	18.29
	BM3D	36.52	**32.47**	**30.30**	**28.83**	**27.72**	**24.54**	**22.82**	**21.57**
	prop1	36.00	31.88	29.72	28.27	27.24	24.02	22.25	21.11
	prop2	36.19	32.01	29.84	28.38	27.34	24.13	22.47	21.44
House	K-SVD	39.33	35.90	34.19	32.97	31.97	28.01	25.27	23.59
	BM3D	**39.84**	**36.63**	**34.87**	**33.72**	**32.79**	29.54	27.42	25.78
	prop1	38.54	35.61	34.08	33.06	32.34	29.28	27.26	25.51
	prop2	38.76	35.80	34.35	33.32	32.54	**29.64**	**27.79**	**26.30**
Peppers	K-SVD	37.80	34.27	32.23	30.88	29.81	26.24	23.54	21.68
	BM3D	**38.09**	**34.70**	**32.77**	**31.37**	**30.26**	26.69	24.71	23.20
	prop1	37.59	34.13	32.21	30.86	29.78	26.28	24.21	22.53
	prop2	37.63	34.26	32.40	31.09	30.01	**26.75**	**24.72**	**23.21**

corresponding to existing and missing pixels, correspondingly. Each patch may contain missing pixels, and we denote by S_i the set of indices of non-missing pixels in the patch $\mathbf{x}_i$. We choose the distance measure between patches $\mathbf{x}_i$ and $\mathbf{x}_j$ to be the average of squared differences between existing pixels that share the same location in both patches, i.e.

$$w(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{k \in S_i \cap S_j} (x_i[k] - x_j[k])^2}{|S_i \cap S_j|}. \tag{8.19}$$

We start by calculating the matrix $\mathbf{P}$ according to the scheme described in Section 8.2.2, with a minor difference: when a patch does not share pixels with any of the unvisited patches, the next patch in the path is chosen to be its nearest spatial neighbor. We next

Figure 8.8: Denoising results (PSNR) for the images Lena, Barbara, Boat and House ($\sigma = 25$, input PSNR=20.18 dB): top row - noisy images, center row - 1 iteration results, and bottom row - 2 iterations results.

apply the obtained matrix to the subimages $\tilde{\mathbf{z}}_j$, and observe that the permuted vectors $\tilde{\mathbf{z}}_j^p = \mathbf{P}\tilde{\mathbf{z}}_j$ contain missing values. We bear in mind that the target signals $\tilde{\mathbf{y}}_j^p = \mathbf{P}\tilde{\mathbf{y}}^j$ should be smooth, and therefore apply on the subimages $\tilde{\mathbf{z}}_j^p$ an operator H which recovers the missing values using cubic spline interpolation. We apply the matrix $\mathbf{P}^{-1}$ on the resulting vectors and obtain the estimated subimages $\hat{\tilde{\mathbf{y}}}_j$. The final estimate is obtained from these subimages using (8.7). We improve our results by applying two additional iterations of a modified version of this inpainting scheme, where the only difference is that we rebuild $\mathbf{P}$ using reconstructed (and thus full) patches.

We demonstrate the performance of our proposed scheme on corrupted versions of the images Lena, Barbara and House, obtained by zeroing 80% of their pixels, which are selected at random. The parameters employed in each of the three iterations are

Table 8.4: Parameters used in the inpainting experiments.

Iteration	K	$\sqrt{n}$	B	ϵ
1	10	16	9	10^2
2	10	8	43	10^4
3	10	5	55	10^8

shown in Table 8.4. In order to demonstrate the advantages of our method over simpler interpolation schemes we compare our results to the ones obtained by the matlab function "griddata" which performs cubic interpolation of the missing pixels based on Delaunay triangulation [65, 66]. We also compare the performance of our algorithm to those of three other patch-based algorithms. The first is the algorithm described in chapter 15 of [25], which employs a patch-based sparse representation reconstruction algorithm with a DCT overcomplete dictionary to recover the image patches. We use a patch size of 16×16 pixels in order to improve the results this method produces. We note that we do not employ the K-SVD based algorithm which was also described in this chapter, as our experiments showed that it produces comparable or only slightly better results than the redundant DCT dictionary, at a higher computational cost. The other two algorithms to which we compare our results are the algorithm proposed by Li [39], which is based on transforming groups of patches, and the state-of-the-art PLE algorithm proposed by Yu et al. [68], which is based on patch clustering. We present here the results reported for these algorithms in [68], for the case of 80% missing pixels. The PSNR values of the results obtained with the different algorithms are shown in Table 8.5. Figure 8.9 shows the corrupted and the reconstructed images, with the corresponding PSNR values, obtained using Delaunay triangulation, overcomplete DCT dictionary, and 1 and 3 iterations of the proposed scheme. First, it can be seen that the second and third iterations greatly improve the results of our proposed algorithm. It can also be seen that the results obtained with three iterations of our proposed scheme are much better than those obtained with Delaunay triangulation and the overcomplete DCT dictionary. Further, it can be seen that our three iterations results are better than the results obtained with the patch grouping algorithm proposed by Li [39] for the images Lena and Barbara, but slightly inferior to its results for the image House. Finally, it can be seen that the stat-of-the-art PLE algorithm

Table 8.5: Inpainting results (PSNR in dB) of corrupted versions of the images Lena, Barbara and House with 80% of their pixels missing, obtained using Delaunay triangulation (DT), overcomplete DCT dictionary, the algorithm in [39], PLE [68], and 1 (P1), 2 (P2), and 3 (P3) iterations of the proposed scheme. For each image the best result is highlighted.

Image	DT	DCT	[39]	[68]	P1	P2	P3
Lena	30.25	29.97	31.62	**32.22**	30.25	31.80	31.96
Barb	22.88	27.15	25.40	**30.94**	27.56	29.34	29.71
House	29.21	29.69	32.87	**33.05**	29.03	32.10	32.71

proposed by Yu et al. [68] outperforms our three iterations results by only 0.3 dB for the images Lena and House, but by more than 1.2 dB for the image Barbara.

8.2.5.3 Computational Complexity

We next evaluate the computational complexity of a single iteration of the two image processing algorithms described above. We note that for the image denoising scheme, we assume that the filters training has been done beforehand, and exclude it from our calculations. First, building the matrix $\mathbf{X}$ which contains the image patches requires $O(nN)$ operations. We assume that when the nearest-neighbor search described above is used with a search window of size $B \times B$, most of the patches do not require to calculate distances outside this neighborhood. Therefore, as calculating each of the distance measures (8.10) and (8.19) requires $O(n)$ operations, the number of operations required to calculate a single reordering matrix $\mathbf{P}_k$ can be bounded by $O(NB^2n)$. Next, applying the matrices $\mathbf{P}_k$ and $\mathbf{P}_k^{-1}$ to the n subimages $\mathbf{z}_j$ require $O(nN)$ operations, and so does applying either one of the operators H described above to the n subimages $\mathbf{z}_j^p$. Finally , constructing an estimate image by averaging the pixel values obtained with the different subimages also requires $O(nN)$ operations, and averaging the estimates obtained with the different matrices $\mathbf{P}_k$ requires $O(KN)$ operations. Therefore when K permutation matrices are employed, the total complexity is

$$O((n+K)N) + K\left[O(NB^2n) + O(nN)\right] = O(NKB^2n) \tag{8.20}$$

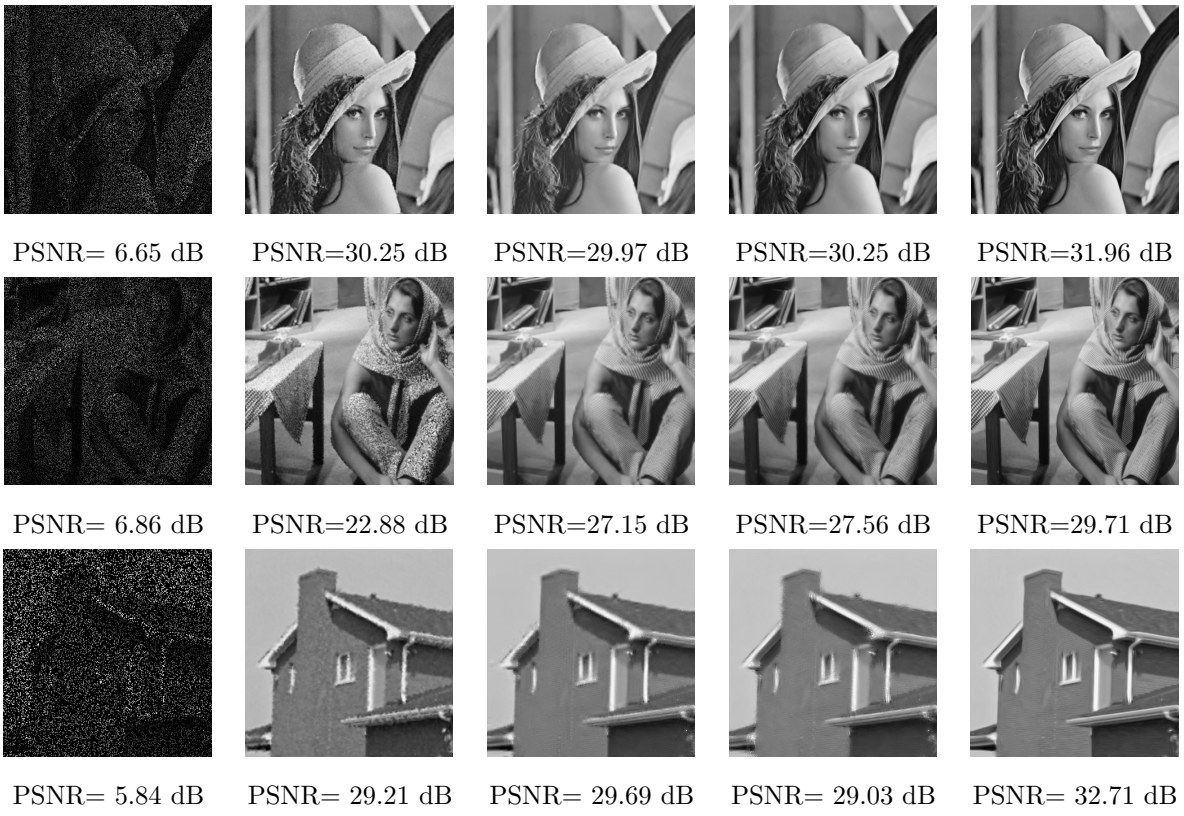


Figure 8.9: Inpainting results (PSNR) of corrupted versions of the images Lena, Barbara and House with 80% of their pixels missing, obtained with different reconstruction methods: : First column - corrupted images, Second column - Delaunay triangulation, Third column - overcomplete DCT dictionary, Fourth column - 1 iteration of the proposed scheme, Fifth column - 3 iterations of the proposed scheme.

operations, which means that, as might be expected, the overall complexity is dominated by the creation of the permutation matrices. For a typical case in our experiments, $N = 512^2$, $K = 10$, $n = 64$ and $B = 61$, the above amounts to $6.24 \cdot 10^{11}$ operations. In order to better illustrate these numbers, we also provide run-times: applying two iterations of our denoising scheme to a 512×512 image with noise level $\sigma = 25$ using a non optimized and non parallel matlab implementation, on an Intel(R) Core(TM) i7-2600 CPU @ 3.40 GHz, takes about 45 minutes. Applying three iterations of our inpainting scheme to a 512×512 image with 80% missing pixels take only about 14 minutes. We should note that while in our experiments we employed exact exhaustive search, approximate nearest neighbor algorithms may be used to alleviate the computational burden.

8.3 Image denoising using NL-Means via Smooth Patch Ordering

The denoising algorithm proposed in Section 8.2.5.1 has two main limitations. First, it ignores altogether the distances between the patches. These distances carry additional information, because they have a key role in the construction of the matrices $\mathbf{P}_k$, as was explained in Section 8.2.2. Additionally, the smoothing filters require a separate training set to be learned from. Our goal is to develop an image denoising algorithm which employs the matrices $\mathbf{P}_k$, takes into account the distances between the patches, and do not require filter learning. To this end we modify the NL-means algorithm, which takes into account these distances in the denoising process.

8.3.1 The Basic Denoising Scheme

We operate in the same setting as the one described in Section 8.2.5.1, i.e the noisy image satisfies $\mathbf{z} = \mathbf{y} + \mathbf{v}$. We start by calculating K permutation matrices $\mathbf{P}_k$ from the image patches $\mathbf{x}_j$, using the scheme described in Section 8.2.2. We apply these matrices to $\mathbf{z}$ and obtain K permuted signals $\mathbf{z}_k^p = \mathbf{P}_k\mathbf{z}$. We wish to modify the NL-means algorithm proposed by Baudes et al. [6] so it will make use of these signals. The NL-means algorithm estimates each pixel y_i as a weighted average of pixels in $\mathbf{Z}$, which reside in a square neighborhood surrounding z_i. The weights are determined by the distances between the patch surrounding the estimated pixel and the patches surrounding the pixels in that neighborhood. More formally, let S_i^{NL} denote the set of indices of the pixels in the square neighborhood surrounding z_i. Then

$$\hat{y}_i = \frac{1}{D_i} \sum_{j \in S_i^{NL}} z_j w_{i,j} \tag{8.21}$$

where the weights $w_{i,j}$ and D_i satisfy

$$w_{i,j} = w(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{n\alpha}\right), \; D_i = \sum_{j \in S_i^{NL}} w_{i,j}. \tag{8.22}$$

We next employ the signals $\mathbf{z}_k^p$ to construct for each image pixel z_i a modified neighborhood, where we denote by S_i the set of indices of the pixels in that neighborhood. Let $\mathbf{p}_k$

denote a vector containing the locations in $\mathbf{z}$ of the pixels in $\mathbf{z}_k^p$. Also, let j_i^k be the index of the pixel z_i in $\mathbf{z}_k^p$, i.e. $z_k^p[j_i^k] = z_i$. As $\mathbf{y}_k^p$ should be smooth, $y_k^p[j_i^k]$ should be close to its L neighboring pixels, and therefore we choose to estimate it from their noisy versions. Thus, we first define $S_{k,i}$ as a set of indices of pixels in the modified neighborhood of z_i

$$S_{k,i} = \{p_k[j_i^k - \frac{L}{2}], \ldots, p_k[j_i^k + \frac{L}{2}]\}. \tag{8.23}$$

Then we define the total set of indices of pixels in the modified neighborhood of z_i as

$$S_i = \bigcup_{k=1}^{K} S_{k,i}. \tag{8.24}$$

We obtain our proposed image denoising algorithm by replacing the neighborhood defined by S_i^{NL} with the one defined by S_i in (8.21) and (8.22). As can be seen, unlike the algorithm in Section 8.2.5.1 we do make use of the distances between the patches, and we do not require a preliminary filter-learning stage. We next describe how the results obtained with our method can be further improved.

8.3.2 Improving the Denoising results

We employ the three following methods in order to further improve the denoising results. First, similarly to what is done in Section 8.2.5.1, we improve our results by treating smooth areas in the image differently than areas with edges or texture. We divide the patches into two sets: S_s - which contains smooth patches, and S_e - which contains patches with edges or texture. Let $\mathrm{std}(\mathbf{x}_i)$ denote the standard deviation of the patch $\mathbf{x}_i$ and let C be a scalar design parameter. Then we use the following classification rule: if $\mathrm{std}(\mathbf{x}_i) < C\sigma$ then $\mathbf{x}_i \in S_s$, otherwise $\mathbf{x}_i \in S_e$. We next divide the image $\mathbf{z}$ into two signals: $\mathbf{z}_s$ - which contains the pixels corresponding to the smooth patches, and $\mathbf{z}_e$ - which contains the pixels corresponding to the patches with edges and texture. We apply the denoising scheme described above to the signals $\mathbf{z}_s$ and $\mathbf{z}_e$, with the sets of patches S_s and S_e, and two sets of parameters L_s, α_s and L_e, α_e for the NL-means, respectively. We recover the two signals $\hat{\mathbf{y}}_s$ and $\hat{\mathbf{y}}_e$ from $\mathbf{z}_s$ and $\mathbf{z}_e$, respectively, and obtain the final estimate $\hat{\mathbf{y}}$ by returning the pixels in each signal to their original place in the image canvas.

Table 8.6: Parameters used in the denoising experiments.

σ	Iter	$\sqrt{n}$	B	C	ϵ	L_s	L_e	α_s	α_e
10	1	7	31	1.2	10	9	5	3.3	1.7
	2	4	231	1.1	10^3	33	3	0.4	1.4
25	1	12	31	1.1	10^2	11	5	4.1	1.7
	2	4	131	0.3	10^7	71	11	0.3	0.5
50	1	16	31	1.1	10^2	11	5	5	5.5
	2	6	141	0.1	10^3	91	19	0.2	0.3

Next, we employ the subimage averaging scheme. Similarly to what is done in Section 8.2.5.1, in order to improve the quality of the recovered image we utilize all the n subimages of a noisy image in its denoising process. We apply the proposed denoising scheme, with the patch classification described above, to each of the subimages $\tilde{\mathbf{z}}_j$, and obtain reconstructed subimages $\hat{\tilde{\mathbf{y}}}_j$. We next reconstruct the image $\mathbf{y}$ from all the subimages $\hat{\tilde{\mathbf{y}}}_j$ by plugging each subimage into its original place in the image canvas and averaging the different values obtained for each pixel.

Finally, we can further improve the quality of the recovered image by applying the noisy image a second iteration of our proposed denoising scheme, with the two improvements described above. In the second iteration, all the processing stages remain the same, but the applied permutations, the standard deviations of the patches, and the weights $w_{i,j}$, are all calculated using patches extracted from the first iteration clean result.

8.3.3 Experimental Results

In order to assess the performance of the proposed image denoising scheme we apply it to noisy versions of the images Lena, Barbara and House, with noise standard deviations $\sigma = 10, 25, 50$. In all our experiments we used $K = 10$ permutation matrices. The rest of the parameters employed by the proposed scheme for the three noise levels are shown in Table 8.6. For comparison, we also apply the NL-means and BM3D [16] algorithms, and two iterations of the algorithm proposed in Section 8.2.5.1, to the noisy images. The PSNR values of the results obtained with the classic NL-means, BM3D, the algorithm in Section 8.2.5.1, and the two iterations of our denoising scheme are shown in Table 8.7.

Table 8.7: Denoising results (PSNR in dB) of noisy versions of the images Lena, Barbara and House,obtained with the NL-means and BM3D algorithms, two iterations of the algorithm proposed in Section 8.2.5.1, and the proposed scheme. For each image and noise level the best result is highlighted.

Image	Method	σ/PSNR		
		10/28.14	25/20.18	50/14.16
Lena	NL-means	34.73	30.43	26.84
	BM3D	**35.93**	**32.08**	29.05
	8.2.5.1 (2 iter.)	35.44	31.78	28.96
	prop. (1 iter.)	35.41	31.50	28.46
	prop. (2 iter.)	35.78	32.03	**29.16**
Barbara	NL-means	33.34	28.35	24.62
	BM3D	**34.98**	30.72	27.23
	8.2.5.1 (2 iter.)	34.49	30.56	27.40
	prop. (1 iter.)	34.46	30.08	26.67
	prop. (2 iter.)	34.75	**30.76**	**27.48**
House	NL-means	35.22	30.66	26.27
	BM3D	**36.71**	32.86	29.69
	8.2.5.1 (2 iter.)	35.91	32.61	29.91
	prop. (1 iter.)	36.2	32.23	28.96
	prop. (2 iter.)	36.55	**33.07**	**30.21**

The noisy and recovered images obtained with our scheme for $\sigma = 25$ are shown in Figure 8.10. First, it can be seen that the second iteration improves the results of our proposed scheme in all the cases. It can also be seen that the results obtained with two iterations of our scheme are better than the ones obtained with NL-means and the algorithm in Section 8.2.5.1. Further, compared with BM3D results, our second iteration results are slightly better for $\sigma = 50$, comparable for $\sigma = 25$, and slightly worse for $\sigma = 10$.

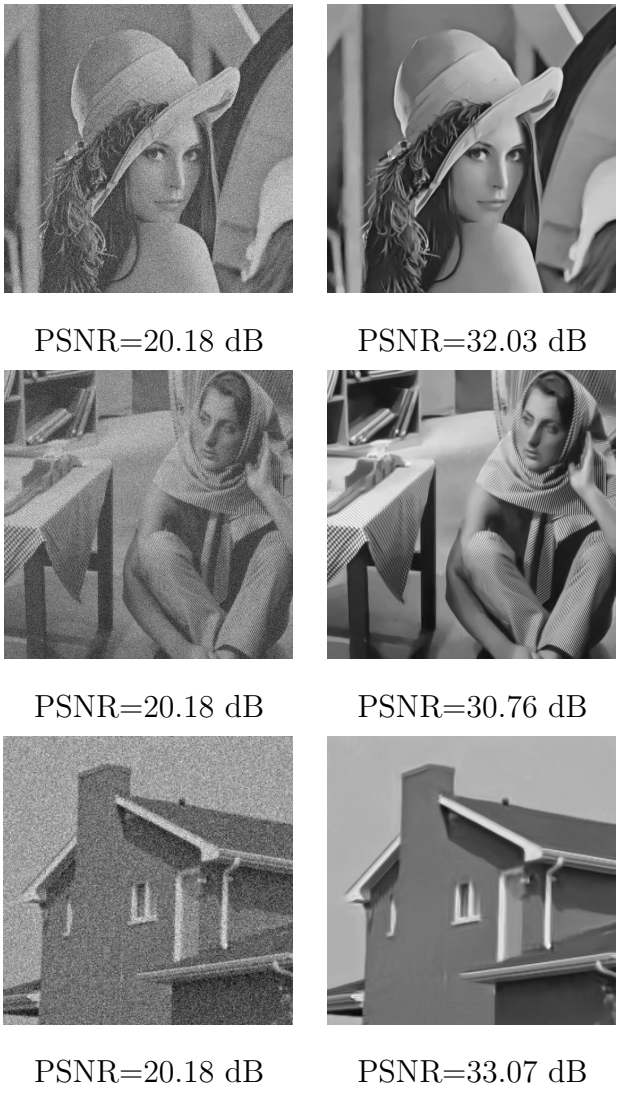


Figure 8.10: Denoising results (PSNR) for the images Lena, Barbara and House ($\sigma = 25$, input PSNR=20.18 dB): Left column - noisy images, Right column - 2 iterations results.

8.4 Conclusions

We have proposed a new image processing scheme which is based on smooth 1D ordering of the pixels in the given image. We have shown that using a carefully designed permutation matrices and simple and intuitive 1D operations such as linear filtering and interpolation, the proposed scheme can be used for image denoising and inpainting, where it achieves high quality results. We have also shown that improved image denoising results are obtained when a modified version of the NL-means algorithm is used instead of linear filtering. We have shown that combined with patch classification and subimage averaging

schemes, applying two iterations of either of our proposed schemes produces results which are close to the state-of-the-art.

Chapter 9

Research Summary and Future Directions

9.1 Research summary

In this thesis we have extended the orthonormal and redundant wavelet transforms, to functions defined on graphs and high dimensional data-points. Both schemes make use of permutations calculated from features points in their different scales, and these permutations enable them to adapt to their input signals and sparsely represent them. We have explained how these transforms may be applied to images, and used them to construct image denoising, deblurring and compression algorithms, which achieved state-of-the-art results. We have also developed an image processing scheme which is based on smooth 1D reordering of the image pixels using similar permutations calculated from the image patches, which led to state-of-the-art image denoising and inpainting results. The main contributions of the thesis chapters are as follows:

In Chapter 3, we introduced the generalized tree-based wavelet transform (GTBWT) applicable to graphs and high dimensional data. This transform is an extension of the multiscale harmonic analysis approach proposed by Gavish et al. [30], which uses Haar-like wavelets. We have designed our transform to use general wavelet filters by adding data-driven operators into the common orthogonal wavelet transform decomposition scheme.

Chapter 4 introduces the redundant tree-based wavelet transform (RTBWT) applicable to scalar functions which is also defined on graphs and high dimensional data clouds.

This transform is the redundant version of the GTBWT introduced in Chapter 3. We have introduced an alternative implementation for this transform which is based on the à trous algorithm, and used it to show that the RTBWT is a frame. We have shown that our proposed transform can be used for the recovery of labels defined on point clouds.

Next, in Chapter 5 we explained how to construct the generalized and redundant tree-based wavelet transforms from images. We demonstrated the ability of the GTBWT to represent images more efficiently than the common 1D and separable 2D wavelet transforms. We have also shown that our proposed transforms can be used for image denoising, and that combined with a subimage averaging and a vector thresholding scheme they achieves denoising results which are close to the state-of-the-art.

In Chapter 6, we showed that the image representation obtained with the RTBWT combined with the subimage averaging scheme constitutes a frame, and calculated the synthesis and analysis operators associated with it. We have proposed image denoising and deblurring algorithms which make use of these operators as sparsity promoting regularizers, and demonstrated state-of-the-art results.

In Chapter 7 we proposed a new face image compression scheme based on RTBWT. We learn the transform from a training set containing aligned face images, and use it as a redundant dictionary when we encode images by applying sparse coding on them. Improved quality results are obtained by using a filtering-based post-processing scheme. We have demonstrated competitive performance compared to other methods, and managed to obtain results of high visual quality for low bit-rates.

Finally, in Chapter 8, we proposed a new image processing scheme which is based on smooth 1D ordering of the pixels in the given image. We have shown that using a carefully designed permutation matrices and simple and intuitive 1D operations such as linear filtering and interpolation, the proposed scheme can be used for image denoising and inpainting, where it achieves high quality results. We have also shown that improved image denoising results are obtained when a modified version of the NL-means algorithm is used instead of linear filtering. We have shown that combined with patch classification and subimage averaging schemes, applying two iterations of either of our proposed schemes produces results which are close to the state-of-the-art.

9.2 Future research directions

In this thesis we developed two data adaptive wavelet transforms, and a patch ordering based image processing method, which are all based on permutations of feature points, or patches in the case of images. We explored the application of the proposed transforms mainly for image processing tasks such as denoising, deblurring, and facial image compression, and the application of the patch ordering based method to image denoising and inpainting.

For future research, we first propose to apply the proposed transforms to different image processing problems, and also find new applications, that involve processing functions on graphs and point clouds. Our patch ordering based image processing method may also be extended to new image processing algorithms which involve optimization problems in which the 1D image reorderings act as regularizers. These may both improve the image denoising and inpainting results, and allow to tackle other applications such as image deblurring, where the operator $\mathbf{M}$ (defined in Section 8.2.1) is no longer restricted to be point-wise local. The patch ordering based image processing method may also be generalized to work on graphs or point clouds by replacing the image patches and their center pixels with feature points and scalar functions defined on them.

Next, one can try to improve the ordering scheme used by all the aforementioned methods to calculate permutations. This greedy method finds an approximation to the shortest path through the feature points or the patches. We have seen in Section 8.2.1 that this scheme performs poorly near the end of the found path, when only a small number of unvisited patches remain. A possible solution could be to develop a scheme which allows patches/feature points to be revisited more than once. It would also be interesting to explore the use of a weighted average in the subimage averaging scheme, that is shared to most of our proposed image processing scheme. Intuitively it seems that the proposed patch-based methods would perform better on subimages which reside closer to the center of the image, and therefore their pixels should receive higher weights when the final estimate is calculated.

The image processing schemes proposed in Chapter 6 may be extended in several manners. First the thresholding operator S_τ (defined in Section 6.4) may be replaced with

an operator that takes into consideration the change in the noise levels in the different wavelet scales. Further, the operators $\mathbf{\Phi}^{SA}$ and $\mathbf{\Psi}^{SA}$ (defined in Section 6.3) may be used to solve different image processing problems such as image inpainting, superresolution, tomographic reconstruction, and more. Finally, it might be interesting to assess the performance gain (if any), obtained by a non-simplified version of our image denoising scheme, that uses finite values of the parameter η (defined in Section 6.4) and more than one iteration in each stage.

Several research directions may be considered to extend the scheme proposed in Chapter 7. A first direction is to learn a set of indices of leading coefficients, which is shared by all encoded images. These indices will be known to the decoder, and therefore only their corresponding values will be sent by the encoder, thus achieving better compression. A different direction is to train different dictionaries for different parts of the image or for different sub-groups of images in order to obtain dictionaries which are more adapted to the data. Such dictionaries may lead to a sparser image representation and improved quality of the reconstructed images. The performance of our scheme may also be improved by replacing the entropy coding technique it uses from Huffman coding to arithmetic coding. Then we may compare its results to those of the advanced HEVC compression scheme [56] which reduces the bit rate by about 20% compared to JPEG2000 [33].

Finally, regarding the schemes proposed in Chapter 8, we note that we have not exhausted the potential of the proposed algorithms, and the choice of different parameters (e.g., B and ϵ defined in Section 8.2) for each set of patches may also improve the produced results. Also, the filtering and NL-means based image denoising schemes may be improved by dividing the patches in each scheme to more than two types, and treating each type differently. One may also try to improve the performance of state-of-the-art algorithms like the BM3D and the algorithm proposed by Mairal et al. [40] by employing our proposed patch neighborhoods in their block matching procedure.

Bibliography

[1] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans. Signal processing*, 54(11):4311, 2006.

[2] G. Beylkin. On the representation of operators in bases of compactly supported wavelets. *SIAM J. Numer. Anal.*, 29(6):1716–1740, 1992.

[3] A. M. Bruckstein, D. L. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review*, 51(1):34–81, 2009.

[4] O. Bryt and M. Elad. Compression of facial images using the k-svd algorithm. *Journal of Visual Communication and Image Representation*, 19(4):270–282, 2008.

[5] O. Bryt and M. Elad. Improving the k-svd facial image compression using a linear deblocking method. In *Proc. 25th IEEE Convention of Electrical and Electronics Engineers in Israel*, pages 533–537. IEEE, 2008.

[6] A. Buades, B. Coll, and J. M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling and Simulation*, 4(2):490–530, 2006.

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

[8] G. Chen and M. Maggioni. Multiscale geometric wavelets for the analysis of point clouds. In *Information Sciences and Systems (CISS), 2010 44th Annual Conference on*, pages 1–6. IEEE, 2010.

[9] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Rev.*, 43(1):59–129, 2001.

[10] F. R. K. Chung. Spectral graph theory. volume 92. CBMS Regional Conference Series in Mathematics, AMS Bookstore, 1997.

[11] R. R. Coifman and D. L. Donoho. *Wavelets and Statistics*, chapter Translation-invariant de-noising, pages 125–150. Springer-Verlag, 1995.

[12] R. R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006.

[13] R. R. Coifman and M. Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, 2006.

[14] T. H. Cormen. *Introduction to algorithms*. The MIT press, 2001.

[15] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 1998.

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

[17] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image restoration by sparse 3d transform-domain collaborative filtering. In *SPIE Electronic Imaging*, volume 6812, 2008.

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

[19] W. Dong, X. Li, L. Zhang, and G. Shi. Sparsity-based image denoising via dictionary learning and structural clustering. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 457–464. IEEE, 2011.

[20] W. Dong, L. Zhang, G. Shi, and X. Wu. Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization. *IEEE Trans. Image Processing*, 20(7):1838–1857, 2011.

[21] D. L. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via $\ell 1$ minimization. *Proc. National Academy of Sciences*, 100(5):2197–2202, 2003.

[22] D. L. Donoho, M. Elad, and V. N. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Trans. Information Theory*, 52(1):6–18, 2006.

[23] D. L. Donoho and X. Huo. Uncertainty principles and ideal atomic decomposition. *IEEE Trans. Information Theory*, 47(7):2845–2862, 2001.

[24] K. Egiazarian and J. Astola. Tree-structured Haar transforms. *Journal of Mathematical Imaging and Vision*, 16(3):269–279, 2002.

[25] M. Elad. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer Verlag, 2010.

[26] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans. Image Processing*, 15(12):3736–3745, 2006.

[27] M. Elad, R. Goldenberg, and R. Kimmel. Low bit-rate compression of facial images. *IEEE Trans. Image Processing*, 16(9):2379–2383, 2007.

[28] M. Elad, P. Milanfar, and R. Rubinstein. Analysis versus synthesis in signal priors. *Inverse problems*, 23(3):947–968, 2007.

[29] J.E. Fowler. The redundant discrete wavelet transform and additive noise. *IEEE Signal Processing Letters*, 12(9):629–632, 2005.

[30] M. Gavish, B. Nadler, and R. R. Coifman. Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning. In *Proceedings of the 27th International Conference on machine Learning*, 2010.

[31] S.P. Ghael, A.M. Sayeed, and R.G. Baraniuk. Improved wavelet denoising via empirical wiener filtering. In *Proc. SPIE*, volume 3169, pages 389–399, 1997.

[32] D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 2010.

[33] P. Hanhart, M. Rerabek, P. Korshunov, and T. Ebrahimi. Subjective evaluation of hevc intra coding for still image compression. In *Seventh International Workshop on Video Processing and Quality Metrics for Consumer Electronics-VPQM 2013*, 2013.

[34] D. Heinen and G. Plonka. Wavelet shrinkage on paths for denoising of scattered data. *Results in Mathematics*, 62(3-4):337–354, 2012.

[35] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets. Time-Frequency Methods and Phase Space*, volume 1, pages 286–297. Springer-Verlag, 1989.

[36] M. Jansen, G. P. Nason, and B. W. Silverman. Multiscale methods for data on graphs and irregular multidimensional situations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(1):97–125, 2009.

[37] S. Lafon, Y. Keller, and R. R. Coifman. Data fusion and multicue data matching by diffusion maps. *IEEE Transactions on pattern analysis and machine intelligence*, 28(11):1784–1797, 2006.

[38] A. B. Lee, B. Nadler, and L. Wasserman. Treelets-An adaptive multi-scale basis for sparse unordered data. *Annals*, 2(2):435–471, 2008.

[39] X. Li. Patch-based image interpolation: algorithms and applications. In *International Workshop on Local and Non-Local Approximation in Image Processing*, 2008.

[40] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2272–2279. IEEE, 2009.

[41] J. Mairal, M. Elad, G. Sapiro, et al. Sparse representation for color image restoration. *IEEE Transactions on Image Processing*, 17(1):53, 2008.

[42] S. Mallat. Geometrical grouplets. *Applied and Computational Harmonic Analysis*, 26(2):161–180, 2009.

[43] S. Mallat. *A Wavelet Tour of Signal Processing, The Sparse Way*. Academic Press, 2009.

[44] S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Trans. Signal Processing*, 41(12):3397–3415, 1993.

[45] F. Murtagh. The Haar wavelet transform of a dendrogram. *Journal of Classification*, 24(1):3–32, 2007.

[46] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *presented at the 27th Annu. Asilomar Conf. Signals, Systems, and Computers*, pages 40–44. IEEE, 1993.

[47] G. Plonka. The Easy Path Wavelet Transform: A New Adaptive Wavelet Transform for Sparse Representation of Two-Dimensional Data. *Multiscale Modeling & Simulation*, 7:1474–1496, 2009.

[48] I. Ram, I. Cohen, and M. Elad. Facial Image Compression using Patch-Ordering-Based Adaptive Wavelet Transform. *IEEE Signal Processing Letters*, 21(10):1270–1274, 2014.

[49] I. Ram, I. Cohen, and M. Elad. Patch-ordering-based wavelet frame and its use in inverse problems. *IEEE Trans. Image Processing*, 33(7):2779–2792, 2014.

[50] I. Ram, M. Elad, and I. Cohen. Generalized Tree-Based Wavelet Transform . *IEEE Trans. Signal Processing*, 59(9):4199–4209, 2011.

[51] I. Ram, M. Elad, and I. Cohen. Redundant Wavelets on Graphs and High Dimensional Data Clouds. *IEEE Signal Processing Letters*, 19(5):291–294, 2012.

[52] I. Ram, M. Elad, and I. Cohen. Image denoising using nl-means via smooth patch ordering. In *Proc. 38th IEEE Internat. Conf. Acoust. Speech Signal Process., ICASSP-2013*, pages 1350–1354. IEEE, 2013.

[53] I. Ram, M. Elad, and I. Cohen. Image processing using smooth ordering of its patches. *IEEE Trans. Image Processing*, 22(7):2764 – 2774, 2013.

[54] M.J. Shensa. The discrete wavelet transform: Wedding the a trous and mallat algorithms. *IEEE Trans. Signal Processing*, 40(10):2464–2482, 1992.

[55] A. Singer, Y. Shkolnisky, and B. Nadler. Diffusion interpretation of non-local neighborhood filters for signal denoising. *SIAM J. on Imag. Sci*, 2(1):118–139, 2009.

[56] G. J. Sullivan, J. Ohm, W. J. Han, and T. Wiegand. Overview of the high efficiency video coding (hevc) standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(12):1649–1668, 2012.

[57] W. Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets. *Applied and Computational Harmonic Analysis*, 3(2):186–200, 1996.

[58] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM Journal on Mathematical Analysis*, 29(2):511, 1998.

[59] A. D. Szlam, M. Maggioni, and R. R. Coifman. Regularization on graphs with function-adapted diffusion processes. *The Journal of Machine Learning Research*, 9:1711–1739, 2008.

[60] D. S. Taubman and M. W. Marcellin. Jpeg 2000: Image compression fundamentals, standards and practice. 2001.

[61] J. A. Tropp. Just relax: Convex programming methods for subset selection and sparse approximation. *IEEE Trans. Information Theory*, 51(3):1030–1051, 2005.

[62] J.A. Tropp. Greed is good: Algorithmic results for sparse approximation. *Information Theory, IEEE Transactions on*, 50(10):2231–2242, 2004.

[63] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[64] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Processing*, 13(4):600–612, 2004.

[65] D.F. Watson. Contouring: a guide to the analysis and display of spatial data: with programs on diskette. 1992.

[66] T.Y. Yang. *Finite element structural analysis*, volume 2. Prentice-Hall, 1986.

[67] G. Yu, G. Sapiro, and S. Mallat. Image modeling and enhancement via structured sparse model selection. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 1641–1644. IEEE, 2010.

[68] G. Yu, G. Sapiro, and S. Mallat. Solving inverse problems with piecewise linear estimators: from gaussian mixture models to structured sparsity. *Image Processing, IEEE Transactions on*, (99):2481–2499, 2012.

[69] J. Zepeda, C. Guillemot, and E. Kijak. Image compression using the iteration-tuned and aligned dictionary. In *Proc. 36th IEEE Internat. Conf. Acoust. Speech Signal Process., ICASSP-2011*, pages 793–796, 2011.

[70] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. *Curves and Surfaces*, pages 711–730, 2012.

[71] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 479–486. IEEE, 2011.

התמרות wavelet עבור גרפים והשימוש בהם לעיבוד תמונה

עידן רם

התמרות wavelet עבור גרפים והשימוש בהם לעיבוד תמונה

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
דוקטור לפילוסופיה

עידן רם

הוגש לסנט הטכניון - מכון טכנולוגי לישראל
אב תשע״ד, חיפה, אוגוסט 2014

המחקר נעשה בהנחיית פרופ׳ ישראל כהן (הפקולטה להנדסת חשמל) ופרופ׳ מיכאל אלעד (הפקולטה למדעי המחשב) בפקולטה להנדסת חשמל.

תודות

ברצוני להביע את תודתי והערכתי העמוקה לפרופ׳ ישראל כהן ופרופ׳ מיכאל אלעד עבור הדרכתם והנחייתם המסורה. תודה על התמיכה המקצועית, העידוד לשלמות והתרומה להתפתחותי האישית.

תודה לצוות של מעבדת SIPL עבור דאגתם ותמיכתם אשר אפשרו את ביצועם של רבים מהניסויים המתוארים בתיזה זו.

ברצוני להודות לאישתי סיגל, ילדי הילה ואלון והורי רותי וזאב, עבור האהבה, התמיכה והעידוד הבלתי פוסקים.

אני מודה לטכניון ולקרן הלאומית למדע (מענק מס׳ 1130/11) על התמיכה הכספית הנדיבה בהשתלמותי.

תקציר

בתיזה זו אנו מציגים שיטות חדישות לעיבוד אותות המוגדרים על גרפים וענני נקודות מממד גבוה. מציאת שיטות יעילות לייצוג ועיבוד אותות מסוג זה היא בעיה מאתגרת למדי, מכיוון שהם דגומים בצורה לא אחידה. מסיבה זו, שיטות עיבוד אותות מסורתיות בדרך כלל אינן מועילות, מכיוון שהן מתוכננות עבור מידע המוגדר על סריגי דגימה אויקלידים ורגולרים. הפתרון שלנו כולל את שינוי האופן שבו אנו מסתכלים על המידע על ידי "יצירת סדר באי הסדר". אנו מציעים גישה חדשה לעיבוד אותות, שהיא למעשה הרעיון המרכזי המשותף לכל השיטות המוצעות בעבודה זו. אנו עוברים מתחום האותות המוגדרים על סריגים בעלי טופולגיה מורכבת לסביבה הנוחה של עיבוד אותות חד ממדי, על ידי סידור מחדש של אותות אלה לאותות חד ממדיים רגולרים. שינוי זה הופך את עיבוד האותות המתקבלים להרבה יותר פשוט ואינטואיטיבי ומאפשר לנו לפתח אלגוריתמים חדשים אשר מבוססים על שיטות חד ממדיות נפוצות. לכל אורך עבודה זו אנו חוקרים את השימוש באלגוריתמים אלו בעיקר עבור עיבוד תמונות, ומדגימים תוצאות במספר יישומים עם תוצאות מהטובות המוכרות בספרות.

העבודה שלנו מתחילה בהצגת התמרת wavelet המבוססת על עצים מוכללים (Generalized Tree-Based Wavelet Transform , או בקיצור GTBWT), המהווה הרחבה של התמרת ה-wavelet הבדידה החד ממדית לפונקציות המוגדרות על גרפים וענני נקודות מממד גבוה. הסיבה לבחירתינו להרחיב את התמרת ה-wavelet היא שהתמרה זו הוכחה להיות כלי יעיל מאוד לעיבוד אותות. היתרון העיקרי של שיטות המבוססות על התמרת ה-wavelet הוא היכולת שלהן לרכז את מיקום תוכן האות בו זמנית גם במרחב וגם בתדר. תכונה זו מאפשרת להן לייצג בצורה קומפקטית אותות כגון כגון מדרגות חד ממדיות או אותות חלקים למקוטעין, אשר אי הרציפויות שבהם מהווים את תוכן המידע העיקרי שלהם. התמרת ה-GTBWT מחושבת על ידי הוספת אופרטורים תלויי מידע המבצעים ערבול (פרמוטציה) לתוך מימוש מערך המסננים הקלאסי של התמרת ה-wavelet החד ממדית. פעולות ערבול אלו מופעלות על מקדמי הקירוב על מנת להגדיל את הרגולריות שלהם ולגרום לייצוג שלהם באמצעות ההתמרה המוצעת להיות דליל יותר. אופרטורי הסידור מחדש מחושבים מתוך מבנה דמוי עץ אותו אנו מכנים "עץ מוכלל" וממנו נגזר שמה של ההתמרה. עץ זה נבנה באופן רקורסיבי על ידי ארגון בכל רמה של קואורדינטות מממד גבוה או וקטורי מאפיינים (features) המשוייכים לקודקודי הגרף, באופן שהם משורשרים למסלול הקצר ביותר. סידור זה למעשה מהווה קירוב לפתרון בעיית הסוכן הנוסע.

אנו ממשיכים ומציגים התמרת wavelet יתירה המבוססת על עצים (Redundant Tree-Based Wavelet Transform , או בקיצור RTBWT) אשר מרחיבה באופן דומה את התמרת ה-wavelet היתירה לפונקציות המוגדרות על גרפים וענני נקודות. התמרה זו מתקבלת תחילה על ידי שינוי מימוש של התמרת wavelet היתירה, המשתמש בפירוק מערך מסננים הדומה לזה של ההתמרה האורתוגונלית, אך בכל דרגה שלו, אף אחד מהמקדמים לא מושלך. גם כאן אנו מוסיפים בכל דרגת פירוק מספר אופרטורים שמסדרים מחדש את מקדמי הקירוב וכך מגדילים את הרגולריות שלהם וגורמים לייצוג שלהם עם התמרת ה-wavelet המוצעת להיות יותר דליל. בהמשך אנו מציעים מימוש פשוט יותר של ה-RTBWT המבוסס על אלגוריתם "à trous`` המוכר. אלגוריתם זה מממש את התמרת ה-wavelet היתירה בצורה שונה, המאפשרת לסכימה שלנו להוסיף לכל דרגת פירוק של ההתמרה אופרטור ערבול יחיד אשר מסדר מחדש את מקדמי הקירוב לפני שמסננים אותם. אנו עושים שימוש בסכימה זו על מנת להראות שה-RTBWT היא מסגרת (frame), בעלת אותם חסמים כמו של התמרת ה- wavelet היתירה הרגילה. כמו כן אנו חוקרים את השימוש של ההתמרה המוצעת לשחזור תוויות (labels) המוגדרות על ענן נקודות ומראים שהיא משיגה תוצאות טובות יותר ממסווג השכן הקרוב .

למרות ששתי התמרות ה-wavelet המוצעות בתיזה זו תוכננו במקור לייצג פונקציות סקלריות המוגדרות על ענני נקודות מממד גבוה וגרפים, אנו מתמקדים במחקר שלנו בהפעלתן לצרכי עיבוד תמונה. ההתמרות מופעלות על תמונות על ידי המרת התמונות למבנה גרף, כאשר הפיקסלים בתמונה משמשים כפונקציה דגומה המוגדרת על קודקודי הגרף, והבלוקים המקיפים אותם משמשים כמאפיינים למשוייכים לדגימות אלו. התמרות ה-multi-scale המתקבלות אדפטיביות לתוכן התמונה ויעילות מאוד לדילול שלה. אנו מדגימים עובדה זו עבור ההתמרת ה-GTBWT על ידי כך שאנו מראים שהיא יעילה יותר לייצוג תמונות מאשר התמרות ה-wavelet החד ממדית והדו ממדית הספרבילית. אנו גם מראים שהאדפטיביות של התמרה זו משתקפת בפונקציות הבסיס שלה, אשר תוכנן נע בין גרסאות חלקות של התמונה לשפות שונות

בתמונה אשר לא בהכרך מתיישרות עם הצירים האופקי והאנכי. בהמשך אנו חוקרים את הפעלת ההתמרות המופקות מתמונה לצרכי הסרת רעש מתמונות. אנו מציעים סכימת מיצוע של תתי תמונות הכוללת הפעלה של התמרה הנבחרת על תתי תמונות שונות של התמונה המטופלת, פעולה משותפת על מקדמי ההתמרה של כל תתי התמונות, הפעלת ההתמרה הפוכה ושחזור התמונה מתוך תתי התמונות הנקיות המתקבלות. אנו מראים שיחד עם סכימת מיצוע תתי תמונות, ההתמרות המוצעות משיגות תוצאות אשר קרובות ל-state-of-the-art .

בהמשך עבודתנו, אנו חוקרים בצורה יסודית יותר את ייצוג התמונה היתיר המתקבל עם התמרת ה-RTBWT ביחד עם סכימת מיצוע תתי התמונות שתוארה לעיל. אנו בונים מטריצות הפועלות בתור אופרטורי אנליזה וסינתיזה המשמשים לחישוב הייצוג ושחזור התמונה ממנו. אנו מראים שמטריצות אלה מהוות מסגרת והמסגרת הדואלית שלה. אנו חוקרים את השימוש בייצוג יתיר זה כ-regularizer רב עוצמה המקדם דלילות בבעיות היפוך כלליות בעיבוד תמונה ומדגימים תוצאות state-of-the-art של הסרת רעש וטשטוש מתמונות. בנוסף, אנו עושים שימוש בהתמרת ה-RTBWT גם לדחיסת תמונות פנים. מעבר להיות בעיה זו מעניינת היישום שלה הוא מועיל בשל השימוש הנרחב בתמונות פנים בשנים האחרונות והצורך לאחסן אותן במאגרי מידע גדולים, שרתים וכו'. בהינתן סט תמונות אימון המכיל תמונות פנים פרונטליות, התכנון של התמרת ה-RTBWT מאפשר לה לדלל באופן מיטבי את כל הסט באמצעות סידור משותף של המאפיינים. סכימת הדחיסה שלנו כוללת קידוד דליל (sparse coding) באמצעות ההתמרה, ואחריו קידוד אנטרופיה של המקדמים המתקבלים. ההתמרה ההופכית ושלב עיבוד שאחרי (post-processing) משמשים לפריסת התמונה הדחוסה.

לבסוף, אנו מציגים סכימת עיבוד תמונה המבוססת רק על ערבול הפיקסלים על ידי סידור מחדש של כל הבלוקים בתמונה וזאת ללא טיפול רב-סקאלה והעיבוד מוכוון הדלילות הנובע מהשימוש בהתמרת wavelet. אנו מחלצים את כל הבלוקים מהתמונה ומסדרים אותם באמצעות אותו אלגוריתם סידור בו עושות שימוש ההתמרות המוצעות, אך הטיפול שלנו מנקודה זו שונה באופן ניכר. אנו עושים שימוש בעובדה שגם כאשר התמונה מעוותת (רועשת, מכילה פיקסלים חסרים וכדומה), הסידור המוצע צפוי להיות חסין לעיוותים ובשל כך מביא לסידור מחדש של הפיקסלים המושחתים ל"מה שאמור להיות" אות חד ממדי רגולרי. לכן, הפעלת פעולות חד ממדיות פשוטות יחסית (כגון סינון או אינטרפולציה) על סט הפיקסלים המסודרים מחדש אמור לאפשר שחזור טוב של התמונה הנקייה. זהו עיקרה של השיטה המוצעת – עבור תמונה פגומה אנו מסדרים מחדש את הפיקסלים בה, פועלים על האות החדש החד ממדי שהתקבל באמצעות אלגורתמים פשוטים, ומחזירים את הפיקסלים שנתקבלו למקומם המקורי בתמונה. אנו חוקרים שתי סכימות המבוססות על שתי גישות שונות של פעולה על אותות חד ממדיים: 1) שימוש בשיטות פשוטות כגון סינון ואינטרפולציה חד ממדיים, 2) הפעלת גירסא חד ממדית של אלגוריתם ה--NL Means. אנו מראים כי כאשר שתי השיטות המוצעות מופעלות עם מספר סידורים הנבנים בצורה אקראית ובצירוף סכימת מיצוע תתי תמונות, הן מסוגלות להוביל לתוצאות מהטובות מאילו המוכרות בספרות למשימת הסרת רעש מתמונות והשלמת פיקסלים חסרים בתמונות.