Imperial College London

Department of Electrical and Electronic Engineering

Quadtree Structured Approximation Algorithms

Adam Scholefield

2013

Supervised by Dr Pier Luigi Dragotti

Submitted in part fulfilment of the requirements for the degree of

Doctor of Philosophy in Electrical and Electronic Engineering of Imperial College London

and the Diploma of Imperial College London

Declaration of Originality

I declare that the contents of this thesis are either my own work - under the guidance of my PhD supervisor Dr Pier Luigi Dragotti - or have been appropriately referenced.

Adam Scholefield

Abstract

The success of many image restoration algorithms is often due to their ability to sparsely describe the original signal. Many sparse promoting transforms exist, including wavelets, the so called 'lets' family of transforms and more recent non-local learned transforms. The first part of this thesis reviews sparse approximation theory, particularly in relation to 2-D piecewise polynomial signals. We also show the connection between this theory and current state of the art algorithms that cover the following image restoration and enhancement applications: denoising, deconvolution, interpolation and multi-view super resolution.

In [63], Shukla et al. proposed a compression algorithm, based on a sparse quadtree decomposition model, which could optimally represent piecewise polynomial images. In the second part of this thesis we adapt this model to image restoration by changing the rate-distortion penalty to a description-length penalty. Moreover, one of the major drawbacks of this type of approximation is the computational complexity required to find a suitable subspace for each node of the quadtree. We address this issue by searching for a suitable subspace much more efficiently using the mathematics of updating matrix factorisations. Novel algorithms are developed to tackle the four problems previously mentioned. Simulation results indicate that we beat state of the art results when the original signal is in the model (e.g. depth images) and are competitive for natural images when the degradation is high.

Acknowledgements

Behind every PhD thesis is a unique journey comprising of many ups and downs; however, I believe, the past few years have been more extreme than most. This journey would not have been possible without my supervisor Dr. Pier Luigi Dragotti. I would like to thank him for his hard work, understanding and guidance that has made this possible.

I have no doubt that he has been an exceptional supervisor to all his students, however I am sure that I have pushed his flexibility more than most. For the vast majority of my PhD I have lived hundreds of miles away from the lab so that I could splash around in a pool for four hours a day, instead of concentrating solely on my studies. This unusual arrangement has resulted in meetings at airports, train stations and conferences; on Skype; and occasionally in the lab. However, despite their unusual locations, I have left every meeting wiser and with an increased motivation and passion for my work, due to the intellect and charisma of my supervisor.

I have no doubt that without his support, flexibility and dedication, I would not have been able to complete both this thesis and compete in the London 2012 Olympics. It was thus a great honour that he was present to support me during this competition and I only hope that this thesis invokes the same excitement. However a word of warning, I very much doubt the contents of this thesis will have the reader out of their seat, unable to resist screaming in support.

Additionally, I would like to thank my friends, including the numerous members of the Morley, Rotherham, Pécs and Great Britain water polo squads, for the their support, humour and complete lack of interest in my PhD, which has made my life that much richer.

Last but definitely not least, I would like to thank my family for their unconditional support and love that have always allowed me to pursue my passions.

Contents

List of Tables

List of Figures

Abbreviations

BM3D:	Block matching with 3-D collaborative filtering [17];
BM3D-SAPCA:	BM3D with shape adaptive PCA [19];
CCTV:	Closed-circuit television;
CWT:	Continuous wavelet transform;
dB:	Decibel;
DCT:	Discrete cosine transform;
DFT:	Discrete Fourier transform;
DWT:	Discrete wavelet transform;
EM algorithm:	Expectation maximisation algorithm;
FISTA:	Fast iterative shrinkage-thresholding algorithm [4];
FRI:	Finite rate of innovation;
ISTA:	Iterative shrinkage-thresholding algorithm;
JPEG2000:	Joint photographic expert group 2000;
K-SVD:	The K-SVD algorithm [29] is an extension of the K-means clustering process [46] that uses the SVD;
MAP:	Maximum a posteriori estimator;
MM algorithm:	Majorise minimise algorithm if minimising a function; minorise maximise algorithm if maximising;
MSE:	Mean squared error;
n-D:	n dimensions;

OMP:	Orthogonal matching pursuit;
PCA:	Principle component analysis;
PLOW:	Patch-based near-optimal image denoising [14];
PSNR:	Peak signal to noise ratio;
SADCT:	Shape adaptive DCT [35];
SIFT:	Scale invariant feature transform;
SSIM:	Structural SIMilarity index;
SVD:	Singular value decomposition;
UDWT:	Undecimated DWT.

Notation

In this thesis scalars are denoted by regular letters, vectors by lowercase bold letters and matrices by uppercase bold. Subscripts and superscripts are used to provide additional naming; e.g., $\boldsymbol{I_N}$ is the $N \times N$ identity. The only exception to this is superscripts on scalars, which are used for powers. We also use the following notation:

$[a,b]$	$\{x \vert a \le x \text{ and } x \le b\}$;
(a,b)	$\{x \vert a < x \text{ and } x < b\}$;
$[a,b)$	$\{x \vert a \le x \text{ and } x < b\}$;
$(a,b]$	$\{x \vert a < x \text{ and } x \le b\}$;
$\boldsymbol{A}^T$	The transpose of the matrix $\boldsymbol{A}$;
$\boldsymbol{A}^*$	The conjugate transpose of the matrix $\boldsymbol{A}$;
$\Vert\boldsymbol{A}\Vert_p$	The l_p matrix norm, which for $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is given by $\max_{\boldsymbol{x} \neq \boldsymbol{0}} \frac{\Vert\boldsymbol{Ax}\Vert_p}{\Vert\boldsymbol{x}\Vert_p}$;
$\mathbb{C}$	The set of complex numbers;
$diag(\boldsymbol{x})$	The $N \times N$ diagonal matrix with $\boldsymbol{x} \in \mathbb{R}^N$ on the diagonal;
$E(A)$	The expected value of A;
$\boldsymbol{\mathcal{F}}$	The DFT matrix;
$\boldsymbol{I_n}$	The $n \times n$ identity matrix;
$\mathbb{N}$	The set of natural numbers;
$\mathcal{N}_N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	An N dimensional normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$;

$O(.)$	$f(x) = O(g(x))$ iff $\exists C, x_0$ s.t. $\lvert f(x)\rvert \le C\lvert g(x)\rvert\ \forall x > x_0$;
$p(A)$	The probability of a particular event A;
$p(A\vert B)$	The probability of a particular event A given B;
$prox_\lambda(\cdot)$	The proximal map of a function f is given by $prox_\lambda(\boldsymbol{x}) = \arg\min_{\boldsymbol{a}} \left\{ \lVert \boldsymbol{x} - \boldsymbol{a} \rVert_2^2 + \lambda f(\boldsymbol{a}) \right\}$;
$\mathbb{R}$	The set of real numbers;
$\boldsymbol{S_{\lambda,p}}(\cdot)$	A thresholding, defined from the proximal map of $\lvert\cdot\rvert^p$, applied to each element of the vector input: $\boldsymbol{S_{\lambda,p}}(\boldsymbol{x})[i] = prox_\lambda(\boldsymbol{x}[i])$ where the *prox* is for $\lvert\boldsymbol{x}[i]\rvert^p$;
$\boldsymbol{x}_{\boldsymbol{\mathcal{F}}}$	$\boldsymbol{x}$ in the DFT domain; i.e., $\boldsymbol{x}_{\boldsymbol{\mathcal{F}}} = \boldsymbol{\mathcal{F}}^* \boldsymbol{x}$;
$\boldsymbol{x}^{(k)}$	$\boldsymbol{x}$ at the k-th iteration;
$\hat{\boldsymbol{x}}$	An estimator of the vector $\boldsymbol{x}$;
$\lVert\boldsymbol{x}\rVert_p$	The l_p vector norm, which for $\boldsymbol{x} \in \mathbb{R}^n$ is given by $\left[\sum_{i \in \mathbb{R}^n} (\boldsymbol{x}[i])^p\right]^{\frac{1}{p}}$;
$\lvert\cdot\rvert$	Absolute value of a scaler or the cardinality of a set;
$\star$	The convolution operator;
$\circledast$	The circular convolution operator;
∇^n	An n-th order partial derivatives: $\nabla^n \boldsymbol{x} = \left[\frac{\partial^n \boldsymbol{x}}{\partial \boldsymbol{x}[1]^n}, \frac{\partial^n \boldsymbol{x}}{\partial \boldsymbol{x}[2]^n}, \cdots\right]^T$.

CHAPTER 1

Introduction

1.1. Background and motivation

In recent years, digital image acquisition devices have become part of our everyday lives: cameras are embedded in smartphones, laptops and tablets; CCTV provides almost 100% coverage of major cities; and more recently depth sensors have started to appear in game consoles and smart TVs. These technologies continue to provide signal processing and computer vision challenges: for example embedded image sensors in portable electronics must be cheap and small, limiting the potential for optical processing. Depth sensors and multi-view images capture depth information, which allows more robust object detection and tracking, and opens up, in theory, a full 3-D free viewpoint experience. The acquisition of these depth images, and their joint processing with an illumination image, is currently receiving much research interest.

In order to solve these problems we need to understand better the underlying structure of the images and any acquisition process. The complexity of images, or in fact any real signal, makes approximation a sensible first step that can provide knowledge of the signals fundamental components and facilitate easier storage and processing. A sparse promoting transform converts a signal into a sum of functions, such that most of the energy is contained in a relatively small number of components. Sparse approximation theory can thus be described as designing transforms that, for signals of interest, compress the energy into as few components as possible and, given a possibly overcomplete transform, select the best components to keep. Since many images are approximately piecewise polynomial, the sparse approximation of this class of signals is an interesting problem that is at the heart of this thesis.

It is well known that wavelets can be constructed to annihilate polynomials and thus sparsely describe piecewise polynomial signals. This has been exploited in numerous image processing applications including the incorporation of wavelets in the JPEG2000 compression standard. The success of the sparse approximation framework has motivated the quest for even sparser approximations of images. Much of this research has focused around wavelets' lack of adaptability to the 2-D discontinuities that are seen in images. In order to better represent these complex edge structures, overcomplete and adaptive techniques have been developed that produce increased sparsity and thus improved performance in practical image processing problems. However there is no free lunch: calculating sparse approximations using these transforms is more complex than in a basis and requires interesting new approaches.

In this thesis we will apply these approximations to image restoration and enhancement applications. More precisely we will investigate denoising, deconvolution, interpolation and super resolution problems. Denoising, deconvolution and interpolation are well studied problems; however, improved performance can be obtained using modern approximation techniques, and they are still relevant in modern applications. For example, many depth sensors only obtain a depth value at certain locations so, in order to have a full depth map for further processing, interpolation is required. Since depth maps have a highly constrained structure, there is great potential for accurate interpolation by properly modelling this class of images. In addition, multi-view super resolution – the problem of combining multiple images into one higher resolution image – can be decomposed into multiple steps, which are mostly traditional restoration problems. As we are embedding image sensors into smaller spaces and reaching the physical resolution limit of digital sensors, super resolution is becoming an attractive alternative that allows greater image resolution.

1.2. Outline of thesis

This thesis has five additional chapters, which are briefly outlined below:

In the first part of Chapter 2 we introduce sparsity and sparse seeking algorithms by studying a system of linear equations. In particular, we focus on a class of shrinkage or thresholding algorithms that have seen widespread use in image processing. The second part of the chapter summarises existing transforms that sparsely represent piecewise polynomial signals. We first highlight the key attributes of the wavelet transform that lead to its suitability for 1-D piecewise polynomial signals. This is followed by an analysis of 2-D sparse promoting transforms. We outline a range of algorithms that aim to improve on wavelets shortcomings, particularly their lack of directional selectivity.

The analysis of Chapter 2 is applied in Chapter 3 to perform image restoration and enhancement. We discuss four problems, denoising, deconvolution, interpolation and super resolution, that will be our main applications throughout this thesis. We will show how each problem can be modelled as a degradation, following the same model, and an estimation of the original signal can be generated using the sparse seeking algorithms of Chapter 2. Existing state of the art algorithms are presented for each problem and, although these algorithms are not our own work, we present our own perspective of how these relate to Chapter 2 and will provide similar principles when, later in the thesis, we adapt our approximation technique to restoration.

Chapter 4 contains the first main novel contribution of the thesis. Here we propose a novel image approximation algorithm that is based on the earlier work of Shukla et al. [63] on image compression. This approach uses a quadtree decomposition to adaptively partition the image and approximates each adaptive region by a piecewise polynomial model with at most one discontinuity. An additional 'joining' step is also applied that allows a more flexible image partitioning than the pure quadtree structure. We make the following novel contributions that make this approach more appropriate for restoration: we replace the rate-distortion penalty with a description-length penalty, which is more appropriate for image restoration, and we propose a new way to quickly calculate a discontinuity, for each region, that exploits the mathematics of updating matrix factorisations. We take the reader through a detailed example that reveals how this vastly improves the computational requirement, often a major shortcoming of this type of technique.

In Chapter 5 we apply similar principles to Chapter 3 to adapt our approximation algorithm to restoration and enhancement. The resulting novel restoration algorithms are extensively compared to the current state of the art, using as performance measures the peak signal-to-noise ratio (PSNR)[1] and structural similarity (SSIM) index[2].

[1] All PSNRs in this thesis are calculated as $10\log_{10}\left(\frac{255^2}{\frac{1}{N}\|\boldsymbol{x_1}-\boldsymbol{x_2}\|_2^2}\right)$, where $\boldsymbol{x_1}, \boldsymbol{x_1} \in \mathbb{R}^N$ are the vectors of the two images been compared, and N is the number of pixels.

[2] The SSIM index [73] attempts to measure the visual quality of an image by measuring its structural similarity to the original image.

Finally, Chapter 6 concludes this thesis, summarising its achievements and highlighting possible directions for future study.

1.3. Original contribution

The main original contributions of this thesis are presented in Chapters 4 and 5. In Chapter 4 we propose a novel approximation algorithm, based on the earlier work of Shukla et al. [63], with the following novel contributions:

- The incorporation of a description length penalty that is more suitable to the restoration and enhancement problem we consider.
- A new approach to calculate a suitable discontinuity over each adaptive region that vastly reduces the computational complexity. Traditionally an exhaustive search is carried that tests a large number of possible discontinuities independently. We maintain the exhaustive search, and thus robust selection, but test new discontinuities very effectively by using previous results.

In Chapter 3 we discuss current state of the art restoration and enhancement algorithms. We introduce these algorithms using our own perspective that then naturally leads to our novel algorithms presented in Chapter 5. Here we present the following novel contributions:

- A novel denoising algorithm that produces state of the art results for depth images and highly degraded natural images.
- An iterative denoising approach to deconvolution that is an extension of iterative shrinkage algorithms to our non-linear model.
- A deconvolution algorithm that combines our novel denoising algorithm with traditional regularised inverses.
- A novel interpolation algorithm that, for irregularly sampled data, produces state of the art results for depth images and highly degraded natural images.

The work presented in this thesis has led to the following publications:

- A. Scholefield and P. L. Dragotti, "Quadtree structured image approximation for denoising and interpolation," *submitted to IEEE Transactions on Image Processing*, May 2013

- A. Scholefield and P. L. Dragotti, “Image restoration using a sparse quadtree decomposition representation,” in *Image Processing, IEEE International Conference on*, 2009, pp. 1473–1476
- A. Scholefield and P. L. Dragotti, “Quadtree structured restoration algorithms for piecewise polynomial images,” in *Acoustics, Speech and Signal Processing, IEEE International Conference on*, IEEE, 2009, pp. 705–708
- A. Scholefield and P. L. Dragotti, *Quadtree structured restoration algorithms for piecewise polynomial images*, at Inspire Workshop on Sparsity and its application to large inverse problems, Cambridge, Dec. 2008

CHAPTER 2

Sparse Representation of 1-D and 2-D Functions

2.1. Introduction

Sparsity – the ability to represent a signal using as few components as possible – has been a key tool in many recent image processing algorithms. In this thesis we are interested in sparse approximations of piecewise polynomial signals, particularly in 2-D. This chapter introduces the concept of sparsity by studying systems of linear equations. Sparsity is introduced as a regularisation technique which produces a minimisation problem of the form

$$arg \min_{\boldsymbol{\theta}} \left\{ \|\boldsymbol{y} - \boldsymbol{B\theta}\|_2^2 + \lambda \|\boldsymbol{\theta}\|_p^p \right\},$$

where $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{B} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{\theta} \in \mathbb{R}^n$ produce the system of m equations $\boldsymbol{y} = \boldsymbol{B\theta}$, with n unknowns. This problem has been extensively studied in the literature over a wide range of disciplines. As is common to problems of an interdisciplinary nature, there are a multitude of algorithms that have been developed from different standpoints, some of which are equivalent. We will concentrate on a class of algorithms that have had a recent flurry of research interest. These are iterative shrinkage algorithms and can be thought of as an extension of gradient descent methods.

Later in the chapter we will see how solving minimisation problems of this form can find sparse approximations of piecewise polynomial signals. The transform, $\boldsymbol{B}$, will be of key importance, particularly its ability to efficiently represent the discontinuities. In 1-D these discontinuities can only be points but in 2-D they are more complex edge structures. Many different transforms have been proposed to deal with these higher dimensional edges and we will summarise the most important. Some of these transforms extend the setting to a non-linear transform as we also do in our novel quadtree decomposition algorithm presented in Chapter 4.

2.2. Sparse approximations via regularisation

2.2.1. Closed-form solutions

To introduce the concept of sparsity and algorithms that promote sparse solutions, consider a system of linear equations

$$\boldsymbol{y} = \boldsymbol{B}\boldsymbol{\theta},$$

where $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{B} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{\theta} \in \mathbb{R}^n$. Clearly if $\boldsymbol{B}$ is invertible then $\boldsymbol{\theta} = \boldsymbol{B}^{-1}\boldsymbol{y}$ is the unique solution but in many practical situations $\boldsymbol{B}$ is singular or very often not square and, thus, not invertible. In these cases we wish to find a single 'sensible' solution. When $n > m$ there are infinitely many solutions but regularisation can be applied to select the most 'appropriate'. Depending on the application the minimum energy solution maybe the most appropriate and is given by

$$\hat{\boldsymbol{\theta}} = arg \min_{\boldsymbol{\theta}} \|\boldsymbol{\theta}\|_2^2 \quad s.t. \quad \boldsymbol{y} = \boldsymbol{B}\boldsymbol{\theta} \qquad \text{where } \|\boldsymbol{\theta}\|_2 = \left(\sum_{i=1}^{n} |\boldsymbol{\theta}[i]|^2 \right)^{\frac{1}{2}}. \tag{2.1}$$

This can be solved using Lagrange multipliers: we first construct the Lagrange function,

$$L(\boldsymbol{\theta}, \boldsymbol{\lambda})^T = \|\boldsymbol{\theta}\|_2^2 + \boldsymbol{\lambda}^T(\boldsymbol{y} - \boldsymbol{B}\boldsymbol{\theta}),$$

where $\boldsymbol{\lambda} \in \mathbb{R}^m$ is the vector of Lagrange multipliers for each equality constraint. The solution of (2.1) occurs at a stationary point of $L(\boldsymbol{\theta}, \boldsymbol{\lambda})$; i.e., at a point where both $\nabla_{\boldsymbol{\theta}} L = 0$ and $\nabla_{\boldsymbol{\lambda}} L = 0$. The partial derivatives are

$$\nabla_{\boldsymbol{\theta}} L = 2\boldsymbol{\theta} - \boldsymbol{B}^T\boldsymbol{\lambda} \quad \text{and} \quad \nabla_{\boldsymbol{\lambda}} L = \boldsymbol{y} - \boldsymbol{B}\boldsymbol{\theta},$$

therefore

$$0 = 2\hat{\boldsymbol{\theta}} - \boldsymbol{B}^T\hat{\boldsymbol{\lambda}} \quad \Rightarrow \quad \hat{\boldsymbol{\theta}} = \frac{\boldsymbol{B}^T\hat{\boldsymbol{\lambda}}}{2}, \text{ and}$$

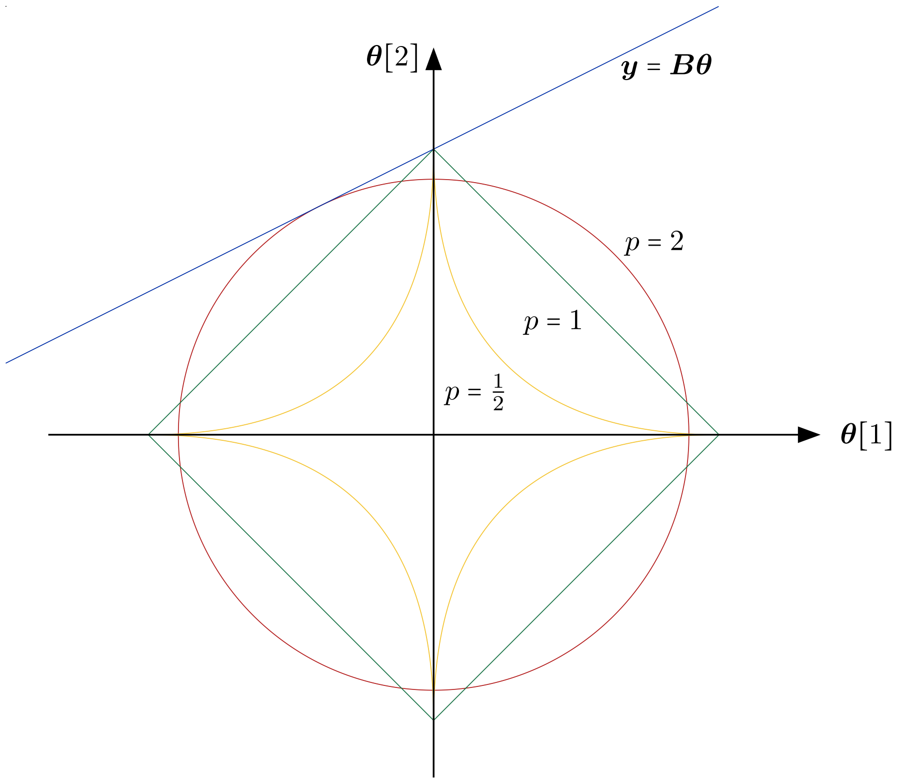


Figure 2.1.: Graphical illustration of the points in the affine subspace, $\boldsymbol{y} = \boldsymbol{B}\boldsymbol{\theta}$, with minimum regularisation norm. The norms are l_p-norms, $\|\boldsymbol{\theta}\|_p$, for $p = \frac{1}{2}, 1, 2$.

$$0 = \boldsymbol{y} - \boldsymbol{B}\hat{\boldsymbol{\theta}} \quad \Rightarrow \quad \boldsymbol{B}\hat{\boldsymbol{\theta}} = \boldsymbol{y}.$$

Combining the two we have

$$\boldsymbol{y} = \frac{\boldsymbol{B}\boldsymbol{B}^T\hat{\boldsymbol{\lambda}}}{2} \quad \Rightarrow \quad \hat{\boldsymbol{\lambda}} = 2\left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1}\boldsymbol{y},$$

and finally (2.1) is solved by

$$\hat{\boldsymbol{\theta}} = \boldsymbol{B}^T\left(\boldsymbol{B}\boldsymbol{B}^T\right)^{-1}\boldsymbol{y}.$$

As we have eluded to, we are often interested in the sparsest solution not the one with minimum energy. We can get closer to this goal by generalising (2.1) to any l_p-norm:

$$\hat{\boldsymbol{\theta}}_{\boldsymbol{p}} = arg\min_{\boldsymbol{\theta}} \|\boldsymbol{\theta}\|_p^p \quad s.t. \quad \boldsymbol{y} = \boldsymbol{B}\boldsymbol{\theta} \qquad \text{where } \|\boldsymbol{\theta}\|_p = \left(\sum_{i=1}^{n} |\boldsymbol{\theta}[i]|^p\right)^{\frac{1}{p}}. \tag{2.2}$$

This family of l_p-norms will be used extensively in the forthcoming discussion with values of p in the range $0 \leq p \leq 2$. Note that, it is the convention to call these norms although formally they do not

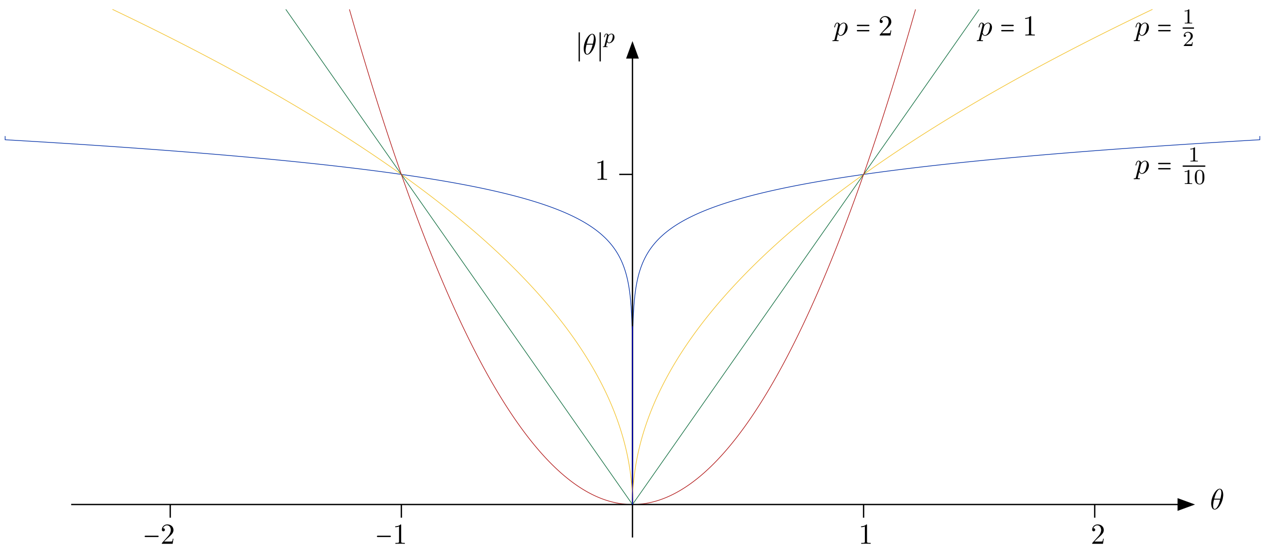


Figure 2.2.: The l_p-norms in 1-D for various values of p.

meet the requirements of a norm for all $0 \leq p \leq 2$[1]. To gain an intuition for the solution of (2.2) for different values of p consider the graphical illustration given in Fig. 2.1. The figure shows various l_p-balls (level sets of an l_p-norm) touching the affine subspace of solutions to $\boldsymbol{y} = \boldsymbol{B\theta}$. For each value of p, the smallest l_p-ball that touches the affine subspace has been plotted and this point of contact is the solution to (2.2). For the $l_{\frac{1}{2}}$ and l_1 balls, these intersections occur on an axis suggesting that these norms promote sparse solutions.

To further examine the properties of these norms, Fig. 2.2 shows a plot of $|\theta|^p$ in 1-D for various values of p. It is clear that as $p \to 0$, $|\theta|^p$ approaches the indicator function

$$1_{\{\mathbb{R}\setminus\{0\}\}}(\theta) = \begin{cases} 1 & \text{if} \quad \theta \neq 0 \\ 0 & \text{if} \quad \theta = 0 \end{cases}.$$

This natural extension leads to the l_0-norm (again this is not strictly a norm), which is defined to be the number of non-zero entries of $\boldsymbol{\theta}$:

$$\|\boldsymbol{\theta}\|_0 = |\{i \mid \boldsymbol{\theta}[i] \neq 0\}|,$$

[1]A norm, $\|\boldsymbol{.}\|$, is a function from a vector space V to $[0,\infty)$ that satisfies the following three properties for all $\boldsymbol{\theta} \in V$:

1. $\|\boldsymbol{\theta}\| = 0 \quad \text{iff} \quad \boldsymbol{\theta} = 0$
2. $\|\boldsymbol{\theta_1} + \boldsymbol{\theta_2}\| \leq \|\boldsymbol{\theta_1}\| + \|\boldsymbol{\theta_2}\|$ (triangle inequality)
3. $\|k\boldsymbol{\theta}\| = |k|\,\boldsymbol{.}\,\|\boldsymbol{\theta}\| \quad \forall k \in \mathbb{C}$ (scalability)

where $|.|$ is the cardinality of the set.

We have now introduced the three norms that will be used throughout this thesis, namely the l_0, l_1 and l_2-norm. The l_0-norm provides the best measure of sparsity but it is non convex leading to difficult minimisation problems. The l_1-norm promotes sparsity and is convex so can sometimes be more appealing than l_0. The l_2-norm does not promote sparsity but, as we will see, its minimisation problems have a closed-form solution. We will see these characteristics in more detail as we progress.

If the system of equations, $\boldsymbol{y} = \boldsymbol{B\theta}$, has no solutions we must introduce a distance measure such as

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= arg \min_{\boldsymbol{\theta}} \|\boldsymbol{y} - \boldsymbol{B\theta}\|_2^2 \\ &= \left(\boldsymbol{B}^T\boldsymbol{B}\right)^{-1}\boldsymbol{B}^T\boldsymbol{y}, \end{aligned} \tag{2.3}$$

which is known as ordinary least squares. This assumes that all the errors are in the measured variable $\boldsymbol{y}$ but if the errors occur in both $\boldsymbol{B}$ and $\boldsymbol{y}$ then total least squares is more appropriate. If $\boldsymbol{B}^T\boldsymbol{B}$ is singular then (2.3) has an infinite number of solutions and we once again require regularisation:

$$\hat{\boldsymbol{\theta}} = arg \min_{\boldsymbol{\theta}} \|\boldsymbol{y} - \boldsymbol{B\theta}\|_2^2 \quad s.t. \quad \|\boldsymbol{\theta}\|_p^p < t \tag{2.4}$$

or in the Lagrangian form

$$\hat{\boldsymbol{\theta}} = arg \min_{\boldsymbol{\theta}} \left\{\|\boldsymbol{y} - \boldsymbol{B\theta}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_p^p\right\}. \tag{2.5}$$

The minimisation problem (2.4), which finds the minimum squared error within the sub level set $\|\boldsymbol{\theta}\|_p^p < t$, has received much less research interest than (2.5). This minimisation problem has, in general, no closed-form solution, however there are some special cases:

i) when $p = 2$, there is a closed-form solution known as Tikhovnov regularisation.

ii) when $p = 1$, there is a closed-form solution if $\boldsymbol{B}$ is unitary ($\boldsymbol{B}^T\boldsymbol{B} = \boldsymbol{I_n}$), and an iterative algorithm that converges to the global minimum for general $\boldsymbol{B}$.

iii) when $p = 0$, there is a closed-form solution if $\boldsymbol{B}$ is unitary, but because the problem is not convex we can only design an iteration that converges to a local minimum.

First we will deal with the three closed-form solutions: the $p = 2$ case leads to Tikhovnov regularisation:

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= arg\min_{\boldsymbol{\theta}} \left\{ \|\boldsymbol{y} - \boldsymbol{B\theta}\|_2^2 + \lambda \|\boldsymbol{\theta}\|_2^2 \right\} \\ &= arg\min_{\boldsymbol{\theta}} \left\{ \boldsymbol{\theta}^T \boldsymbol{B}^T \boldsymbol{B\theta} - 2\boldsymbol{\theta}^T \boldsymbol{B}^T \boldsymbol{y} + \lambda \boldsymbol{\theta}^T \boldsymbol{\theta} \right\},\end{aligned}$$

where we have neglected terms that do not depend on $\boldsymbol{\theta}$ because they do not affect the minimisation. The solution is found by setting the derivative to zero:

$$\begin{aligned}0 &= 2\boldsymbol{B}^T \boldsymbol{B}\hat{\boldsymbol{\theta}} - 2\boldsymbol{B}^T \boldsymbol{y} + 2\lambda \hat{\boldsymbol{\theta}} \\ \hat{\boldsymbol{\theta}} &= \left(\boldsymbol{B}^T \boldsymbol{B} + \lambda \boldsymbol{I_n} \right)^{-1} \boldsymbol{B}^T \boldsymbol{y}. \qquad (2.6)\end{aligned}$$

It is clear that as $\lambda \to 0$, (2.6) approaches (2.3); i.e., the case with no regularisation.

When $\boldsymbol{B}$ is unitary the problem separates:

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= arg\min_{\boldsymbol{\theta}} \left\{ \|\boldsymbol{y} - \boldsymbol{B\theta}\|_2^2 + \lambda \|\boldsymbol{\theta}\|_p^p \right\} \\ &= arg\min_{\boldsymbol{\theta}} \left\{ \boldsymbol{\theta}^T \boldsymbol{B}^T \boldsymbol{B\theta} - 2\boldsymbol{\theta}^T \boldsymbol{B}^T \boldsymbol{y} + \lambda \|\boldsymbol{\theta}\|_p^p \right\} \\ &= arg\min_{\boldsymbol{\theta}} \left\{ \|\boldsymbol{B}^T \boldsymbol{y} - \boldsymbol{\theta}\|_2^2 + \lambda \|\boldsymbol{\theta}\|_p^p \right\} \\ &= arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{m} \left[(\boldsymbol{y_B}[i] - \boldsymbol{\theta}[[i])^2 + \begin{cases} \lambda |\boldsymbol{\theta}[i]|^p & \text{if} \quad \boldsymbol{\theta}[i] \neq 0 \\ 0 & \text{if} \quad \boldsymbol{\theta}[i] = 0 \end{cases} \right] \quad \text{where } \boldsymbol{y_B} = \boldsymbol{B}^T \boldsymbol{y},\end{aligned}$$

so that we can solve for each element independently,

$$\hat{\boldsymbol{\theta}}[i] = arg\min_{\boldsymbol{\theta}[i]} \left[(\boldsymbol{y_B}[i] - \boldsymbol{\theta}[[i])^2 + \begin{cases} \lambda |\boldsymbol{\theta}[i]|^p & \text{if} \quad \boldsymbol{\theta}[i] \neq 0 \\ 0 & \text{if} \quad \boldsymbol{\theta}[i] = 0 \end{cases} \right]. \qquad (2.7)$$

When $p = 0$, (2.7) becomes

$$\hat{\boldsymbol{\theta}}[i] = arg\min_{\boldsymbol{\theta}[i]} \left[(\boldsymbol{y_B}[i] - \boldsymbol{\theta}[i])^2 + \begin{cases} \lambda & \text{if} \quad \boldsymbol{\theta}[i] \neq 0 \\ 0 & \text{if} \quad \boldsymbol{\theta}[i] = 0 \end{cases} \right],$$

which, by inspection, is solved by hard thresholding:

$$\hat{\boldsymbol{\theta}}[i] = S_{\lambda,0}\left(\boldsymbol{y_B}[i]\right) = \begin{cases} \boldsymbol{y_B}[i] & \text{if} \quad (\boldsymbol{y_B}[i])^2 \geq \lambda \\ 0 & \text{if} \quad (\boldsymbol{y_B}[i])^2 < \lambda \end{cases},$$

or extending the notation

$$\hat{\boldsymbol{\theta}} = \boldsymbol{S_{\lambda,0}}\left(\boldsymbol{y_B}\right). \tag{2.8}$$

Similarly, when $p = 1$, (2.7) becomes

$$\hat{\boldsymbol{\theta}}[i] = arg\min_{\boldsymbol{\theta}[i]} \left\{ (\boldsymbol{y_B}[i] - \boldsymbol{\theta}[i])^2 + \lambda \left|\boldsymbol{\theta}[i]\right| \right\},$$

which, as proved in Appendix A.1, is solved by the soft thresholding operator:

$$\hat{\boldsymbol{\theta}}[i] = S_{\lambda,1}\left(\boldsymbol{y_B}[i]\right) = \begin{cases} \boldsymbol{y_B}[i] - \frac{\lambda}{2} & \text{if} \quad \boldsymbol{y_B}[i] \geq \frac{\lambda}{2} \\ 0 & \text{if} \quad -\frac{\lambda}{2} < \boldsymbol{y_B}[i] < \frac{\lambda}{2} \\ \boldsymbol{y_B}[i] + \frac{\lambda}{2} & \text{if} \quad \boldsymbol{y_B}[i] \leq -\frac{\lambda}{2} \end{cases},$$

or extending the notation

$$\hat{\boldsymbol{\theta}} = \boldsymbol{S_{\lambda,1}}\left(\boldsymbol{y_B}\right). \tag{2.9}$$

These two thresholding operators will play a key role in the next subsection on iterative algorithms and, as we will see, they are the proximal operator of their respective norms. The reader may also note that, the above thresholding operators are often used in practise when $\boldsymbol{B}$ is not unitary and satisfactory results obtained. As we will see in the next subsection, this is because this thresholding is the first step of an iterative approach to solve (2.5) with more general $\boldsymbol{B}$.

2.2.2. Iterative algorithms

We are interested in a class of algorithms, known as iterative shrinkage algorithms, that can be used to tackle (2.5) for $p = 0, 1$ and general $\boldsymbol{B}$. There are many different ways to arrive at the same algorithm but we will highlight two approaches: first we will formulate the problem using majorise minimise (MM) ideas, which is an extension of expectation maximisation (EM) (for a good introduction to the MM algorithm see [41]). This approach is essentially equivalent to optimisation transfer using surrogate functions which is the approach taken in the seminal work of Daubechies

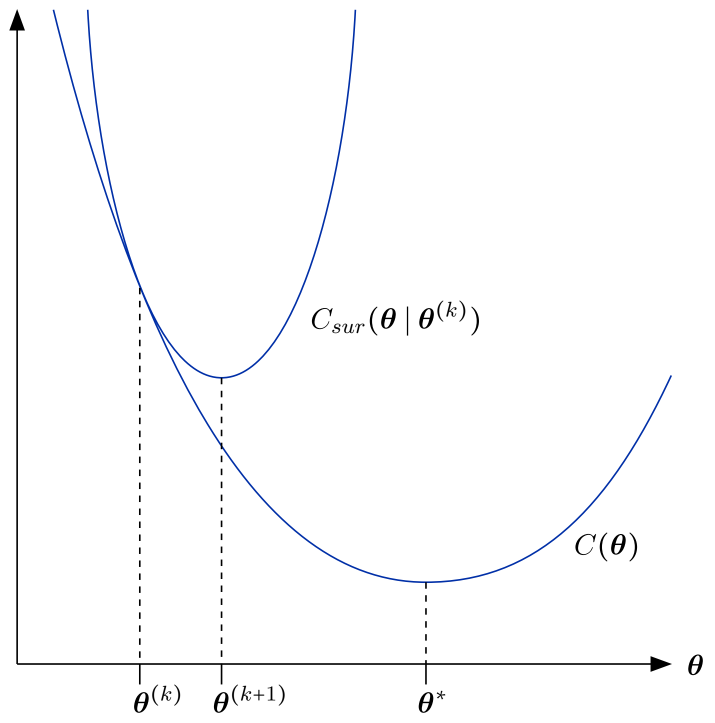

$C_{sur}(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(k)})$
$C(\boldsymbol{\theta})$
$\boldsymbol{\theta}$
$\boldsymbol{\theta}^{(k)}$
$\boldsymbol{\theta}^{(k+1)}$
$\boldsymbol{\theta}^{*}$

Figure 2.3.: The MM approach to minimise a function, $C(\boldsymbol{\theta})$, is to construct a maximiser, $C_{sur}(\boldsymbol{\theta})$, at the current estimate. This maximiser is constructed in such a way that it can be minimised and the location of this minimum gives the next estimate.

et al. [22]. In this paper the authors show that iterative soft thresholding converges to the global minimum of (2.5) for $p = 1$ and general $\boldsymbol{B}$. We will also show this but using the well established optimisation framework of proximal gradient descent, which is equivalent to Combettes and Wajs' Forward-Backward Splitting [16]. As we will see, proximal gradient descent is in fact a special case of an MM algorithm with more stringent conditions that allow stronger convergence proofs. This framework also allows us to analyse acceleration techniques such as that used in the fast iterative shrinkage-thresholding algorithm (FISTA) [4]. For the $p = 0$ case, iterative hard thresholding converges to a local minimum of the non convex cost function, as proved in [7]. More information on this topic can be found in the literature [6, 8, 16, 28, 31–34, 43, 55] and is nicely summarised by Elad et al. [30].

As illustrated in Fig. 2.3, the basic idea of the MM approach is to transfer the minimisation of a complicated function into the iterative minimisation of simpler functions that maximise the original function. More precisely, if we have a function $C(\boldsymbol{\theta})$ that we wish to minimise, then we construct a maximiser $C_{sur}(\boldsymbol{\theta} \mid \boldsymbol{a})$ that by definition satisfies the following two properties:

$$C_{sur}(\boldsymbol{\theta} \mid \boldsymbol{a}) \geq C(\boldsymbol{\theta}) \qquad \forall \boldsymbol{\theta} \tag{2.10}$$

$$C_{sur}(\boldsymbol{a} \mid \boldsymbol{a}) = C(\boldsymbol{a}). \tag{2.11}$$

Then, the MM algorithm is defined as the sequence

$$\boldsymbol{\theta}^{(\boldsymbol{k+1})} = arg \min_{\boldsymbol{\theta}} C_{sur}(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(\boldsymbol{k})}), \tag{2.12}$$

and is guaranteed to be non increasing since

$$C(\boldsymbol{\theta}^{(\boldsymbol{k+1})}) \leq C_{sur}(\boldsymbol{\theta}^{(\boldsymbol{k+1})} \mid \boldsymbol{\theta}^{(\boldsymbol{k})}) \leq C_{sur}(\boldsymbol{\theta}^{(\boldsymbol{k})} \mid \boldsymbol{\theta}^{(\boldsymbol{k})}) = C(\boldsymbol{\theta}^{(\boldsymbol{k})}).$$

Here the first inequality comes from (2.10), the second from (2.12) and the equality from (2.11).

For example, to solve (2.5) we would set

$$C(\boldsymbol{\theta}) = \|\boldsymbol{y} - \boldsymbol{B\theta}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_p^p, \tag{2.13}$$

and could use the maximiser

$$C_{sur}(\boldsymbol{\theta} \mid \boldsymbol{a}) = C(\boldsymbol{\theta}) - \|\boldsymbol{B\theta} - \boldsymbol{Ba}\|_2^2 + \alpha\|\boldsymbol{\theta} - \boldsymbol{a}\|_2^2. \tag{2.14}$$

This is a maximiser of (2.13), since (2.11) is trivially satisfied and (2.10) is satisfied if $\alpha \geq \|\boldsymbol{B}\|_2^2$. This can be shown by substituting (2.14) into (2.10):

$$C(\boldsymbol{\theta}) - \|\boldsymbol{B\theta} - \boldsymbol{Ba}\|_2^2 + \alpha\|\boldsymbol{\theta} - \boldsymbol{a}\|_2^2 \geq C(\boldsymbol{\theta}) \qquad \forall\boldsymbol{\theta},$$

which yields

$$\alpha\|\boldsymbol{\theta} - \boldsymbol{a}\|_2^2 \geq \|\boldsymbol{B\theta} - \boldsymbol{Ba}\|_2^2 \qquad \forall\boldsymbol{\theta}$$

and

$$\alpha \geq \|\boldsymbol{B}\|_2^2.$$

Here the matrix norm is given by

$$\|\boldsymbol{B}\| = \max_{\boldsymbol{\theta}\neq 0} \frac{\|\boldsymbol{B\theta}\|}{\|\boldsymbol{\theta}\|}.$$

All that remains is to show that we can minimise this surrogate cost function. In fact,

$$\begin{aligned}
arg\min_{\boldsymbol{\theta}} C_{sur}(\boldsymbol{\theta} \mid \boldsymbol{a}) &= arg\min_{\boldsymbol{\theta}} \left\{ \|\boldsymbol{y} - \boldsymbol{B\theta}\|_2^2 - \|\boldsymbol{B\theta} - \boldsymbol{Ba}\|_2^2 + \alpha\|\boldsymbol{\theta} - \boldsymbol{a}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_p^p \right\} \\
&= arg\min_{\boldsymbol{\theta}} \left\{ -2\boldsymbol{\theta}^T\boldsymbol{B}^T\boldsymbol{y} + 2\boldsymbol{\theta}^T\boldsymbol{B}^T\boldsymbol{Ba} + \alpha\boldsymbol{\theta}^T\boldsymbol{\theta} - 2\alpha\boldsymbol{\theta}^T\boldsymbol{a} + \lambda\|\boldsymbol{\theta}\|_p^p \right\} \\
&= arg\min_{\boldsymbol{\theta}} \left\{ \boldsymbol{\theta}^T\boldsymbol{\theta} - 2\boldsymbol{\theta}^T\left(\frac{\boldsymbol{B}^T\boldsymbol{y}}{\alpha} - \frac{\boldsymbol{B}^T\boldsymbol{Ba}}{\alpha} + \boldsymbol{a}\right) + \frac{\lambda}{\alpha}\|\boldsymbol{\theta}\|_p^p \right\} \\
&= arg\min_{\boldsymbol{\theta}} \left\{ \left\| \boldsymbol{\theta} - \left(\boldsymbol{a} + \frac{\boldsymbol{B}^T(\boldsymbol{y} - \boldsymbol{Ba})}{\alpha}\right) \right\|_2^2 + \frac{\lambda}{\alpha}\|\boldsymbol{\theta}\|_p^p \right\} \\
&= \boldsymbol{S_{\lambda/\alpha,p}}\left(\boldsymbol{a} + \frac{\boldsymbol{B}^T(\boldsymbol{y} - \boldsymbol{Ba})}{\alpha}\right) \quad \text{for } p = 0, 1.
\end{aligned}$$

Therefore, the MM iteration (2.12), applied to (2.13), is an iterative thresholding:

$$\boldsymbol{\theta^{(k+1)}} = \boldsymbol{S_{\lambda/\alpha,p}}\left(\boldsymbol{\theta^{(k)}} + \frac{\boldsymbol{B}^T}{\alpha}\left(\boldsymbol{y} - \boldsymbol{B\theta^{(k)}}\right)\right). \tag{2.15}$$

Using the MM construction, we have a very intuitive way to construct these iterative thresholding algorithms and have a proof that the sequence is non increasing. Let us now consider the problem from the perspective of proximal gradient descent. This class of algorithms can minimise cost functions of the form

$$C(\boldsymbol{\theta}) = C_1(\boldsymbol{\theta}) + C_2(\boldsymbol{\theta}) \tag{2.16}$$

where

i) $C_1(\boldsymbol{\theta})$ is convex, differentiable and ∇C_1 is Lipschitz continuous with constant L; i.e., $\|\nabla C_1(\boldsymbol{\theta_1}) - \nabla C_1(\boldsymbol{\theta_2})\|_2 \le L\|\boldsymbol{\theta_1} - \boldsymbol{\theta_2}\|_2$ for all $\boldsymbol{\theta_1}, \boldsymbol{\theta_2}$.

ii) $C_2(\boldsymbol{\theta})$ is convex and the proximal map of C_2 can be evaluated: the proximal map is given by $prox_t(\boldsymbol{\theta}) = arg\min_{\boldsymbol{a}} \left\{ \|\boldsymbol{\theta} - \boldsymbol{a}\|_2^2 + t\, C_2(\boldsymbol{a}) \right\}$.

To construct the algorithm using the MM framework, consider the following surrogate cost function:

$$C_{prox}(\boldsymbol{\theta} \mid \boldsymbol{a}) = C_1(\boldsymbol{a}) + \nabla C_1(\boldsymbol{a})^T(\boldsymbol{\theta} - \boldsymbol{a}) + \frac{1}{2t}\|\boldsymbol{\theta} - \boldsymbol{a}\|_2^2 + C_2(\boldsymbol{\theta}). \tag{2.17}$$

Essentially this function has two parts: the first part (everything except the $C_2(\boldsymbol{\theta})$ term) is a quadratic approximation of C_1 around $\boldsymbol{a}$ but with the $\nabla^2 C_1$ quadratic factor, of the Taylor expansion, replaced with $\frac{1}{t}$. If we iteratively minimised this first part we would obtain gradient descent for C_1. The second part is simply the $C_2(\boldsymbol{\theta})$ term, which is added to deal with this additional, non-differentiable, penalty. One can also view this cost function as the sum of three terms: a linear approximation of C_1 around $\boldsymbol{a}$, a term promoting proximity between $\boldsymbol{\theta}$ and $\boldsymbol{a}$, and finally the $C_2(\boldsymbol{\theta})$ term.

In Appendix A.2 we prove that (2.17) is a maximiser of C if $t \leq \frac{1}{L}$; therefore, using the MM results, the following sequence is guaranteed to be non increasing if $t \leq \frac{1}{L}$:

$$\begin{aligned}
\boldsymbol{\theta}^{(k+1)} &= arg\min_{\boldsymbol{\theta}} \left\{ C_1\left(\boldsymbol{\theta}^{(\boldsymbol{k})}\right) + \nabla C_1\left(\boldsymbol{\theta}^{(\boldsymbol{k})}\right)^T \left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(\boldsymbol{k})}\right) + \frac{1}{2t}\left\|\boldsymbol{\theta} - \boldsymbol{\theta}^{(\boldsymbol{k})}\right\|_2^2 + C_2(\boldsymbol{\theta}) \right\} \\
&= arg\min_{\boldsymbol{\theta}} \left\{ 2t\nabla C_1\left(\boldsymbol{\theta}^{(\boldsymbol{k})}\right)^T \boldsymbol{\theta} + \boldsymbol{\theta}^T\boldsymbol{\theta} - 2\boldsymbol{\theta}^T\boldsymbol{\theta}^{(\boldsymbol{k})} + 2tC_2(\boldsymbol{\theta}) \right\} \\
&= arg\min_{\boldsymbol{\theta}} \left\{ \left\|\boldsymbol{\theta} - \left(\boldsymbol{\theta}^{(\boldsymbol{k})} - t\nabla C_1\left(\boldsymbol{\theta}^{(\boldsymbol{k})}\right)\right)\right\|_2^2 + 2tC_2(\boldsymbol{\theta}) \right\} \\
&= prox_{2t}\left(\boldsymbol{\theta}^{(\boldsymbol{k})} - t\nabla C_1\left(\boldsymbol{\theta}^{(\boldsymbol{k})}\right)\right).
\end{aligned}$$

Furthermore, the above sequence satisfies the following error bound:

$$C\left(\boldsymbol{\theta}^{(\boldsymbol{k})}\right) - C\left(\boldsymbol{\theta}^*\right) \leq \frac{\left\|\boldsymbol{\theta}^{(\mathbf{0})} - \boldsymbol{\theta}^*\right\|_2^2}{2tk}, \tag{2.18}$$

where $\boldsymbol{\theta}^*$ is the optimal solution (proof given in Appendix A.3). I.e., the iteration converges in C, at a rate of $O\left(\frac{1}{k}\right)$.

To apply this theory to (2.5) we need to check the two requirements of proximal gradient descent: the two norm is convex and differentiable with Lipschitz gradient ($L = 2$) and the l_1-norm is convex. Therefore, (2.5) with $p = 1$ can be minimised by proximal gradient descent if $t \leq \frac{1}{2}$. As a special case of (2.9), the proximal map of the l_1-norm is given by

$$prox_t(\boldsymbol{\theta}) = arg\min_{\boldsymbol{a}} \left\{ \|\boldsymbol{\theta} - \boldsymbol{a}\|_2^2 + t\|\boldsymbol{a}\|_1 \right\} = \boldsymbol{S}_{\boldsymbol{t},\mathbf{1}}(\boldsymbol{\theta}).$$

Therefore, the proximal gradient descent for this problem is

$$\boldsymbol{\theta}^{(\boldsymbol{k}+\mathbf{1})} = \boldsymbol{S}_{\mathbf{2}\boldsymbol{t}\boldsymbol{\lambda},\mathbf{1}}\left(\boldsymbol{\theta}^{(\boldsymbol{k})} + 2t\boldsymbol{B}^T\left(\boldsymbol{y} - \boldsymbol{B}\boldsymbol{\theta}^{(\boldsymbol{k})}\right)\right) \quad \text{with} \quad t \leq \frac{1}{2}.$$

This is the same as we had for the MM construction, but with α replaced with $\frac{1}{2t}$. This shows us that for $p = 1$, iterative soft thresholding converges to the global minimum of (2.13) at a rate of

$O\left(\frac{1}{k}\right)$.

Acceleration methods are well known in the field of gradient descent with Nesterov being a key protagonist in this field [52]. In 2007 [53] he extended these ideas to the proximal gradient descent algorithm and obtained a convergence rate of $O\left(\frac{1}{k^2}\right)$. The proposed acceleration method requires knowledge of all previous steps and two prox evaluations per iteration. In 2009 Beck and Teboulle [4] independently published a slightly different acceleration method that obtained the same convergence rate, but only required the two previous steps and one prox evaluation per iteration. This algorithm, known as the fast iterative shrinkage-thresholding algorithm (FISTA) as opposed to the standard iterative shrinkage-thresholding algorithm (ISTA), is an extension of Nesterov's original work [52].

As we have seen, the standard proximal gradient descent is given by the sequence

$$\boldsymbol{\theta}^{(k+1)} = prox_{2t}\left(\boldsymbol{\theta}^{(\boldsymbol{k})} - t\nabla C_1\left(\boldsymbol{\theta}^{(\boldsymbol{k})}\right)\right).$$

Instead of applying this update to the current point $\boldsymbol{\theta}^{(k)}$, the FISTA first calculates a linear combination of the current and previous points:

$$\boldsymbol{\omega} = \boldsymbol{\theta}^{(k)} + \frac{k-2}{k+1}\left(\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}\right),$$

and then calculates the update using this linear combination:

$$\boldsymbol{\theta}^{(k+1)} = prox_{2t}\left(\boldsymbol{\omega} - t\nabla C_1(\boldsymbol{\omega})\right)^2.$$

The addition of this properly weighted memory term, $\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k-1)}$, gives the proposed, $O\left(\frac{1}{k^2}\right)$, convergence rate. One downside of these acceleration techniques is that they are not decent methods; i.e., the resulting iterations are no longer guaranteed to decrease at each step.

2.2.3. The non-convex problem and uniqueness of solutions

In the preceding subsection we saw that proximal gradient descent could exactly minimise (2.5) for the convex, $p = 1$, case, but in the non-convex, $p = 0$, case this analysis breaks down. However, the MM analysis is still valid so we know that iterative hard thresholding will produce a monotonically decreasing sequence. Therefore, we intuitively expect the algorithm to converge to a local minimum

[2]Note that this formulation of FISTA is taken from [38] and is slightly different to the presentation given in [4].

of this non-convex cost function and, as we have already stated, this was formally proved in [7].

Interestingly, if $\boldsymbol{B}$ is very sparse then the l_0 constrained problem ((2.2) with $p = 0$) is equivalent to the l_1 constrained problem ((2.2) with $p = 1$); i.e., the solution of the non convex problem is the same as the solution of the convex problem that we can solve. Therefore, in this case FISTA can be used to find the sparsest solution at a rate of $O\left(\frac{1}{k^2}\right)$.

There is also a large family of greedy algorithms that can be applied to the non convex problem. In [69] Tropp shows that, like iterative soft thresholding, orthogonal matching pursuit (OMP) solves the l_1 constrained problem when $\boldsymbol{B}$ satisfies the same sparsity conditions. Recall that OMP is a simple greedy algorithm that seeks sparse solutions as follows: first the current approximation is initialised to zero ($\boldsymbol{\theta}^{(0)} \leftarrow \mathbf{0}$); then at each iteration, the column from $\boldsymbol{B}$ that best approximates the residual is selected and the current approximation is given by the projection of the signal onto the space spanned by all columns that, up to this iteration, have been selected.

For more information on exact sparse recovery we refer the reader to the compressed sensing (or compressive sampling) literature.

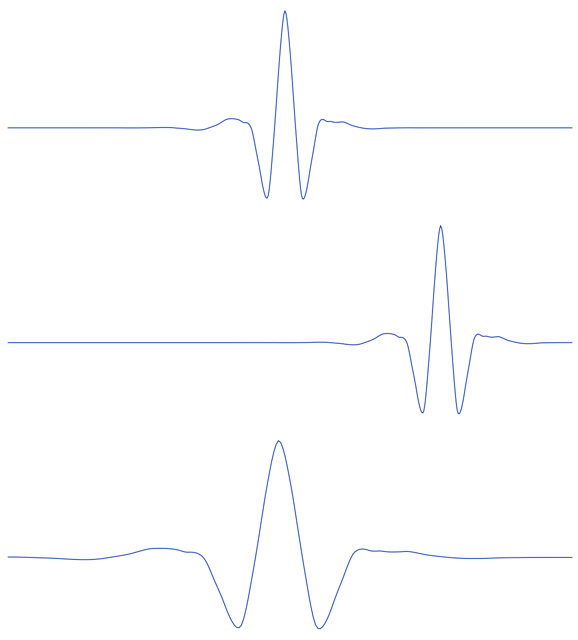
2.3. Sparse approximation of 1-D functions using wavelets

In the rest of this chapter we will show how the preceding approximation algorithms can be used to find sparse approximations of signals. We will be particularly interested in piecewise polynomial signals because this class of signals can be very effective at approximating images and are central to the work of this thesis. For simplicity we will first consider sparse approximations of 1-D signals using well established wavelet theory [21, 48, 67, 71].

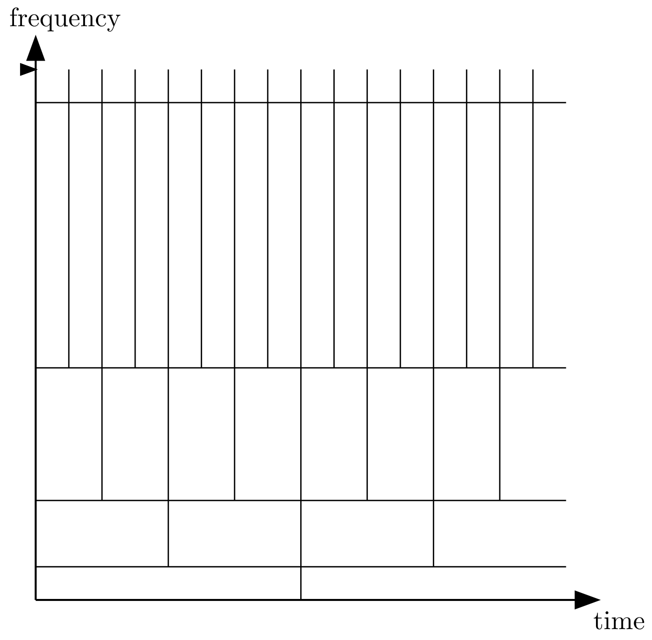
2.3.1. Introduction to wavelets

Wavelets can be constructed with desirable properties that lead to sparse approximations of piecewise polynomial signals: specifically they can be constructed to be smooth (bounded in frequency), have compact support (bounded in time or space), and have vanishing moments (their inner product with polynomials is zero). Unlike the traditional Fourier transform, they are well localised in both time and frequency.

The continuous wavelet transform (CWT) is an infinite combination of shifts and dilations of a



(a) Wavelets generated from dyadic shifts and dilations



(b) Dyadic time-frequency partitioning

Figure 2.4.: 1-D continuous time wavelets with the corresponding dyadic time-frequency partitioning.

mother wavelet, $\psi(t)$:

$$f_{\mathcal{W}}(u,s) = \frac{1}{\sqrt{s}} \int_{-\infty}^{\infty} f(x)\psi^*\left(\frac{x-u}{s}\right) dx, \tag{2.19}$$

where the $\frac{1}{\sqrt{s}}$ term ensures that all shifted and dilated versions have equal norm. If $s = 2^m$ and $u = n2^m$ then (2.19) becomes the dyadic wavelet series:

$$f_{\mathcal{W}}[m,n] = 2^{-m/2} \int_{-\infty}^{\infty} f(x)\psi^*\left(2^{-m}x - n\right) dx \qquad m,n \in \mathbb{Z}.$$

Wavelets can be constructed, see [21], such that the functions, $2^{-m/2}\psi^*\left(2^{-m}x - n\right)$, form a basis of $L_2(\mathbb{R})$, with the favourable time frequency partitioning shown in Fig. 2.4(b). Here the transform has good frequency but poor time localisation at low frequencies and vice-versa at high frequencies because of the dilation of the waveform: when the mother wavelet is stretched it contains a small range of lower frequencies and covers a larger time window, whereas when it is compressed it contains a larger range of higher frequencies but covers a smaller time interval. An example of a mother wavelet and a shift and scale of the form $s = 2^m$ and $u = n2^m$ are shown in Fig. 2.4(a).

As well as producing this favourable time-frequency partitioning, wavelets can be constructed with vanishing moments, which means that they annihilate polynomials up to a certain degree.

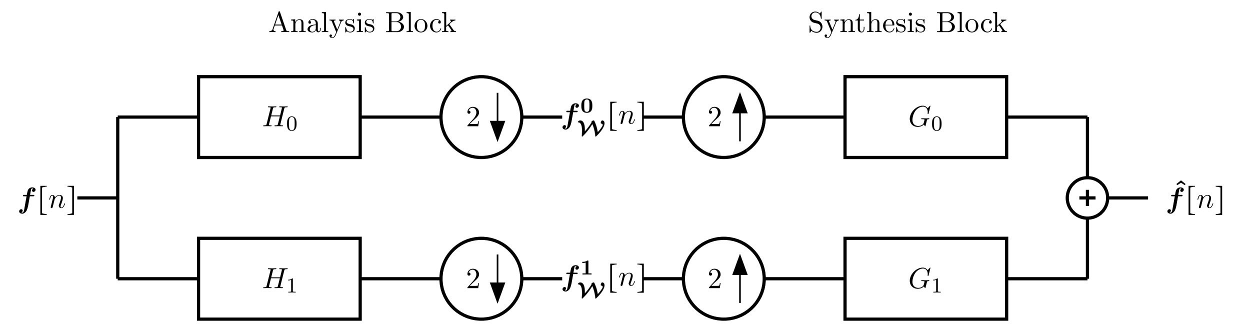

Analysis Block
Synthesis Block
H_0
H_1
G_0
G_1
$2\downarrow$
$2\uparrow$
$\boldsymbol{f}[n]$
$\boldsymbol{f}^{\boldsymbol{0}}_{\mathcal{W}}[n]$
$\boldsymbol{f}^{\boldsymbol{1}}_{\mathcal{W}}[n]$
$\hat{\boldsymbol{f}}[n]$

Figure 2.5.: A two channel filter bank.

More formally, a wavelet with n vanishing moments satisfies

$$\int_{-\infty}^{\infty} \psi(x)x^k dx = 0 \qquad 0 \leq k < n.$$

Importantly, this is also true for all shifts and dilations; therefore, the wavelet transform, for a wavelet with n vanishing moments, satisfies

$$f_{\mathcal{W}}(u,s) = \frac{1}{\sqrt{s}} \int_{-\infty}^{\infty} x^k \psi^* \left(\frac{x-u}{s}\right) dx = 0 \qquad 0 \leq k < n. \tag{2.20}$$

This can be seen by substituting $x' = \dfrac{x-u}{s}$ into (2.20):

$$f_{\mathcal{W}}(u,s) = \frac{1}{\sqrt{s}} \int_{-\infty}^{\infty} (sx'+u)^k \psi^*(x')dx',$$

and then taking the binomial expansion of $(sx'+u)^k$:

$$\begin{aligned} f_{\mathcal{W}}(u,s) &= \frac{1}{\sqrt{s}} \int_{-\infty}^{\infty} \sum_{i=0}^{k} \left\{ \binom{k}{i} (sx')^i u^{k-i} \right\} \psi^*(x')dx' \\ &= \frac{1}{\sqrt{s}} \sum_{i=0}^{k} \left[\binom{k}{i} s^i u^{k-i} \int_{-\infty}^{\infty} x'^i \psi^*(x')dx' \right] = 0 \qquad \text{if} \quad 0 \leq k < n. \end{aligned}$$

Therefore if we take the wavelet transform of a piecewise polynomial signal using a compactly supported wavelet with vanishing moments, the only non-zero samples will come from the regions where the compactly supported wavelet intersects a discontinuity. Therefore if the piecewise polynomial signal does not contain too many discontinuities, the transformed signal will have many zeros and thus be sparse. This is shown in Fig. 2.7, using the discrete wavelet transform (DWT).

The DWT can be constructed by iterating the two channel filter bank, shown in Fig. 2.5, to

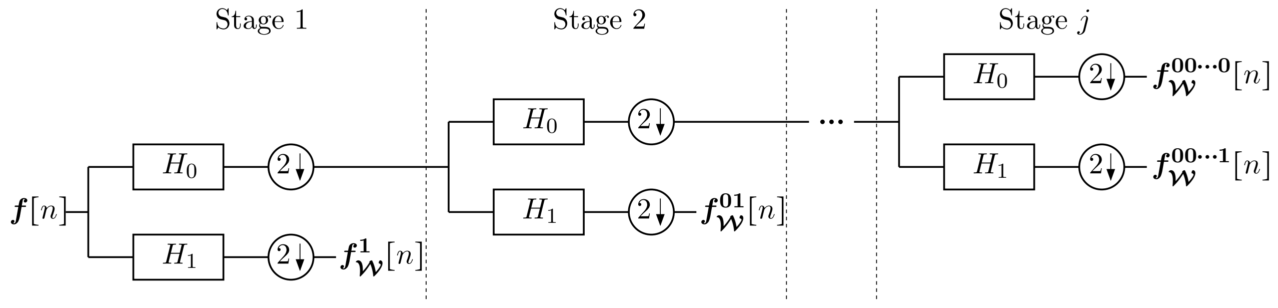

Stage 1
Stage 2
Stage j
$\boldsymbol{f}[n]$
H_0
H_1
$2\downarrow$
$\boldsymbol{f}_{\mathcal{W}}^{\mathbf{1}}[n]$
$\boldsymbol{f}_{\mathcal{W}}^{\mathbf{01}}[n]$
$\boldsymbol{f}_{\mathcal{W}}^{\mathbf{00\cdots0}}[n]$
$\boldsymbol{f}_{\mathcal{W}}^{\mathbf{00\cdots1}}[n]$
…

Figure 2.6.: The analysis discrete wavelet transform, constructed from a dyadic tree-structured filter bank.

produce the dyadic tree structured filter bank shown in Fig. 2.6. Figure 2.6 just shows the analysis, forward transform; however, it is clear that each stage can be processed with a synthesis block from the two channel case. In fact we can design the filters, H_0, H_1, G_0 and G_1, to produce perfect reconstruction filter banks so that $\hat{\boldsymbol{f}} = \boldsymbol{f}$ for any $\boldsymbol{f} \in l_2(\mathbb{Z})$. This was understood by the signal processing community well before the connection with continuous time wavelets was suggested by Mallat [49, 50]. It was this unification of independent, and seemingly unrelated, results in mathematics and signal processing (and in fact a number of other disciplines) that caused much of the excitement around wavelets.

Just like the continuous time wavelet, a discrete time highpass filter with vanishing moments annihilates polynomials. Therefore, as shown in Fig. 2.7, the highpass coefficients of the DWT of a piecewise polynomial signal are zero except for a few non zero coefficients in the cone of influence of the discontinuity. Of course, the width of this cone is determined by the support of the wavelet. In this example a wavelet with at least two vanishing moments was required to annihilate polynomials up to the linear degree. The piecewise linear signal is, then, decomposed into a sum of a few non-zero samples, from the highpass coefficients, and the low pass coefficients, $\boldsymbol{f}_{\mathcal{W}}^{\mathbf{00\cdots0}}[n]$, at the coarsest scale. This sparsity is particularly desirable: for example, in Chapter 3 we will see how it can be exploited to develop state of the art image restoration and enhancement algorithms. In [27] an even sparser representation using wavelets was obtained by exploiting the fact that these non-zero coefficients in the discontinuity's cone of influence are related to each other and are thus predictable.

In reality, signals are not normally exactly piecewise polynomial but are often well approximated by piecewise polynomials. For example Fig. 2.8 shows one line of the cameraman image along with the highpass coefficients of its wavelet transform.

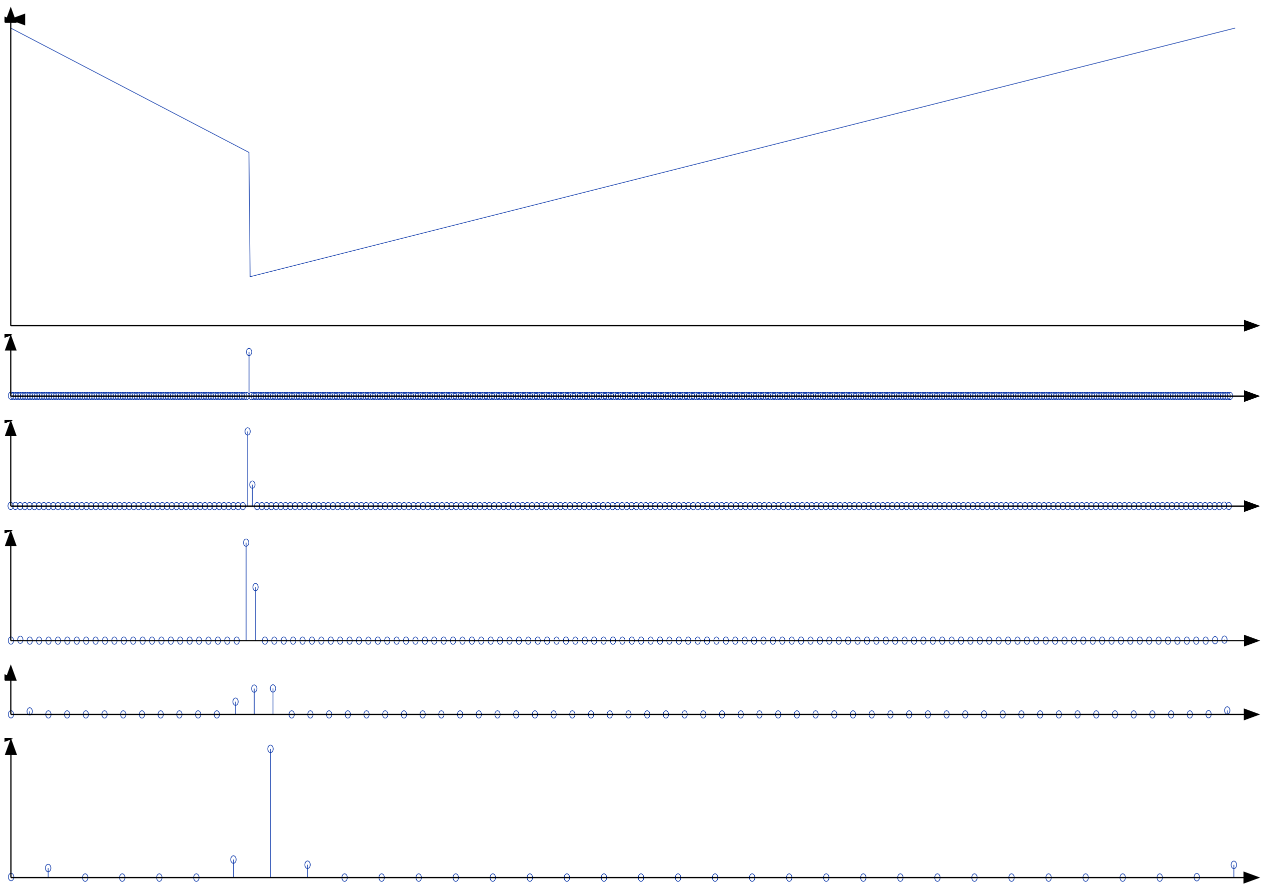


Figure 2.7.: A piecewise linear 1-D signal decomposed into highpass subbands using a wavelet with two vanishing moments. From top to bottom is the time domain signal, followed by the amplitude of the highpass coefficients at decreasing scales.

Very few of these coefficients are identically zero, but the majority of them has a very small amplitude and almost all of the signal's energy is contained in a small number of, larger, coefficients. Therefore, we can retain almost all of the signal's information by using a sparse approximation. In this case, since the transform is orthogonal, hard thresholding produces the sparsest solution. However as we know from the previous analysis, if the transform was biorthogonal, hard thresholding would not be the optimum sparsest solution in the least squares sense, but would just be the first step of an iteration that would converge to a local minimum of this problem.

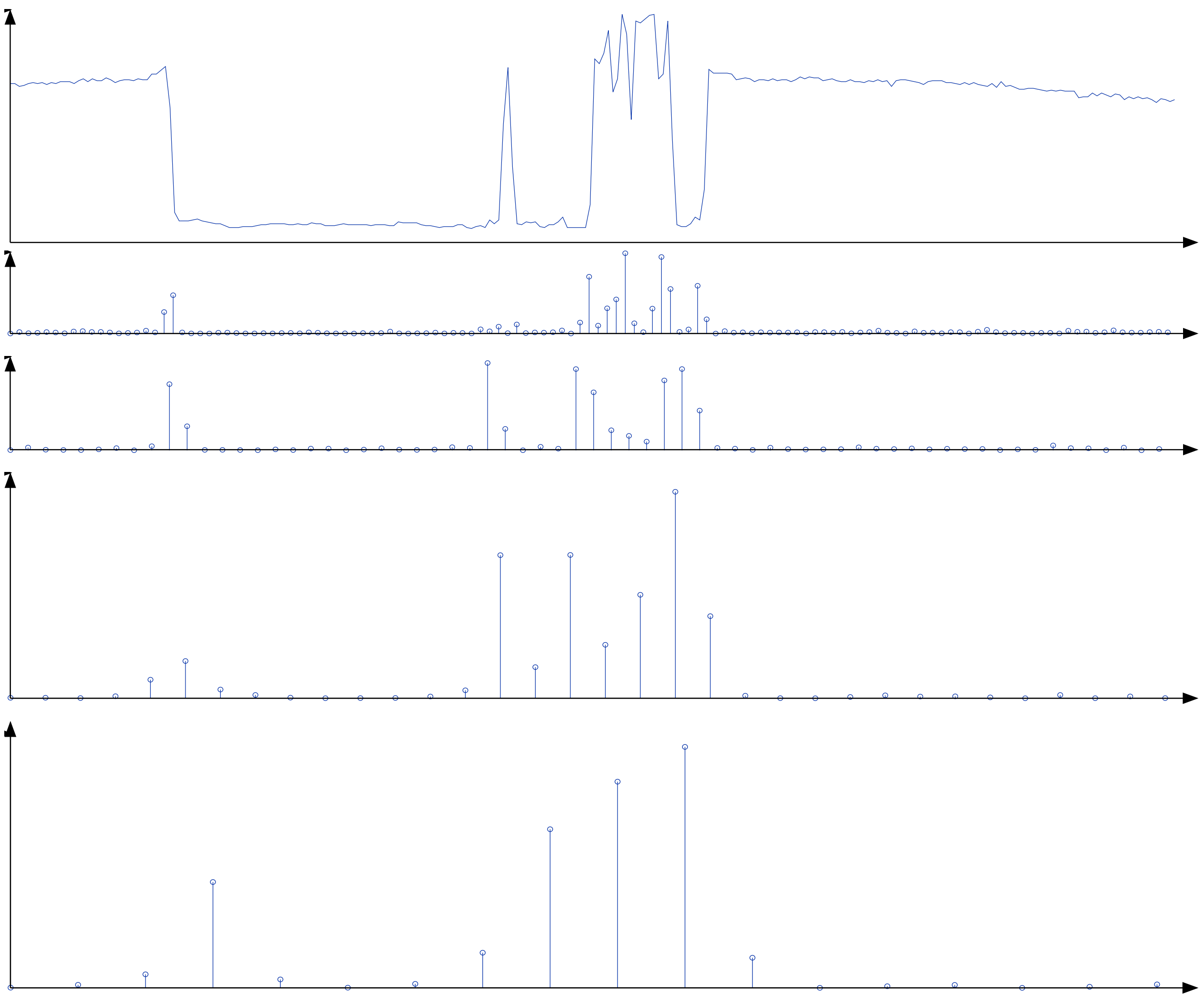
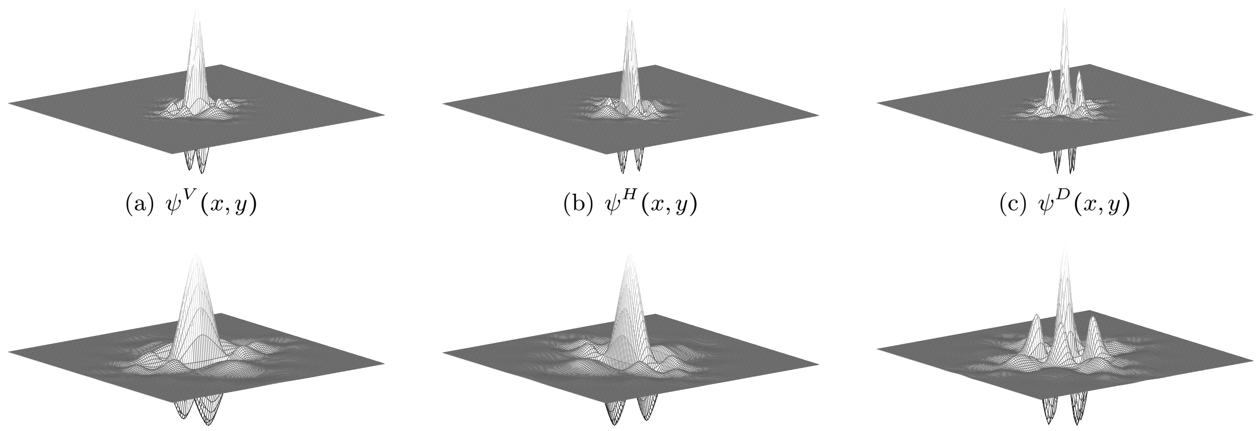


Figure 2.8.: A scanline of the cameraman image decomposed into highpass subbands using a wavelet with two vanishing moments. From top to bottom is the time domain signal, followed by the amplitude of the highpass coefficients at decreasing scales.

2.4. Sparse approximation of 2-D functions using wavelets and beyond

The inclusion of the 2-D wavelet transform in the JPEG2000 image compression standard is a testament to not only wavelet's sparse promoting properties, but also their speed and stability. However, they are far from perfect: they are shift variant[3] and because of their separability in 2-D they only efficiently represent point singularities and struggle to capture the higher dimensional discontinuities, such as the edges and contours, that are seen in images. This problem has been

[3]If $y[m,n]$ is the response of a shift invariant system to $x[m,n]$ then $y[m-k_m, n-k_n]$ is the response to $x[m-k_m, n-k_n]$. A shift variant system is a system that is not shift invariant.



(a) $\psi^V(x,y)$ (b) $\psi^H(x,y)$ (c) $\psi^D(x,y)$

(d) $\psi^V(x,y)$ dilated (e) $\psi^H(x,y)$ dilated (f) $\psi^D(x,y)$ dilated

Figure 2.9.: Plots of the Daubechies 2nd order 2-D continuous time wavelet to show the lack of directional flexibility of the transform.

noted many times in the literature and many improved transforms have been suggested, including Ridgelets [10], Curvelets [11], Contourlets [23], Wedgelets [26], Bandlets [44] and Directionlets [70]. In this section we will highlight these shortcomings and briefly summarise some of these improved transforms.

2.4.1. Two dimensional wavelets

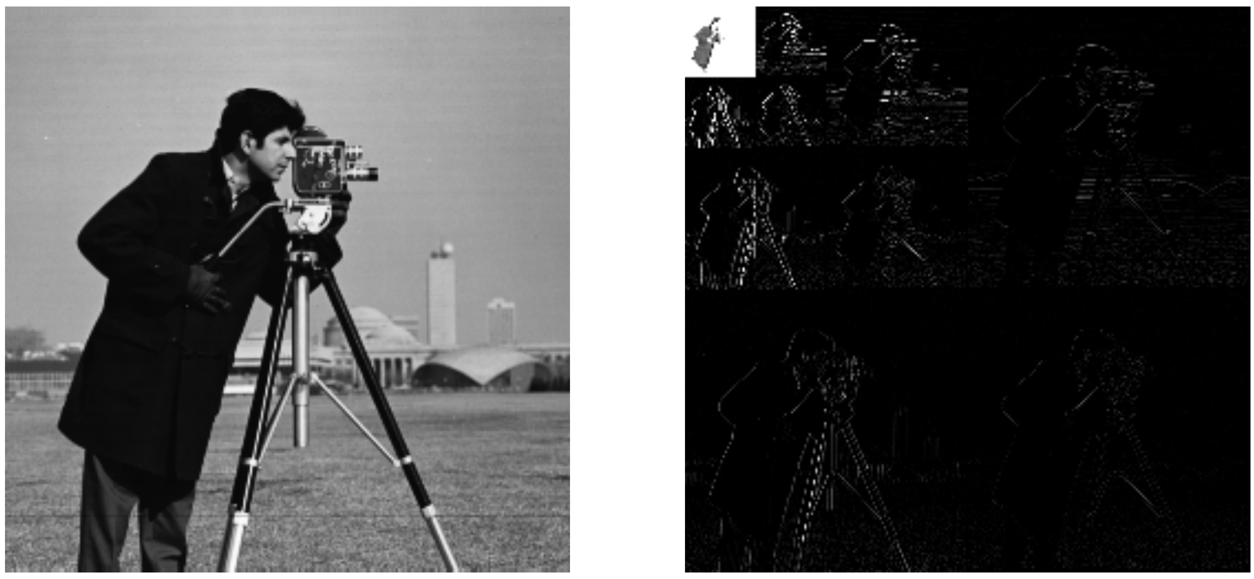
The 2-D DWT is constructed by applying the 1-D DWT to the rows followed by columns (or vice-versa) producing three subbands per stage. Although there are more flexible ways to generate the 2-D CWT, we will consider it in its analogous separable form, because this will highlight the inefficiencies that are present in the 2-D DWT. Given a 1-D mother wavelet $\psi(x)$ and associated scaling function $\phi(x)$, the separable 2-D wavelet transform is given by

$$f_{\mathcal{W}}(u_x, u_y, s, k) = \frac{1}{\sqrt{s}} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) \psi^k \left(\frac{x - u_x}{s}, \frac{y - u_y}{s} \right) dx dy,$$

where $\psi^k(x,y)$ is one of three wavelets constructed, from separable products, to produce wavelets orientated in the vertical, horizontal and diagonal directions:

$$\psi^V(x,y) = \phi(x)\psi(y), \quad \psi^H(x,y) = \psi(x)\phi(y), \quad \psi^D(x,y) = \psi(x)\psi(y).$$

An example of these three functions is shown in Fig. 2.9, along with a dilation of each. It is clear that dilating these waveforms is not enough to capture edges that occur at different orientations.



(a) Cameraman image. (b) Wavelet decomposition of the Cameraman image.

Figure 2.10.: The original cameraman image and its wavelet decomposition. The wavelet decomposition shows both its sparsity and also the large samples that occur around discontinuities, at all levels.

This is further highlighted when, in Fig. 2.10, we take the 2-D DWT of the cameraman image. There is a large number of large samples around edges that would not be present if the wavelets were better orientated. The task of designing better oriented functions has been studied extensively leading to a large number of overcomplete and sometimes adaptive transforms, which we will summarise in the next subsection.

2.4.2. Beyond wavelets

As already stated, discrete time wavelets are shift variant and lack good directional adaptability in 2-D.

Shift variance is not so important for compression, where redundancy is detrimental, but in restoration and enhancement problems not exploiting this variance leads to blocking artefacts and poor performance. One solution to this problem is to introduce redundancy by removing the decimation steps of the DWT. This produces the undecimated, stationary or à trous DWT which in 1-D can be calculated by removing the downsamplers from Fig. 2.6 and upsampling the filter coefficients at the j-th stage by a factor of 2^{j-1}. Since each subband is now the same length, this introduces a redundancy of $j+1$ but, unlike the decimated transform, has the advantageous property of shift-invariance.

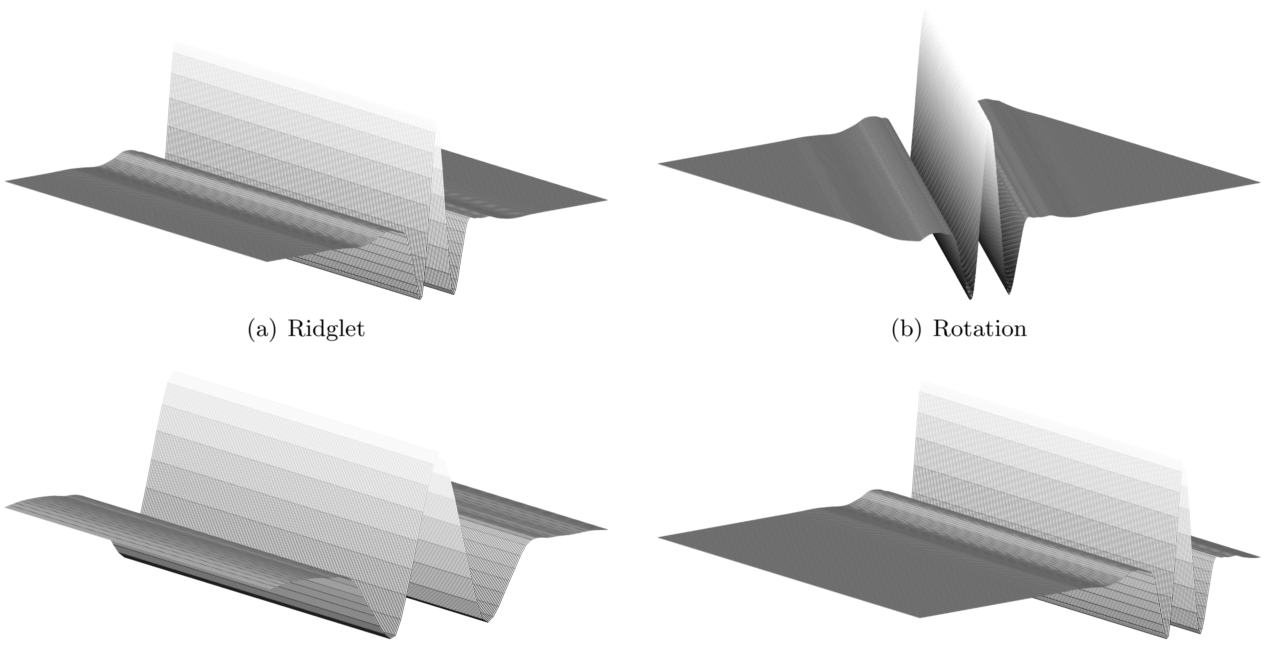

(a) Ridglet (b) Rotation (c) Dilation (d) Shift

Figure 2.11.: A Ridgelet along with a rotation, dilation and shift.

The 2-D undecimated DWT (UDWT) is, like its decimated counterpart, obtained from 1-D transforms of the rows followed by columns; therefore, since all subbands are the same size as the input, it has a redundancy of $3j + 1$.

The isotropic – meaning equal in all directions – undecimated DWT is a 2-D transform that can be obtained by summing, at each stage, the three subbands of the 2-D undecimated DWT. Its redundancy is thus reduced to $j + 1$. This transform is particularly useful for astronomical data and will also be used in the first generation Curvelet transform.

The second problem of directional adaptability can be solved by constructing better oriented transforms. In 1999 Candes et al. [10] proposed the Ridgelet transform, which is an overcomplete frame of $L_2(\mathbb{R})$:

$$f_{Ridgelet}(u, s, \theta) = \frac{1}{\sqrt{s}} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \psi \left(\frac{x \cos(\theta) + y \sin(\theta) - u}{s} \right) dxdy.$$

As shown in Fig. 2.11, the frame vectors or ridgelets are constant along lines parallel to $x \cos(\theta) + y \sin(\theta) = u$ and are wavelets perpendicular to these lines. Introducing this rotation parameter, θ, obviously improves the directional sensitivity of the transform. The reader may observe the similarity between ridgelets and the Radon transform,

$$f_{Radon}(u,\theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta\left(u - x\cos(\theta) - y\sin(\theta)\right) dxdy.$$

In fact, the Ridgelet transform is nothing more than 1-D wavelet transforms calculated along rays of the Radon transform. Recall that the Radon transform converts straight lines into points, which the 1-D wavelet transform can efficiently represent. The Radon transform provides a very effective strategy to calculate the continuous Ridgelet transform and similar approaches have been developed for the discrete time case. We refer the interested reader to [65, 66].

Ridgelets, although optimal for global straight edges, still do not efficiently capture the more flexible edges seen in images. Locally curves are approximately straight, so a natural extension is to apply the transform locally. This is exactly what Candes et al. [11] propose in their first generation Curvelet transform. This transform first applies an isotropic undecimated DWT and then, on each of the, j, highpass subbands, block Ridgelet transforms of different sizes are applied locally. More precisely the $j = 1$ subband is partitioned into blocks at a minimum size and the $j = 2$ and $j = 3$ subbands are partitioned into blocks that are twice this minimum size. For later subbands the block size is doubled every two subbands (i.e. it is doubled when j is odd). The coefficients of the first generation Curvelet transform are given by the Ridgelet transform of each of these blocks and the unchanged lowpass subband. Partitioning the subbands in this way makes the curvelet frame vectors have a length proportional to 2^j and width proportional to 2^{2j} as well as the orientation θ. This makes the transform particularly suited to C^2 singularities. The second generation Curvelet transform [12, 13] maintains the advantageous features of the first but is less redundant and computationally cheaper to compute.

The sparse approximation that curvelets provide of C^2 discontinuities is surprising for a non adaptive transform, however they still lack efficient adaptability to other edge structures. So far, the only way to efficiently represent more flexible edge structures has been to use non adaptive transforms. This is exactly what we will do in Chapter 4 when we develop our novel quadree decomposition approximation algorithm. Other adaptive quadtree decomposition models include wedgelets and bandlets. The Wedgelet transform [26] represents each adaptive region by two constant regions separated by a straight edge. This was later extended, by Shukla et al. [63], to polynomial regions and continuous boundaries. Furthermore, the authors added an additional joining step that allowed the adaptive regions to be much more flexible. It is this transform that we extend in Chapter 4.

Finally, the Bandlet transform, of Le Pennec and Mallat [44] and later refined by Peyré and Mallat [54], are another non adaptive algorithm based on a quadtree decompostion. However unlike our model, which uses a very low dimensional model on each region, Bandlets approximate each adaptive region using a basis. Specifically, Bandlets exploit some of the redundancy that is present in the wavelet transform of a 2-D signal. As can be seen from Fig. 2.10, the non-zero coefficients in the highpass subbands occur along the edge discontinuities at all levels. The Bandlet transform computes a quadtree decomposition of each highpass subband and then on adaptive regions computes an Alpert basis [2] oriented along the geometric flow of the region. Thresholding these Alpert bases allows the edge discontinuities to be efficiently represented.

2.5. Summary

In this chapter we have introduced the concept of sparsity as a regularisation technique and studied the resulting optimisation problems. We have seen that an l_1-norm constraint promotes sparsity and produces convex optimisation problems that can be exactly solved using iterative soft thresholding. Furthermore, an exact sparsity measure is given by an l_0-norm constraint but this produces non convex optimisation problems that, in general, can not be exactly solved. However, iterative hard thresholding converges to a local minimum and if the desired solution is sparse enough the l_1-norm and l_0-norm constrained problems become equivalent so that iterative soft thresholding can find the global optimum solution. As a special case, when the transform $\boldsymbol{B}$ is unitary these iterations simplify to exact closed-form solutions; i.e soft thresholding for l_1 and hard thresholding for l_0.

Later in the chapter we saw that, using a wavelet with enough vanishing moments, the wavelet transform of a 1-D piecewise polynomial signal is sparse and, for many images, the 1-D wavelet transform of a scanline compresses the signals energy into a relatively small number of coefficients. Despite the success of the 2-D wavelet transform, it fails to efficiently represent the complex edge structures present in images. Proposed solutions to this problem include over complete representations such as ridgelets, which give sparse representations of global straight edges; curvelets, which give sparse representations of C^2 discontinuities and adaptive techniques such as the Bandlet transform, which is effective at most edge structures.

However, the Bandlet transform uses a basis for each adaptive region, which can, in many applications, provide too much flexibility. In Chapter 4 we will propose our novel approximation algorithm that uses a very low dimensional model for each region of an adaptive partitioning. This model provides very sparse approximations of piecewise polynomial images, which has many practical applications in restoration and enhancement that we will explore.

In the next chapter we will see how the material developed in this chapter can be used to develop state of the art image restoration and enhancement algorithms.

CHAPTER 3

State of the Art Image Restoration and Problem Setup

3.1. Introduction

In many practical scenarios one measures a degraded version of an image and restoration is required to estimate the original. In this thesis we will consider four restoration problems, namely denoising, deconvolution, interpolation and multi-view super resolution, which span a wide range of applications. For example noise is present in all applications and, interestingly, many algorithms related to other problems often include a denoising step. For example, we will show that the deconvolution problem can be solved by first applying a regularised inverse, to invert the convolution, and then denoising, or, alternatively, by iterative denoising. Interpolation of even spaced samples allows resolution enhancement. We will consider the case of irregularly spaced samples, which has interesting applications in depth image acquisition. Finally we will consider multi-view image super resolution, which aims to combine multiple low resolution images of the same scene into one high resolution image. This complex problem involves many stages including interpolation and deconvolution steps.

All these problems can, if $\boldsymbol{H}$ is properly constructed, be modelled by the linear degradation model

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{z}. \tag{3.1}$$

Here the vectors $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{R}^N$ are the desired, measured and noise images respectively (N is the number of pixels). The matrix $\boldsymbol{H}$ will be chosen according to the type of problem: for example if $\boldsymbol{H} = \boldsymbol{I_N}$, (3.1) models an additive noise process, and if $\boldsymbol{H} \in \mathbb{R}^{N\times N}$ is a circulant matrix corresponding to a particular filtering, (3.1) models a noisy filtering process, with the filtering most commonly creating blur.

Assuming that $\boldsymbol{x}$ is sparse in a proper domain, we can approximate $\boldsymbol{x}$ from $\boldsymbol{y}$, given in (3.1), by using the minimisation techniques of the previous chapter. In the following subsections we will show this relationship for the four problems and show the analysis that leads to state of the art algorithms.

3.2. Denoising

Noise is present in all practical restoration problems, therefore we must develop restoration algorithms that are robust to noise. Furthermore, many other restoration problems can be constructed such that they have a denoising step. In this subsection we will show how the denoising problem can be solved using sparse optimisation theory. We will first consider the simplest case where the noise is additive white and Gaussian and later extend this noise model to coloured noise. We will show examples of results obtained using some of the transforms discussed in the previous chapter, as well as one additional result that exploits non-local similarity of regions to obtain an even sparser approximation.

The denoising problem can be defined as approximating $\boldsymbol{x}$ from $\boldsymbol{y}$ where

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{z},$$

with $\boldsymbol{z}$ the noise vector; i.e., (3.1) with $\boldsymbol{H} = \boldsymbol{I_N}$.

Assuming $\boldsymbol{x}$ is sparse in a proper domain; i.e., $\boldsymbol{x} = \boldsymbol{B\theta}$ with $\boldsymbol{\theta}$ sparse and $\boldsymbol{B} \in \mathbb{R}^{N\times d}$, a possible solution to (3.1) is given by

$$\hat{\boldsymbol{x}} = \boldsymbol{B}\hat{\boldsymbol{\theta}}, \tag{3.2}$$

where

$$\hat{\boldsymbol{\theta}} = arg\min_{\boldsymbol{\theta}} \left\{ \|\boldsymbol{y} - \boldsymbol{B\theta}\|_2^2 + \lambda \|\boldsymbol{\theta}\|_p^p \right\}, \tag{3.3}$$

and we have the usual choices for p to promote sparse solutions. In the previous chapter we extensively studied this minimisation problem, which we will now use for restoration. Intuitively we are searching for a sparse solution that is close to the measured signal $\boldsymbol{y}$. As we will show in the next subsection, it can be interpreted as a maximum a posteriori (MAP) estimator for the case when $\boldsymbol{z}$ is additive white Gaussian noise.

3.2.1. White noise

If $\boldsymbol{z}$ is white Gaussian, $\boldsymbol{z} \sim \mathcal{N}_N(\boldsymbol{0}, \sigma_z^2 \boldsymbol{I_N})$,

$$p(\boldsymbol{z} = \grave{\boldsymbol{z}}) = \frac{1}{\sigma_z^N (2\pi)^{N/2}} exp\left(-\frac{\grave{\boldsymbol{z}}^T \grave{\boldsymbol{z}}}{2\sigma_z^2}\right),$$

and

$$p(\boldsymbol{y} \mid \boldsymbol{\theta}) = p(\boldsymbol{z} = \boldsymbol{y} - \boldsymbol{B\theta}) = \frac{1}{\sigma_z^N (2\pi)^{N/2}} exp\left[-\frac{(\boldsymbol{y} - \boldsymbol{B\theta})^T (\boldsymbol{y} - \boldsymbol{B\theta})}{2\sigma_z^2}\right].$$

The inverse probability $p(\boldsymbol{\theta} \mid \boldsymbol{y})$ can be found using Bayesian's theorem:

$$p(\boldsymbol{\theta} \mid \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\boldsymbol{y})} = \frac{p(\boldsymbol{\theta})}{p(\boldsymbol{y}) \sigma_z^N (2\pi)^{N/2}} exp\left[-\frac{(\boldsymbol{y} - \boldsymbol{B\theta})^T (\boldsymbol{y} - \boldsymbol{B\theta})}{2\sigma_z^2}\right].$$

Let

$$p(\boldsymbol{\theta} = \grave{\boldsymbol{\theta}}) = Aexp\left[-\frac{\zeta \|\grave{\boldsymbol{\theta}}\|_p^p}{2}\right],$$

so that the sparser the solution the greater the assigned probability. Here the constant, A, is set such that

$$\sum_{\boldsymbol{\theta} \in \mathbb{R}^N} Aexp\left[-\frac{\zeta \|\boldsymbol{\theta}\|_p^p}{2}\right] = 1.$$

Under these assumptions the MAP estimator is

$$\begin{aligned}
\hat{\boldsymbol{\theta}} &= arg \max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \boldsymbol{y}) = arg \max_{\boldsymbol{\theta}} log(p(\boldsymbol{\theta} \mid \boldsymbol{y})) \\
&= arg \max_{\boldsymbol{\theta}} \left\{ K - \frac{(\boldsymbol{y} - \boldsymbol{B\theta})^T (\boldsymbol{y} - \boldsymbol{B\theta})}{2\sigma_z^2} - \frac{\zeta \|\boldsymbol{\theta}\|_p^p}{2} \right\} \\
&= arg \min_{\boldsymbol{\theta}} \left\{ \|\boldsymbol{y} - \boldsymbol{B\theta}\|_2^2 + \zeta \sigma_z^2 \|\boldsymbol{\theta}\|_p^p \right\}, \qquad (3.4)
\end{aligned}$$

which is equivalent to (2.1) with $\lambda = \zeta \sigma_z^2$, proportional to the noise variance. Recall from the previous chapter that if $\boldsymbol{B}$ is unitary and $p = 0$, (3.4) is solved by hard thresholding each coefficient:

$$\hat{\boldsymbol{\theta}}[i] = S_{\zeta\sigma_z^2,0}\left(\boldsymbol{y_B}[i]\right) = \begin{cases} \boldsymbol{y_B}[i] & \text{if} \quad (\boldsymbol{y_B}[i])^2 \geq \zeta\sigma_z^2 \\ 0 & \text{if} \quad (\boldsymbol{y_B}[i])^2 < \zeta\sigma_z^2 \end{cases}, \tag{3.5}$$

where $\boldsymbol{y_B} = \boldsymbol{B}^T\boldsymbol{y}$.

We can gain some intuition for the parameter ζ by considering the sparse promoting denoising problem as a hypothesis test. Since $\boldsymbol{z}$ is white, $\boldsymbol{y_B}$ has the following distribution:

$$\begin{aligned} \boldsymbol{y_B} = \boldsymbol{B}^T\boldsymbol{y} = \boldsymbol{B}^T(\boldsymbol{x}+\boldsymbol{z}) &\sim \mathcal{N}_d\left(\boldsymbol{B}^T\boldsymbol{x}, \boldsymbol{B}^T\sigma_z^2\boldsymbol{I_N}\boldsymbol{B}\right) \\ &\sim \mathcal{N}_d\left(\boldsymbol{B}^T\boldsymbol{x}, \boldsymbol{B}^T\boldsymbol{B}\sigma_z^2\right), \end{aligned}$$

and is, thus, independent and identically distributed (iid) when $\boldsymbol{B}$ is unitary:

$$\boldsymbol{y_B} \sim \mathcal{N}_d\left(\boldsymbol{B}^T\boldsymbol{x}, \boldsymbol{I_d}\sigma_z^2\right),$$

which, if $\boldsymbol{b_i}$ is the i-th column of $\boldsymbol{B}$, can be written individually for each component as follows:

$$\boldsymbol{y_B}[i] \sim \mathcal{N}\left(\boldsymbol{b_i}^T\boldsymbol{x}, \sigma_z^2\right).$$

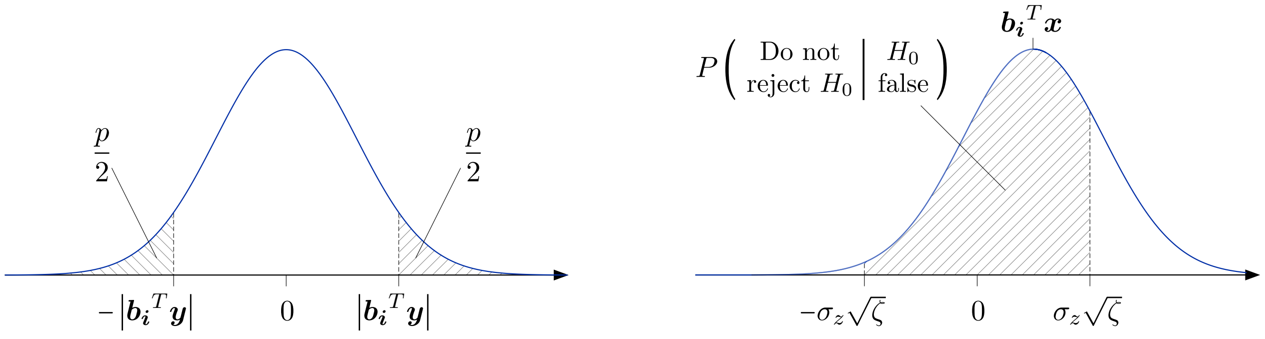
To formulate this thresholding as a hypothesis test, let the null hypotheses, H_0, be the expected case where, since the transform is sparse, the original coefficient is zero and the alternative hypothesis, H_1 be the contrary:

$$\begin{aligned} H_0 &: \quad \boldsymbol{b_i}^T\boldsymbol{x} = 0, \\ H_1 &: \quad \boldsymbol{b_i}^T\boldsymbol{x} \neq 0. \end{aligned}$$

Assuming the null hypothesis is true, the probability of observing a coefficient at least as extreme as $\boldsymbol{b_i}^T\boldsymbol{y}$ is

$$p = 2 - 2\Phi\left(\frac{\boldsymbol{b_i}^T\boldsymbol{y}}{\sigma_z}\right), \tag{3.6}$$

where Φ is the cumulative distribution function for the standard normal distribution. The p-value, given in (3.6), is graphically depicted in Fig. 3.1(a). The smaller the value of p, the more evidence



(a) Graphical illustration of the p-value. (b) Graphical illustration of a type II error.

Figure 3.1.: Graphical illustrations of the p-value given in (3.6) and the probability of a type II error given in (3.9).

there is to reject the null hypothesis. Therefore if we decide to reject the null hypothesis if p is less than some small predetermined probability α, we will reject the null hypothesis if

$$p = 2 - 2\Phi\left(\frac{\boldsymbol{b_i}^T\boldsymbol{y}}{\sigma_z}\right) \leq \alpha.$$

Equivalently, there is sufficient evidence to reject H_0 if

$$\left|\boldsymbol{b_i}^T\boldsymbol{y}\right| \geq \sigma_z\Phi^{-1}\left(1 - \frac{\alpha}{2}\right). \tag{3.7}$$

When there is insufficient evidence to suggest the original coefficient was not zero we hard threshold the coefficient; therefore, comparing (3.5) with (3.7), we deduce that

$$\zeta = \left[\Phi^{-1}\left(1 - \frac{\alpha}{2}\right)\right]^2.$$

This result allows us to select ζ from an appropriate p-value threshold: for example if we choose to reject the null hypothesis if $p \leq 5\%$ then $\zeta = 3.84$. Alternatively we can experimentally tune ζ and then deduce the probability of type I and II errors: the probability of a type I error, i.e. a false positive, is

$$p(\text{Reject } H_0|H_0 \text{ true}) = \alpha = 2 - 2\Phi\left(\sqrt{\zeta}\right), \tag{3.8}$$

and, as shown graphically in Fig. 3.1(b), the probability of a type II error, i.e. a false negative, is

$$p(\text{Do not reject } H_0|H_0 \text{ false}) = \Phi\left(\sqrt{\zeta} - \frac{\boldsymbol{b_i}^T\boldsymbol{x}}{\sigma_z}\right) - \Phi\left(-\sqrt{\zeta} - \frac{\boldsymbol{b_i}^T\boldsymbol{x}}{\sigma_z}\right). \tag{3.9}$$

As depicted in Fig. 3.1(b), there is a large probability of a type II error, when $\boldsymbol{b_i}^T\boldsymbol{x}$ is small; however, these small coefficients carry little power and in most cases $|\boldsymbol{b_i}^T\boldsymbol{x}| >> |\sigma_z\sqrt{\zeta}|$ and the probability of a type II error is small.

3.2.2. Beyond white noise

In the above analysis we made the assumption that the noise was additive white and Gaussian. In this subsection we will slightly extend this noise model to coloured or filtered Gaussian noise, which will be particularly relevant to the deconvolution problem. This coloured denoising problem can be formulated as approximating $\boldsymbol{x}$ from $\boldsymbol{y}$ where

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{W}\boldsymbol{z},$$

$\boldsymbol{W} \in \mathbb{R}^{N\times N}$ is a circulant matrix, since it models circular convolution, and $\boldsymbol{z}$ is once again white Gaussian. The filtered noise vector $\boldsymbol{W}\boldsymbol{z} \sim \mathcal{N}_N(\boldsymbol{0}, \boldsymbol{W}\sigma_z^2\boldsymbol{W}^T)$, therefore

$$p(\boldsymbol{W}\boldsymbol{z} = \dot{\boldsymbol{z}}) = \frac{1}{\sigma_z^N(2\pi)^{N/2}} exp\left(-\frac{\dot{\boldsymbol{z}}^T\left(\boldsymbol{W}\boldsymbol{W}^T\right)^{-1}\dot{\boldsymbol{z}}}{2\sigma_z^2}\right).$$

By similar analysis to the white noise case, the resulting MAP estimator is given by

$$\hat{\boldsymbol{\theta}} = arg\min_{\boldsymbol{\theta}}\left\{(\boldsymbol{y}-\boldsymbol{B}\boldsymbol{\theta})^T\left(\boldsymbol{W}\boldsymbol{W}^T\right)^{-1}(\boldsymbol{y}-\boldsymbol{B}\boldsymbol{\theta}) + \zeta\sigma_z^2\|\boldsymbol{\theta}\|_p^p\right\}.$$

Even if $\boldsymbol{B}$ is unitary, hard thresholding is no longer optimal. This can also be seen by looking at the distribution of $\boldsymbol{y_B}$:

$$\boldsymbol{y_B} = \boldsymbol{B}^T\boldsymbol{y} = \boldsymbol{B}^T(\boldsymbol{x}+\boldsymbol{W}\boldsymbol{z}) \sim \mathcal{N}_d\left(\boldsymbol{B}^T\boldsymbol{x}, \boldsymbol{B}^T\boldsymbol{W}\sigma_z^2\boldsymbol{W}^T\boldsymbol{B}\right).$$

Clearly the covariance matrix, $\boldsymbol{B}^T\boldsymbol{W}\sigma_z^2\boldsymbol{W}^T\boldsymbol{B}$, is no longer diagonal. Despite this, satisfactory results can be obtained by assuming it is diagonal and hard thresholding each individual component depending on its variance, e.g. [18]. Operating on each element individually yields

$$\begin{aligned}\boldsymbol{y_B}[i] = \boldsymbol{b_i}^T\boldsymbol{y} = \boldsymbol{b_i}^T(\boldsymbol{x}+\boldsymbol{W}\boldsymbol{z}) &\sim \mathcal{N}\left(\boldsymbol{b_i}^T\boldsymbol{x}, \boldsymbol{b_i}^T\boldsymbol{W}\sigma_z^2\boldsymbol{W}^T\boldsymbol{b_i}\right)\\ &\sim \mathcal{N}\left(\boldsymbol{b_i}^T\boldsymbol{x}, \sigma_z^2\|\boldsymbol{W}^T\boldsymbol{b_i}\|_2^2\right).\end{aligned}$$

Since $\boldsymbol{W}$ is circulant, it can be diagonalised by the DFT: $\boldsymbol{W} = \boldsymbol{\mathcal{F}} diag(\boldsymbol{w_{\mathcal{F}}})\boldsymbol{\mathcal{F}}^*$, where $\boldsymbol{\mathcal{F}}$ is the DFT matrix. It is often more efficient to exploit this and calculate the variance in the DFT domain:

$$\begin{aligned}\sigma_z^2 \|\boldsymbol{W}^T \boldsymbol{b_i}\|_2^2 &= \sigma_z^2 \|\boldsymbol{\mathcal{F}} diag(\boldsymbol{w_{\mathcal{F}}}^*)\boldsymbol{\mathcal{F}}^* \boldsymbol{b_i}\|_2^2 \\ &= \sigma_z^2 \|diag(\boldsymbol{w_{\mathcal{F}}}^*)\boldsymbol{b_{\mathcal{F}i}}\|_2^2 \\ &= \sigma_z^2 \left(\boldsymbol{w_{\mathcal{F}}}^* \boldsymbol{b_{\mathcal{F}i}}\right)^2,\end{aligned}$$

where $\boldsymbol{b_{\mathcal{F}i}} = \boldsymbol{\mathcal{F}}^* \boldsymbol{b_i}$.

The resulting hard thresholding has a different threshold for each element:

$$\hat{\boldsymbol{\theta}}[i] = S_{\lambda_i,0}\left(\boldsymbol{y_B}[i]\right) \quad \text{where} \quad \lambda_i = \zeta \sigma_z^2 \|\boldsymbol{W}^T \boldsymbol{b_i}\|_2^2,$$

which accounts for the different noise variances of the coefficients.

3.2.3. Transforms

The preceding analysis shows that the minimisation algorithms presented in Chapter 2 can be used to remove noise. In Chapter 2 we also discussed different transforms that could provide sparse representations of images, including wavelets, curvelets and bandlets. A simple denoising strategy is, thus, to hard threshold in the transform domain where the signal is sparse. For example, the case of thresholding in the wavelet domain is shown in Fig. 3.3(c). The results of this approach, however, suffer from blocking artifacts, which are due to the shift variance of the transform. This variance can be removed by using the UDWT, however this is an overcomplete transform so hard thresholding is no longer optimal.

The alternative is to exploit the shift variance using cycle spinning [15][1]. Figure 3.3(d) shows the performance of cycle spinning the DWT, which, given enough shifts, creates the same shift invariance as the UDWT. It is clear that this produces a large visual improvement, due to reduced blocking, and also a significant PSNR improvement.

Figure 3.3(e) shows the results of hard thresholding the curvelet transform. Of course, with this overcomplete frame, hard thresholding is only the first step of an iteration that converges to a local minimum; however, this still produces satisfactory results. Note that Figs. 3.3(c) and 3.3(e) employ

[1]Cycle spinning is the process of averaging multiple approximations, each calculated from a different shift. It has also been shown that improved results can be obtained by cycle spinning with weighted averaging [39].

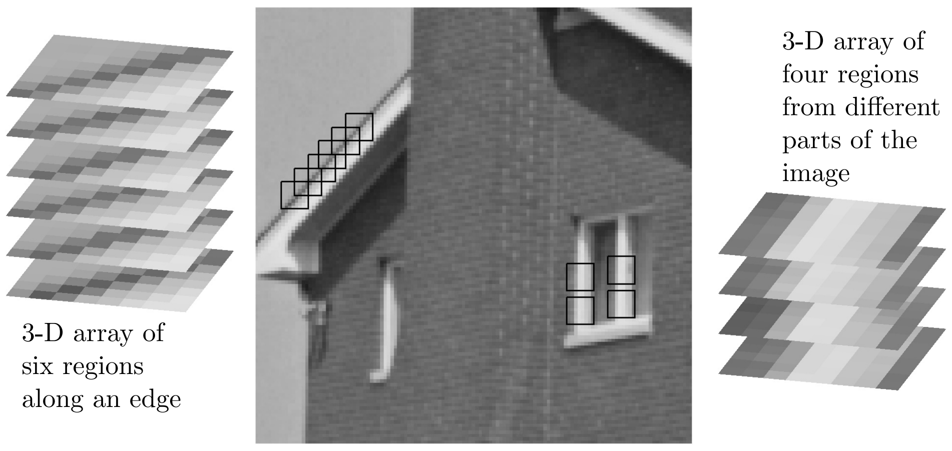


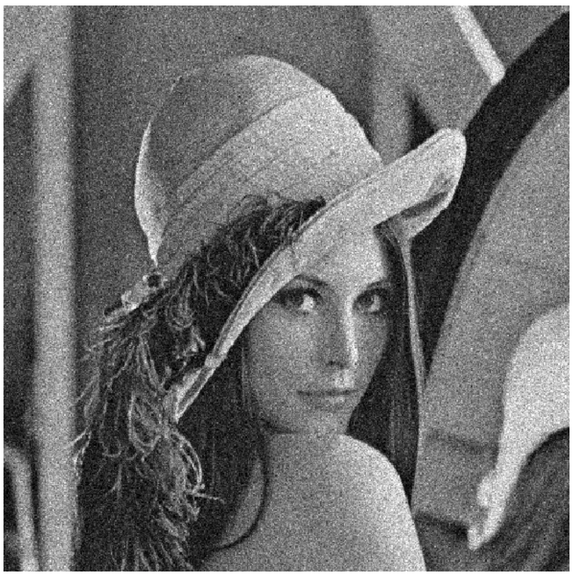
Figure 3.2.: Examples of, similar, non local regions in an image stacked into 3-D arrays.

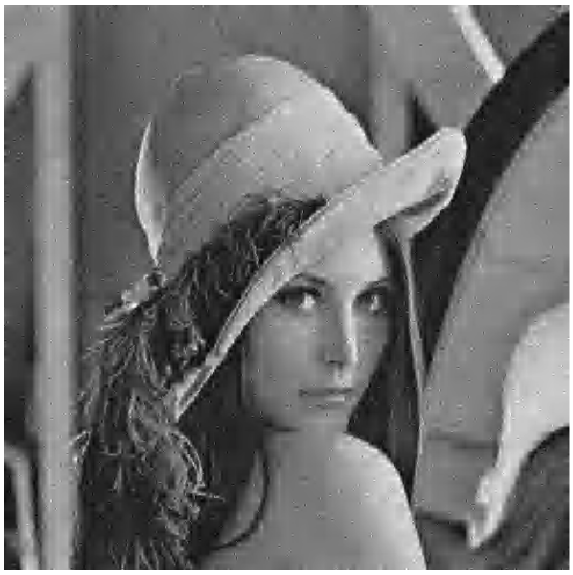
the same thresholding strategy but using two different transforms. One can see the significant performance improvement that is obtained by using the curvelet transform, which provides sparser representations of images.

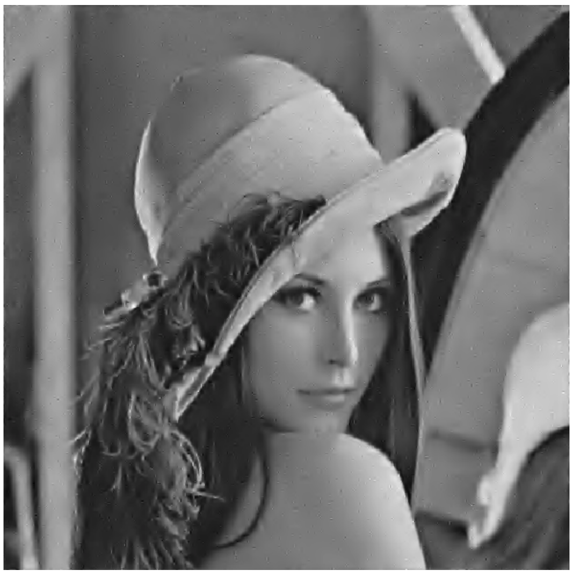
As summarised in [42], there has recently been a trend towards non local approximations [9, 18, 47]. These transforms exploit the similarity between different regions of an image. For example the Block Matching with 3-D Collaborative Filtering (BM3D) algorithm of Dabov et al. [17] combines similar regions by stacking them into a 3-D array, computing a 3-D transform and thresholding the result. This additional dimension of regularity provides an even sparser representation and very impressive results. This idea is highlighted in Fig. 3.2: six similar regions along an edge and four similar non local regions have been stacked into 3-D arrays on the left and right respectively. The similarity of these regions will be captured by applying a transform, containing a constant basis function, along this new dimension.

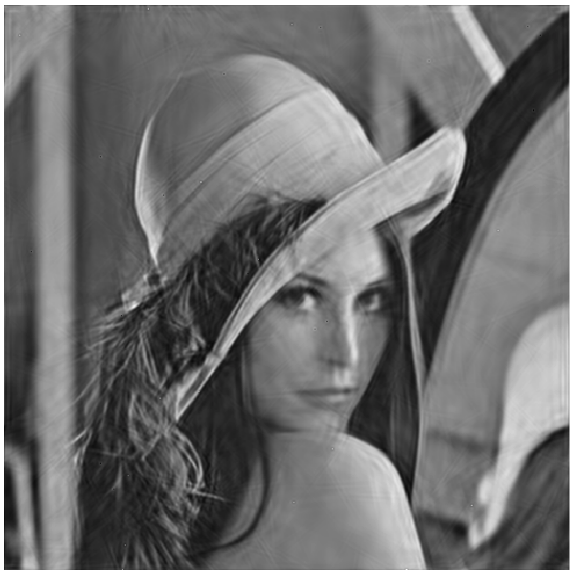
Another recent trend in image denoising is to adaptively select a basis. We have already seen this with non-linear approximations such as bandlets, but this idea can be extended to learning the basis from the image. For example the KSVD method [1] calculates a sparse dictionary from a training set and has successfully been used for image denoising [29]. Furthermore, Dabov et al. [19] extended their BM3D algorithm to adaptively choose a basis for each 3-D array using a principle component analysis (PCA). An example of this algorithm, which is basically regarded as the leading denoising algorithm on natural images, is shown in Fig. 3.3(f). The advantage of adaptive basis selection algorithms is that they search for the basis that leads to the sparsest solution, for each individual signal. This provides very good results over a wide range of images; however since the model can

(a) Lena image.


(b) Noisy image. PSNR=20.18dB.


(c) Hard thresholding in an orthogonal wavelet basis. PSNR=27.34dB.


(d) Cycle spinning using hard thresholding in an orthogonal wavelet basis. PSNR=30.25dB.


(e) Hard thresholding the curvelet transform. PSNR=30.07dB.

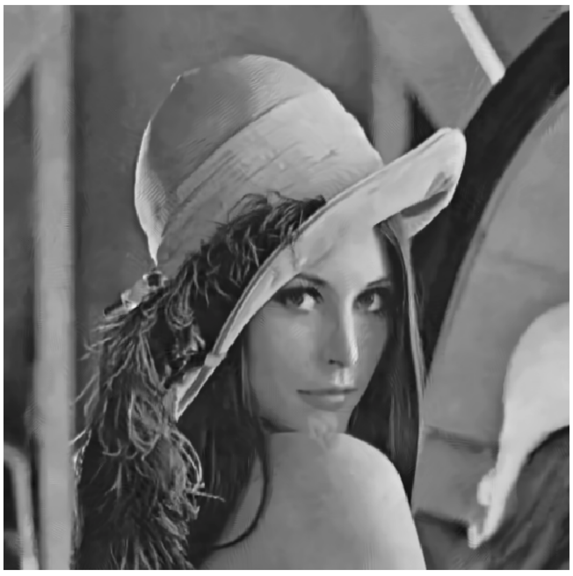

(f) Restored using the BM3D-SAPCA algorithm. PSNR=32.22dB.

Figure 3.3.: An example of the performance of various denoising algorithms on the Lena image with a noise standard deviation of 25.

only be generated from the measured degraded image, it may become unreliable if the degradation is high. It is also possible that, for images that can be well approximated with a strong prior, some of this prior knowledge will not be exploited. For example, depth images are piecewise smooth and an algorithm that exploits this would be better suited to this class of images. Depth images will be introduced in Section 3.4, which looks at interpolation, because there are some interesting applications in this area.

3.3. Deconvolution

Blur is a common degradation that is normally due to motion, atmospheric effects and camera lenses. Deblurring is a deconvolution problem that, once again, can be modelled by

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{z}, \tag{3.10}$$

if $\boldsymbol{H} \in \mathbb{R}^{N\times N}$ is the circular convolution matrix constructed from $\boldsymbol{h}$, the impulse response of the system. Under this construction (3.10) is equivalent to

$$\boldsymbol{y} = \boldsymbol{h} \circledast \boldsymbol{x} + \boldsymbol{z}.$$

To consider the inversion of $\boldsymbol{H}$, in the general case, let $\boldsymbol{H} = \boldsymbol{U} diag(\boldsymbol{s})\boldsymbol{V}^T$ be the singular value decomposition (SVD) of $\boldsymbol{H}$, where $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{N\times N}$ are orthogonal matrices and $\boldsymbol{s} \in \mathbb{R}^N$ is the vector of singular values. Using this notation the pure inverse solution can be written as

$$\begin{aligned} \boldsymbol{H}^{-1}\boldsymbol{y} &= \boldsymbol{V} diag(\boldsymbol{s})^{-1}\boldsymbol{U}^T\boldsymbol{y} \\ &= \boldsymbol{x} + \boldsymbol{V} diag(\boldsymbol{s})^{-1}\boldsymbol{U}^T\boldsymbol{z} \\ &= \boldsymbol{x} + \sum_{i=1}^{N} \frac{\boldsymbol{v_i}\boldsymbol{u_i}^T\boldsymbol{z}}{\boldsymbol{s}[i]}, \end{aligned} \tag{3.11}$$

where $\boldsymbol{u_i}$ and $\boldsymbol{v_i}$ are the i-th columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ respectively. It is clear from (3.11) that, if $\boldsymbol{H}$ has small singular values, the pure inverse solution will be ineffective, since it will be dominated by the noise term. Just like the rest of this thesis, we will use regularisation to obtain a good solution. Assuming $\boldsymbol{z}$ is white and Gaussian, and the original signal is sparse in a proper domain, the same

MAP analysis as the denoising case results in the estimate

$$\hat{\boldsymbol{x}} = \boldsymbol{B}\hat{\boldsymbol{\theta}},$$

where

$$\hat{\boldsymbol{\theta}} = arg\min_{\boldsymbol{\theta}} \left\{ \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{B}\boldsymbol{\theta}\|_2^2 + \lambda \|\boldsymbol{\theta}\|_p^p \right\}. \tag{3.12}$$

The minimisation problems (3.3) and (3.12) differ in just the presence of the matrix $\boldsymbol{H}$, however this simple difference prevents closed-form solutions. This is because the product $\boldsymbol{HB}$ will almost certainly not be unitary, even if $\boldsymbol{B}$ is. In the previous chapter we saw that iterative soft, and hard, thresholding could be used to obtain sparse solutions in the generally, non-unitary, case. Also, recall that FISTA is a fast iterative soft thresholding algorithm, with improved convergence rate. Figures 3.5(c) and 3.5(d) show the result of restoring a blurred noisy image with 100 iterations of FISTA using the wavelet and curvelet transforms respectively.

We have seen how the SVD diagonalises $\boldsymbol{H}$. In fact since $\boldsymbol{H}$ is a circular matrix, it is diagonalised by the discrete Fourier transform (DFT) matrix, $\boldsymbol{\mathcal{F}}$:

$$\boldsymbol{H} = \boldsymbol{\mathcal{F}} diag(\boldsymbol{h}_{\boldsymbol{\mathcal{F}}}) \boldsymbol{\mathcal{F}}^*, \tag{3.13}$$

where $\boldsymbol{h}_{\boldsymbol{\mathcal{F}}} = \boldsymbol{\mathcal{F}}^* \boldsymbol{h}$. Therefore, by choosing $\boldsymbol{B} = \boldsymbol{\mathcal{F}}$, we no longer sparsely represent the signal (unless the signal is sparse in the DFT domain) but we do diagonalise the filtering process. In this case, instead of a sparsity promoting prior we would like a different prior that will allow us to obtain a closed-form solution. Recall the minimum energy Tikhonov regularisation has the following closed-form solution:

$$\begin{aligned} \hat{\boldsymbol{\theta}} &= arg\min_{\boldsymbol{\theta}} \left\{ \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{\mathcal{F}}\boldsymbol{\theta}\|_2^2 + \zeta\sigma_z^2 \|\boldsymbol{\theta}\|_2^2 \right\}, \\ &= \left(\boldsymbol{\mathcal{F}}^* \boldsymbol{H}^* \boldsymbol{H} \boldsymbol{\mathcal{F}} + \zeta\sigma_z^2 \boldsymbol{I_N} \right)^{-1} \boldsymbol{\mathcal{F}}^* \boldsymbol{H}^* \boldsymbol{y}. \end{aligned} \tag{3.14}$$

Substituting (3.13) into (3.14) yields

$$\begin{aligned} \hat{\boldsymbol{\theta}} &= \left(\boldsymbol{\mathcal{F}}^* \boldsymbol{\mathcal{F}} diag(\boldsymbol{h}_{\boldsymbol{\mathcal{F}}})^* \boldsymbol{\mathcal{F}}^* \boldsymbol{\mathcal{F}} diag(\boldsymbol{h}_{\boldsymbol{\mathcal{F}}}) \boldsymbol{\mathcal{F}}^* \boldsymbol{\mathcal{F}} + \zeta\sigma_z^2 \boldsymbol{I_N} \right)^{-1} \boldsymbol{\mathcal{F}}^* \boldsymbol{\mathcal{F}} diag(\boldsymbol{h}_{\boldsymbol{\mathcal{F}}})^* \boldsymbol{\mathcal{F}}^* \boldsymbol{y} \\ &= \left(diag(\boldsymbol{h}_{\boldsymbol{\mathcal{F}}})^* diag(\boldsymbol{h}_{\boldsymbol{\mathcal{F}}}) + \zeta\sigma_z^2 \boldsymbol{I_N} \right)^{-1} diag(\boldsymbol{h}_{\boldsymbol{\mathcal{F}}})^* \boldsymbol{y}_{\boldsymbol{\mathcal{F}}}, \end{aligned}$$

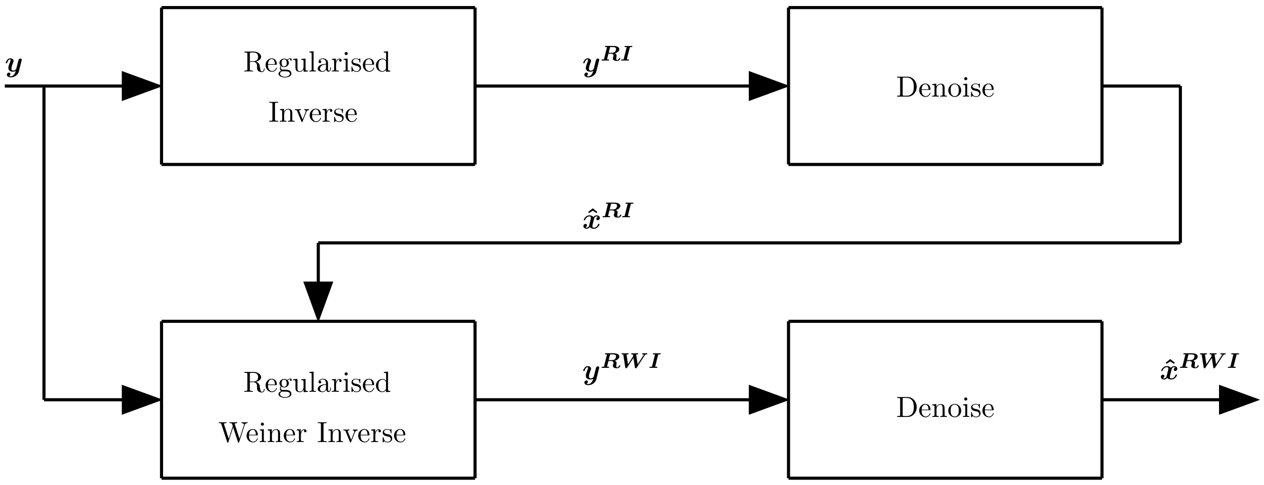

$\boldsymbol{y}$
Regularised Inverse
$\boldsymbol{y}^{\boldsymbol{RI}}$
Denoise
$\hat{\boldsymbol{x}}^{\boldsymbol{RI}}$
Regularised Weiner Inverse
$\boldsymbol{y}^{\boldsymbol{RWI}}$
Denoise
$\hat{\boldsymbol{x}}^{\boldsymbol{RWI}}$

Figure 3.4.: The two-step deconvolution approach using a regularised inverse and a regularised Wiener inverse.

so that each element can be calculated independently:

$$\hat{\boldsymbol{\theta}}[i] = \frac{\boldsymbol{h}_{\mathcal{F}}{}^*[i]}{|\boldsymbol{h}_{\mathcal{F}}[i]|^2 + \zeta\sigma_z^2}\boldsymbol{y}_{\mathcal{F}}[i]. \tag{3.15}$$

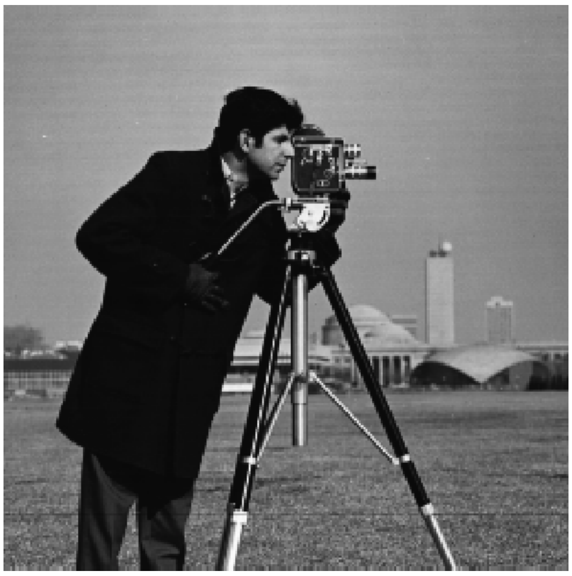
This regularised inverse is just the pure inverse, with the additional $\zeta\sigma_z^2$ term to prevent the denominator becoming too small. Traditionally, ζ in (3.15) is set to optimise the PSNR of the solution $\hat{\boldsymbol{x}} = \boldsymbol{\mathcal{F}}\hat{\boldsymbol{\theta}}$, which heavily suppresses the noise. Consequently this solution is not as sharp as it could be, which, in images, degrades a large amount of the visual information occurring around discontinuities. More recently, (3.15) has been used with a smaller regularisation parameter, ζ, that maintains a sharper image and suppresses less noise. To obtain a satisfactory result a post processing denoising is applied to remove the unsuppressed coloured noise term, which in the DFT domain is given by

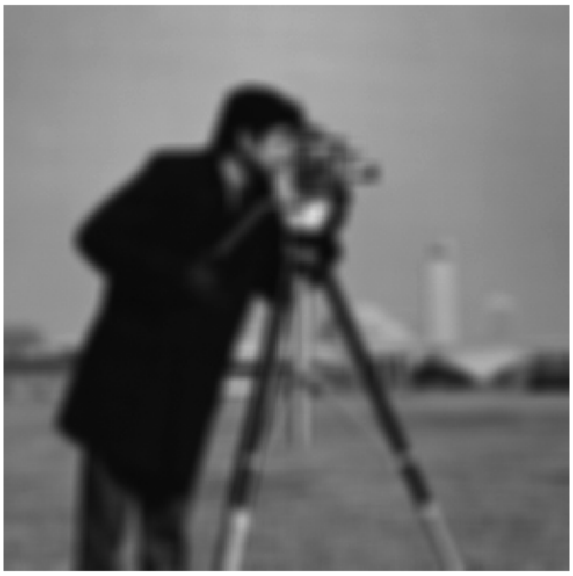
$$\frac{\boldsymbol{h}_{\mathcal{F}}{}^*[i]}{|\boldsymbol{h}_{\mathcal{F}}[i]|^2 + \zeta\sigma_z^2}\boldsymbol{z}_{\mathcal{F}}[i].$$

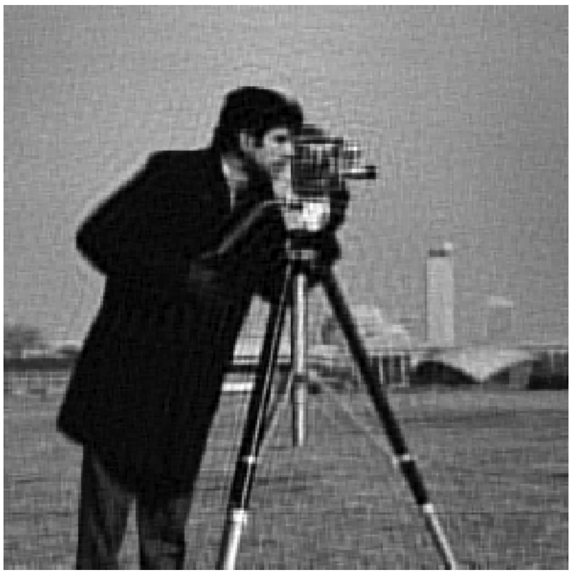
Note that this noise fits the filtered noise model that we studied in the previous section.

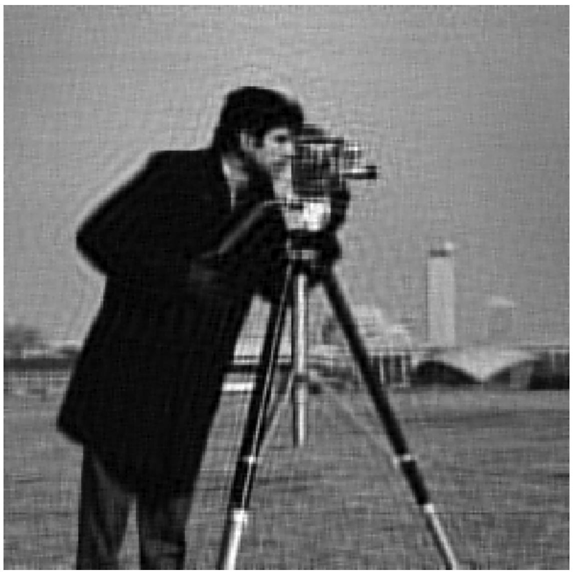
Neelamani et al. [51] use this technique in their Fourier Wavelet Regularised Deconvolution (ForWaRD) algorithm, which denoises by thresholding in the wavelet domain. They were able to derive the optimum thresholding parameters in both the Fourier and wavelet domains and, with a non-iterative algorithm, achieve much better results than the simple regularised filter.

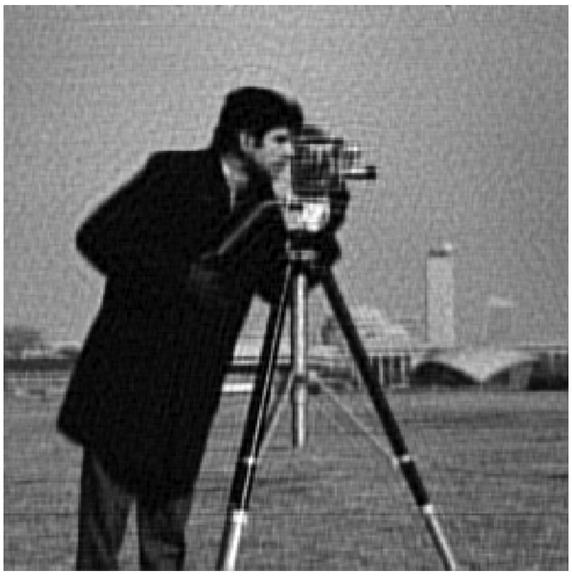
This idea can be further extended to the two-step process shown in Fig. 3.4. Here the first step is as previously described but the output is used to provide an estimate of the energy spectrum of


(a) Cameraman image.


(b) Noisy blurred image. PSNR=21.37.


(c) FISTA in the wavelet domain. PSNR=26.82dB, SSIM=0.7376.


(d) FISTA in the curvelet domain. PSNR=26.83dB, SSIM=0.7656.


(e) Wiener filter with oracle energy spectrum. PSNR=27.19, SSIM=0.7235.

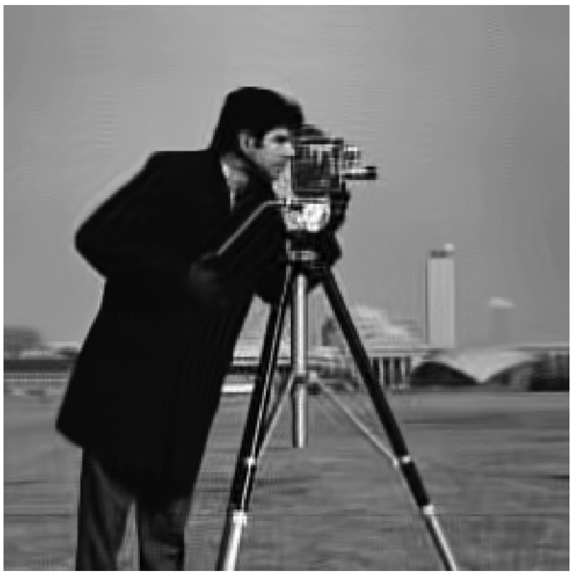

(f) Two-step approach with BM3D denoising. PSNR=28.59dB, SSIM=0.8598.

Figure 3.5.: An example of the performance of various deconvolution algorithms on the cameraman image.

the original signal. This energy spectrum is used to construct a more accurate regularised Wiener inverse in the second step. Recall the Wiener filter is the optimum linear filter in terms of PSNR:

$$\hat{\boldsymbol{\theta}}[i] = \frac{|\boldsymbol{x}_{\mathcal{F}}[n]|^2 \boldsymbol{h}_{\mathcal{F}}{}^*[i]}{|\boldsymbol{h}_{\mathcal{F}}[i]|^2 \cdot |\boldsymbol{x}_{\mathcal{F}}[n]|^2 + \sigma_z^2} \boldsymbol{y}_{\mathcal{F}}[i]. \tag{3.16}$$

The Wiener filter produces the same over-smoothing effect as the traditional application of (3.15), so the two-step process tunes ζ in the regularised Wiener filter,

$$\hat{\boldsymbol{\theta}}[i] = \frac{|\hat{\boldsymbol{x}}_{\mathcal{F}}^{\boldsymbol{RI}}[n]|^2 \boldsymbol{h}_{\mathcal{F}}{}^*[i]}{|\boldsymbol{h}_{\mathcal{F}}[i]|^2 \cdot |\hat{\boldsymbol{x}}_{\mathcal{F}}^{\boldsymbol{RI}}[n]|^2 + \zeta\sigma_z^2} \boldsymbol{y}_{\mathcal{F}}[i], \tag{3.17}$$

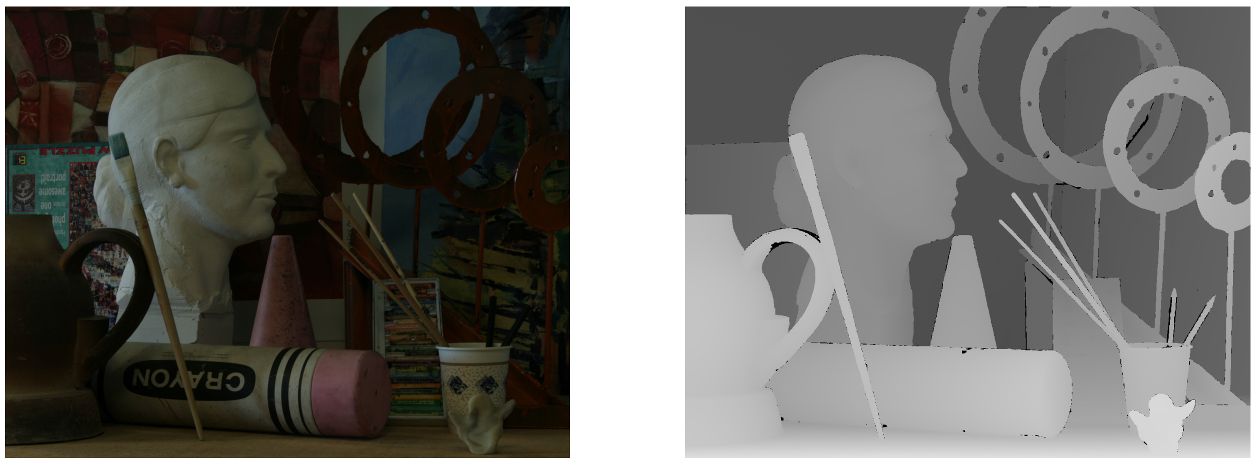
to, once again, produce a sharper, and consequently noisier, result, which is then denoised. Note that, since $\boldsymbol{x}$ is unknown, it has been replaced, in (3.17), by the approximation from the first step, $\hat{\boldsymbol{x}}$.

Figures 3.5(e) and 3.5(f) show a deconvolution result for the Oracle Wiener filter and two-step process with BM3D denoising. The Oracle Wiener filter, unrealistically, uses the ideal energy spectrum of the original signal. A more realistic Wiener filter using a non ideal energy spectrum would produce significantly poorer result; however, the two-step algorithm still out performs the Oracle Wiener filter even though it operates without oracle knowledge.

3.4. Interpolation

In many digital imaging applications we have a set of samples and wish, to interpolate, to obtain an estimate of the signal at a new unknown data point. For example the image upscale problem can be viewed as interpolating between a regularly sampled grid of data points, some depth sensors obtain samples at irregular locations, and, as we will see in the next section, the multi-view image super resolution problem produces an irregularly sampled grid of blurred data.

Recently depth image problems have received a large amount of research interest due to their increased use in applications. As shown in Fig. 3.6, each pixel of a depth image corresponds to the distance from the camera image plane to the object, instead of the luminance value in a traditional image. A depth image is thus inherently piecewise smooth and therefore ideally suited to algorithms that can sparsely represent this class of images. The interpolation problem is particularly relevant to these images because many depth sensors, e.g. the Kinect sensor, only calculate the depth at an



(a) Luminance image.

(b) Corresponding depth image.

Figure 3.6.: A luminance image and its corresponding depth image.

irregularly sampled grid of locations. Note that all depth images used in this thesis come from the Middlebury stereo datasets [56–58].

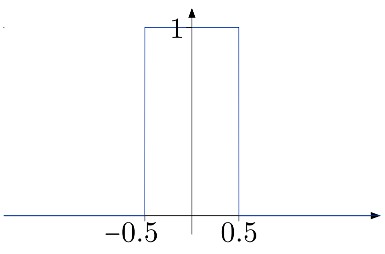
We can model the interpolation problem as a degradation if we assume that $\boldsymbol{x} \in \mathbb{R}^N$ is the desired image containing all N samples and $\boldsymbol{y} \in \mathbb{R}^{N_v}$ is the measured signal containing $N_v < N$ visible samples. Let $\boldsymbol{H} \in \mathbb{R}^{N_v \times N}$ be the $N \times N$ identity with the $N - N_v$ rows that correspond to an unknown pixel removed, so that

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{z}$$

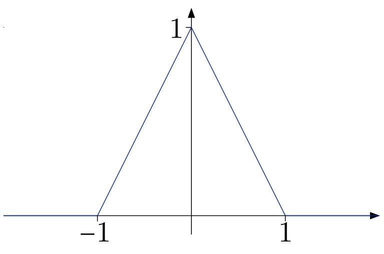
models the degradation. Here $\boldsymbol{z} \in \mathbb{R}^{N_v}$ can either be $\mathbf{0}$ or, in the case of noisy interpolation, an additional white Gaussian noise vector.

Although the following algorithms are very general, they are designed for cases where the missing pixels are randomly dispersed. In cases where the missing pixels occur in large blocks, such as inpainting a damaged region of a photo, there are specific, more tailored, algorithms that rely on copying regions of the image into the unknown regions. These algorithms will not be covered in this thesis and we refer the interested reader to the image inpainting literature.

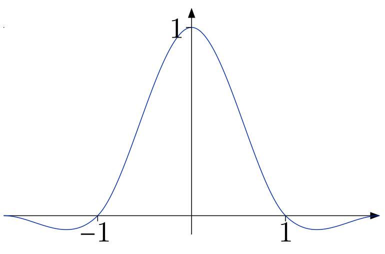
Traditional interpolation algorithms perform a linear filtering on the known samples. Figures 3.7(d) - 3.7(f) show examples of convolving the same 1-D regularly sampled signal with the constant, linear and cubic interpolating functions shown in Figs. 3.7(a) - 3.7(c). Convolving the samples with the constant interpolation spline is equivalent to nearest neighbour interpolation and connecting the samples by straight lines is equivalent to convolving with the linear spline. The Catmull-Rom



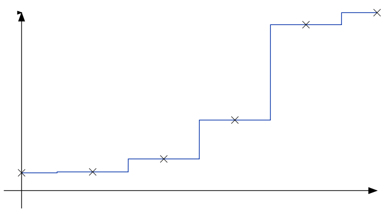
(a) A constant interpolation spline.



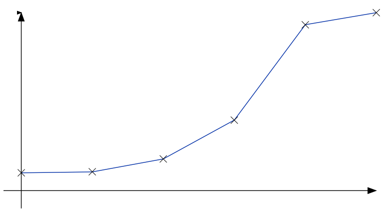
(b) A linear interpolation spline.



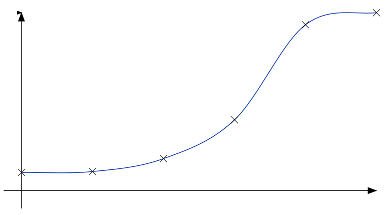
(c) A cubic interpolation spline.



(d) Interpolation of the discrete points using the constant interpolation spline.



(e) Interpolation of the discrete points using the linear interpolation spline.



(f) Interpolation of the discrete points using the cubic interpolation spline.

Figure 3.7.: Interpolation of the same data points by a linear convolution with three different splines.

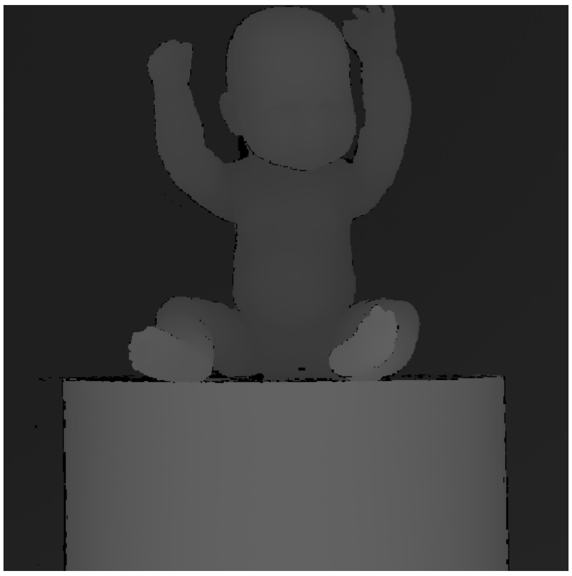
cubic[2], shown in Fig. 3.7(c), is one of many possible cubic splines, and is generally regarded as the best for interpolation [24]. As shown in Fig. 3.7(f), it produces visually pleasing interpolation through the samples (not all splines produce an interpolation that goes through the data points). It is very easy to extend these interpolation techniques to 2-D, with bi-linear and bi-cubic interpolation being the most commonly used techniques on images. Note that, although rarely used, it is possible to construct quadratic splines that are suitable for interpolation [25].

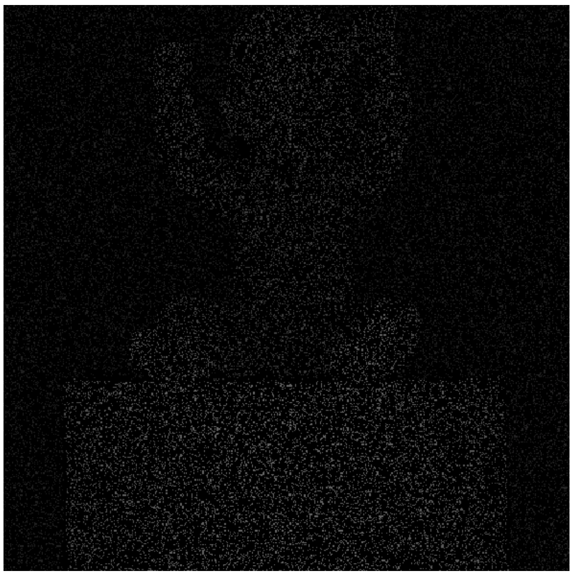
Figure 3.8(c) shows the bilinear interpolation of an irregularly sampled depth image. For this example, bilinear interpolation is significantly superior to nearest neighbour and slightly better than bicubic.

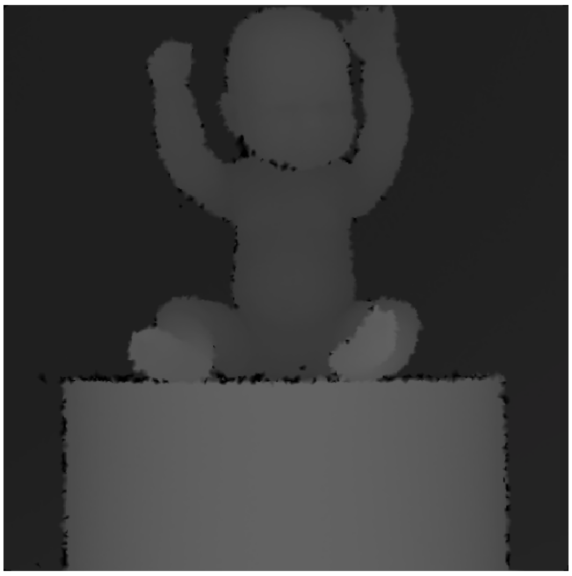
Despite there prevalence, linear interpolation algorithms suffer from their inability to adapt to the signal, which causes inaccuracies around image features such as edges. Takeda et al. [68] developed a data adaptive kernel regression technique that produces very effective interpolation. Around each desired point, kernel regression calculates an n-term Taylor expansion using local samples. The effect of each local sample is weighted with a kernel, such as a Gaussian function, so that nearby samples have more effect. Traditional kernel regression uses isotropic kernels across the whole

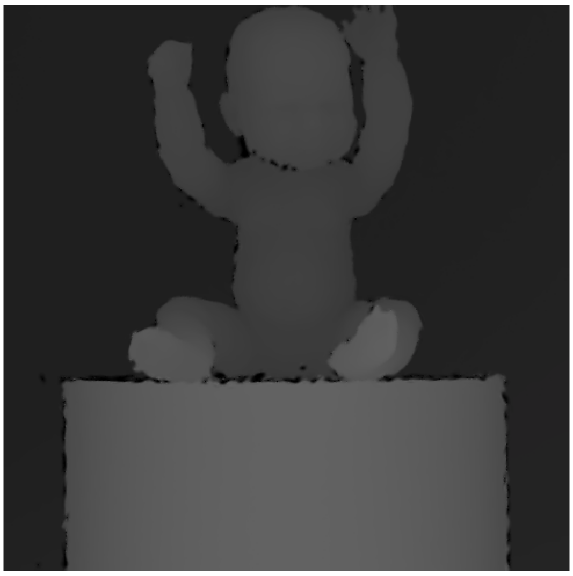
[2]The Catmull-Rom cubic is given by

$$\psi(x) = \begin{cases} 1.5|x|^3 - 2.5|x|^2 + 1 & \text{if } |x| \le 1 \\ -0.5|x|^3 + 2.5|x|^2 - 4|x| + 2 & \text{if } 1 < |x| \le 2 \\ 0 & \text{otherwise.} \end{cases}$$


(a) Cameraman image.


(b) Irregularly sampled image. 85% of the pixels randomly removed.


(c) Bilinear. PSNR=33.03dB.


(d) Adaptive kernel regression. PSNR=33.97dB.

(e) Close up of bilinear.

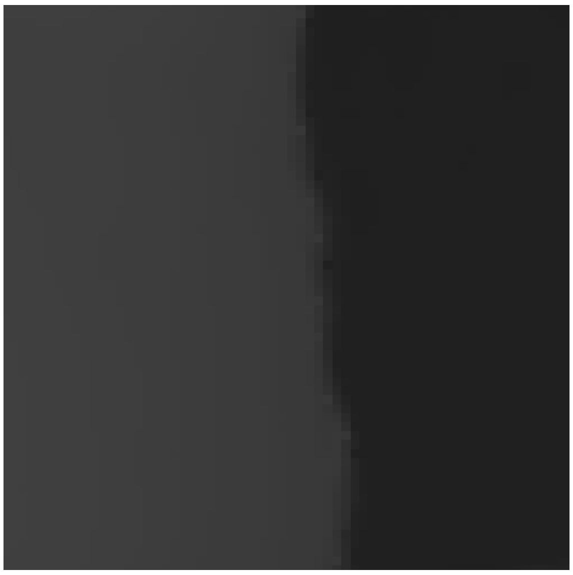

(f) Close up of adaptive kernel regression.

Figure 3.8.: An example of interpolating the baby depth image after 85% of the pixels have been randomly removed, using various interpolation algorithms.

image, but the data adaptive kernels of Takeda et al. are elongated along smooth regions of the image to enhance performance. This kernel regression algorithm can be used for many restoration problems including interpolation. When interpolating, the kernel size is dependent not only on the structure of the image but also on the density of visible samples. An example of the performance of interpolating with adaptive kernel regression is shown in Fig. 3.8(d). Figures 3.8(e) and 3.8(f) show closeups of the bilinear and adaptive kernel results. Along this edge, it is clear how adaptive interpolation algorithms can better fit the data.

Since the problem has been modelled as a restoration problem, with the usual degradation model, another way to obtain data adaptive interpolation is to use the same sparse promoting algorithms that aim to solve

$$\hat{\boldsymbol{\theta}} = arg \min_{\boldsymbol{\theta}} \left\{ \|\boldsymbol{y} - \boldsymbol{HB\theta}\|_2^2 + \lambda \|\boldsymbol{\theta}\|_p^p \right\}.$$

Later in this thesis we will construct an interpolation algorithm by employing this principle.

3.5. Super resolution

Multi-view image super resolution is the problem of combining multiple low resolution images of the same scene into one high resolution image. It is currently receiving a large amount of research interest as we reach the physical resolution limits of image sensors. In order to perform super resolution we need to accurately model the image acquisition process. Figure 3.9 shows a possible model that consists of a lens, that blurs the image, and an image sensor, that samples the blurred image. The problem is to combine a number of these digital images, taken from different viewpoints, into one high resolution image.

This problem has multiple steps, which are summarised in Fig. 3.10. The first step is to take the set of low resolution images and register them onto a high resolution grid at sub-pixel accuracy. Then the blurred irregularly sampled data can be interpolated to produce a noisy blurred high resolution image. Finally this image is deconvolved to remove the blur generated by the camera lens.

Multi-view image super resolution is thus a three step process: registration, interpolation and deconvolution. Of these three problems, interpolation and deconvolution have already been covered in detail and the following subsection will briefly summarise image registration.

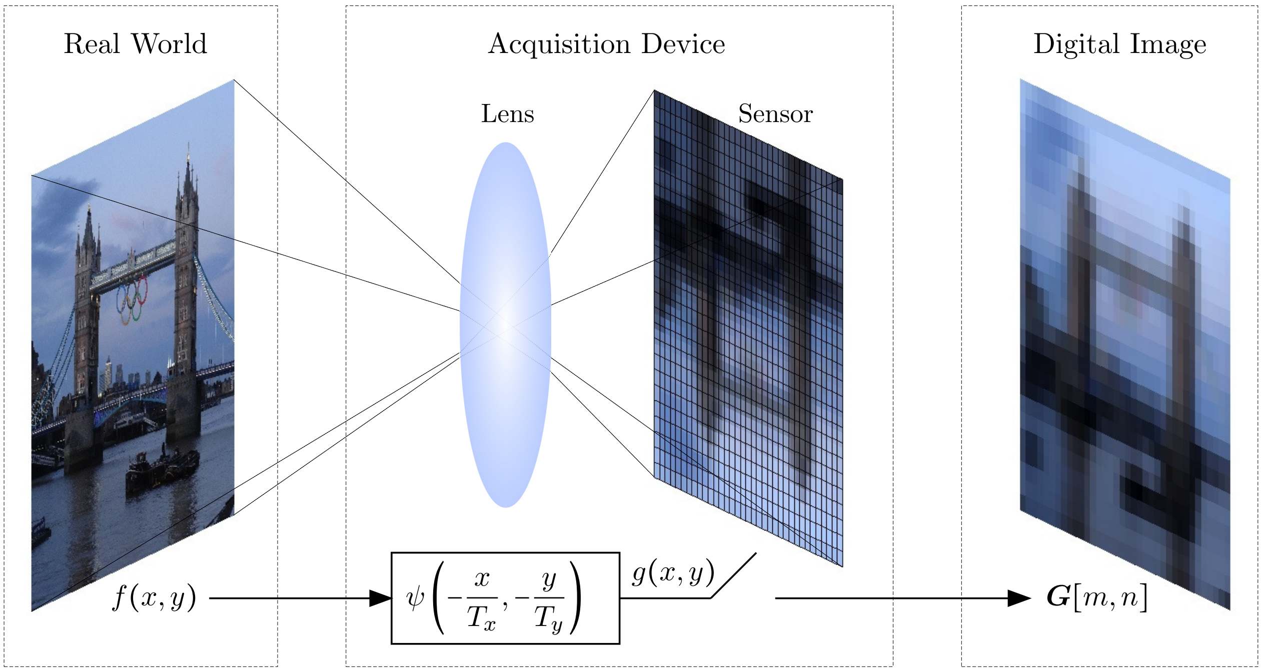


Figure 3.9.: An image acquisition model consisting of a lens, which blurs the image, and an image sensor, which samples the blurred image.

3.5.1. Image registration

Given a set of low resolution images we wish to calculate the transformations between each image and one of the images, which we will take to be the reference. A simple registration strategy is to match image key points, using for example a scale invariant feature transform (SIFT) detector, and average the translations between these points. The key points can only be calculated at pixel accuracy, however averaging many estimates produces a reasonable registration at sub-pixel accuracy. More accurate registration techniques, using finite rate of innovation (FRI) principles [72], have been presented by Baboulaz et al. [3]. Their techniques use the acquisition model depicted in Fig. 3.9 yielding

$$g(x,y) = f(x,y) \star \psi\left(-\frac{x}{T_x},-\frac{y}{T_y}\right),$$

and

$$\begin{aligned}\boldsymbol{G}[m,n] = g(m,n) &= \int\int f(x,y)\psi(x/T_x - m, y/T_y - n)dxdy \\ &= \langle f(x,y), \psi(x/T_x - m, y/T_y - n)\rangle . \end{aligned} \tag{3.18}$$

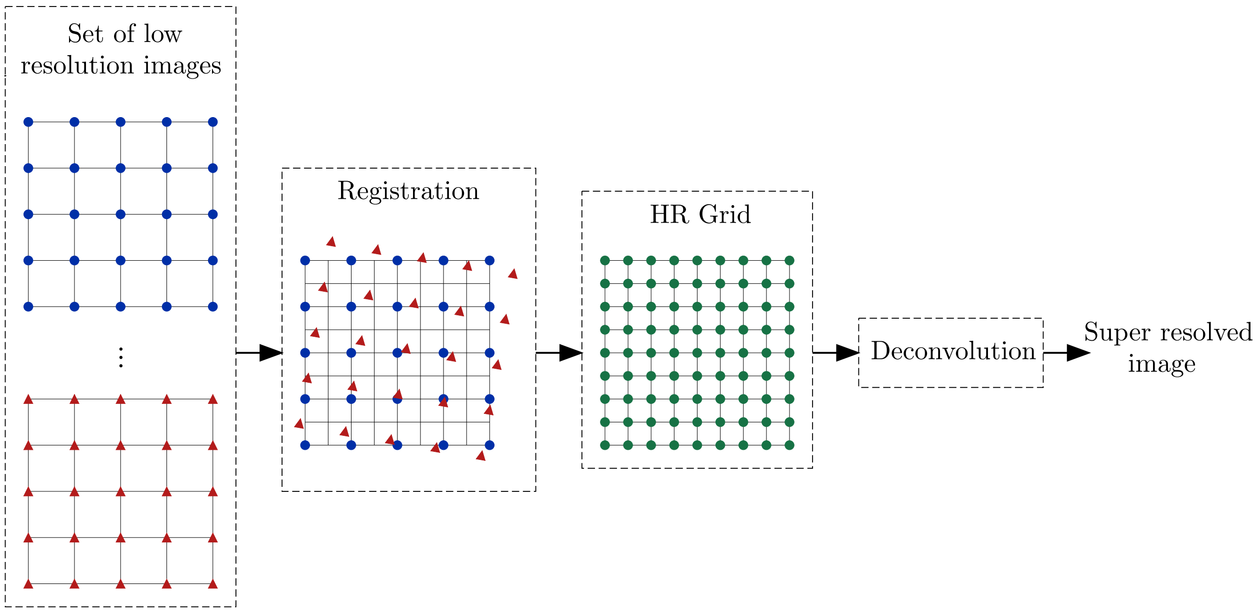


Figure 3.10.: The steps necessary to perform image super resolution. First a set of low resolution images are registered, revealing the location of each sample. Then, these samples are interpolated to produce one blurred high resolution image. Finally, the blur is suppressed with a deconvolution algorithm.

Here T_x and T_y are the sampling periods in the x and y directions, which for simplicity we will assume to be one. They then calculate the continuous geometric moments $\boldsymbol{M}[p,q]$ of the signal $f(x,y)$ from the samples $\boldsymbol{F}[m,n]$, by modelling the sampling kernel as a B-spline, $\psi(x,y) = \beta_\rho(x,y)$. This is possible because a B-spline, β_ρ, of order ρ, and its shifts, can reproduce polynomials up to degree $\rho - 1$:

$$\sum_m \sum_n \boldsymbol{C_{p,q}}[m,n]\beta_\rho(x-m, y-n) = x^p y^q, \quad \text{where} \quad p,q < \rho, \tag{3.19}$$

and $\boldsymbol{C_{p,q}}[m,n]$ are the polynomial reproducing coefficients. The continuous geometric moments are defined as

$$\boldsymbol{M}[p,q] = \int\int f(x,y)x^p y^q dxdy. \tag{3.20}$$

Substituting (3.19) into (3.20) yields

$$\begin{aligned}\boldsymbol{M}[p,q] &= \int\int f(x,y)\sum_m \sum_n \boldsymbol{C_{p,q}}[m,n]\beta_\rho(x-m,y-n)dxdy\\ &= \sum_m \sum_n \boldsymbol{C_{p,q}}[m,n]\int\int f(x,y)\beta_\rho(x-m,y-n)dxdy\\ &= \sum_m \sum_n \boldsymbol{C_{p,q}}[m,n]\boldsymbol{G}[m,n],\end{aligned}$$

where $\boldsymbol{G}[m,n] = \int\int f(x,y)\beta_\rho(x-m,y-n)dxdy$ are the digital image samples given in (3.18), since $\psi(x,y) = \beta_\rho(x,y)$.

To relate this result to image registration, suppose there are two signals, $f_1(x,y)$ and $f_2(x,y)$, that are related by an affine transformation; i.e., a rotation θ, scale $[X_{scale}, Y_{scale}]^T$, sheer $[X_{shear}, Y_{shear}]^T$ and translation $[t_x, t_y]^T$:

$$\begin{bmatrix} \grave{x} \\ \grave{y} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X_{scale} & X_{shear} \\ Y_{scale} & Y_{shear} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix},$$

where x and y are the arguments for the first signal, f_1, and $\grave{x}$ and $\grave{y}$ are the arguments for the second, f_2. Baboulaz et al. [3] showed, using the results of Heikkilä [40] and Sprinzak et al. [64], that the exact parameters of this affine transformation can be found from the continuous geometric moments of f_1 and f_2. Therefore in theory, we can calculate the exact continuous geometric moments of each signal from its discrete samples and use these moments to calculate the exact affine transformation between any of the signals. In practice this technique produces state of the art registration, even when the sampling kernel and transformation are only approximately a B-spline and affine respectively.

Figure 3.11 shows the various images that are obtained along the different stages of a super resolution simulation. In this example the above FRI based registration, bicubic interpolation and two-step deconvolution using BM3D denoising were used.

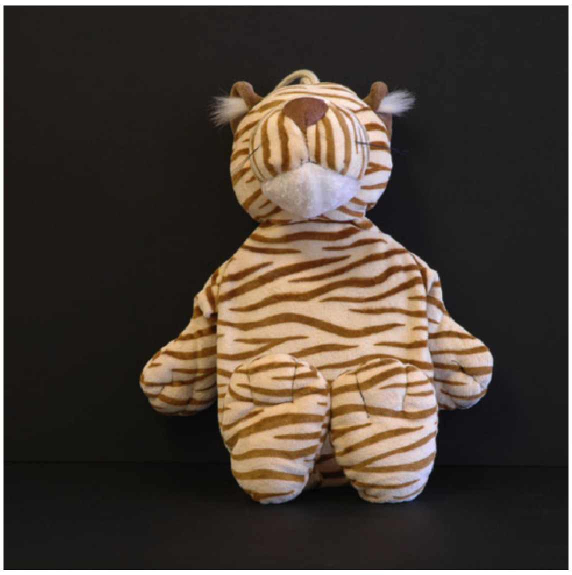
3.6. Summary

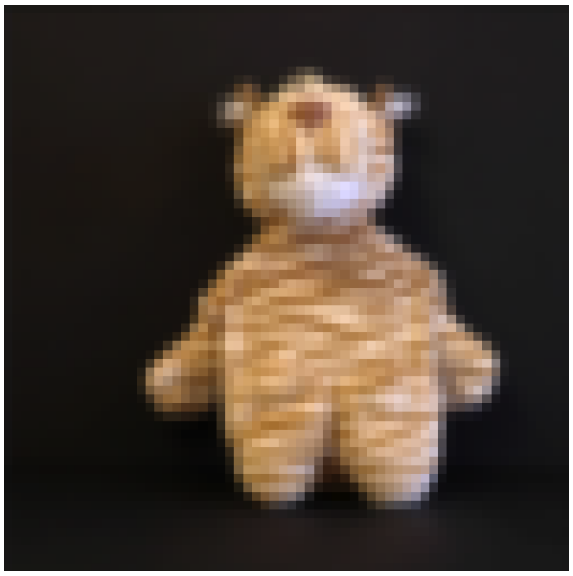
In this chapter we have seen how sparse promoting algorithms can be used to perform image restoration. We have looked at a wide range of problems, from traditional denoising to the more complex multi-step super resolution problem, and given a brief idea of the performance of various algorithms. Throughout all the problems there have been two key ingredients to successful restoration:

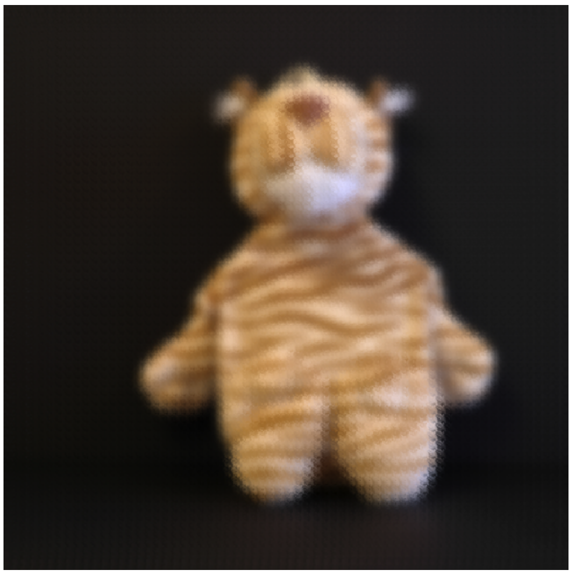
1. The accurate modelling of the degradation.

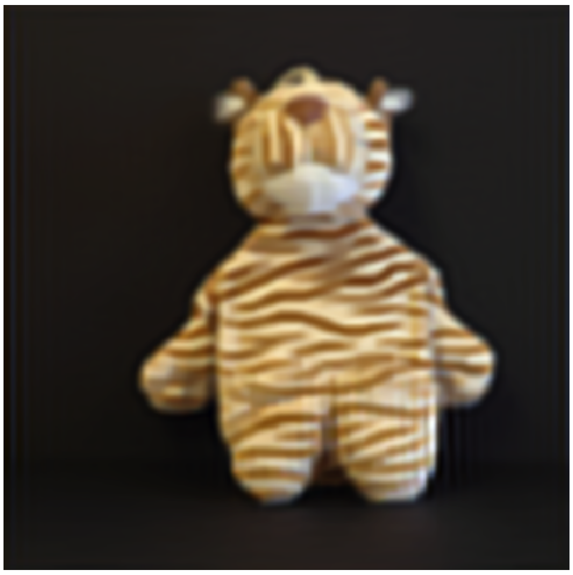
2. The ability to sparsely represent the original signal with a shift invariant transform.

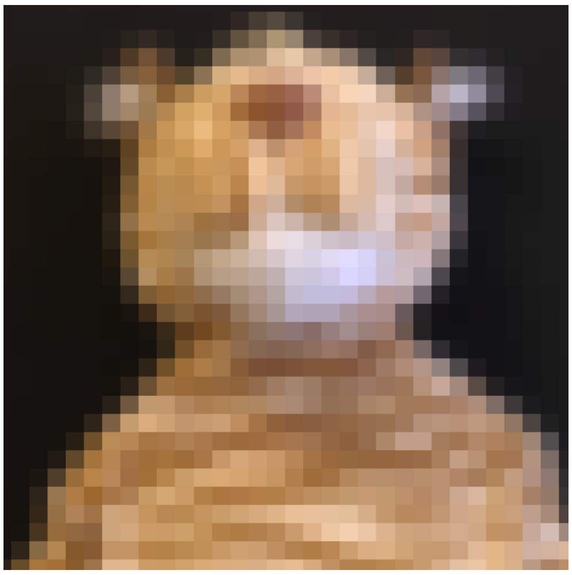
In the next chapter we will propose a new transform that can sparsely represent piecewise polynomial signals and in the following chapter we will use this transform to develop state of the art restoration


(a) Tiger image.


(b) One of 64 low resolution images (64×64).


(c) Bicubic interpolation of the irregularly sampled image (512×512). PSNR=21.60dB.


(d) Final super resolved image after BM3D deconvolution of the interpolated image (512×512). PSNR=23.87dB.


(e) Close up of a low resolution image.

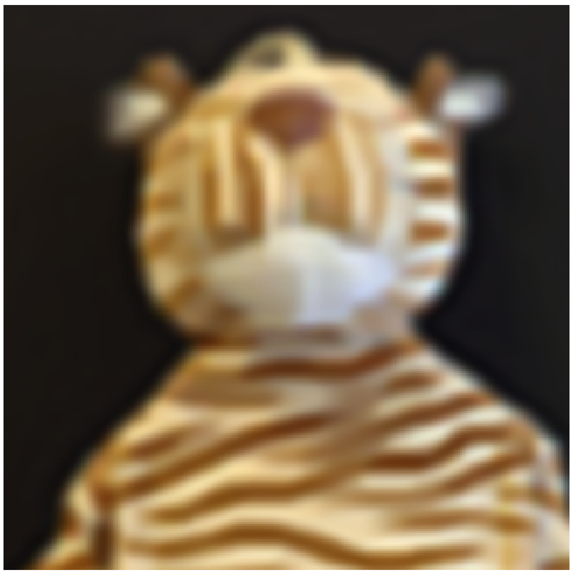

(f) Close up of the super resolved image.

Figure 3.11.: An example of a super resolution simulation result. Here the registration has been achieved using FRI principles, the interpolation with a bi-cubic spline and the deconvolution using the two-step approach with BM3D denoising.

algorithms. These algorithms will tackle the same problems presented in this chapter and will rely on many of the principles. We will also present a more thorough analysis of the performance of different algorithms.

CHAPTER 4

A Novel Algorithm to Approximate Piecewise Polynomial Images based on a Quadtree Decomposition

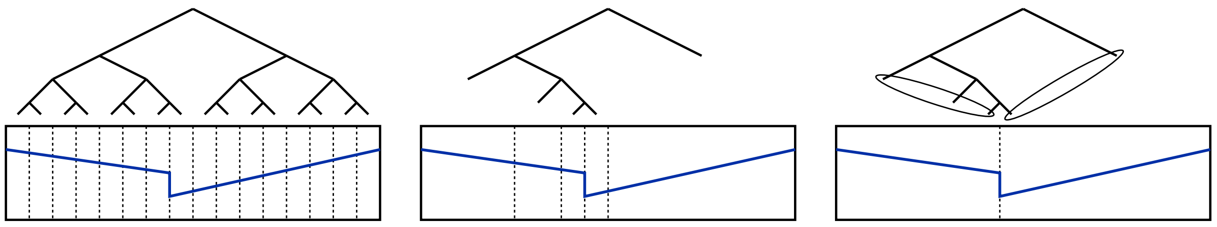
4.1. Introduction

In Chapter 2 we saw numerous attempts to sparsely approximate 2-D piecewise polynomial images and in this chapter we will describe our novel approximation algorithm, which is particularly suited to this class of functions. Our algorithm is based on the compression algorithm of Shukla et al. [63], but we make the following novel contributions: we replace the bit rate constraint with a description length penalty, which is more suitable for restoration. Furthermore, we vastly improve the computational efficiency of the algorithm by using the mathematics of updating matrix factorisations.

We are interested in approximating 2-D signals, but in the aid of clarity we will begin by considering the simpler 1-D case.

4.2. 1-D piecewise polynomial approximation using a binary tree

Our aim is to sparsely describe a 1-D piecewise polynomial function. To do this we dyadically partition the signal space using a binary tree, where each leaf represents a polynomial. Therefore, to represent the 1-D piecewise linear signal shown in Fig. 4.1, we could use any of the trees shown



(a) The binary tree at its deepest depth.
(b) The pruned representation.
(c) The prune-joined representation.

Figure 4.1.: Comparison between the prune and prune-join algorithms for a piecewise linear 1-D signal.

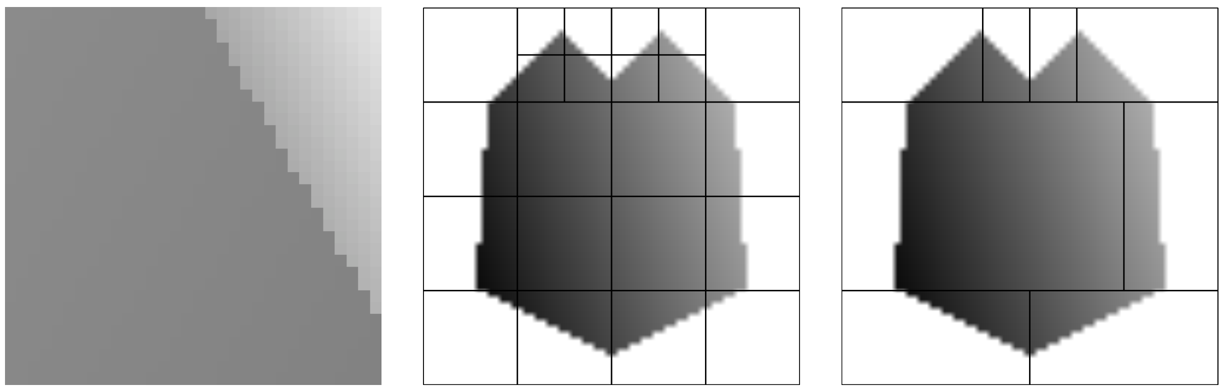
in Figs. 4.1(a), 4.1(b) or 4.1(c). To calculate these representations we define a cost function that has two terms: the first is a two-norm data fitting term and the second is a term to penalise the description length. In 1-D we define the description length to be the sum of the degrees of the polynomials in the approximation; therefore, the cost function is

$$\|\boldsymbol{y} - \boldsymbol{x}\|_2^2 + \lambda \sum_{i \in T} d_i, \tag{4.1}$$

where $\boldsymbol{y}$ is the 1-D signal we are trying to approximate, T is the set of all leaves in the approximation $\boldsymbol{x}$, and d_i is the degree of the polynomial at node i. Here λ is used to provide a tradeoff between the two terms: it is set according to the quality of approximation that is required and in later chapters will be set depending on the degradation of the restoration problem.

To obtain the deepest depth solution shown in Fig. 4.1(a) we visit each node at this deepest depth and minimise (4.1) locally. Note that the approximation of a single node is a simple linear approximation problem. The pruned representation shown in Fig. 4.1(b) is obtained using a bottom up approach that starts from the deepest depth solution. The parent nodes of this deepest depth solution are approximated using the same approach and then two sibling leaves are pruned if the sum of their costs is greater than the cost of their parent. This process is repeated all the way up the tree to produce the final pruned tree.

The pruned representation is suboptimal due to the limitations of the binary tree structure: for example, because of the location of the discontinuity in Fig. 4.1(b), five regions are required to represent a piecewise linear signal containing only one discontinuity. By allowing neighbouring regions of the tree to jointly represent just one polynomial region we can overcome this limitation, as shown in Fig. 4.1(c). This prune-join representation is calculated as follows: the leaves of the pruned tree are visited, in a top-down left-right fashion, and tested to see if they can be joined to



(a) A possible tile with an edge. (b) The pruned representation. (c) The prune-joined representation.

Figure 4.2.: Comparison between the prune and prune-join algorithms for a piecewise linear 2-D signal.

neighbouring leaves which have already been checked (i.e. nodes that are at a higher depth, or the same depth but further to the left in the tree). Two leaves are joined if their combined cost is less than the sum of their individual costs. After two leaves have been joined, the joined representation is used in place of the individual leaves for the rest of the joining algorithm.

4.3. 2-D piecewise polynomial approximation using a quadtree

Now let us move to the 2-D case: a quadtree is constructed where each leaf is either a global polynomial or two polynomials separated by a continuous boundary. Figure 4.2(a) shows a possible node which we will also call a tile. Prune and prune-join representations are generated in almost the same way as the 1-D case and examples are shown in Figs. 4.2(b) and 4.2(c). In 2-D the penalty for the description length, $P^x(\boldsymbol{x})$, is slightly more complex: it is defined to be separable across each node so we can write

$$P^x(\boldsymbol{x}) = \sum_{i \in L} P_i^x(\boldsymbol{x}), \tag{4.2}$$

where $\boldsymbol{x}$ is the image vector, $P_i^x(\boldsymbol{x})$ is the penalty of tile i and L is the set of all leaves in the approximation. In 1-D the description length of a tile is the degree of the polynomial; in 2-D it is very similar: we penalise a polynomial region of degree d by $2d + 1$ because this is the dimension of the space of 2-D polynomials of degree d. For example 2-D polynomials of degree one span the space which can be defined by a constant basis function and two linear basis functions, one per direction. The total description length penalty also needs to deal with the description of the edge discontinuity. This can be done by defining the penalty for the i-th node to be

$$P_i^x(\boldsymbol{x}) = \begin{cases} 2d + 1 & \text{if a global tile,} \\ 2d_1 + 2d_2 + 2 + ln(N_i) & \text{if an edge tile,} \end{cases}$$

where d_1, d_2 are the degrees of the polynomials either side of the edge and N_i is the number of pixels in tile i. Finally $ln(N_i)$ is present to penalise edges of larger tiles more harshly because there are more possible discrete edges to choose from.

Just like the 1-D case we use a two-norm data fitting term producing the final cost function

$$\|\boldsymbol{y} - \boldsymbol{x}\|_2^2 + \lambda P^x(\boldsymbol{x}), \tag{4.3}$$

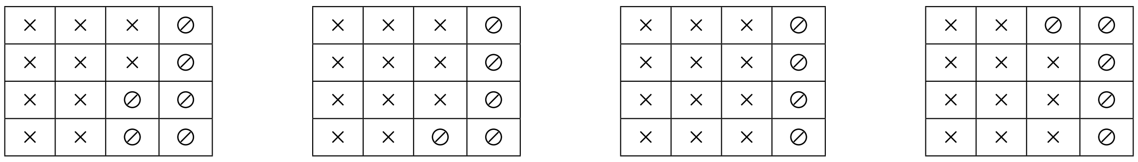
which is separable over each tile. To find the best approximation for a particular tile we exhaustively search a dictionary of possible edges (including the global 'no edge' case) and choose the edge with the cheapest (4.3). This process is explained in detail in Section 4.4. The prune and prune-join algorithms are identical to the 1-D case and their goal is to find

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \left\{ \|\boldsymbol{y} - \boldsymbol{x}\|_2^2 + \lambda P^x(\boldsymbol{x}) \right\}. \tag{4.4}$$

Note that when joining in 1-D, each leaf can only have two neighbours: one left and one right. In 2-D the situation is slightly more complex. However, since we only join two regions that have already been checked, these regions must be either the same size or larger. Therefore, although there could be multiple smaller neighbours at one side, each leaf will have a maximum of four joining candidates: one up, one down, one left and one right.

Since the quadtree piecewise polynomial model is nonlinear, we will use the notation $\boldsymbol{x} = D(\boldsymbol{\theta})$ where $\boldsymbol{\theta}$ is the parameter set which describes the tile structure, edge discontinuities and polynomial coefficients. We also use the notation that the penalty $P^x(D(\boldsymbol{\theta})) = P^\theta(\boldsymbol{\theta})$ and therefore (4.4) is equivalent to

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \left\{ \|\boldsymbol{y} - D(\boldsymbol{\theta})\|_2^2 + \lambda P^\theta(\boldsymbol{\theta}) \right\}.$$



(a) First edge ($e1$). (b) Second edge ($e2$). (c) Third edge ($e3$). (d) Fourth edge ($e4$).

Figure 4.3.: Four edges that differ from the previous edge in only one pixel.

4.4. Finding the best approximation for a node

The prune and prune-join algorithms assume we can calculate an approximation for each node of the tree. In 2-D this involves calculating a suitable edge discontinuity, which may be 'no edge', and the polynomial coefficients.

The tile approximation given a particular edge is a linear problem that is solved by projecting onto the polynomial subspace and hard thresholding. Traditionally a suitable edge is found by exhaustively searching a large number of possible edge discontinuities by approximating each one using two linear projections, one either side of the edge. The edge leading to the minimum cost is chosen. This approach is inefficient and can become unfeasible for large tiles. In this case, due to complexity, the search space is reduced so that only a small number of straight edges are tested. To overcome this limitation, we present a fast method which allows us to exhaustively search edges and also, if we wish, relax the constraint that the edge be straight. Our inspiration comes from the mathematics of updating matrix factorisations: it is often more efficient to update a factorisation than recalculate it from scratch if the original matrix has only undergone a small change. We can use this to quickly check different edges by changing just one pixel at a time. We will see that the complexity of approximating a new edge is independent of the tile size and instead depends only on the maximum degree of the polynomials.

In what follows we demonstrate the process by approximating a 4×4 tile using the edges shown in Fig. 4.3. We will first approximate the tile using the edge shown in Fig. 4.3(a) by calculating the QR decomposition from scratch and then update this factorisation to approximate the tile using the edge shown in Fig. 4.3(b). Later in the chapter we will demonstrate rank deficiencies using the edges shown in Figs. 4.3(c) and 4.3(d).

All the approximations will require a vector representation for the tile which is obtained by lexicographically stacking the tile into a vector $\boldsymbol{t} \in \mathbb{R}^{N_i}$ ($N_i = 16$ in this example):

$$\begin{array}{|c|c|c|c|}\hline t_{11} & t_{12} & t_{13} & t_{14} \\ \hline t_{21} & t_{22} & t_{23} & t_{24} \\ \hline t_{31} & t_{32} & t_{33} & t_{34} \\ \hline t_{41} & t_{42} & t_{43} & t_{44} \\ \hline \end{array} \quad \Rightarrow \quad \boldsymbol{t} = \begin{bmatrix} t_{11} \\ t_{21} \\ \vdots \\ t_{41} \\ t_{12} \\ \vdots \\ t_{44} \end{bmatrix}.$$

4.4.1. Approximating the tile by calculating QR decompositions

In this subsection we will approximate the tile using the edge of Fig. 4.3(a) by computing a QR decomposition either side of the edge.

Let the maximum degree of the polynomials be one, resulting in polynomial subspaces of dimension three. We lexicographically stack the three biorthogonal linear polynomial basis functions either side of the edge and use these vectors as the columns of two matrices $\boldsymbol{B}_{\boldsymbol{1}}^{\boldsymbol{e1}}$ and $\boldsymbol{B}_{\boldsymbol{2}}^{\boldsymbol{e1}}$. We then calculate the thin QR decomposition for these matrices:

$$\begin{array}{c}\begin{array}{|c|c|c|c|}\hline 1 & 1 & 1 & \oslash \\ \hline 1 & 1 & 1 & \oslash \\ \hline 1 & 1 & \oslash & \oslash \\ \hline 1 & 1 & \oslash & \oslash \\ \hline \end{array} \\ \\ \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & \oslash \\ \hline 1 & 2 & 3 & \oslash \\ \hline 1 & 2 & \oslash & \oslash \\ \hline 1 & 2 & \oslash & \oslash \\ \hline \end{array} \\ \\ \begin{array}{|c|c|c|c|}\hline 1 & 1 & 1 & \oslash \\ \hline 2 & 2 & 2 & \oslash \\ \hline 3 & 3 & \oslash & \oslash \\ \hline 4 & 4 & \oslash & \oslash \\ \hline \end{array}\end{array} \Rightarrow \underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 1 & 3 \\ 1 & 1 & 4 \\ 1 & 2 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 4 \\ 1 & 3 & 1 \\ 1 & 3 & 2 \end{bmatrix}}_{\boldsymbol{B}_{\boldsymbol{1}}^{\boldsymbol{e1}}} = \underbrace{\begin{bmatrix} \frac{\sqrt{10}}{10} & -\frac{2\sqrt{35}}{35} & -\frac{23\sqrt{2170}}{2170} \\ \frac{\sqrt{10}}{10} & -\frac{2\sqrt{35}}{35} & -\frac{9\sqrt{2170}}{2170} \\ \frac{\sqrt{10}}{10} & -\frac{2\sqrt{35}}{35} & \frac{5\sqrt{2170}}{2170} \\ \frac{\sqrt{10}}{10} & -\frac{2\sqrt{35}}{35} & \frac{19\sqrt{2170}}{2170} \\ \frac{\sqrt{10}}{10} & \frac{\sqrt{35}}{70} & -\frac{17\sqrt{2170}}{2170} \\ \frac{\sqrt{10}}{10} & \frac{\sqrt{35}}{70} & -\frac{3\sqrt{2170}}{2170} \\ \frac{\sqrt{10}}{10} & \frac{\sqrt{35}}{70} & \frac{11\sqrt{2170}}{2170} \\ \frac{\sqrt{10}}{10} & \frac{\sqrt{35}}{70} & \frac{25\sqrt{2170}}{2170} \\ \frac{\sqrt{10}}{10} & \frac{3\sqrt{35}}{35} & -\frac{11\sqrt{2170}}{2170} \\ \frac{\sqrt{10}}{10} & \frac{3\sqrt{35}}{35} & \frac{3\sqrt{2170}}{2170} \end{bmatrix}}_{\boldsymbol{Q}_{\boldsymbol{1}}^{\boldsymbol{e1}}} \underbrace{\begin{bmatrix} \sqrt{10} & \frac{9\sqrt{10}}{5} & \frac{23\sqrt{10}}{10} \\ 0 & \frac{2\sqrt{35}}{5} & -\frac{6\sqrt{35}}{35} \\ 0 & 0 & \frac{\sqrt{2170}}{14} \end{bmatrix}}_{\boldsymbol{R}_{\boldsymbol{1}}^{\boldsymbol{e1}}}$$

⊘	⊘	⊘	1
⊘	⊘	⊘	1
⊘	⊘	1	1
⊘	⊘	1	1

⊘	⊘	⊘	4
⊘	⊘	⊘	4
⊘	⊘	3	4
⊘	⊘	3	4

⊘	⊘	⊘	1
⊘	⊘	⊘	2
⊘	⊘	3	3
⊘	⊘	4	4

$$\Rightarrow \underbrace{\begin{bmatrix} 1 & 3 & 3 \\ 1 & 3 & 4 \\ 1 & 4 & 1 \\ 1 & 4 & 2 \\ 1 & 4 & 3 \\ 1 & 4 & 4 \end{bmatrix}}_{\boldsymbol{B_2^{e1}}} = \underbrace{\begin{bmatrix} \frac{\sqrt{6}}{6} & -\frac{\sqrt{3}}{3} & -\frac{\sqrt{22}}{22} \\ \frac{\sqrt{6}}{6} & -\frac{\sqrt{3}}{3} & \frac{\sqrt{22}}{22} \\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & -\frac{3\sqrt{22}}{22} \\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & -\frac{\sqrt{22}}{22} \\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & \frac{\sqrt{22}}{22} \\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & \frac{3\sqrt{22}}{22} \end{bmatrix}}_{\boldsymbol{Q_2^{e1}}} \underbrace{\begin{bmatrix} \sqrt{6} & \frac{11\sqrt{6}}{3} & \frac{17\sqrt{6}}{6} \\ 0 & \frac{2\sqrt{3}}{3} & -\frac{2\sqrt{3}}{3} \\ 0 & 0 & \frac{\sqrt{22}}{2} \end{bmatrix}}_{\boldsymbol{R_2^{e1}}}.$$

Having the orthogonal Q matrices makes approximating $\boldsymbol{t}$ using this edge easy. We first split $\boldsymbol{t}$ into two corresponding vectors, $\boldsymbol{t_1^{e1}}$ and $\boldsymbol{t_2^{e1}}$, either side of the edge. We will use the notation $\boldsymbol{t_1^{e1}} \cup \boldsymbol{t_2^{e1}} = \boldsymbol{t}$ to allow us to combine vectors from different regions of the image; similarly, we use '$\cap$' for the intersection of two image vectors. Since $\boldsymbol{t_1^{e1}} \cap \boldsymbol{t_2^{e1}} = \oslash$, the approximation can be found independently for each side:

$$\begin{aligned} \boldsymbol{\hat{t}_1^{e1}} &= \boldsymbol{Q_1^{e1}} \boldsymbol{\hat{\theta}}_{\boldsymbol{Q_1^{e1}}} \text{ where } \boldsymbol{\hat{\theta}}_{\boldsymbol{Q_1^{e1}}} = \boldsymbol{Q_1^{e1}}^T \boldsymbol{t_1^{e1}}, \\ \text{and } \boldsymbol{\hat{t}_2^{e1}} &= \boldsymbol{Q_2^{e1}} \boldsymbol{\hat{\theta}}_{\boldsymbol{Q_2^{e1}}} \text{ where } \boldsymbol{\hat{\theta}}_{\boldsymbol{Q_2^{e1}}} = \boldsymbol{Q_2^{e1}}^T \boldsymbol{t_2^{e1}}. \end{aligned}$$

The error is

$$\begin{aligned} \|\boldsymbol{t} - \boldsymbol{\hat{t}^{e1}}\| &= \|\boldsymbol{t_1^{e1}} - \boldsymbol{\hat{t}_1^{e1}}\|_2^2 + \|\boldsymbol{t_2^{e1}} - \boldsymbol{\hat{t}_2^{e1}}\|_2^2 \\ &= \|\boldsymbol{t}\|_2^2 - \|\boldsymbol{\hat{\theta}}_{\boldsymbol{Q_1^{e1}}}\|_2^2 - \|\boldsymbol{\hat{\theta}}_{\boldsymbol{Q_2^{e1}}}\|_2^2, \end{aligned} \tag{4.5}$$

which can be simplified by neglecting the $\|\boldsymbol{t}\|_2^2$ term when comparing the errors of different edges for the same tile.

4.4.2. Approximating the tile by adding and removing rows to and from QR decompositions

Looking at the two edges in Figs. 4.3(a) and 4.3(b) and the two matrices, $\boldsymbol{B_1^{e1}}$ and $\boldsymbol{B_2^{e1}}$, it is clear that $\boldsymbol{B_1^{e2}}$ can be constructed by adding a row to $\boldsymbol{B_1^{e1}}$ and, similarly, $\boldsymbol{B_2^{e2}}$ can be constructed by removing the same row from $\boldsymbol{B_2^{e1}}$. In the 1970s efficient algorithms were developed for applying rank one updates to matrix factorisations, see for example Gill et al. [36], and the process is now documented in many books including [5, 37]. In [20] Daniel et al. propose numerical stable algorithms to update a QR factorisation which are particular relevant to our problem. In what follows, we will show the technique of adding and removing rows, as proposed in [20], but in the context of our problem. This will lead to our key insight that allows us to obtain the coefficients and error of the approximation without recalculating the Q matrix of the QR decomposition. This vastly improves the efficiency because we no longer have to operate at the dimension of the Q matrix but can instead work at the much smaller dimension of the R matrix.

Adding a row

We will first consider the process of adding a row to the QR decomposition of $\boldsymbol{B_1^{e1}}$ to obtain the QR decomposition for $\boldsymbol{B_1^{e2}}$. The first step of this process is to construct $\boldsymbol{B_1^{e2}}$ as we did before but we do not calculate the QR decomposition. Instead we proceed as follows: we first modify the matrices $\boldsymbol{Q_1^{e1}}$ and $\boldsymbol{R_1^{e1}}$ so that the equality still holds after the new row is added (the new row of scalar values has been shown in bold for emphasis; this goes against the convention of the rest of this thesis where lowercase bold is reserved for vectors):

$$
\begin{array}{|c|c|c|c|}
\hline 1 & 1 & 1 & \oslash \\
\hline 1 & 1 & 1 & \oslash \\
\hline 1 & 1 & \mathbf{1} & \oslash \\
\hline 1 & 1 & \oslash & \oslash \\
\hline
\end{array}
\quad
\begin{array}{|c|c|c|c|}
\hline 1 & 2 & 3 & \oslash \\
\hline 1 & 2 & 3 & \oslash \\
\hline 1 & 2 & \mathbf{3} & \oslash \\
\hline 1 & 2 & \oslash & \oslash \\
\hline
\end{array}
\quad
\begin{array}{|c|c|c|c|}
\hline 1 & 1 & 1 & \oslash \\
\hline 2 & 2 & 2 & \oslash \\
\hline 3 & 3 & \mathbf{3} & \oslash \\
\hline 4 & 4 & \oslash & \oslash \\
\hline
\end{array}
\Rightarrow
\underbrace{\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 2 \\
1 & 1 & 3 \\
1 & 1 & 4 \\
1 & 2 & 1 \\
1 & 2 & 2 \\
1 & 2 & 3 \\
1 & 2 & 4 \\
1 & 3 & 1 \\
1 & 3 & 2 \\
\mathbf{1} & \mathbf{3} & \mathbf{3}
\end{bmatrix}}_{\boldsymbol{B_1^{e2}}}
=
\underbrace{\begin{bmatrix}
\frac{\sqrt{10}}{10} & -\frac{2\sqrt{35}}{35} & -\frac{23\sqrt{2170}}{2170} & 0 \\
\frac{\sqrt{10}}{10} & -\frac{2\sqrt{35}}{35} & -\frac{9\sqrt{2170}}{2170} & 0 \\
\frac{\sqrt{10}}{10} & -\frac{2\sqrt{35}}{35} & \frac{5\sqrt{2170}}{2170} & 0 \\
\frac{\sqrt{10}}{10} & -\frac{2\sqrt{35}}{35} & \frac{19\sqrt{2170}}{2170} & 0 \\
\frac{\sqrt{10}}{10} & \frac{\sqrt{35}}{70} & -\frac{17\sqrt{2170}}{2170} & 0 \\
\frac{\sqrt{10}}{10} & \frac{\sqrt{35}}{70} & -\frac{3\sqrt{2170}}{2170} & 0 \\
\frac{\sqrt{10}}{10} & \frac{\sqrt{35}}{70} & \frac{11\sqrt{2170}}{2170} & 0 \\
\frac{\sqrt{10}}{10} & \frac{\sqrt{35}}{70} & \frac{25\sqrt{2170}}{2170} & 0 \\
\frac{\sqrt{10}}{10} & \frac{3\sqrt{35}}{35} & -\frac{11\sqrt{2170}}{2170} & 0 \\
\frac{\sqrt{10}}{10} & \frac{3\sqrt{35}}{35} & \frac{3\sqrt{2170}}{2170} & 0 \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1}
\end{bmatrix}}_{\boldsymbol{\tilde{Q}_1^{e2}}}
\underbrace{\begin{bmatrix}
\sqrt{10} & \frac{9\sqrt{10}}{5} & \frac{23\sqrt{10}}{10} \\
0 & \frac{2\sqrt{35}}{5} & -\frac{6\sqrt{35}}{35} \\
0 & 0 & \frac{\sqrt{2170}}{14} \\
\mathbf{1} & \mathbf{3} & \mathbf{3}
\end{bmatrix}}_{\boldsymbol{\tilde{R}_1^{e2}}}
\qquad (4.6)
$$

Currently, $\boldsymbol{B_1^{e2}}$ is factored into an orthogonal matrix, $\boldsymbol{\tilde{Q}_1^{e2}}$, multiplied by a matrix, $\boldsymbol{\tilde{R}_1^{e2}}$, which is not upper triangular. To get the factorisation into the desired thin QR form we introduce an orthogonal matrix, $\boldsymbol{G} \in \mathbb{R}^{4\times4}$, which is the product of three Givens rotation matrices: $\boldsymbol{G} = \boldsymbol{G_3 G_2 G_1}$. Since $\boldsymbol{G}^T\boldsymbol{G} = \boldsymbol{I_4}$, we can write

$$\boldsymbol{B_1^{e2}} = \boldsymbol{\tilde{Q}_1^{e2}}\boldsymbol{G}^T\boldsymbol{G}\boldsymbol{\tilde{R}_1^{e2}}.$$

The aim of $\boldsymbol{G}$ is to make $\boldsymbol{G\tilde{R}_1^{e2}}$ upper triangular. To do this the three Givens rotation matrices are constructed to reflect $\begin{bmatrix} 1 & 3 & 3 \end{bmatrix}$ into the diagonal entries of $\boldsymbol{R_1^{e1}}$. The first rotation matrix, $\boldsymbol{G_1}$, reflects the first element of $\begin{bmatrix} 1 & 3 & 3 \end{bmatrix}$ into the top left element of $\boldsymbol{R_1^{e1}}$:

$$
\underbrace{\begin{bmatrix}
\frac{\sqrt{110}}{11} & 0 & 0 & \frac{\sqrt{11}}{11} \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
\frac{\sqrt{11}}{11} & 0 & 0 & -\frac{\sqrt{110}}{11}
\end{bmatrix}}_{\boldsymbol{G_1}}
\underbrace{\begin{bmatrix}
\sqrt{10} & \frac{9\sqrt{10}}{5} & \frac{23\sqrt{10}}{10} \\
0 & \frac{2\sqrt{35}}{5} & -\frac{6\sqrt{35}}{35} \\
0 & 0 & \frac{\sqrt{2170}}{14} \\
1 & 3 & 3
\end{bmatrix}}_{\begin{bmatrix} \boldsymbol{R_1^{e1}} \\ \boldsymbol{\beta}^T \end{bmatrix}}
=
\begin{bmatrix}
\sqrt{11} & \frac{21\sqrt{11}}{11} & \frac{26\sqrt{11}}{11} \\
0 & \frac{2\sqrt{35}}{5} & -\frac{6\sqrt{35}}{35} \\
0 & 0 & \frac{\sqrt{2170}}{14} \\
0 & -\frac{6\sqrt{110}}{55} & -\frac{7\sqrt{110}}{110}
\end{bmatrix}.
$$

Similarly $\boldsymbol{G_2}$ and $\boldsymbol{G_3}$ are constructed to reflect the second and third elements into the other two diagonal elements of $\boldsymbol{R_1^{e1}}$ leading to

$$\boldsymbol{G_3G_2G_1}\begin{bmatrix} \sqrt{10} & \frac{9\sqrt{10}}{5} & \frac{23\sqrt{10}}{10} \\ 0 & \frac{2\sqrt{35}}{5} & -\frac{6\sqrt{35}}{35} \\ 0 & 0 & \frac{\sqrt{2170}}{14} \\ 1 & 3 & 3 \end{bmatrix} = \underbrace{\begin{bmatrix} \sqrt{11} & \frac{21\sqrt{11}}{11} & \frac{26\sqrt{11}}{11} \\ 0 & \frac{2\sqrt{209}}{11} & -\frac{9\sqrt{209}}{209} \\ 0 & 0 & \frac{\sqrt{4389}}{19} \\ 0 & 0 & 0 \end{bmatrix}}_{\begin{bmatrix} \boldsymbol{R_1^{e2}} \\ \boldsymbol{0}^T \end{bmatrix}}.$$

We see that the Givens matrices have ‘zeroed’ the bottom row so that $\boldsymbol{G\tilde{R}_1^{e2}}$ is upper triangular.

The product $\boldsymbol{\tilde{Q}_1^{e2}G}^T = \begin{bmatrix} \boldsymbol{Q_1^{e2}} & \boldsymbol{q} \end{bmatrix}$ is orthogonal (since both $\boldsymbol{G}$ and $\boldsymbol{\tilde{Q}_1^{e2}}$ are orthogonal) so $\boldsymbol{B_1^{e2}}$ is now factored into an orthogonal matrix times an upper triangular matrix:

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 1 & 3 \\ 1 & 1 & 4 \\ 1 & 2 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 4 \\ 1 & 3 & 1 \\ 1 & 3 & 2 \\ 1 & 3 & 3 \end{bmatrix}}_{\boldsymbol{B_1^{e2}}} = \underbrace{\begin{bmatrix} \frac{\sqrt{11}}{11} & -\frac{5\sqrt{209}}{209} & -\frac{10\sqrt{4389}}{1463} & -\frac{13\sqrt{35805}}{11935} \\ \frac{\sqrt{11}}{11} & -\frac{5\sqrt{209}}{209} & -\frac{\sqrt{4389}}{399} & -\frac{2\sqrt{35805}}{3255} \\ \frac{\sqrt{11}}{11} & -\frac{5\sqrt{209}}{209} & \frac{8\sqrt{4389}}{4389} & -\frac{\sqrt{35805}}{7161} \\ \frac{\sqrt{11}}{11} & -\frac{5\sqrt{209}}{209} & \frac{9\sqrt{4389}}{1463} & \frac{4\sqrt{35805}}{11935} \\ \frac{\sqrt{11}}{11} & -\frac{5\sqrt{209}}{209} & -\frac{17\sqrt{4389}}{2926} & \frac{\sqrt{35805}}{23870} \\ \frac{\sqrt{11}}{11} & -\frac{5\sqrt{209}}{209} & -\frac{13\sqrt{4389}}{8778} & \frac{37\sqrt{35805}}{71610} \\ \frac{\sqrt{11}}{11} & -\frac{5\sqrt{209}}{209} & \frac{25\sqrt{4389}}{8778} & \frac{71\sqrt{35805}}{71610} \\ \frac{\sqrt{11}}{11} & -\frac{5\sqrt{209}}{209} & \frac{3\sqrt{4389}}{418} & \frac{\sqrt{35805}}{682} \\ \frac{\sqrt{11}}{11} & -\frac{5\sqrt{209}}{209} & -\frac{\sqrt{4389}}{209} & \frac{2\sqrt{35805}}{1705} \\ \frac{\sqrt{11}}{11} & -\frac{5\sqrt{209}}{209} & -\frac{2\sqrt{4389}}{4389} & \frac{59\sqrt{35805}}{35805} \\ \frac{\sqrt{11}}{11} & -\frac{5\sqrt{209}}{209} & \frac{17\sqrt{4389}}{4389} & -\frac{\sqrt{35805}}{231} \end{bmatrix}}_{\begin{bmatrix} \boldsymbol{Q_1^{e2}} & \boldsymbol{q} \end{bmatrix}} \underbrace{\begin{bmatrix} \sqrt{11} & \frac{21\sqrt{11}}{11} & \frac{26\sqrt{11}}{11} \\ 0 & \frac{2\sqrt{209}}{11} & -\frac{9\sqrt{209}}{209} \\ 0 & 0 & \frac{\sqrt{4389}}{19} \\ 0 & 0 & 0 \end{bmatrix}}_{\begin{bmatrix} \boldsymbol{R_1^{e2}} \\ \boldsymbol{0}^T \end{bmatrix}}.$$

Since the bottom row of the upper triangular matrix is zero we can simplify to $\boldsymbol{B_1^{e2}} = \boldsymbol{Q_1^{e2}R_1^{e2}}$, which is the required thin QR decomposition.

One may be tempted to calculate the polynomial coefficients by using the updated $\boldsymbol{Q_1^{e2}}$ to directly calculate $\boldsymbol{Q}_{\mathbf{1}}^{\boldsymbol{e2}^T}\boldsymbol{t}$; there is, however, a much more efficient way. Since

$$\begin{bmatrix} \boldsymbol{Q}_{\mathbf{1}}^{\boldsymbol{e2}} & \boldsymbol{q} \end{bmatrix}^T \boldsymbol{t}_{\mathbf{1}}^{\boldsymbol{e2}} = \left(\begin{bmatrix} \boldsymbol{Q}_{\mathbf{1}}^{\boldsymbol{e1}} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \boldsymbol{G}^T \right)^T \begin{bmatrix} \boldsymbol{t}_{\mathbf{1}}^{\boldsymbol{e1}} \\ t_{33} \end{bmatrix},$$

we can calculate the coefficients from

$$\hat{\boldsymbol{\theta}}_{\boldsymbol{Q}_{\mathbf{1}}^{\boldsymbol{e2}}} = \boldsymbol{Q}_{\mathbf{1}}^{\boldsymbol{e2}^T} \boldsymbol{t}_{\mathbf{1}}^{\boldsymbol{e2}} = \underline{\boldsymbol{G}} \begin{bmatrix} \hat{\boldsymbol{\theta}}_{\boldsymbol{Q}_{\mathbf{1}}^{\boldsymbol{e1}}} \\ t_{33} \end{bmatrix}, \tag{4.7}$$

where $\underline{\boldsymbol{G}}$ is $\boldsymbol{G}$ with the last row removed. This is the key result because it allows us to update the coefficients without calculating $\boldsymbol{Q}_{\mathbf{1}}^{\boldsymbol{e2}}$. All that is needed is to construct the Givens matrices using $\boldsymbol{R}_{\mathbf{1}}^{\boldsymbol{e1}}$ and then the updated coefficients and upper triangular matrix are easily found. Once the coefficients have been calculated for both sides of the edge, the error can efficiently be calculated from (4.5). The computational cost of the update is independent of the tile size and instead dependent on the degree of the polynomials used, which in our case is very small. This is because we require d Givens matrices, constructed to reflect the new row into the diagonal elements of $\boldsymbol{R}_{\mathbf{1}}^{\boldsymbol{e1}} \in \mathbb{R}^{d\times d}$, to update the coefficients.

Removing a row

So far we have only told half the story: we now need to remove the same row from $\boldsymbol{B_2^{e1}}$ to obtain $\boldsymbol{B_2^{e2}}$. We recall that we have the following thin QR decomposition for $\boldsymbol{B_2^{e1}}$:

∅	∅	∅	1
∅	∅	∅	1
∅	∅	1	1
∅	∅	1	1

∅	∅	∅	4
∅	∅	∅	4
∅	∅	3	4
∅	∅	3	4

∅	∅	∅	1
∅	∅	∅	2
∅	∅	3	3
∅	∅	4	4

$$\Rightarrow \underbrace{\begin{bmatrix} 1 & 3 & 3 \\ 1 & 3 & 4 \\ 1 & 4 & 1 \\ 1 & 4 & 2 \\ 1 & 4 & 3 \\ 1 & 4 & 4 \end{bmatrix}}_{\boldsymbol{B_2^{e1}} = \begin{bmatrix} \boldsymbol{\beta}^T \\ \boldsymbol{B_2^{e2}} \end{bmatrix}} = \underbrace{\begin{bmatrix} \frac{\sqrt{6}}{6} & -\frac{\sqrt{3}}{3} & -\frac{\sqrt{22}}{22} \\ \frac{\sqrt{6}}{6} & -\frac{\sqrt{3}}{3} & \frac{\sqrt{22}}{22} \\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & -\frac{3\sqrt{22}}{22} \\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & -\frac{\sqrt{22}}{22} \\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & \frac{\sqrt{22}}{22} \\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & \frac{3\sqrt{22}}{22} \end{bmatrix}}_{\boldsymbol{Q_2^{e1}} = \begin{bmatrix} \boldsymbol{\rho}^T \\ \boldsymbol{\tilde{Q}_2^{e2}} \end{bmatrix}} \underbrace{\begin{bmatrix} \sqrt{6} & \frac{11\sqrt{6}}{3} & \frac{17\sqrt{6}}{6} \\ 0 & \frac{2\sqrt{3}}{3} & -\frac{2\sqrt{3}}{3} \\ 0 & 0 & \frac{\sqrt{22}}{2} \end{bmatrix}}_{\boldsymbol{R_2^{e1}}}$$

We wish to remove the first row, $\boldsymbol{\beta}^T$, to obtain the QR decomposition for $\boldsymbol{B_2^{e2}}$. The process of removing a row can be thought of as the reverse of adding a row.

We first append a vector, $\boldsymbol{q}$, to $\boldsymbol{Q_2^{e1}}$ such that $\begin{bmatrix} \boldsymbol{Q_2^{e1}} & \boldsymbol{q} \end{bmatrix}$ is orthogonal and $\left\| \begin{bmatrix} \boldsymbol{\rho}^T & \boldsymbol{q}[1] \end{bmatrix} \right\|_2 = 1$; i.e., this new matrix is orthogonal and its first row (the row to be removed) has unit norm.

These criterion allow us to use three Givens rotation matrices, $\boldsymbol{G} = \boldsymbol{G_3 G_2 G_1}$, to reflect the three elements of $\boldsymbol{\rho}$ into $\boldsymbol{q}[1]$ and, since Givens matrices preserve length, the resulting row, $\begin{bmatrix} \boldsymbol{\rho}^T & \boldsymbol{q}[1] \end{bmatrix} \boldsymbol{G}^T$ will be $\begin{bmatrix} \boldsymbol{0}^T & 1 \end{bmatrix}$.

Furthermore, $\begin{bmatrix} \boldsymbol{Q_2^{e1}} & \boldsymbol{q} \end{bmatrix} \boldsymbol{G}^T$ will be orthogonal and we have just seen that the first element of the last column will be one therefore the rest of this column must be zeros. Thus if we operate correctly, we will be in an analogous position to (4.6) with the Givens matrices, in effect, playing the reverse of the role they played in adding a row. This will become much clearer when, continuing our example, we get to (4.8), which is in the desired form; i.e., it is in a form analogous to (4.6) that allows the row to be removed whilst maintaining the equality.

The vector, $\boldsymbol{q}$, satisfying the above criterion can be constructed by using the Gram-Schmidt process to orthonormalise a vector, $\boldsymbol{v}$, that is all zeros except for a 1 in the row, which we wish to remove:

$$\boldsymbol{q} = \frac{1}{r_{dd}}\left(\underbrace{\begin{bmatrix}1\\0\\0\\0\\0\\0\end{bmatrix}}_{\boldsymbol{v}} - \underbrace{\begin{bmatrix}\frac{\sqrt{6}}{6} & -\frac{\sqrt{3}}{3} & -\frac{\sqrt{22}}{22}\\ \frac{\sqrt{6}}{6} & -\frac{\sqrt{3}}{3} & \frac{\sqrt{22}}{22}\\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & -\frac{3\sqrt{22}}{22}\\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & -\frac{\sqrt{22}}{22}\\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & \frac{\sqrt{22}}{22}\\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & \frac{3\sqrt{22}}{22}\end{bmatrix}}_{\boldsymbol{Q_2^{e1}}} \underbrace{\begin{bmatrix}\frac{\sqrt{6}}{6} & -\frac{\sqrt{3}}{3} & -\frac{\sqrt{22}}{22}\\ \frac{\sqrt{6}}{6} & -\frac{\sqrt{3}}{3} & \frac{\sqrt{22}}{22}\\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & -\frac{3\sqrt{22}}{22}\\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & -\frac{\sqrt{22}}{22}\\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & \frac{\sqrt{22}}{22}\\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & \frac{3\sqrt{22}}{22}\end{bmatrix}^T}_{\boldsymbol{Q_2^{e1}}^T} \underbrace{\begin{bmatrix}1\\0\\0\\0\\0\\0\end{bmatrix}}_{\boldsymbol{v}}\right)$$

$$= \frac{1}{r_{dd}}\left(\underbrace{\begin{bmatrix}1\\0\\0\\0\\0\\0\end{bmatrix}}_{\boldsymbol{v}} - \underbrace{\begin{bmatrix}\frac{\sqrt{6}}{6} & -\frac{\sqrt{3}}{3} & -\frac{\sqrt{22}}{22}\\ \frac{\sqrt{6}}{6} & -\frac{\sqrt{3}}{3} & \frac{\sqrt{22}}{22}\\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & -\frac{3\sqrt{22}}{22}\\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & -\frac{\sqrt{22}}{22}\\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & \frac{\sqrt{22}}{22}\\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & \frac{3\sqrt{22}}{22}\end{bmatrix}}_{\begin{bmatrix}\boldsymbol{\rho}^T\\ \tilde{\boldsymbol{Q}}_{\boldsymbol{2}}^{\boldsymbol{e2}}\end{bmatrix}} \underbrace{\begin{bmatrix}\frac{\sqrt{6}}{6}\\ -\frac{\sqrt{3}}{3}\\ -\frac{\sqrt{22}}{22}\end{bmatrix}}_{\boldsymbol{\rho}}\right),$$

where r_{dd} ensures that $\|\boldsymbol{q}\|_2 = 1$; i.e., $r_{dd} = \|\boldsymbol{v} - \boldsymbol{Q_2^{e1}}\boldsymbol{\rho}\| = \sqrt{1 - \boldsymbol{\rho}^T\boldsymbol{\rho}}$. Therefore

$$\boldsymbol{q} = \frac{1}{\sqrt{1-\boldsymbol{\rho}^T\boldsymbol{\rho}}}\left(\boldsymbol{v} - \boldsymbol{Q_2^{e1}}\boldsymbol{\rho}\right)$$
$$= \frac{1}{\sqrt{1-\boldsymbol{\rho}^T\boldsymbol{\rho}}}\begin{bmatrix}1-\boldsymbol{\rho}^T\boldsymbol{\rho}\\ -\tilde{\boldsymbol{Q}}_{\boldsymbol{2}}^{\boldsymbol{e2}}\boldsymbol{\rho}\end{bmatrix}$$

$$= \begin{bmatrix} r_{dd} \\ -\frac{\tilde{\boldsymbol{Q}}_{\boldsymbol{2}}^{\boldsymbol{e2}}\boldsymbol{\rho}}{r_{dd}} \end{bmatrix}.$$

Appending $\boldsymbol{q}$ to $\boldsymbol{Q}_{\boldsymbol{2}}^{\boldsymbol{e1}}$ yields

$$\underbrace{\begin{bmatrix} 1 & 3 & 3 \\ 1 & 3 & 4 \\ 1 & 4 & 1 \\ 1 & 4 & 2 \\ 1 & 4 & 3 \\ 1 & 4 & 4 \end{bmatrix}}_{\boldsymbol{B}_{\boldsymbol{2}}^{\boldsymbol{e1}}} = \underbrace{\begin{bmatrix} \frac{\sqrt{6}}{6} & -\frac{\sqrt{3}}{3} & -\frac{\sqrt{22}}{22} & \frac{\sqrt{55}}{11} \\ \frac{\sqrt{6}}{6} & -\frac{\sqrt{3}}{3} & \frac{\sqrt{22}}{22} & -\frac{\sqrt{55}}{11} \\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & -\frac{3\sqrt{22}}{22} & -\frac{3\sqrt{55}}{110} \\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & -\frac{\sqrt{22}}{22} & -\frac{\sqrt{55}}{110} \\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & \frac{\sqrt{22}}{22} & \frac{\sqrt{55}}{110} \\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & \frac{3\sqrt{22}}{22} & \frac{3\sqrt{55}}{110} \end{bmatrix}}_{\begin{bmatrix} \boldsymbol{Q}_{\boldsymbol{2}}^{\boldsymbol{e1}} & \boldsymbol{q} \end{bmatrix}} \underbrace{\begin{bmatrix} \sqrt{6} & \frac{11\sqrt{6}}{3} & \frac{17\sqrt{6}}{6} \\ 0 & \frac{2\sqrt{3}}{3} & -\frac{2\sqrt{3}}{3} \\ 0 & 0 & \frac{\sqrt{22}}{2} \\ 0 & 0 & 0 \end{bmatrix}}_{\begin{bmatrix} \boldsymbol{R}_{\boldsymbol{2}}^{\boldsymbol{e1}} \\ \boldsymbol{0}^T \end{bmatrix}},$$

and it is clear that the first row of $\begin{bmatrix} \boldsymbol{Q}_{\boldsymbol{2}}^{\boldsymbol{e1}} & \boldsymbol{q} \end{bmatrix}$, which is $\begin{bmatrix} \boldsymbol{\rho}^T & r_{dd} \end{bmatrix}$, has unit norm since $r_{dd} = \sqrt{1 - \boldsymbol{\rho}^T\boldsymbol{\rho}}$. We required this so that we could use three Givens rotation matrices to reflect the three elements of $\boldsymbol{\rho}$ into r_{dd} and obtain $\begin{bmatrix} \boldsymbol{0}^T & 1 \end{bmatrix}$; i.e., we construct $\boldsymbol{G} = \boldsymbol{G_3 G_2 G_1}$, so that $\boldsymbol{G}\begin{bmatrix} \boldsymbol{\rho} \\ r_{dd} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ 1 \end{bmatrix}$. For our example, the Givens matrices are

$$\underbrace{\begin{bmatrix} \frac{\sqrt{30}}{6} & 0 & 0 & -\frac{\sqrt{6}}{6} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{\sqrt{6}}{6} & 0 & 0 & \frac{\sqrt{30}}{6} \end{bmatrix}}_{\boldsymbol{G_3}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{15}}{5} & 0 & \frac{\sqrt{10}}{5} \\ 0 & 0 & 1 & 0 \\ 0 & -\frac{\sqrt{10}}{5} & 0 & \frac{\sqrt{15}}{5} \end{bmatrix}}_{\boldsymbol{G_2}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{110}}{11} & \frac{\sqrt{11}}{11} \\ 0 & 0 & -\frac{\sqrt{11}}{11} & \frac{\sqrt{110}}{11} \end{bmatrix}}_{\boldsymbol{G_1}} \underbrace{\begin{bmatrix} \frac{\sqrt{6}}{6} \\ -\frac{\sqrt{3}}{3} \\ -\frac{\sqrt{22}}{22} \\ \frac{\sqrt{55}}{11} \end{bmatrix}}_{\begin{bmatrix} \boldsymbol{\rho} \\ r_{dd} \end{bmatrix}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Notice that $\boldsymbol{G_1}$ operates on the last two elements of $\begin{bmatrix} \boldsymbol{\rho} \\ r_{dd} \end{bmatrix}$, $\boldsymbol{G_2}$ the second and last, and $\boldsymbol{G_3}$ the first and last. The elements are zeroed in this order so that $\underline{\boldsymbol{G}}\begin{bmatrix} \boldsymbol{R}_{\boldsymbol{2}}^{\boldsymbol{e1}} \\ \boldsymbol{0}^T \end{bmatrix}$ remains upper triangular.

Therefore inserting $\boldsymbol{G}^T\boldsymbol{G}$ in between the two factors yields

$$\underbrace{\begin{bmatrix} 1 & 3 & 3 \\ 1 & 3 & 4 \\ 1 & 4 & 1 \\ 1 & 4 & 2 \\ 1 & 4 & 3 \\ 1 & 4 & 4 \end{bmatrix}}_{\boldsymbol{B}_{\boldsymbol{2}}^{\boldsymbol{e1}}} = \underbrace{\begin{bmatrix} \frac{\sqrt{6}}{6} & -\frac{\sqrt{3}}{3} & -\frac{\sqrt{22}}{22} & \frac{\sqrt{55}}{11} \\ \frac{\sqrt{6}}{6} & -\frac{\sqrt{3}}{3} & \frac{\sqrt{22}}{22} & -\frac{\sqrt{55}}{11} \\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & -\frac{3\sqrt{22}}{22} & -\frac{3\sqrt{55}}{110} \\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & -\frac{\sqrt{22}}{22} & -\frac{\sqrt{55}}{110} \\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & \frac{\sqrt{22}}{22} & \frac{\sqrt{55}}{110} \\ \frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{6} & \frac{3\sqrt{22}}{22} & \frac{3\sqrt{55}}{110} \end{bmatrix}}_{\begin{bmatrix} \boldsymbol{Q}_{\boldsymbol{2}}^{\boldsymbol{e1}} & \boldsymbol{q} \end{bmatrix}} \boldsymbol{G}^T\boldsymbol{G} \underbrace{\begin{bmatrix} \sqrt{6} & \frac{11\sqrt{6}}{3} & \frac{17\sqrt{6}}{6} \\ 0 & \frac{2\sqrt{3}}{3} & -\frac{2\sqrt{3}}{3} \\ 0 & 0 & \frac{\sqrt{22}}{2} \\ 0 & 0 & 0 \end{bmatrix}}_{\begin{bmatrix} \boldsymbol{R}_{\boldsymbol{2}}^{\boldsymbol{e1}} \\ \boldsymbol{0}^T \end{bmatrix}},$$

$$\underbrace{\begin{bmatrix} 1 & 3 & 3 \\ 1 & 3 & 4 \\ 1 & 4 & 1 \\ 1 & 4 & 2 \\ 1 & 4 & 3 \\ 1 & 4 & 4 \end{bmatrix}}_{\begin{bmatrix} \boldsymbol{b}^T \\ \boldsymbol{B}_{\boldsymbol{2}}^{\boldsymbol{e2}} \end{bmatrix}} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 \\ \frac{\sqrt{5}}{5} & -\frac{2\sqrt{5}}{5} & 0 & 0 \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{5}}{10} & -\frac{3\sqrt{5}}{10} & 0 \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{5}}{10} & -\frac{\sqrt{5}}{10} & 0 \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{5}}{10} & \frac{\sqrt{5}}{10} & 0 \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{5}}{10} & \frac{3\sqrt{5}}{10} & 0 \end{bmatrix}}_{\begin{bmatrix} \boldsymbol{0}^T & 1 \\ \boldsymbol{Q}_{\boldsymbol{2}}^{\boldsymbol{e2}} & \boldsymbol{0} \end{bmatrix}} \underbrace{\begin{bmatrix} \sqrt{5} & \frac{19\sqrt{5}}{5} & \frac{14\sqrt{5}}{5} \\ 0 & \frac{2\sqrt{5}}{5} & -\frac{3\sqrt{5}}{5} \\ 0 & 0 & \sqrt{5} \\ 1 & 3 & 3 \end{bmatrix}}_{\begin{bmatrix} \boldsymbol{R}_{\boldsymbol{2}}^{\boldsymbol{e2}} \\ \boldsymbol{\beta}^T \end{bmatrix}}. \tag{4.8}$$

Notice that the first row and last column of the left factor are all zeros except the one in the upper right element. This was our goal when constructing $\boldsymbol{q}$ and $\boldsymbol{G}$ because it allows us to easily remove the row. Also note that the bottom row of the right factor will always be $\boldsymbol{\beta}^T$ or the equality would not be valid. We have manipulated the matrices so that we are in an analogous position to (4.6). We can obtain the required factorisation by simplifying to $\boldsymbol{B}_{\boldsymbol{2}}^{\boldsymbol{e2}} = \boldsymbol{Q}_{\boldsymbol{2}}^{\boldsymbol{e2}}\boldsymbol{R}_{\boldsymbol{2}}^{\boldsymbol{e2}}$.

Again we can devise a very efficient algorithm to calculate the coefficients $\boldsymbol{\theta}_{\boldsymbol{2}}^{\boldsymbol{e2}}$ from the previous coefficients $\boldsymbol{\theta}_{\boldsymbol{2}}^{\boldsymbol{e1}}$:

$$\begin{aligned} \begin{bmatrix} \boldsymbol{0}^T & 1 \\ \boldsymbol{Q}_{\boldsymbol{2}}^{\boldsymbol{e2}} & \boldsymbol{0} \end{bmatrix}^T \boldsymbol{t}_{\boldsymbol{2}}^{\boldsymbol{e1}} &= \left(\begin{bmatrix} \boldsymbol{Q}_{\boldsymbol{2}}^{\boldsymbol{e1}} & \boldsymbol{q} \end{bmatrix} \boldsymbol{G}^T \right)^T \boldsymbol{t}_{\boldsymbol{2}}^{\boldsymbol{e1}} \\ &= \left(\begin{bmatrix} \boldsymbol{\rho}^T & r_{dd} \\ \tilde{\boldsymbol{Q}}_{\boldsymbol{2}}^{\boldsymbol{e2}} & -\frac{\tilde{\boldsymbol{Q}}_{\boldsymbol{2}}^{\boldsymbol{e2}}\boldsymbol{\rho}}{r_{dd}} \end{bmatrix} \boldsymbol{G}^T \right)^T \boldsymbol{t}_{\boldsymbol{2}}^{\boldsymbol{e1}} \end{aligned}$$

$$= \boldsymbol{G}\begin{bmatrix} \boldsymbol{\rho} & \tilde{\boldsymbol{Q}}_{\boldsymbol{2}}^{\boldsymbol{e2}^T} \\ r_{dd} & -\frac{\boldsymbol{\rho}^T\tilde{\boldsymbol{Q}}_{\boldsymbol{2}}^{\boldsymbol{e2}^T}}{r_{dd}} \end{bmatrix}\boldsymbol{t}_{\boldsymbol{2}}^{\boldsymbol{e1}}$$

$$= \boldsymbol{G}\begin{bmatrix} \boldsymbol{Q}_{\boldsymbol{2}}^{\boldsymbol{e1}^T}\boldsymbol{t}_{\boldsymbol{2}}^{\boldsymbol{e1}} \\ r_{dd}t_{33} - \frac{\boldsymbol{\rho}^T\tilde{\boldsymbol{Q}}_{\boldsymbol{2}}^{\boldsymbol{e2}^T}\boldsymbol{t}_{\boldsymbol{2}}^{\boldsymbol{e2}}}{r_{dd}} \end{bmatrix}$$

$$= \boldsymbol{G}\begin{bmatrix} \boldsymbol{Q}_{\boldsymbol{2}}^{\boldsymbol{e1}^T}\boldsymbol{t}_{\boldsymbol{2}}^{\boldsymbol{e1}} \\ \frac{r_{dd}^2 t_{33} - \boldsymbol{\rho}^T(\boldsymbol{Q}_{\boldsymbol{2}}^{\boldsymbol{e1}^T}\boldsymbol{t}_{\boldsymbol{2}}^{\boldsymbol{e1}} - \boldsymbol{\rho}t_{33})}{r_{dd}} \end{bmatrix}.$$

Consequently

$$\boldsymbol{\theta}_{\boldsymbol{2}}^{\boldsymbol{e2}} = \boldsymbol{Q}_{\boldsymbol{2}}^{\boldsymbol{e2}^T}\boldsymbol{t}_{\boldsymbol{2}}^{\boldsymbol{e2}} = \underline{\boldsymbol{G}}\begin{bmatrix} \boldsymbol{\theta}_{\boldsymbol{2}}^{\boldsymbol{e1}} \\ \frac{t_{33} - \boldsymbol{\rho}^T\boldsymbol{\theta}_{\boldsymbol{2}}^{\boldsymbol{e1}}}{\sqrt{1-\boldsymbol{\rho}^T\boldsymbol{\rho}}} \end{bmatrix}. \tag{4.9}$$

Again we do not need to calculate $\boldsymbol{Q}_{\boldsymbol{2}}^{\boldsymbol{e2}}$ and the update is independent of the tile size. In this case we need knowledge of $\boldsymbol{\rho}$, both to construct $\boldsymbol{G}$ and for the update in (4.9). This can be found efficiently from

$$\boldsymbol{\rho} = \left(\boldsymbol{R}_{\boldsymbol{2}}^{\boldsymbol{e1}^{-1}}\right)^T \boldsymbol{\beta},$$

because $\boldsymbol{R}_{\boldsymbol{2}}^{\boldsymbol{e1}}$ is upper triangular. Now that we have the coefficients for both sides of the edge the error can be calculated using (4.5).

We have developed efficient algorithms to calculate the coefficients and error after adding and removing rows. If we constructed a dictionary of edges where each edge differs from the previous one in only one pixel, then we could exhaustively search this dictionary by adding a pixel to one side of the edge and removing the same pixel from the other side. In most cases, this can be done using the two updates given in (4.7) and (4.9); however, in some special cases rank deficiencies can occur. In the next subsection we will address these issues and present the final algorithms to add and remove a pixel from either side of the edge.

4.4.3. Handling rank deficiencies

Removing columns so that (4.9) is always well defined

It is clear that update (4.9) requires $\boldsymbol{\rho}^T\boldsymbol{\rho} \neq 1$. The case $\boldsymbol{\rho}^T\boldsymbol{\rho} = 1$ occurs if removing the row, $\boldsymbol{\rho}$, makes the system rank deficient. In our context this occurs if one of the linear basis functions becomes equivalent to the constant basis function.

For example, consider what would happen if we tried to obtain $\boldsymbol{B}_{\mathbf{2}}^{\boldsymbol{e3}}$ by removing the row $\boldsymbol{\beta}^T = \begin{bmatrix} 1 & 3 & 4 \end{bmatrix}$ from $\boldsymbol{B}_{\mathbf{2}}^{\boldsymbol{e2}}$:

$$\begin{array}{c}
\begin{array}{|c|c|c|c|}\hline \oslash & \oslash & \oslash & 1 \\ \hline \oslash & \oslash & \oslash & 1 \\ \hline \oslash & \oslash & \oslash & 1 \\ \hline \oslash & \oslash & 1 & 1 \\ \hline \end{array} \\
\begin{array}{|c|c|c|c|}\hline \oslash & \oslash & \oslash & 4 \\ \hline \oslash & \oslash & \oslash & 4 \\ \hline \oslash & \oslash & \oslash & 4 \\ \hline \oslash & \oslash & 3 & 4 \\ \hline \end{array} \\
\begin{array}{|c|c|c|c|}\hline \oslash & \oslash & \oslash & 1 \\ \hline \oslash & \oslash & \oslash & 2 \\ \hline \oslash & \oslash & \oslash & 3 \\ \hline \oslash & \oslash & 4 & 4 \\ \hline \end{array}
\end{array}
\Rightarrow
\underbrace{\begin{bmatrix} 1 & 3 & 4 \\ 1 & 4 & 1 \\ 1 & 4 & 2 \\ 1 & 4 & 3 \\ 1 & 4 & 4 \end{bmatrix}}_{\boldsymbol{B}_{\mathbf{2}}^{\boldsymbol{e2}} = \begin{bmatrix} \boldsymbol{\beta}^T \\ \boldsymbol{B}_{\mathbf{2}}^{\boldsymbol{e3}} \end{bmatrix}}
=
\underbrace{\begin{bmatrix} \frac{\sqrt{5}}{5} & -\frac{2\sqrt{5}}{5} & 0 \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{5}}{10} & -\frac{3\sqrt{5}}{10} \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{5}}{10} & -\frac{\sqrt{5}}{10} \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{5}}{10} & \frac{\sqrt{5}}{10} \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{5}}{10} & \frac{3\sqrt{5}}{10} \end{bmatrix}}_{\boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e2}} = \begin{bmatrix} \boldsymbol{\rho}^T \\ \tilde{\boldsymbol{Q}}_{\mathbf{2}}^{\boldsymbol{e3}} \end{bmatrix}}
\underbrace{\begin{bmatrix} \sqrt{5} & \frac{19\sqrt{5}}{5} & \frac{14\sqrt{5}}{5} \\ 0 & \frac{2\sqrt{5}}{5} & -\frac{3\sqrt{5}}{5} \\ 0 & 0 & \sqrt{5} \end{bmatrix}}_{\boldsymbol{R}_{\mathbf{2}}^{\boldsymbol{e2}}}.$$

The constant and x basis functions would become linearly dependent and the system would become rank deficient. As a consequence $\boldsymbol{\rho}^T\boldsymbol{\rho}$ would be one and update (4.9) would not be defined. In order to be able to use (4.9), rank deficiencies can be prevented by removing the second column before removing the row. As a first step, we remove the second column from $\boldsymbol{B}_{\mathbf{2}}^{\boldsymbol{e2}}$ and $\boldsymbol{R}_{\mathbf{2}}^{\boldsymbol{e2}}$:

$$\begin{array}{c}
\begin{array}{|c|c|c|c|}\hline \oslash & \oslash & \oslash & 1 \\ \hline \oslash & \oslash & \oslash & 1 \\ \hline \oslash & \oslash & \oslash & 1 \\ \hline \oslash & \oslash & 1 & 1 \\ \hline \end{array} \\
\begin{array}{|c|c|c|c|}\hline \oslash & \oslash & \oslash & 1 \\ \hline \oslash & \oslash & \oslash & 2 \\ \hline \oslash & \oslash & \oslash & 3 \\ \hline \oslash & \oslash & 4 & 4 \\ \hline \end{array}
\end{array}
\Rightarrow
\underbrace{\begin{bmatrix} 1 & 4 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}}_{{}_{\text{-}x}\boldsymbol{B}_{\mathbf{2}}^{\boldsymbol{e2}}}
=
\underbrace{\begin{bmatrix} \frac{\sqrt{5}}{5} & -\frac{2\sqrt{5}}{5} & 0 \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{5}}{10} & -\frac{3\sqrt{5}}{10} \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{5}}{10} & -\frac{\sqrt{5}}{10} \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{5}}{10} & \frac{\sqrt{5}}{10} \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{5}}{10} & \frac{3\sqrt{5}}{10} \end{bmatrix}}_{\boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e2}}}
\underbrace{\begin{bmatrix} \sqrt{5} & \frac{14\sqrt{5}}{5} \\ 0 & -\frac{3\sqrt{5}}{5} \\ 0 & \sqrt{5} \end{bmatrix}}_{{}_{\text{-}x}\tilde{\boldsymbol{R}}_{\mathbf{2}}^{\boldsymbol{e2}}}.$$

Here we have used the rather unusual notation that ${}_{\text{-}x}\boldsymbol{B}_{\mathbf{2}}^{\boldsymbol{e2}}$ is the B matrix for the second side of the edge, $e2$, with the x column removed. Similarly ${}_{\text{-}x}\boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e2}}$ and ${}_{\text{-}x}\boldsymbol{R}_{\mathbf{2}}^{\boldsymbol{e2}}$ will be used for the thin QR decomposition of this matrix.

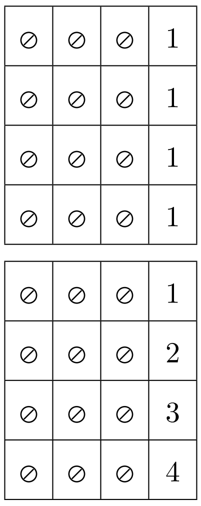
If we had needed to remove the last column from $\boldsymbol{B}_{\mathbf{2}}^{\boldsymbol{e2}}$ things would be simpler because, in this case, the last row of ${}_{\text{-}x}\tilde{\boldsymbol{R}}_{\mathbf{2}}^{\boldsymbol{e2}}$ would be all zeros. This would allow us to remove this row of zeros and the last column of $\boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e2}}$. In this more complex case we need to use a Givens matrix to zero the element below the diagonal before we can remove this row and column. In general, to remove the i-th column we would need to use $d-i$ Givens matrices to remove the $d-i$ elements below the diagonal. For our example we have

$$\underbrace{\begin{bmatrix} 1 & 4 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}}_{{}_{\text{-}x}\boldsymbol{B}_{\mathbf{2}}^{\boldsymbol{e2}}} = \underbrace{\begin{bmatrix} \frac{\sqrt{5}}{5} & -\frac{2\sqrt{5}}{5} & 0 \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{5}}{10} & -\frac{3\sqrt{5}}{10} \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{5}}{10} & -\frac{\sqrt{5}}{10} \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{5}}{10} & \frac{\sqrt{5}}{10} \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{5}}{10} & \frac{3\sqrt{5}}{10} \end{bmatrix}}_{\boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e2}}} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & -\frac{3\sqrt{134}}{134} & \frac{5\sqrt{134}}{134} \\ 0 & \frac{5\sqrt{134}}{134} & \frac{3\sqrt{134}}{134} \end{bmatrix}}_{\boldsymbol{G}^T} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & -\frac{3\sqrt{134}}{134} & \frac{5\sqrt{134}}{134} \\ 0 & \frac{5\sqrt{134}}{134} & \frac{3\sqrt{134}}{134} \end{bmatrix}}_{\boldsymbol{G}} \underbrace{\begin{bmatrix} \sqrt{5} & \frac{14\sqrt{5}}{5} \\ 0 & -\frac{3\sqrt{5}}{5} \\ 0 & \sqrt{5} \end{bmatrix}}_{{}_{\text{-}x}\tilde{\boldsymbol{R}}_{\mathbf{2}}^{\boldsymbol{e2}}}$$

$$= \underbrace{\begin{bmatrix} \frac{\sqrt{5}}{5} & \frac{3\sqrt{670}}{335} & -\frac{\sqrt{670}}{67} \\ \frac{\sqrt{5}}{5} & -\frac{9\sqrt{670}}{670} & -\frac{\sqrt{670}}{335} \\ \frac{\sqrt{5}}{5} & -\frac{2\sqrt{670}}{335} & \frac{\sqrt{670}}{670} \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{670}}{670} & \frac{2\sqrt{670}}{335} \\ \frac{\sqrt{5}}{5} & \frac{3\sqrt{670}}{335} & \frac{7\sqrt{670}}{670} \end{bmatrix}}_{\begin{bmatrix} {}_{\text{-}x}\boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e2}} & \boldsymbol{q} \end{bmatrix}} \underbrace{\begin{bmatrix} \sqrt{5} & \frac{14\sqrt{5}}{5} \\ 0 & \frac{17\sqrt{670}}{335} \\ 0 & 0 \end{bmatrix}}_{\begin{bmatrix} {}_{\text{-}x}\boldsymbol{R}_{\mathbf{2}}^{\boldsymbol{e2}} \\ \mathbf{0}^T \end{bmatrix}}$$

$$= \underbrace{\begin{bmatrix} \frac{\sqrt{5}}{5} & \frac{3\sqrt{670}}{335} \\ \frac{\sqrt{5}}{5} & -\frac{9\sqrt{670}}{670} \\ \frac{\sqrt{5}}{5} & -\frac{2\sqrt{670}}{335} \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{670}}{670} \\ \frac{\sqrt{5}}{5} & \frac{3\sqrt{670}}{335} \end{bmatrix}}_{{}_{\text{-}x}\boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e2}}} \underbrace{\begin{bmatrix} \sqrt{5} & \frac{14\sqrt{5}}{5} \\ 0 & \frac{17\sqrt{670}}{335} \end{bmatrix}}_{{}_{\text{-}x}\boldsymbol{R}_{\mathbf{2}}^{\boldsymbol{e2}}}.$$

Now we can remove the first row from ${}_{-x}\boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e2}}$ as before and maintain full rank. This would result in



$$\Rightarrow \underbrace{\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}}_{{}_{-x}\boldsymbol{B}_{\mathbf{2}}^{\boldsymbol{e3}}} = \underbrace{\begin{bmatrix} \frac{1}{2} & -\frac{3\sqrt{5}}{10} \\ \frac{1}{2} & -\frac{\sqrt{5}}{10} \\ \frac{1}{2} & \frac{\sqrt{5}}{10} \\ \frac{1}{2} & \frac{3\sqrt{5}}{10} \end{bmatrix}}_{{}_{-x}\boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e3}}} \underbrace{\begin{bmatrix} 2 & 5 \\ 0 & \sqrt{5} \end{bmatrix}}_{{}_{-x}\boldsymbol{R}_{\mathbf{2}}^{\boldsymbol{e3}}}.$$

Of course we do not actually want to calculate these decompositions we just want to update the coefficients like before. We can find the coefficients after removing a column by using the fact that ${}_{-x}\boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e2}} = \boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e2}}\underline{\boldsymbol{G}}^{T}$, therefore

$${}_{-x}\boldsymbol{\theta}_{\mathbf{2}}^{\boldsymbol{e2}} = {}_{-x}\boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e2}^T}\boldsymbol{t}_{\mathbf{2}}^{\boldsymbol{e2}} = \underline{\boldsymbol{G}}\boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e2}^T}\boldsymbol{t}_{\mathbf{2}}^{\boldsymbol{e2}}.$$

This can be used in conjunction with (4.9) to give the final coefficients

$${}_{-x}\boldsymbol{\theta}_{\mathbf{2}}^{\boldsymbol{e3}} = \begin{bmatrix} {}_{-x}\boldsymbol{\theta}_{\mathbf{2}}^{\boldsymbol{e2}} \\ \frac{t_{14} - \boldsymbol{\rho}^T\left({}_{-x}\boldsymbol{\theta}_{\mathbf{2}}^{\boldsymbol{e2}}\right)}{\sqrt{1-\boldsymbol{\rho}^T\boldsymbol{\rho}}} \end{bmatrix},$$

where $\boldsymbol{\rho} = \left({}_{-x}\boldsymbol{R}_{\mathbf{2}}^{\boldsymbol{e2}^T}\right)^{-1}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

Adding a column so that the system spans the full space of linear polynomials

When adding a row the system will never become rank deficient; however, it is possible that adding a row will allow a previously removed column to be added back to the system whilst maintaining full rank. If this column was not added then the system would no longer span all the desired space.

For example, suppose we have the QR decomposition of ${}_{-x}\boldsymbol{B}_{\mathbf{2}}^{\boldsymbol{e3}}$ and we wish to add a row to obtain the QR decomposition for the edge, $e4$, shown in Fig. 4.3(d). We would first add the row

$\begin{bmatrix} 1 & 1 \end{bmatrix}$ as usual to obtain

$$
\begin{array}{|c|c|c|c|}
\hline
\oslash & \oslash & 1 & 1 \\ \hline
\oslash & \oslash & \oslash & 1 \\ \hline
\oslash & \oslash & \oslash & 1 \\ \hline
\oslash & \oslash & \oslash & 1 \\ \hline
\end{array}
\quad
\begin{array}{|c|c|c|c|}
\hline
\oslash & \oslash & 1 & 1 \\ \hline
\oslash & \oslash & \oslash & 2 \\ \hline
\oslash & \oslash & \oslash & 3 \\ \hline
\oslash & \oslash & \oslash & 4 \\ \hline
\end{array}
\Rightarrow
\underbrace{\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}}_{{}_{\text{-}\boldsymbol{x}}\boldsymbol{B}_{\mathbf{2}}^{\boldsymbol{e4}}}
=
\underbrace{\begin{bmatrix}
\frac{\sqrt{5}}{5} & -\frac{3\sqrt{170}}{85} \\
\frac{\sqrt{5}}{5} & -\frac{3\sqrt{170}}{85} \\
\frac{\sqrt{5}}{5} & -\frac{\sqrt{170}}{170} \\
\frac{\sqrt{5}}{5} & \frac{2\sqrt{170}}{85} \\
\frac{\sqrt{5}}{5} & \frac{9\sqrt{170}}{170}
\end{bmatrix}}_{{}_{\text{-}\boldsymbol{x}}\boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e4}}}
\underbrace{\begin{bmatrix}
\sqrt{5} & \frac{11\sqrt{5}}{5} \\
0 & \frac{\sqrt{170}}{5}
\end{bmatrix}}_{{}_{\text{-}\boldsymbol{x}}\boldsymbol{R}_{\mathbf{2}}^{\boldsymbol{e4}}}.
$$

Of course, in practice we would actually just calculate ${}_{\text{-}\boldsymbol{x}}\boldsymbol{\theta}_{\mathbf{2}}^{\boldsymbol{e4}}$ and ${}_{\text{-}\boldsymbol{x}}\boldsymbol{R}_{\mathbf{2}}^{\boldsymbol{e4}}$ using the updates.

Visually, it is clear that in this example we can add the x column back to the system whilst maintaining full rank and mathematically this is possible because $\|\boldsymbol{b}\|_2 > \left\|\left({}_{\text{-}\boldsymbol{x}}\boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e4}}\right)^T \boldsymbol{b}\right\|_2$, where $\boldsymbol{b} = \begin{bmatrix} 3 & 4 & 4 & 4 & 4 \end{bmatrix}^T$ is the column we are trying to add.

We could use the Gram Schmidt process to add the column to the right hand side of ${}_{\text{-}\boldsymbol{x}}\boldsymbol{B}_{\mathbf{2}}^{\boldsymbol{e4}}$:

$$
\begin{bmatrix} {}_{\text{-}\boldsymbol{x}}\boldsymbol{B}_{\mathbf{2}}^{\boldsymbol{e4}} & \boldsymbol{b} \end{bmatrix} = \begin{bmatrix} {}_{\text{-}\boldsymbol{x}}\boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e4}} & \boldsymbol{q} \end{bmatrix} \begin{bmatrix} {}_{\text{-}\boldsymbol{x}}\boldsymbol{R}_{\mathbf{2}}^{\boldsymbol{e4}} & \boldsymbol{r_x} \\ \mathbf{0}^T & r_{dd} \end{bmatrix},
$$

where $\boldsymbol{q} = \dfrac{\boldsymbol{b} - \left({}_{\text{-}\boldsymbol{x}}\boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e4}}\right)\left({}_{\text{-}\boldsymbol{x}}\boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e4}}\right)^T \boldsymbol{b}}{r_{dd}}$, $\boldsymbol{r_x} = \left({}_{\text{-}\boldsymbol{x}}\boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e4}}\right)^T \boldsymbol{b}$ and $r_{dd} = \left\|\boldsymbol{b} - \left({}_{\text{-}\boldsymbol{x}}\boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e4}}\right)\left({}_{\text{-}\boldsymbol{x}}\boldsymbol{Q}_{\mathbf{2}}^{\boldsymbol{e4}}\right)^T \boldsymbol{b}\right\|_2 = \sqrt{\|\boldsymbol{b}\|^2 - \|\boldsymbol{r_x}\|^2}$.

In fact, we would like to keep the coefficients in the same order so we insert the x column back into the middle location. This can be done by splitting ${}_{\text{-}\boldsymbol{x}}\boldsymbol{R}_{\mathbf{2}}^{\boldsymbol{e4}}$ into two matrices, $\boldsymbol{R_L}$ and $\boldsymbol{R_R}$, either side of this location. We can then insert $\boldsymbol{b}$ in between the two columns of ${}_{\text{-}\boldsymbol{x}}\boldsymbol{B}_{\mathbf{2}}^{\boldsymbol{e4}}$ by also inserting $\begin{bmatrix} \boldsymbol{r_x} \\ r_{dd} \end{bmatrix}$ in between $\boldsymbol{R_L}$ and $\boldsymbol{R_R}$ (note that $\boldsymbol{R_L}$ and $\boldsymbol{R_R}$ are vectors in our 3×3 example but are uppercase because in general they are matrices):

∅	∅	1	1
∅	∅	∅	1
∅	∅	∅	1
∅	∅	∅	1

∅	∅	3	4
∅	∅	∅	4
∅	∅	∅	4
∅	∅	∅	4

∅	∅	1	1
∅	∅	∅	2
∅	∅	∅	3
∅	∅	∅	4

$$\Rightarrow \underbrace{\begin{bmatrix}1&3&1\\1&4&1\\1&4&2\\1&4&3\\1&4&4\end{bmatrix}}_{\boldsymbol{B_2^{e4}}} = \underbrace{\begin{bmatrix}\frac{\sqrt{5}}{5} & -\frac{3\sqrt{170}}{85} & -\frac{\sqrt{170}}{17}\\ \frac{\sqrt{5}}{5} & -\frac{3\sqrt{170}}{85} & \frac{7\sqrt{170}}{170}\\ \frac{\sqrt{5}}{5} & -\frac{\sqrt{170}}{170} & \frac{2\sqrt{170}}{85}\\ \frac{\sqrt{5}}{5} & \frac{2\sqrt{170}}{85} & \frac{\sqrt{170}}{170}\\ \frac{\sqrt{5}}{5} & \frac{9\sqrt{170}}{170} & -\frac{\sqrt{170}}{85}\end{bmatrix}}_{\begin{bmatrix} {}_{\text{-}x}\boldsymbol{Q_2^{e4}} & \boldsymbol{q}\end{bmatrix}} \underbrace{\begin{bmatrix}\sqrt{5} & \frac{19\sqrt{5}}{5} & \frac{11\sqrt{5}}{5}\\ 0 & \frac{3\sqrt{170}}{85} & \frac{\sqrt{170}}{5}\\ 0 & \frac{\sqrt{170}}{17} & 0\end{bmatrix}}_{\begin{bmatrix}\boldsymbol{R_L} & \boldsymbol{r_x} & \boldsymbol{R_R}\\ 0 & r_{dd} & 0\end{bmatrix}}.$$

Currently the right hand factor is not upper triangular but this can be easily fixed with a Givens rotation matrix:

$$\underbrace{\begin{bmatrix}1 & 0 & 0\\ 0 & \frac{3\sqrt{34}}{34} & \frac{5\sqrt{34}}{34}\\ 0 & \frac{5\sqrt{34}}{34} & -\frac{3\sqrt{34}}{34}\end{bmatrix}}_{\boldsymbol{G}} \underbrace{\begin{bmatrix}\sqrt{5} & \frac{19\sqrt{5}}{5} & \frac{11\sqrt{5}}{5}\\ 0 & \frac{3\sqrt{170}}{85} & \frac{\sqrt{170}}{5}\\ 0 & \frac{\sqrt{170}}{17} & 0\end{bmatrix}}_{\begin{bmatrix}\boldsymbol{R_L} & \boldsymbol{r_x} & \boldsymbol{R_R}\\ 0 & r_{dd} & 0\end{bmatrix}} = \underbrace{\begin{bmatrix}\sqrt{5} & \frac{19\sqrt{5}}{5} & \frac{11\sqrt{5}}{5}\\ 0 & \frac{2\sqrt{5}}{5} & \frac{3\sqrt{5}}{5}\\ 0 & 0 & \sqrt{5}\end{bmatrix}}_{\boldsymbol{R_2^{e4}}}.$$

The final QR decomposition is then given by

$$\underbrace{\begin{bmatrix} 1 & 3 & 1 \\ 1 & 4 & 1 \\ 1 & 4 & 2 \\ 1 & 4 & 3 \\ 1 & 4 & 4 \end{bmatrix}}_{\boldsymbol{B_2^{e4}}} = \underbrace{\begin{bmatrix} \frac{\sqrt{5}}{5} & -\frac{3\sqrt{170}}{85} & -\frac{\sqrt{170}}{17} \\ \frac{\sqrt{5}}{5} & -\frac{3\sqrt{170}}{85} & \frac{7\sqrt{170}}{170} \\ \frac{\sqrt{5}}{5} & -\frac{\sqrt{170}}{170} & \frac{2\sqrt{170}}{85} \\ \frac{\sqrt{5}}{5} & \frac{2\sqrt{170}}{85} & \frac{\sqrt{170}}{170} \\ \frac{\sqrt{5}}{5} & \frac{9\sqrt{170}}{170} & -\frac{\sqrt{170}}{85} \end{bmatrix}}_{\begin{bmatrix} {}_{\boldsymbol{-x}}\boldsymbol{Q_2^{e4}} & \boldsymbol{q} \end{bmatrix}} \boldsymbol{G^T G} \underbrace{\begin{bmatrix} \sqrt{5} & \frac{19\sqrt{5}}{5} & \frac{11\sqrt{5}}{5} \\ 0 & \frac{3\sqrt{170}}{85} & \frac{\sqrt{170}}{5} \\ 0 & \frac{\sqrt{170}}{17} & 0 \end{bmatrix}}_{\begin{bmatrix} \boldsymbol{R_L} & \boldsymbol{r_x} & \boldsymbol{R_R} \\ 0 & r_{dd} & 0 \end{bmatrix}}$$

$$= \underbrace{\begin{bmatrix} \frac{\sqrt{5}}{5} & -\frac{2\sqrt{5}}{5} & 0 \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{5}}{10} & -\frac{3\sqrt{5}}{10} \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{5}}{10} & -\frac{\sqrt{5}}{10} \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{5}}{10} & \frac{\sqrt{5}}{10} \\ \frac{\sqrt{5}}{5} & \frac{\sqrt{5}}{10} & \frac{3\sqrt{5}}{10} \end{bmatrix}}_{\boldsymbol{Q_2^{e4}}} \underbrace{\begin{bmatrix} \sqrt{5} & \frac{19\sqrt{5}}{5} & \frac{11\sqrt{5}}{5} \\ 0 & \frac{2\sqrt{5}}{5} & \frac{3\sqrt{5}}{5} \\ 0 & 0 & \sqrt{5} \end{bmatrix}}_{\boldsymbol{R_2^{e4}}}.$$

We can also generate the final coefficient updating formula as we have done before:

$$\begin{aligned} \boldsymbol{\theta_2^{e4}} &= \boldsymbol{Q_2^{e4}}^T \boldsymbol{t_2^{e4}} \\ &= \left(\begin{bmatrix} {}_{\boldsymbol{-x}}\boldsymbol{Q_2^{e4}} & \frac{\boldsymbol{b} - \left({}_{\boldsymbol{-x}}\boldsymbol{Q_2^{e4}}\right)\left({}_{\boldsymbol{-x}}\boldsymbol{Q_2^{e4}}\right)^T \boldsymbol{b}}{r_{dd}} \end{bmatrix} \boldsymbol{G}^T \right)^T \boldsymbol{t_2^{e4}} \\ &= \boldsymbol{G} \begin{bmatrix} \left({}_{\boldsymbol{-x}}\boldsymbol{Q_2^{e4}}\right)^T \boldsymbol{t_2^{e4}} \\ \left(\frac{\boldsymbol{b} - \left({}_{\boldsymbol{-x}}\boldsymbol{Q_2^{e4}}\right)\boldsymbol{r_x}}{r_{dd}} \right)^T \boldsymbol{t_2^{e4}} \end{bmatrix} \\ &= \boldsymbol{G} \begin{bmatrix} {}_{\boldsymbol{-x}}\boldsymbol{\theta_2^{e4}} \\ \frac{\boldsymbol{b}^T \boldsymbol{t_2^{e4}} - \boldsymbol{r_x}^T \left({}_{\boldsymbol{-x}}\boldsymbol{\theta_2^{e4}}\right)}{\sqrt{\|\boldsymbol{b}\|^2 - \|\boldsymbol{r_x}\|^2}} \end{bmatrix}. \end{aligned}$$

For this update we need $\boldsymbol{b}^T \boldsymbol{t_2^{e4}}$, $\|\boldsymbol{b}\|_2^2$ and $\boldsymbol{r_x} = \left({}_{\boldsymbol{-x}}\boldsymbol{Q_2^{e4}}\right)^T \boldsymbol{b}$ which are computationally expensive to calculate from scratch every time. This can be prevented by observing that these expressions can be extracted from the QR decomposition very easily: let $\boldsymbol{B} = \boldsymbol{QR}$ be a general thin QR decomposition, then

$$\boldsymbol{B}^T \boldsymbol{t} = (\boldsymbol{QR})^T \boldsymbol{t} = \boldsymbol{R}^T \boldsymbol{\theta},$$

therefore $\boldsymbol{b_i}^T \boldsymbol{t} = \boldsymbol{r_i}^T \boldsymbol{\theta}$ where $\boldsymbol{b_i}$ and $\boldsymbol{r_i}$ are the i-th columns of $\boldsymbol{B}$ and $\boldsymbol{R}$ respectively. Furthermore, using the same notation,

$$\boldsymbol{Q}^T\boldsymbol{B} = \boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{R} = \boldsymbol{R},$$

and we, thus, have $\boldsymbol{Q}^T\boldsymbol{b_i} = \boldsymbol{r_i}$. Finally, $\boldsymbol{B} = \boldsymbol{Q}\boldsymbol{R}$ implies that $\boldsymbol{b_i} = \boldsymbol{Q}\boldsymbol{r_i}$, therefore $\|\boldsymbol{b_i}\|_2^2 = \|\boldsymbol{r_i}\|_2^2$.

When we are adding a column, the column we are trying to add is by definition not present, so at first glance these results are useless. However, we can extract these expressions just before we remove the column from the system and then keep them up to date while the column is not part of the system. The column can then be added back to the system whenever it is needed. Updating $\boldsymbol{b}^T\boldsymbol{t}$ and $\|\boldsymbol{b}\|_2^2$ is trivial and $\boldsymbol{r_x} = \boldsymbol{Q}^T\boldsymbol{x}$ can be updated in the same way as the coefficients (note that $\boldsymbol{Q}^T\boldsymbol{x}$ is in the same form as the coefficients $\boldsymbol{Q}^T\boldsymbol{t}$).

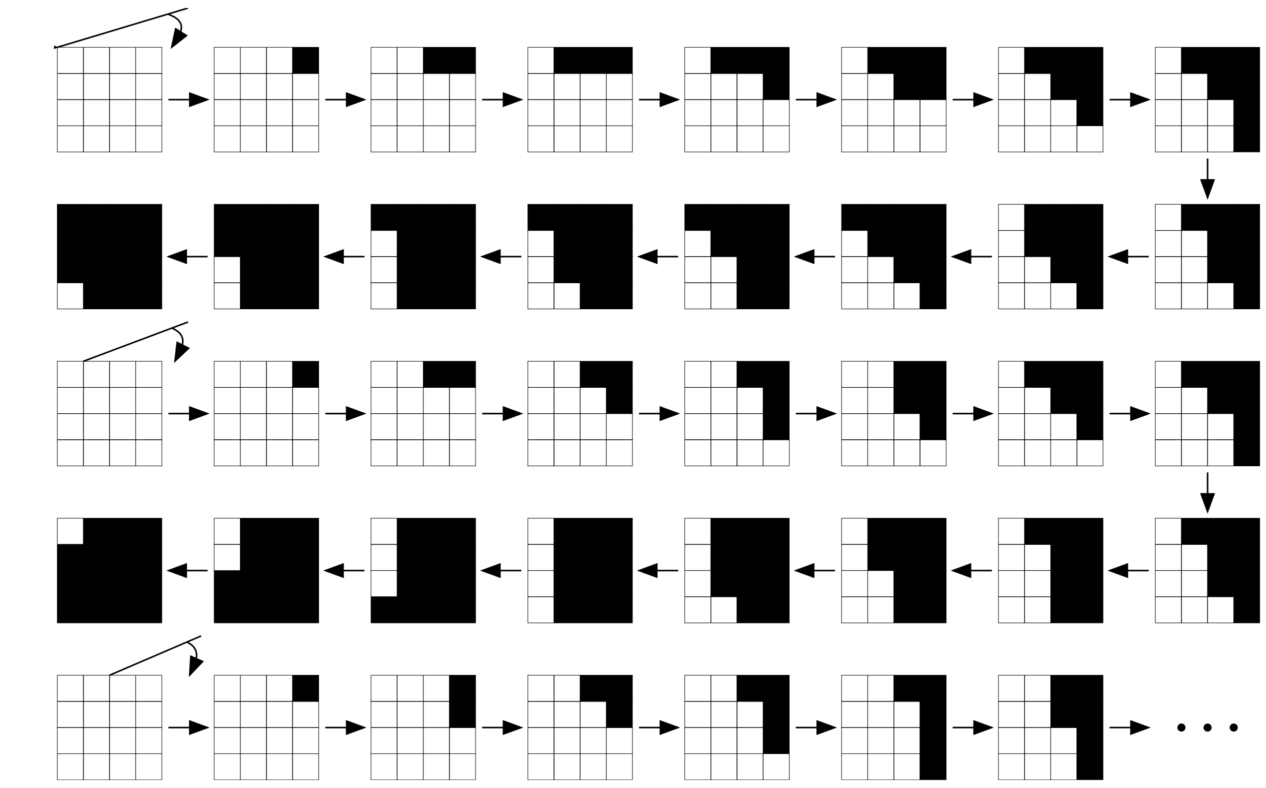
In the next subsection we will construct a dictionary of edges that can be exhaustively searched using this idea of adding and removing pixels.

4.4.4. Constructing a suitable dictionary of edges

We now describe a possible dictionary that can be exhaustively searched efficiently using the previous analysis.

There are many possible ways to construct a dictionary of edges that meet the above requirement. Figure 4.4 shows part of a possible dictionary of 4×4 tiles. This dictionary is created by initialising the tile so that all pixels are on the same side of the edge (the global 'no edge' case). Then, a straight line is rotated clockwise around a particular point on the boundary and when the line crosses the centre point of a pixel it is moved to the other side of the edge. When all pixels have been moved to the other side of the edge, we are back to the global 'no edge' case. The point of rotation is then moved one discrete step clockwise around the tile boundary and the process repeated. For example, the first two rows of Fig. 4.4(a) correspond to rotating the line around the top left corner, $(0, 0)$, and the next two rows correspond to rotating the edge around the point, $(0, 1)$, one step to the right.

Note that it is possible to check all these edges in one continuous chain; however, resetting to the 'no edge' case, when possible, reduces rounding errors. Additionally this strategy allows the dictionary size to be reduced. As previously proposed, the dictionary has $4n^3$ edges for an $n \times n$ tile, since there are $4n$ rotation points and n^2 edges per point. This can be reduced to $2n^3+n^2/2$ if we stop rotating the edge when we reach a boundary point we have already used as a rotation point. The


(a) The start of a possible 4 × 4 dictionary of edges.

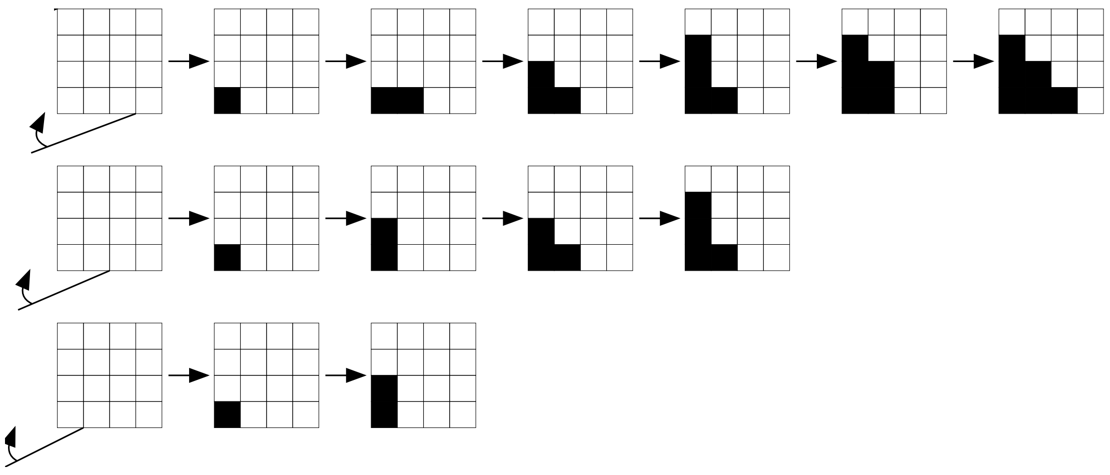

(b) The end of a possible 4 × 4 dictionary of edges.

Figure 4.4.: Part of a possible dictionary of 4 × 4 edge tiles that can be searched with the proposed strategy.

intuition is that, since we have already checked edges starting and ending from approximately these locations, the vast majority of future edges will already have been checked. Figure 4.4(b) shows the last few edges of the same dictionary, when this strategy is employed. Despite the reduced size, the number of edges still has cubic growth, so we only exhaustively search tiles up to a size of 32 × 32. Larger tiles are down sampled and then exhaustively searched. This produces a rough approximation of the edge discontinuity which is refined at the larger, original tile size.

As can be seen in Fig. 4.4, constructing a dictionary of edges such that each edge differs from the previous entry in only one pixel results in checking some edges more than once. The reduced dictionary, just described, reduces this replication but it is still present. However, since the computational cost of moving a pixel from one side of the edge to the other is so cheap, we can tolerate this replication.

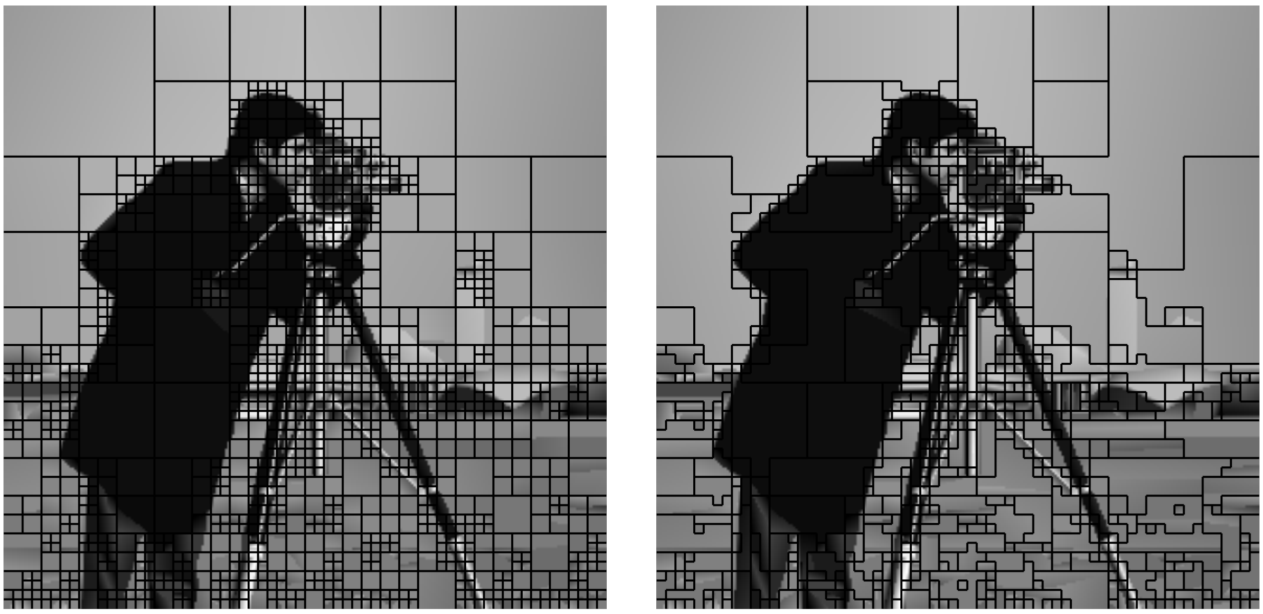
In many cases, we can further reduce the computation by pre-computing and storing the Givens matrices, for the whole dictionary of possible edges, offline. This makes the computation required to update the coefficients from one edge to the next very small. In practise we precompute and store these Givens matrices for square tiles up to 32×32 used in the pruning algorithm, but not for the more flexible regions that can be obtained when joining.

4.5. Example of the speed and sparsity capabilities of the proposed method

To conclude this section we show an example of the speed improvement that is obtained by updating the QR decomposition in the way that we have presented. We also use the example to demonstrate the sparsity of our model. Figure 4.5 shows two approximations of the cameraman image that have a PSNR of 30dB. The first approximation was calculated in 1.0 seconds using the prune only model and the second was calculated in 8.3 seconds using the more complex prune-join model[1]; for comparison these approximations would take around 100 and 10000 seconds if we calculated the QR decomposition from scratch each time. In both cases $d = 1$; namely, the tiles are either piecewise linear or a single linear polynomial.

For a rough comparison of sparsity, the prune and prune-join models use 3602 and 2753 polynomial coefficients respectively, whereas a Daubechies 4 tap wavelet decomposition would require 4712 coefficients to achieve the same approximation error. This greater sparsity should aid us in restoration, particularly in cases of high degradation where a strong prior is required.

[1]All calculations were made in MATLAB on a 2.2GHz Intel Core i7 Macbook Pro with 4GB of RAM (no multi-threading).



(a) Reconstruction using the prune model only. (b) Reconstruction using the prune-join model.

Figure 4.5.: Approximation of the cameraman image, with associated tilings, to a PSNR of 30dB using the prune and prune-join models.

4.6. Summary

In this chapter we have proposed a novel approximation algorithm that generates very sparse approximations of piecewise polynomial images. The algorithm uses a quadtree decomposition with an additional joining to adaptively partition the image. Each of these adaptive regions is approximated by a very low dimensional model, namely a 2-D piecewise polynomial with at most one continuous discontinuity. Traditionally it is very computationally expensive to search for a suitable discontinuity for each region. We propose a fast exhaustive search that, given the previous approximation, calculates the next approximation by very efficiently moving one pixel to the other side of the discontinuity. This vastly reduces the computation time, which, when we look at restoration, will make cycle spinning feasible.

Since images are often well approximated by piecewise polynomial functions, our method can be used over a wide range of images. In the next chapter, we will develop restoration and enhancement algorithms that exploit the sparsity of our model and present results for a wide range of images.

CHAPTER 5

Image Restoration and Enhancement using a Quadtree Decomposition to Sparsely Represent Piecewise Polynomial Functions

5.1. Introduction

In Chapters 2 and 3 we presented sparse approximation techniques for images and showed how they could be used for restoration and enhancement. In Chapter 4 we proposed a novel algorithm that provided very sparse approximations of piecewise polynomial images. In this chapter, using a similar approach to Chapter 3, we will show how this new transform can be used to tackle the same restoration and enhancement problems. Although our algorithms are particularly suited to piecewise polynomial images, we will provide a thorough analysis for natural and depth images, which have practical applications. Since depth images are approximately piecewise smooth, we expect to perform well on these images. Furthermore, in cases of high degradation, where a strong prior is required, a piecewise polynomial model will be more appropriate to a wider range of images.

5.2. Denoising

Recall that the denoising problem can be modelled by

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{z},$$

where $\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^N$ are the measured, desired and noise images respectively (N is the number of pixels). The approximation algorithm of the previous chapter solved the minimisation problem

$$\hat{\boldsymbol{x}} = arg \min_{\boldsymbol{x}} \left\{ \|\boldsymbol{y} - \boldsymbol{x}\|_2^2 + \lambda P^x(\boldsymbol{x}) \right\}. \tag{5.1}$$

Equation (5.1), with $\lambda = \zeta \sigma_z^2$, can be interpreted as the MAP estimator of the denoising problem, when $\boldsymbol{z}$ is white Gaussian, if

$$p(\boldsymbol{x}) = Aexp\left[-\frac{\zeta P^x(\boldsymbol{x})}{2} \right],$$

where $P^x(\boldsymbol{x})$ is the penalty given in (4.2) and A is a constant so that

$$\sum_{\boldsymbol{x} \in \mathbb{R}^N} Aexp\left[-\frac{\zeta P^x(\boldsymbol{x})}{2} \right] = 1.$$

Therefore, using the $\boldsymbol{\theta}$ notation of the previous chapter, our denoising algorithm aims to solve

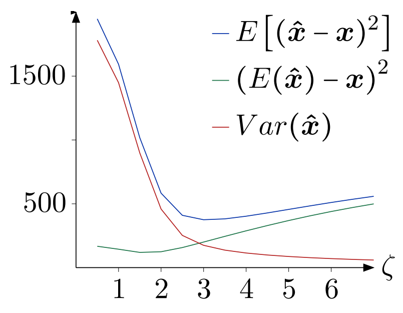
$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta}} \left\{ \|\boldsymbol{y} - D(\boldsymbol{\theta})\|_2^2 + \zeta \sigma_z^2 P^\theta(\boldsymbol{\theta}) \right\}, \tag{5.2}$$

using the same approximation described in Chapter 4.

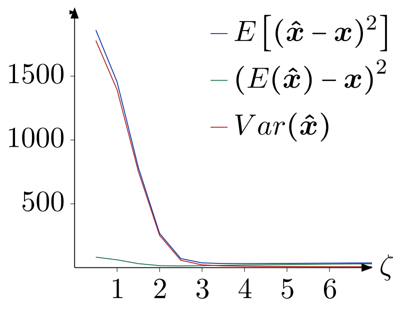
Our quadtree decomposition approximation technique is shift variant, which can be exploited with cycle spinning. We can reduce the complexity of computing an approximation for each new shift by noticing that $N \times N$ tiles only have N^2 unique shifts. For example 2×2 tiles only have four unique shifts. This means that only the first four shifts have to calculate 2×2 tiles and all future shifts can simply look up these results from previous trees.

When measuring the performance of our estimators, we will mainly be concerned with the PSNR and SSIM, but the bias and variance are also of interest. It is well known and easy to prove that the sum of the squared bias and the variance is equal to the MSE. Figure 5.1 shows the MSE, squared bias and variance plotted against ζ for different noise levels and images[1]. When ζ is small we use a more precise model and thus have a low bias and a high variance and for larger ζ the opposite is true. As ζ increases, the variance drops suddenly before slowly decreasing, and the squared bias slowly increases. Consequently, the optimum MSE occurs when ζ is just above 3 but there is not a huge performance difference for $3 < \zeta < 6$. It is favourable that the optimum ζ is similar for the three different scenarios, since this suggests that a fixed value can be used for all denoising problems. In

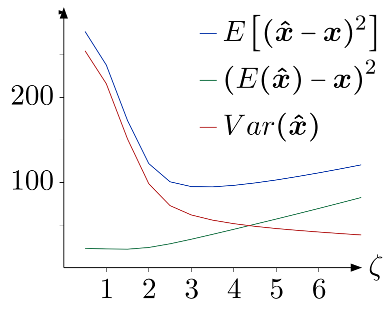
[1]These results were calculated with a Monte-Carlo simulation with 50 noise realisations.



(a) Calculated over a 128×128 section of the cameraman image with $\sigma_z = 75$.



(b) Calculated over a 128×128 section of the baby image with $\sigma_z = 75$.



(c) Calculated over a 128×128 section of the cameraman image with $\sigma_z = 25$.

Figure 5.1.: MSE, squared bias and variance for the proposed denoising estimator, plotted against the regularisation parameter ζ. Here 16^2 shifts of cycle spinning were used with the prune only model.

the following denoising simulations we use $\zeta = 3.3$, 16^2 shifts of cycle spinning and, for increased speed, the prune only model.

Tables 5.1 and 5.2 show PSNR and SSIM comparisons of the proposed method with a number of leading denoising algorithms for natural images. In terms of PSNR, our algorithm is competitive for all images and state of the art when the degradation is high. The SSIM index favours our algorithm even more, suggesting that our algorithm produces visually pleasing results. This can be further verified in Figs. 5.2 - 5.5, which show examples of the denoising results. For low noise the PCA of the BM3D-SAPCA algorithm performs very well. This is demonstrated, in Fig. 5.3, by its improved reconstruction of the hair and nasolabial line in the man image. However, when the noise degradation is increased the PCA fails to choose a suitable basis and, as shown in Fig. 5.5, the BM3D-SAPCA's result is heavily distorted. In this case, the proposed method and BM3D filter produce the best SSIM index and PSNR respectively. One can observe that the proposed method maintains smooth regions with sharp discontinuities, which will be particularly appropriate for depth images.

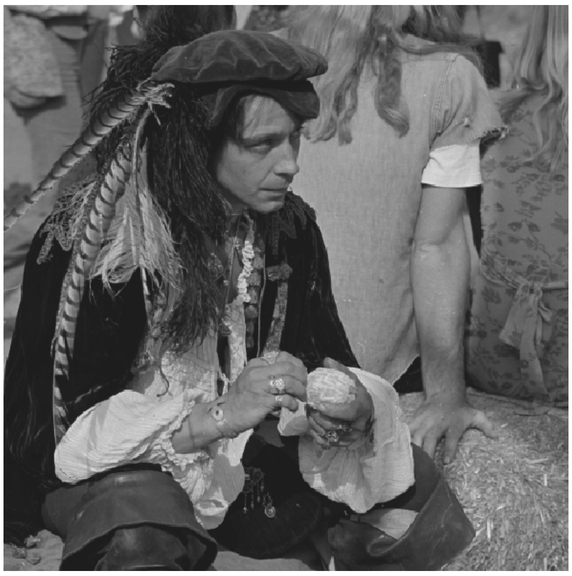
Tables 5.3 - 5.4 and Figs. 5.6 - 5.7 show similar analysis for depth images. Since this class of images is closer to our model, state of the art results are obtained in almost all cases. The performance improvement achieved by our algorithm is clearly visible in the figures, particularly at high noise levels.

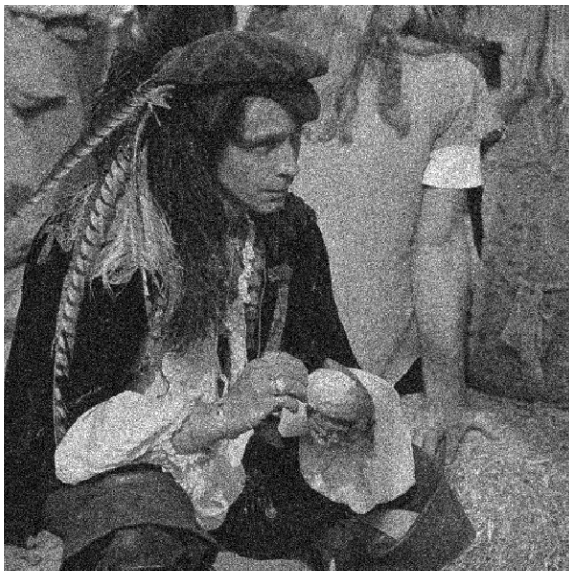
	Degraded		Denoising Results								
Image	σ_z	Degraded	BM3DSAPCA	BM3D	Proposed	SADCT	PLOW	KSVD	Wavelet CS	Curvelet	Wavelet
Cameraman (256 × 256)	10	28.10	**34.59**	34.18	33.24	33.70	33.17	33.74	32.59	29.39	30.02
	25	20.14	**29.81**	29.45	29.00	28.96	28.59	29.00	27.49	25.66	24.90
	50	14.12	**26.58**	26.12	26.05	25.67	25.61	25.76	23.98	23.48	21.48
	75	10.60	**24.41**	24.33	24.35	23.67	23.61	23.41	22.07	22.16	19.49
	100	8.10	22.88	23.07	**23.08**	22.34	22.22	21.56	20.71	21.10	18.09
Lena (512 × 512)	10	28.14	**36.07**	35.93	35.24	35.57	35.32	35.57	34.50	33.37	31.74
	25	20.18	**32.22**	32.08	31.40	31.65	31.87	31.37	30.21	30.07	27.34
	50	14.16	**29.07**	29.05	28.65	28.54	28.66	27.82	26.85	27.19	23.88
	75	10.63	26.83	**27.26**	27.11	26.78	26.55	25.82	24.85	25.44	21.68
	100	8.14	25.37	25.95	**26.00**	25.60	25.08	24.54	23.39	24.22	20.05
Boat (512 × 512)	10	28.14	**34.10**	33.92	33.34	33.44	33.02	33.69	32.66	30.60	29.97
	25	20.18	**30.03**	29.91	29.35	29.31	29.53	29.34	28.28	27.51	25.54
	50	14.16	**26.89**	26.78	26.55	26.15	26.60	25.92	25.08	25.18	22.35
	75	10.63	24.92	**25.12**	25.01	24.54	24.69	24.02	23.31	23.78	20.52
	100	8.14	23.69	**23.97**	23.97	23.51	23.33	22.85	22.07	22.79	19.17
Hill (512 × 512)	10	28.14	**33.83**	33.62	33.05	33.20	32.62	33.38	32.21	30.55	29.78
	25	20.18	**29.96**	29.85	29.36	29.36	29.56	29.22	28.25	27.79	25.82
	50	14.16	**27.20**	27.19	26.94	26.82	26.95	26.30	25.46	25.64	22.84
	75	10.63	25.42	**25.68**	25.63	25.46	25.24	24.89	23.83	24.31	21.00
	100	8.14	24.27	24.58	**24.66**	24.54	24.06	24.01	22.61	23.33	19.58
Man (512 × 512)	10	28.14	**34.25**	33.98	33.57	33.55	32.98	33.64	32.66	30.81	29.97
	25	20.18	**29.81**	29.62	29.39	29.16	29.33	29.12	28.17	27.57	25.70
	50	14.16	**26.94**	26.81	26.75	26.33	26.56	26.08	25.20	25.12	22.68
	75	10.63	25.13	**25.32**	25.28	24.89	24.86	24.44	23.54	23.70	20.81
	100	8.14	23.96	24.22	**24.31**	23.96	23.68	23.42	22.32	22.72	19.44
Peppers (256 × 256)	10	28.10	**34.94**	34.68	34.45	34.37	33.60	34.29	33.62	30.81	30.60
	25	20.14	**30.43**	30.16	30.11	29.83	29.63	29.71	28.54	25.93	25.28
	50	14.12	**27.00**	26.68	26.83	26.33	26.38	26.08	24.68	23.29	21.47
	75	10.60	24.74	24.73	**24.94**	24.35	24.26	23.64	22.40	22.01	19.37
	100	8.10	23.24	23.39	**23.61**	22.98	22.76	21.96	20.95	20.99	18.00
Average			**28.29**	28.25	28.04	27.82	27.68	27.49	26.42	25.88	23.62

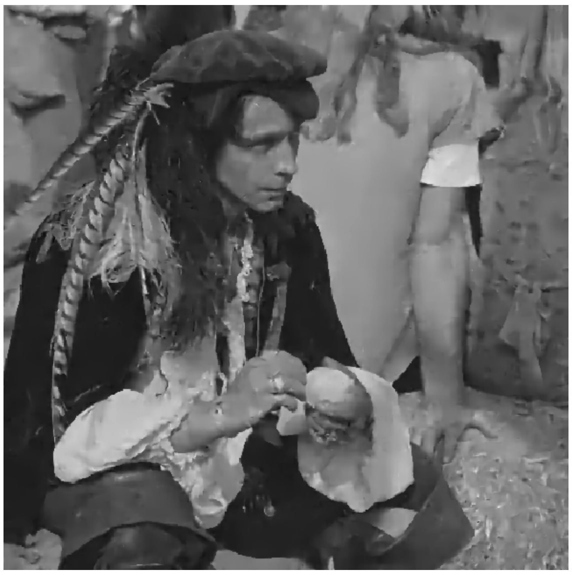
Table 5.1.: A PSNR comparison of various denoising algorithms for natural images, with the best result shown in bold. The algorithms, in order of average performance, are BM3D-SAPCA [19], BM3D [17], Proposed, SADCT [35], PLOW [14], KSVD [29], Wavelet CS (hard thresholding with cycle spinning), Curvelet [11] and Wavelet (hard thresholding). Finally, σ_z is the standard deviation of the noise.

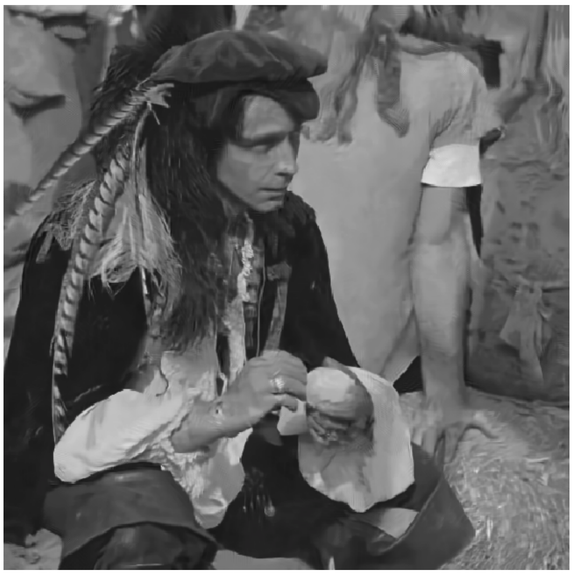
	Degraded		Denoising Results								
Image	σ_z	Degraded	BM3D	Proposed	BM3DSAPCA	SADCT	KSVD	PLOW	Curvelet	Wavelet CS	Wavelet
Cameraman (256 × 256)	10	0.6326	0.9319	0.9261	**0.9352**	0.9250	0.9270	0.9187	0.8502	0.9050	0.8519
	25	0.3360	0.8544	0.8484	**0.8642**	0.8473	0.8402	0.8400	0.7409	0.7920	0.7069
	50	0.1787	0.7824	0.7834	**0.7868**	0.7733	0.7497	0.7453	0.6493	0.6674	0.5774
	75	0.1125	0.7341	**0.7442**	0.7051	0.7186	0.6636	0.6224	0.5881	0.5788	0.5006
	100	0.0762	0.6924	**0.7102**	0.6445	0.6713	0.5775	0.5294	0.5285	0.5041	0.4419
Lena (512 × 512)	10	0.6136	0.9166	0.9075	**0.9183**	0.9129	0.9118	0.9077	0.8858	0.8939	0.8385
	25	0.2709	0.8607	0.8517	**0.8650**	0.8555	0.8436	0.8571	0.8203	0.8115	0.7222
	50	0.1120	0.7994	0.7962	**0.8014**	0.7908	0.7612	0.7754	0.7426	0.7059	0.6102
	75	0.0600	0.7516	**0.7597**	0.7247	0.7457	0.6973	0.6943	0.6805	0.6282	0.5422
	100	0.0367	0.7090	**0.7317**	0.6747	0.7114	0.6451	0.6312	0.6284	0.5704	0.5020
Boat (512 × 512)	10	0.6913	0.8878	0.8763	**0.8923**	0.8761	0.8834	0.8661	0.8300	0.8574	0.7916
	25	0.3452	0.8014	0.7832	**0.8039**	0.7827	0.7723	0.7909	0.7323	0.7351	0.6362
	50	0.1549	0.7053	0.6905	**0.7082**	0.6781	0.6571	0.6910	0.6338	0.6071	0.5053
	75	0.0864	**0.6410**	0.6341	0.6276	0.6156	0.5812	0.6024	0.5689	0.5258	0.4408
	100	0.0542	0.5936	**0.5948**	0.5789	0.5750	0.5286	0.5308	0.5203	0.4695	0.4039
Hill (512 × 512)	10	0.6945	0.8834	0.8683	**0.8896**	0.8722	0.8778	0.8565	0.8137	0.8405	0.7672
	25	0.3254	0.7748	0.7522	**0.7788**	0.7532	0.7392	0.7633	0.7002	0.6996	0.6018
	50	0.1288	0.6747	0.6559	**0.6756**	0.6512	0.6222	0.6567	0.6011	0.5746	0.4803
	75	0.0662	**0.6118**	0.6034	0.5983	0.5946	0.5644	0.5764	0.5395	0.4999	0.4218
	100	0.0395	0.5650	**0.5654**	0.5492	0.5576	0.5279	0.5171	0.4905	0.4477	0.3855
Man (512 × 512)	10	0.6797	0.9076	0.9008	**0.9125**	0.8970	0.9020	0.8879	0.8443	0.8769	0.8039
	25	0.3303	0.8047	0.7961	**0.8111**	0.7849	0.7798	0.7967	0.7249	0.7413	0.6366
	50	0.1407	0.7056	0.6995	**0.7107**	0.6805	0.6652	0.6858	0.6187	0.6095	0.5062
	75	0.0758	**0.6445**	0.6421	0.6308	0.6236	0.5967	0.6002	0.5518	0.5296	0.4415
	100	0.0465	0.5978	**0.6049**	0.5793	0.5867	0.5502	0.5397	0.5033	0.4741	0.4051
Peppers (256 × 256)	10	0.6782	0.9282	0.9272	**0.9287**	0.9235	0.9241	0.9186	0.8792	0.9142	0.8558
	25	0.3511	0.8676	**0.8710**	0.8690	0.8600	0.8563	0.8571	0.7912	0.8253	0.7077
	50	0.1678	0.7936	**0.8039**	0.7942	0.7800	0.7715	0.7570	0.6931	0.7110	0.5724
	75	0.0967	0.7368	**0.7575**	0.7220	0.7223	0.6906	0.6595	0.6304	0.6253	0.4887
	100	0.0615	0.6881	**0.7207**	0.6726	0.6782	0.6237	0.5901	0.5752	0.5587	0.4407
Average			**0.7615**	0.7602	0.7551	0.7482	0.7244	0.7222	0.6786	0.6727	0.5862

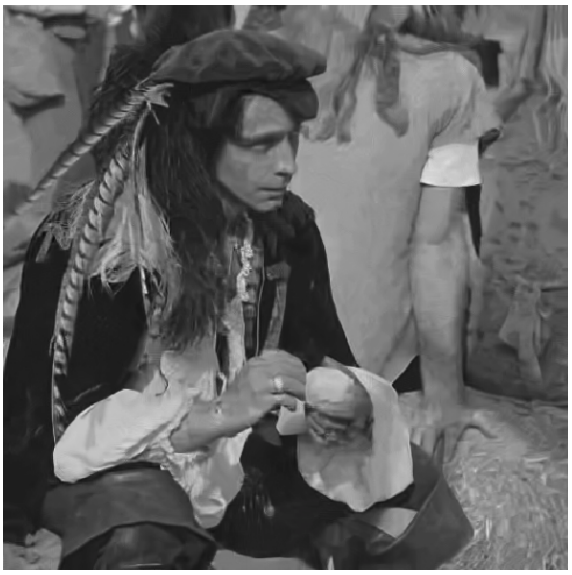
Table 5.2.: A SSIM index comparison of various denoising algorithms for natural images, with the best result shown in bold. The algorithms, in order of average performance, are Proposed, BM3D [17], BM3DSAPCA [19], SADCT [35], KSVD [29], PLOW [14], Curvelet [11], Wavelet CS (hard thresholding with cycle spinning) and Wavelet (hard thresholding). Finally, σ_z is the standard deviation of the noise.


(a) Original.


(b) Noisy, PSNR = 10.63dB, SSIM = 0.3303 (σ_z = 25).


(c) Denoised as proposed. PSNR = 29.39dB, SSIM = 0.7961.


(d) Denoised with BM3D-SAPCA [19]. PSNR = 29.81dB, SSIM = 0.8111.


(e) Denoised with BM3D [17], PSNR = 29.62dB, SSIM = 0.8047.

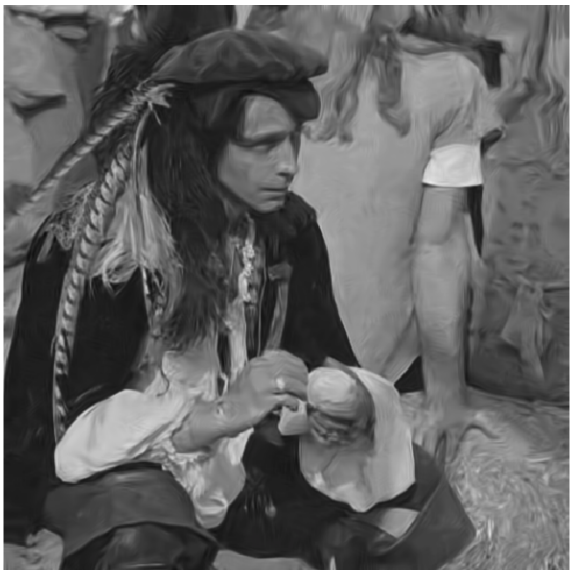
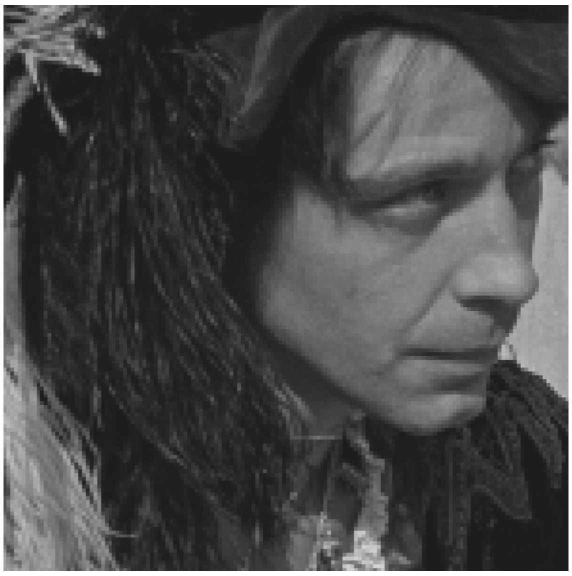
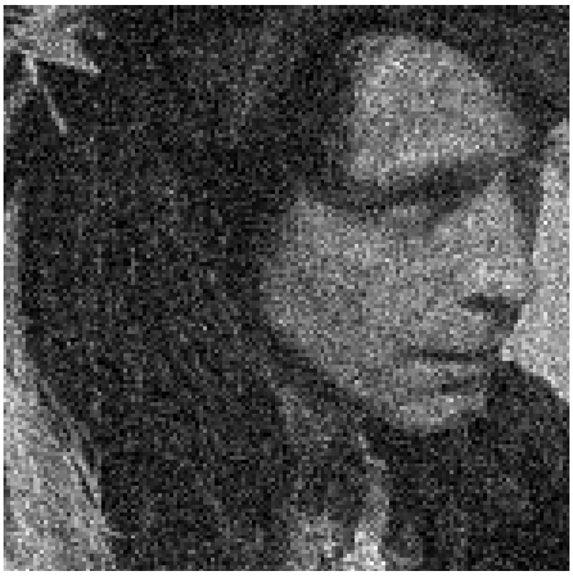
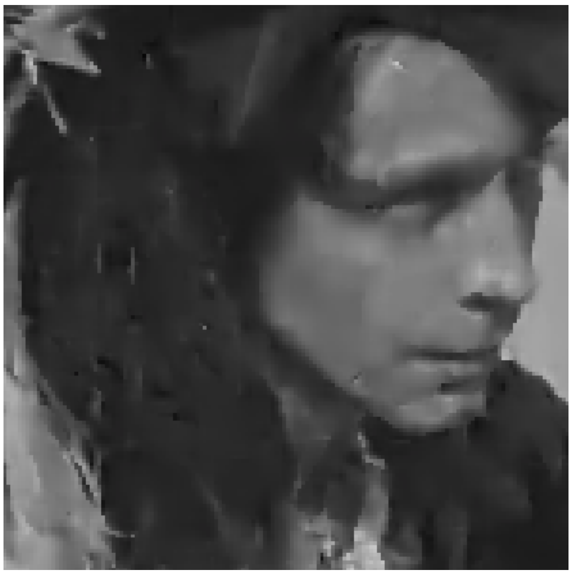
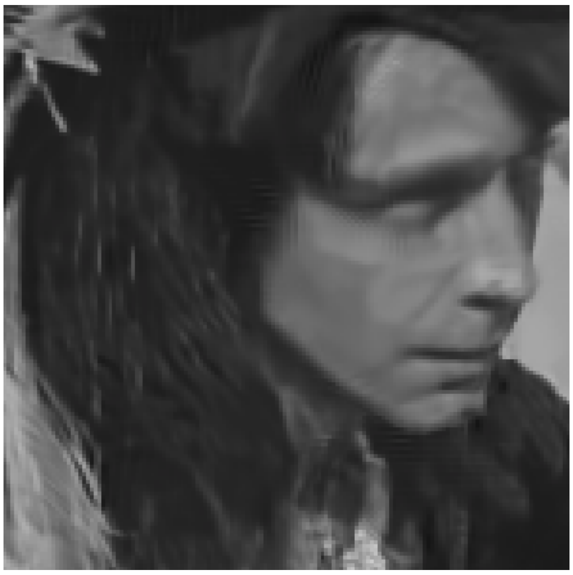

(f) Denoised with PLOW [14]. PSNR = 29.33dB, SSIM = 0.7967.

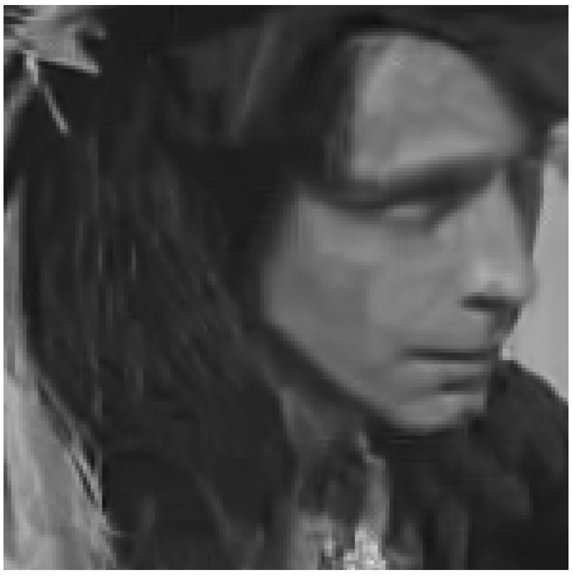
Figure 5.2.: Visual comparison of the four top performing denoising algorithms for the man image, with $\sigma_z = 25$.


(a) Closeup of original.


(b) Closeup of noisy ($\sigma_z = 25$).


(c) Closeup of proposed denoising.


(d) Closeup of BM3D-SAPCA denoising.


(e) Closeup of BM3D denoising.

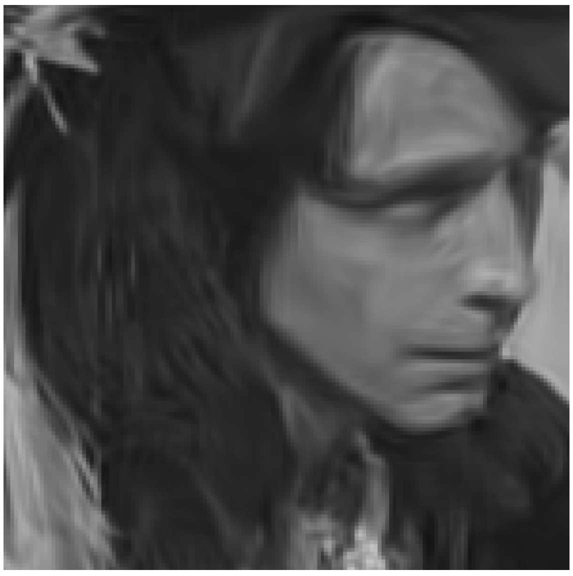
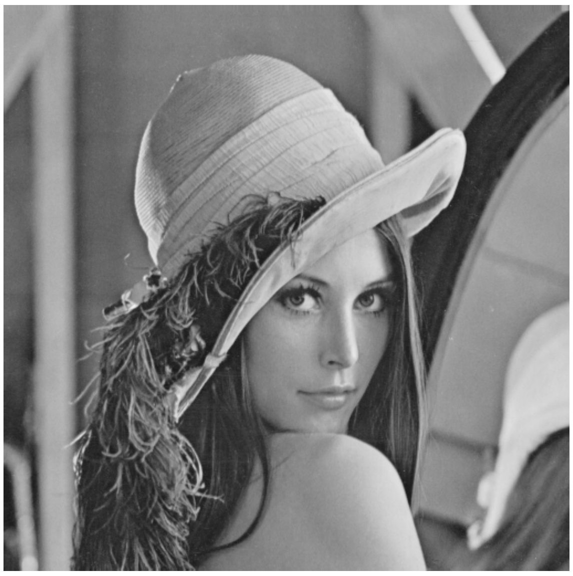
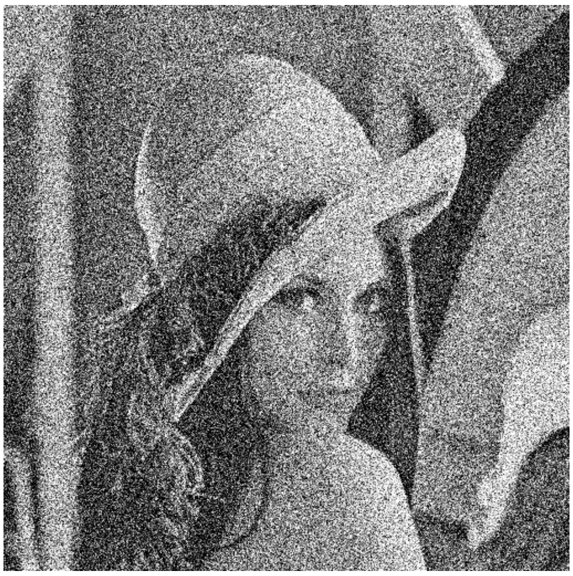
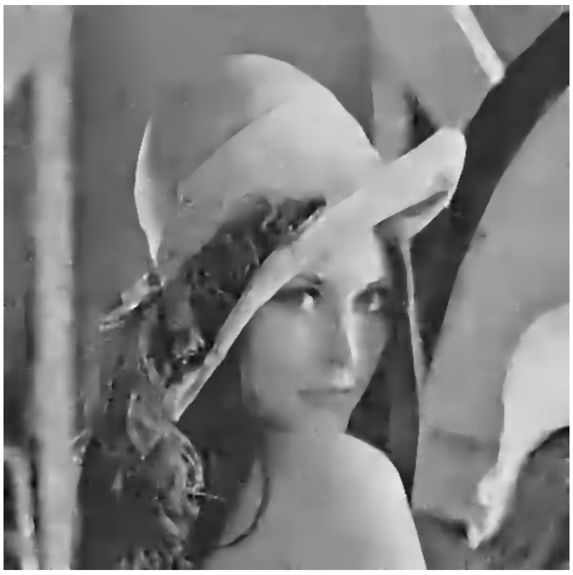
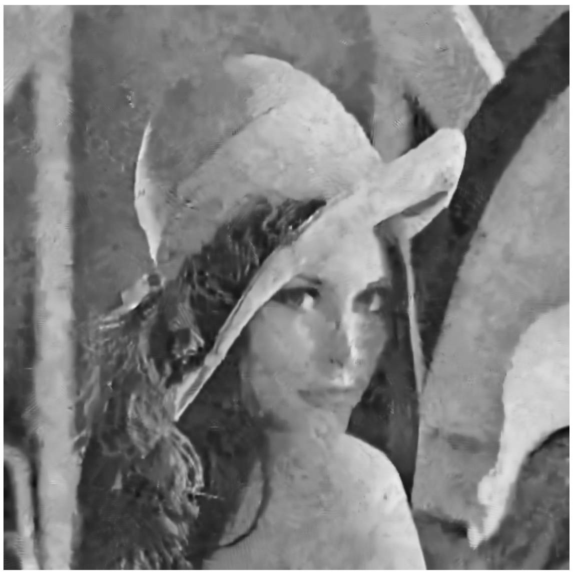

(f) Closeup of PLOW denoising.

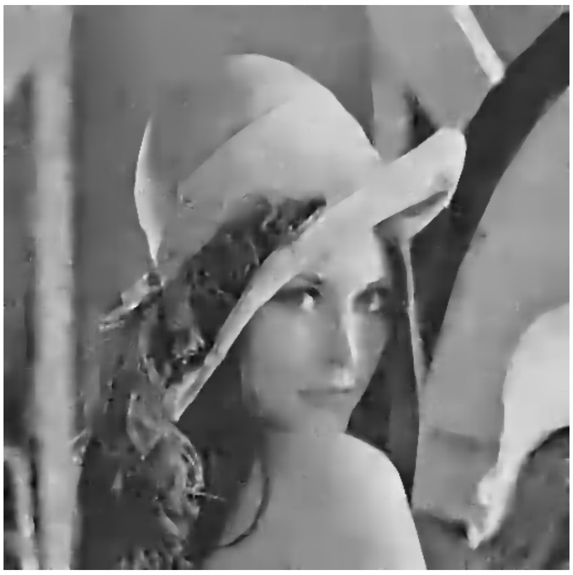
Figure 5.3.: Visual comparison of the four top performing denoising algorithms for the man image, with $\sigma_z = 25$.


(a) Original.


(b) Noisy, PSNR = 10.63dB, SSIM = 0.0600 (σ_z = 75).


(c) Denoised as proposed. PSNR = 27.11dB, SSIM = 0.7597.


(d) Denoised with BM3D-SAPCA [19]. PSNR = 26.83dB, SSIM = 0.7247.


(e) Denoised with BM3D [17], PSNR = 27.26dB, SSIM = 0.7516.

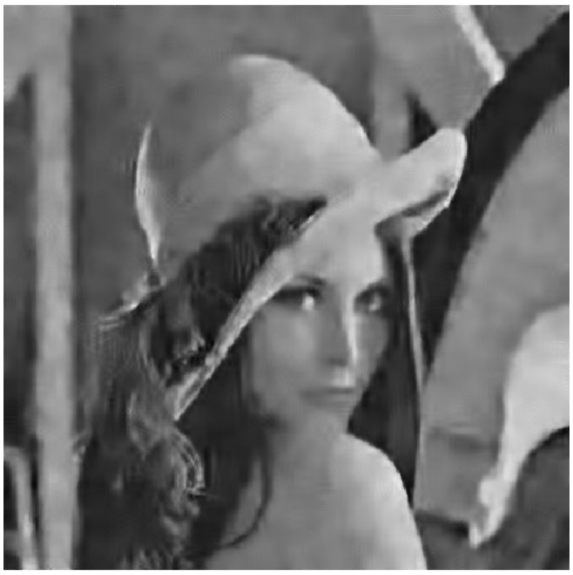
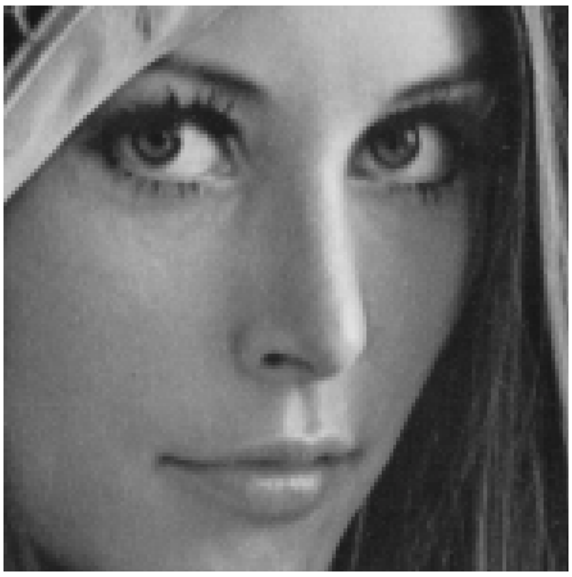
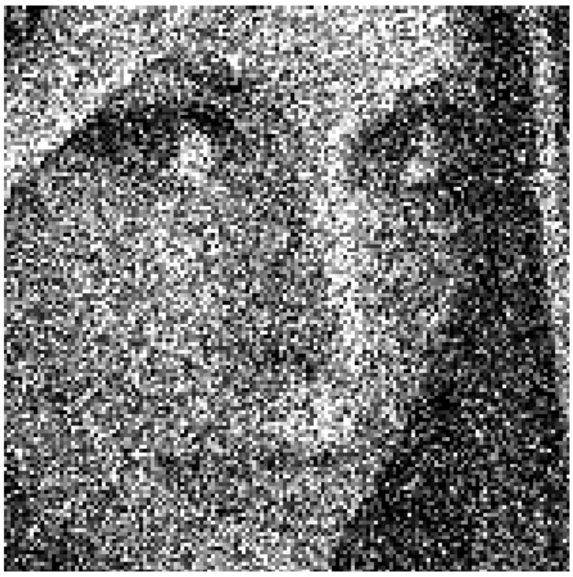
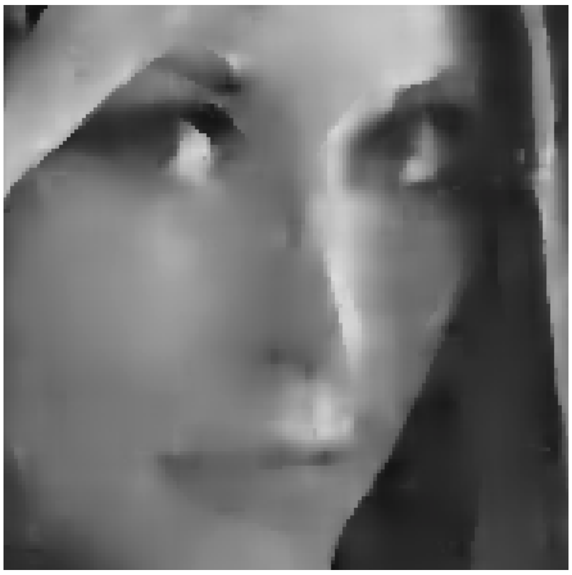
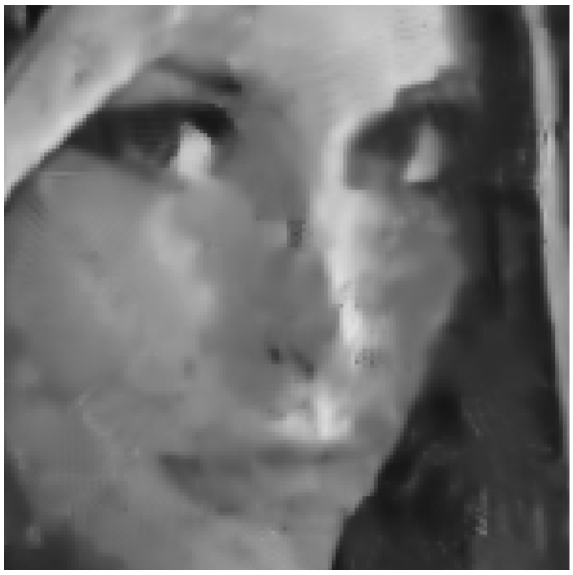

(f) Denoised with SADCT [35]. PSNR = 26.78dB, SSIM = 0.7457.

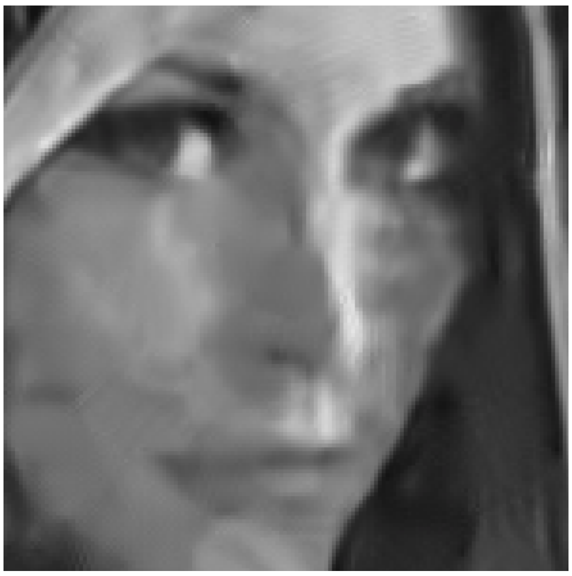
Figure 5.4.: Visual comparison of the four top performing denoising algorithms for the Lena image, with σ_z = 75.


(a) Closeup of original.


(b) Closeup of noisy, ($\sigma_z = 75$).


(c) Closeup of proposed denoising.


(d) Closeup of BM3D-SAPCA denoising.


(e) Closeup of BM3D denoising.

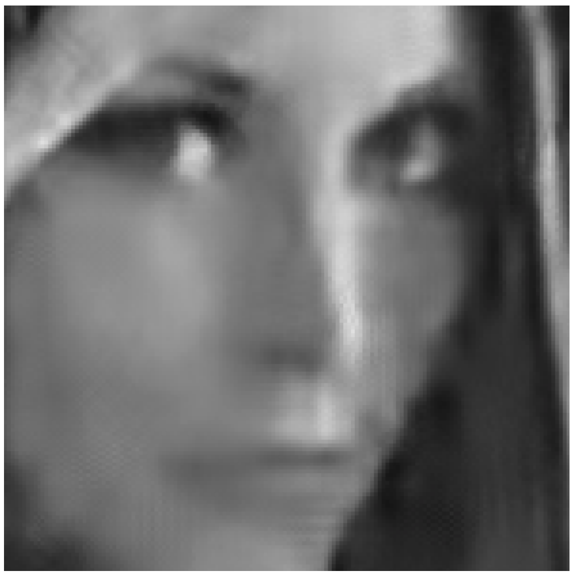

(f) Closeup of SAPCA denoising.

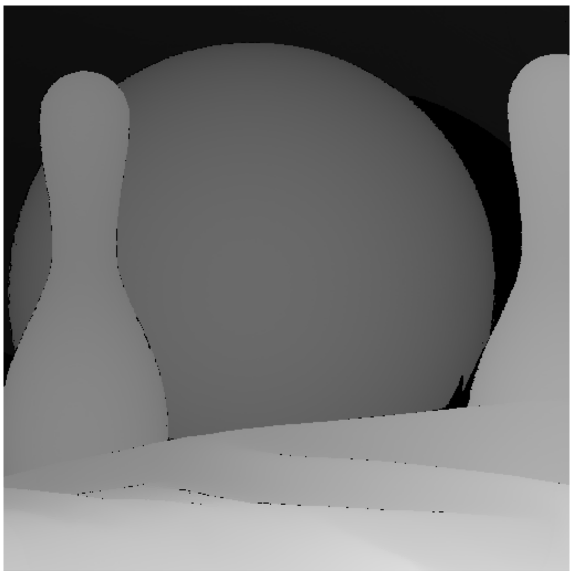
Figure 5.5.: Visual comparison of the four top performing denoising algorithms for the Lena image, with $\sigma_z = 75$.

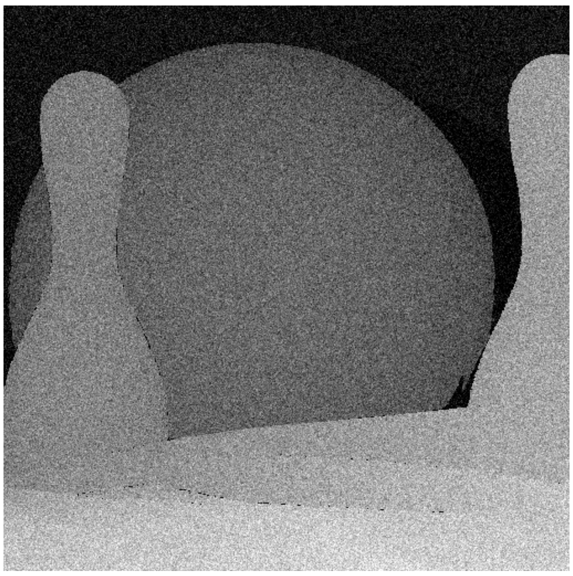
	Degraded		Denoising Results								
Image	σ_z	Degraded	Proposed	BM3DSAPCA	BM3D	SADCT	KSVD	PLOW	Wavelet CS	Curvelet	Wavelet
Aloe	10	28.14	41.83	**42.35**	39.68	38.21	41.16	36.17	37.51	33.65	34.13
(512 × 512)	25	20.18	**35.51**	34.24	33.21	32.07	32.91	31.84	31.08	29.87	28.21
	50	14.16	**30.91**	30.00	29.82	29.06	28.77	28.91	27.52	27.70	24.67
	75	10.63	**28.80**	27.50	28.17	27.55	26.64	27.07	25.64	26.46	22.51
	100	8.14	**27.57**	25.95	26.96	26.52	25.27	25.76	24.26	25.46	20.83
Art	10	28.14	41.80	**42.82**	40.10	38.86	41.08	36.58	37.41	33.57	34.07
(512 × 512)	25	20.18	**35.58**	34.77	32.82	31.19	33.65	31.17	31.04	29.49	28.11
	50	14.16	**30.33**	29.35	29.03	28.06	28.11	28.23	27.05	26.81	24.33
	75	10.63	**28.17**	27.03	27.48	26.72	26.06	26.50	25.08	25.58	22.21
	100	8.14	**26.81**	25.59	26.31	25.78	24.84	25.28	23.70	24.66	20.62
Baby	10	28.14	**45.40**	44.65	42.87	42.47	42.73	39.57	40.92	36.71	37.41
(512 × 512)	25	20.18	**38.44**	37.08	36.86	35.87	35.71	35.88	34.35	33.20	31.32
	50	14.16	**34.44**	32.62	33.80	32.85	31.53	32.27	30.42	31.04	27.04
	75	10.63	**32.89**	29.67	31.81	31.23	29.20	30.05	28.15	29.46	24.37
	100	8.14	**31.77**	27.72	30.40	29.96	27.49	28.58	26.40	28.24	22.35
Bowling bowl	10	28.14	45.91	**46.23**	43.62	43.09	44.12	39.80	41.04	36.30	37.42
(512 × 512)	25	20.18	**39.75**	38.07	36.47	35.26	36.24	34.93	34.21	31.46	31.05
	50	14.16	**34.72**	32.71	32.90	32.15	31.25	31.62	29.48	29.32	26.69
	75	10.63	**32.69**	29.81	31.22	30.54	28.82	29.44	27.01	27.84	23.98
	100	8.14	**31.33**	27.99	29.96	29.36	27.04	27.88	25.31	26.61	21.96
Average			**34.91**	33.59	33.34	32.50	32.40	31.56	30.65	29.83	27.44

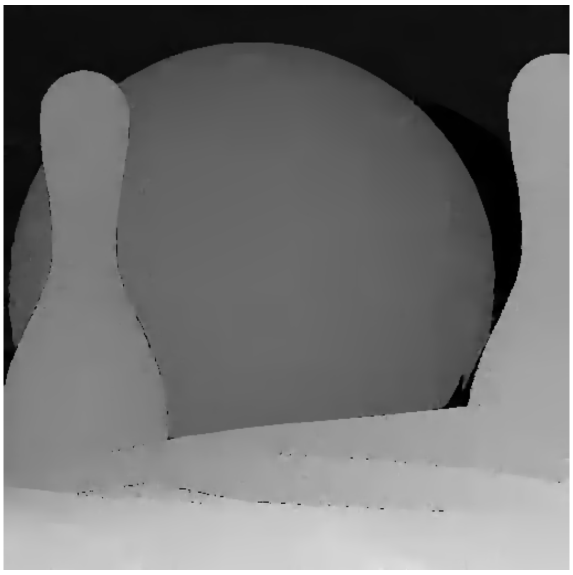
Table 5.3.: A PSNR comparison of various denoising algorithms for depth images, with the best result shown in bold. The algorithms, in order of average performance, are Proposed, BM3D-SAPCA [19], BM3D [17], SADCT [35], KSVD [29], PLOW [14], Wavelet CS (hard thresholding with cycle spinning), Curvelet [11] and Wavelet (hard thresholding). Finally, σ_z is the standard deviation of the noise.

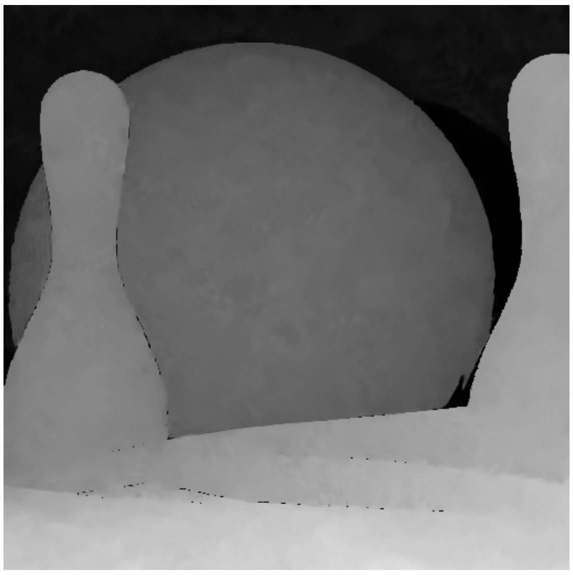
	Degraded		Denoising Results								
Image	σ_z	Degraded	Proposed	BM3D	SADCT	BM3DSAPCA	KSVD	PLOW	Curvelet	Wavelet CS	Wavelet
Aloe	10	0.4833	**0.9908**	0.9844	0.9763	0.9901	0.9832	0.9699	0.9324	0.9643	0.9176
(512 × 512)	25	0.1840	**0.9652**	0.9446	0.9270	0.9549	0.9308	0.9263	0.8735	0.8785	0.7932
	50	0.0759	**0.9202**	0.8938	0.8784	0.8949	0.8438	0.8386	0.8082	0.7650	0.6939
	75	0.0413	**0.8866**	0.8464	0.8442	0.7910	0.7683	0.7454	0.7522	0.6871	0.6421
	100	0.0255	**0.8636**	0.8052	0.8157	0.7298	0.7039	0.6790	0.6997	0.6300	0.6124
Art	10	0.4865	**0.9930**	0.9877	0.9804	0.9917	0.9854	0.9771	0.9296	0.9673	0.9208
(512 × 512)	25	0.1882	**0.9725**	0.9497	0.9265	0.9634	0.9465	0.9293	0.8753	0.8894	0.8030
	50	0.0799	**0.9277**	0.8948	0.8751	0.8959	0.8465	0.8397	0.8000	0.7736	0.6972
	75	0.0448	**0.8947**	0.8476	0.8450	0.7985	0.7721	0.7474	0.7474	0.6935	0.6464
	100	0.0283	**0.8674**	0.8036	0.8182	0.7421	0.7106	0.6838	0.6953	0.6348	0.6176
Baby	10	0.4251	**0.9958**	0.9913	0.9903	0.9941	0.9878	0.9851	0.9639	0.9783	0.9473
(512 × 512)	25	0.1276	**0.9825**	0.9682	0.9652	0.9733	0.9542	0.9572	0.9269	0.9149	0.8614
	50	0.0430	**0.9634**	0.9437	0.9410	0.9301	0.8894	0.8846	0.8776	0.8186	0.7860
	75	0.0212	**0.9518**	0.9115	0.9197	0.8309	0.8233	0.8066	0.8231	0.7464	0.7437
	100	0.0123	**0.9421**	0.8815	0.8976	0.7635	0.7562	0.7535	0.7677	0.6902	0.7160
Bowling bowl	10	0.4248	**0.9953**	0.9919	0.9913	0.9947	0.9873	0.9862	0.9583	0.9765	0.9427
(512 × 512)	25	0.1290	**0.9858**	0.9692	0.9668	0.9777	0.9558	0.9540	0.9154	0.9140	0.8560
	50	0.0458	**0.9673**	0.9412	0.9396	0.9342	0.8853	0.8829	0.8477	0.8135	0.7738
	75	0.0239	**0.9550**	0.9100	0.9154	0.8362	0.8136	0.8007	0.7861	0.7363	0.7236
	100	0.0146	**0.9438**	0.8786	0.8915	0.7820	0.7422	0.7339	0.7289	0.6757	0.6839
Average			**0.9482**	0.9172	0.9153	0.8885	0.8643	0.8541	0.8355	0.8074	0.7689

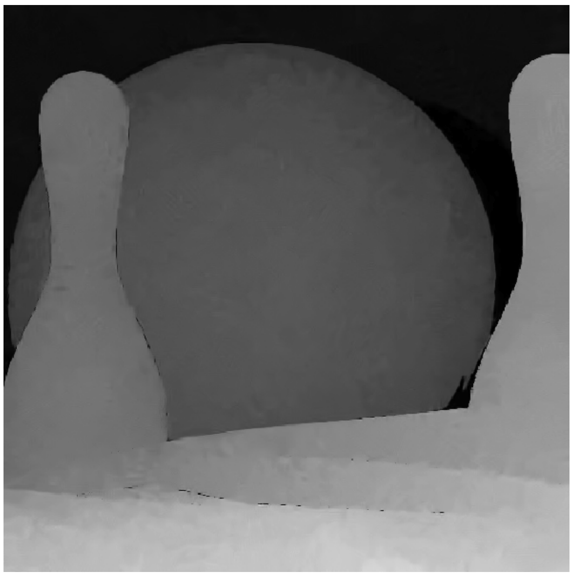
Table 5.4.: PSNR (dB) comparison of various denoising algorithms for depth images, with the best result in bold. The algorithms, in order of average performance, are BM3D-SAPCA [19], BM3D [17], Proposed, SADCT [35], PLOW [14], KSVD [29], Wavelet CS (hard thresholding with cycle spinning), Curvelet [11] and Wavelet (hard thresholding). Finally, σ_z is the standard deviation of the noise.


(a) Original


(b) Noisy, PSNR = 20.18dB, SSIM = 0.1290 (σ_z = 25).


(c) Denoised as proposed, PSNR = 39.75dB, SSIM = 0.9858.


(d) Denoised with BM3D-SAPCA, PSNR = 38.07dB, SSIM = 0.9777.


(e) Denoised with BM3D, PSNR = 36.47dB, SSIM = 0.9642.

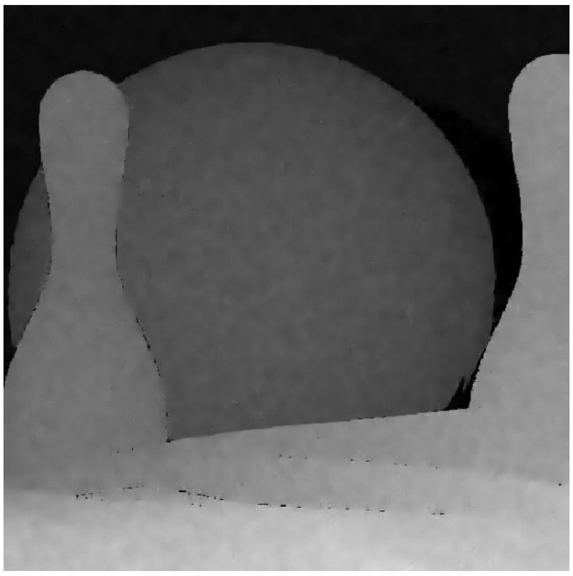
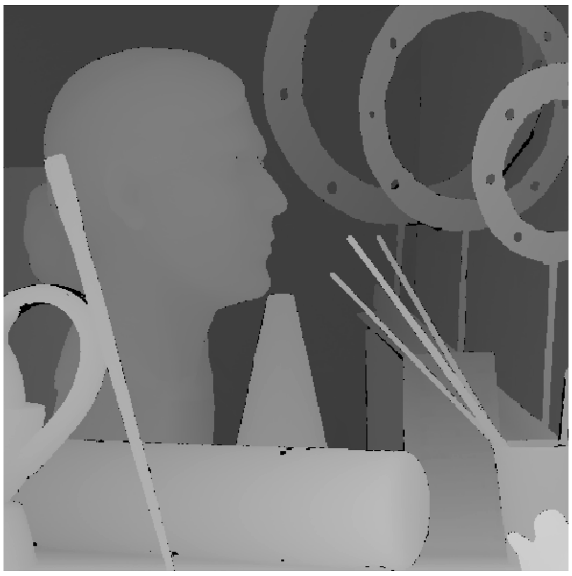
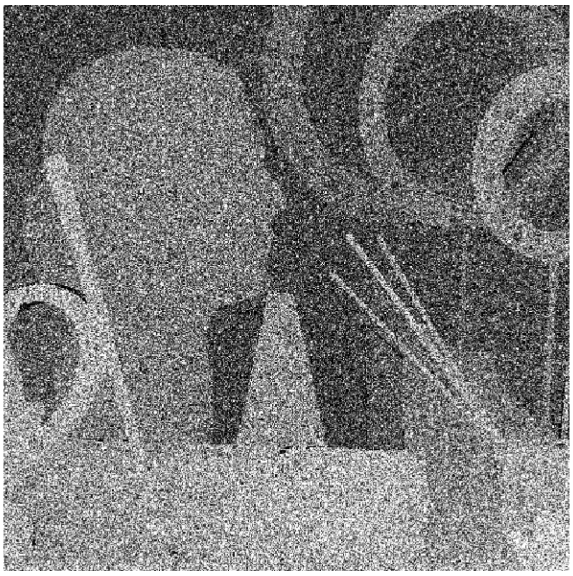
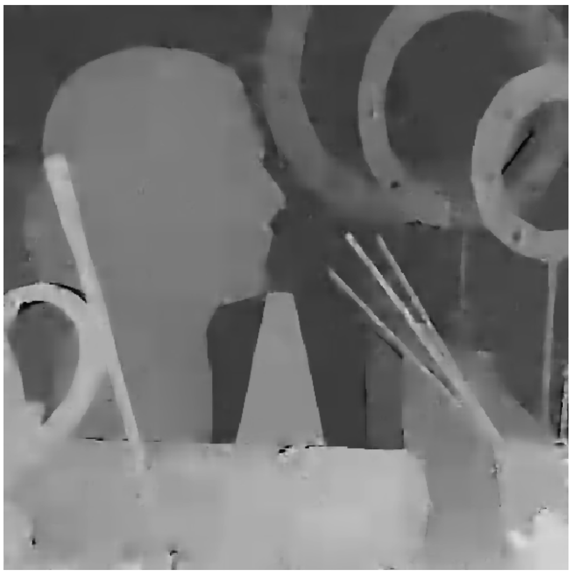
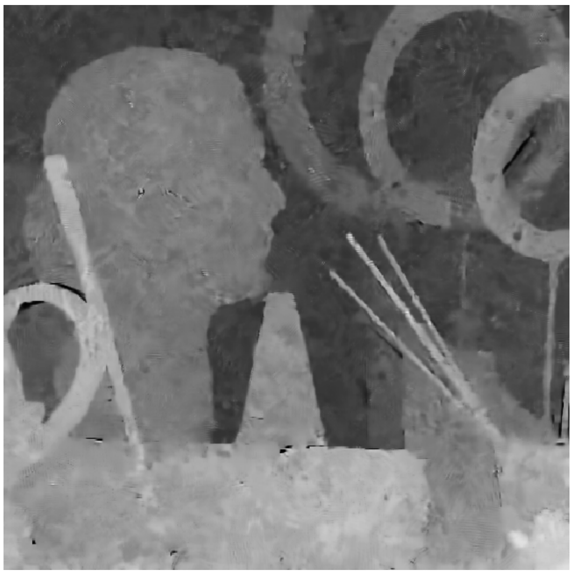

(f) Denoised with KSVD, PSNR = 36.24dB, SSIM = 0.9558.

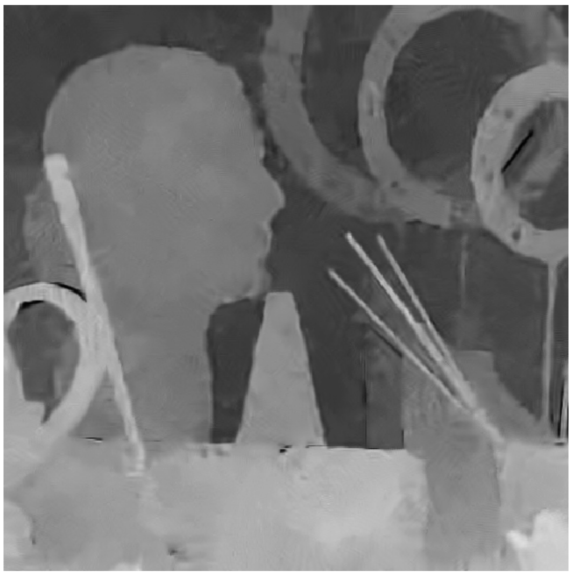
Figure 5.6.: Visual comparison of the four top performing denoising algorithms for the bowling ball image, with σ_z = 25.


(a) Original


(b) Noisy, PSNR = 10.63dB, SSIM = 0.0448 (σ_z = 75).


(c) Denoised as proposed, PSNR = 28.17dB, SSIM = 0.8947.


(d) Denoised with BM3D-SAPCA, PSNR = 27.03dB, SSIM = 0.7985.


(e) Denoised as BM3D, PSNR = 27.48dB, SSIM = 0.8476.

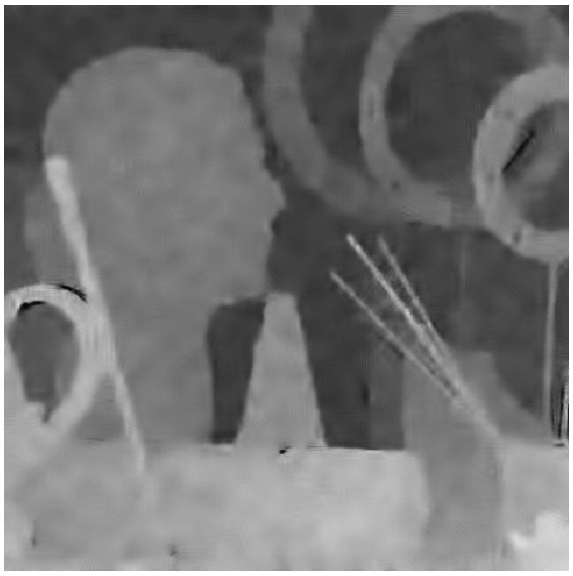

(f) Denoised with SADCT, PSNR = 26.72dB, SSIM = 0.8450.

Figure 5.7.: Visual comparison of the four top performing denoising algorithms for the art image, with $\sigma_z = 27$.

5.3. Deconvolution

For the deconvolution linear inverse problem, $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{z}$, the MAP estimator is given by

$$\hat{\boldsymbol{x}} = D(\hat{\boldsymbol{\theta}}),$$

where

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{y} - \boldsymbol{H}D(\boldsymbol{\theta})\|_2^2 + \zeta\sigma_z^2 P^\theta(\boldsymbol{\theta}). \tag{5.3}$$

Here we have made the same assumptions as the denoising case; i.e., the noise is additive white Gaussian and

$$p(\boldsymbol{x}) = Aexp\left[-\frac{\zeta P^x(\boldsymbol{x})}{2}\right].$$

When calculating an approximation using the quadtree model we operate on each tile independently. This is possible because, given a particular tile structure, the regions do not overlap. The prune and join algorithms just provide an efficient way to choose between the possible tile structures.

Unfortunetely, analogous to the non-unitary linear transform case, the presence of $\boldsymbol{H}$ in (5.3) prevents us from solving for each tile independently. In the following subsections we will show that, like the linear transform case, we can solve using regularised inverses followed by a denoising step, or using an iterative approach.

5.3.1. Fourier regularised inverses

Chapter 3 introduced Fourier regularised inverses for deconvolution. It was shown that if the amount of regularisation is reduced, a sharper but consequently noisier image is produced, which could be denoised to produce a high quality deconvolution. We also saw the two-step process depicted in Fig. 3.4, which applied this process twice. The result of the first step is used to obtain an estimate of the original signal's energy spectrum. This energy spectrum is used, in the second step, to construct an improved regularised Wiener filter. The output of this Wiener filter is denoised to give the final deconvolution result.

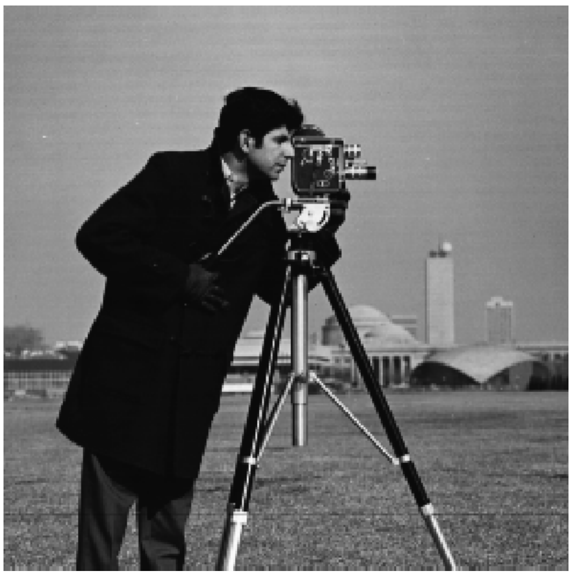
In this subsection we will apply this two-step process using our denoising algorithm. We can cater for the coloured noise by using a different λ for each coefficient:

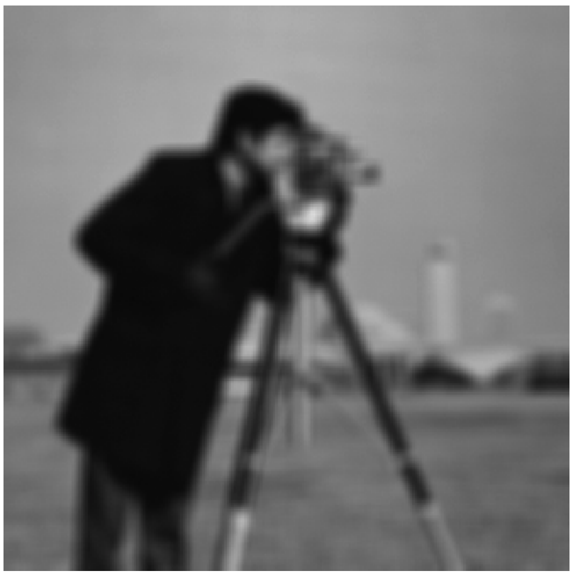
$$\lambda_i = \zeta\sigma_z^2\|\boldsymbol{W}^T\boldsymbol{b_i}\|_2^2.$$

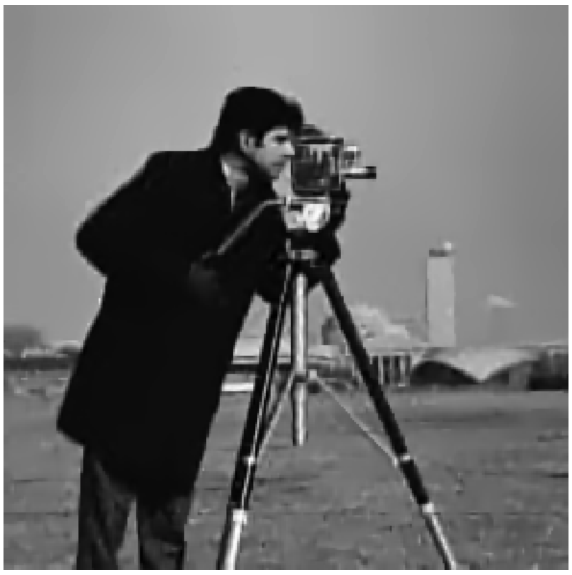
This essentially applies a weight, $\|\boldsymbol{W}^T\boldsymbol{b_i}\|_2^2$, to the usual $\lambda = \zeta\sigma_z^2$. These weights can be computationally expensive to compute due to the large number of possible edge orientations. If we restrict ourselves to pruning we can compute these weights, for a particular $\boldsymbol{W}$, offline. In the case of joining the number of possible, not necessarily square, tile sizes is too large to process offline. Furthermore, $\boldsymbol{W}$ corresponds to the regularised filter, so for the Wiener filter this is dependent on the measured signal and can not be computed offline. A compromise is to simply use the regularised Fourier inverse weights for both steps and restrict ourselves to the prune-only model. This approach can produce satisfactory results, however it is often sufficient to just assume the noise is white and denoise as proposed in the previous section.

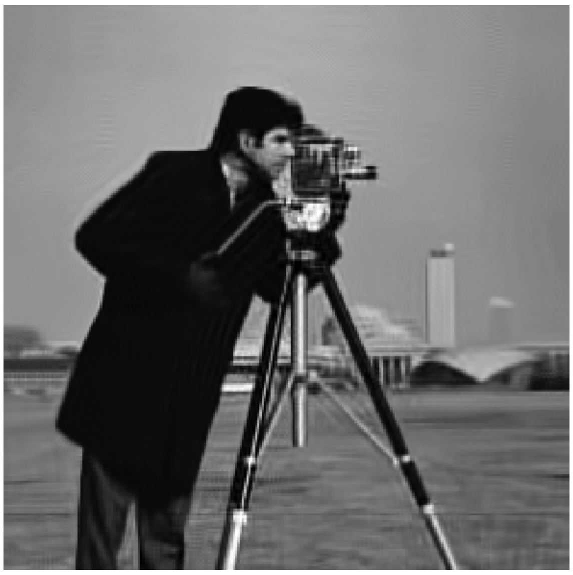
Figures 5.8 - 5.10 show some deconvolution results of the proposed method in comparison to some state of the art algorithms. In the first experiment the point spread function (PSF) was a 9×9 Gaussian with standard deviation of 4 and the noise standard deviation was 0.5. For the second experiment the PSF was a 51×51 uniform kernel and the noise standard deviation was, once again, 0.5.

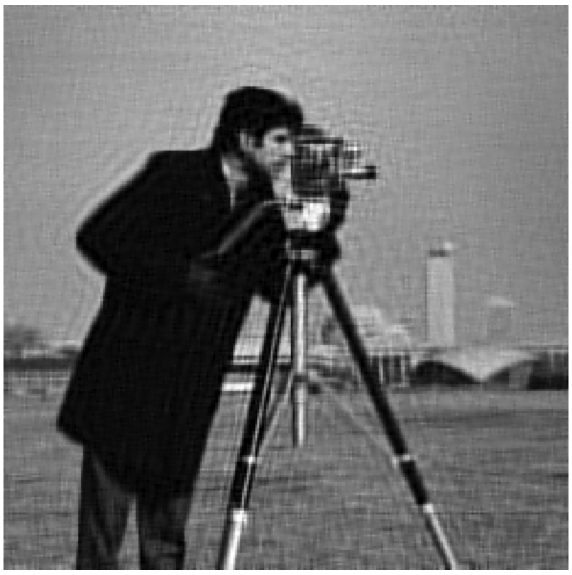
The proposed results were calculated with the standard denoising algorithm; i.e. assuming the noise is white. Unfortunately, applying the coloured noise weights rarely improves performance; therefore, since they are also computationally expensive to calculate, we neglect this approach. The coloured noise weights are not advantageous to our algorithm because of its low dimension. We use polynomials up to degree one and do not decompose the signal into frequency bands as well as other transforms that use a full basis. Consequently, it is difficult to correctly threshold different signal frequencies when tho noise is coloured. Despite this, we still obtain satisfactory results in these examples.


(a) Original image.


(b) Noisy blurred image. PSNR=21.37.


(c) Two-step deconvolution using proposed denoising. PSNR=28.15dB, SSIM=0.8534.


(d) Two-step deconvolution using BM3D denoising. PSNR=28.59dB, SSIM=0.8598.


(e) FISTA deconvolution in the curvelet domain. PSNR=26.83dB, SSIM=0.7656.

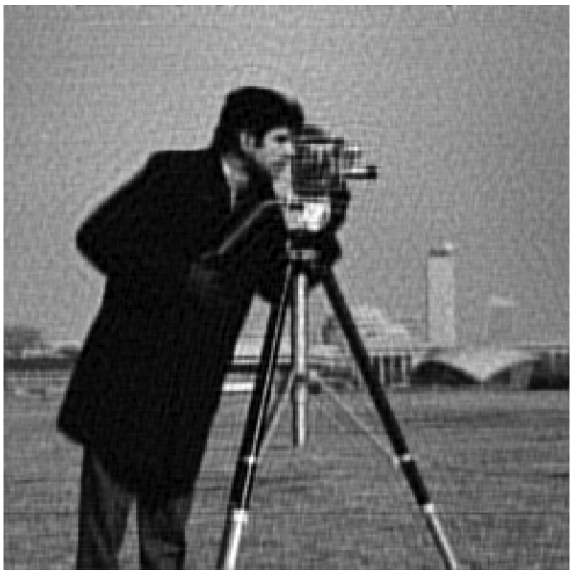
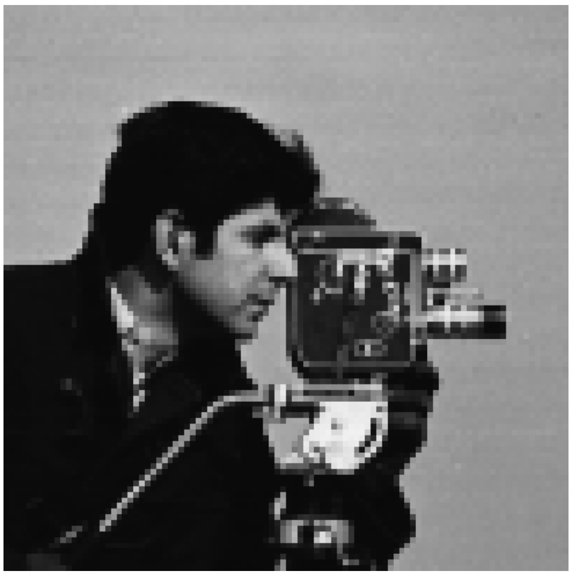
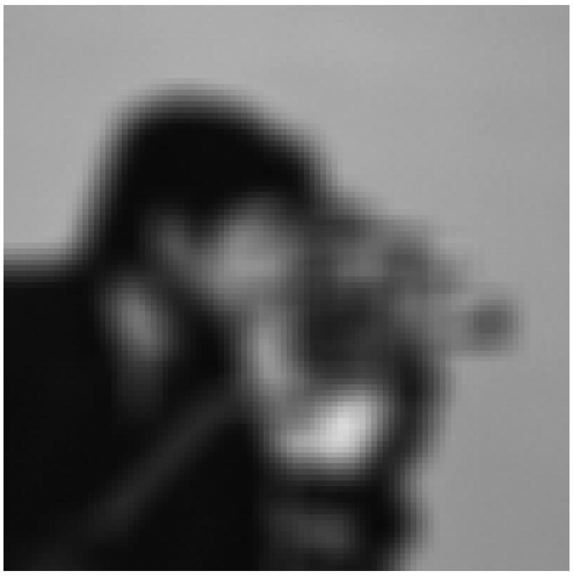
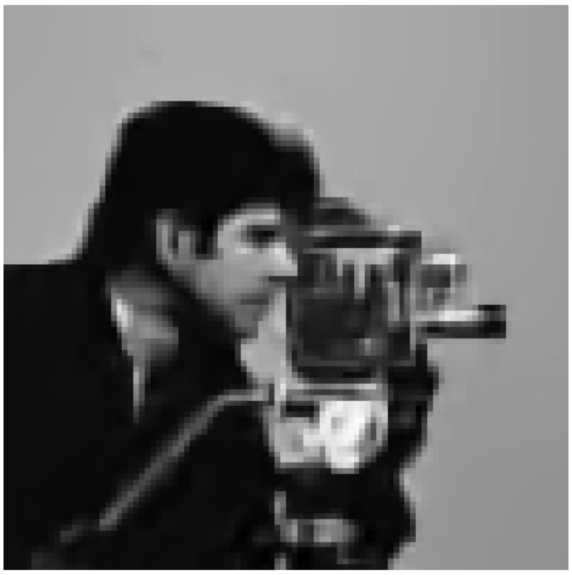
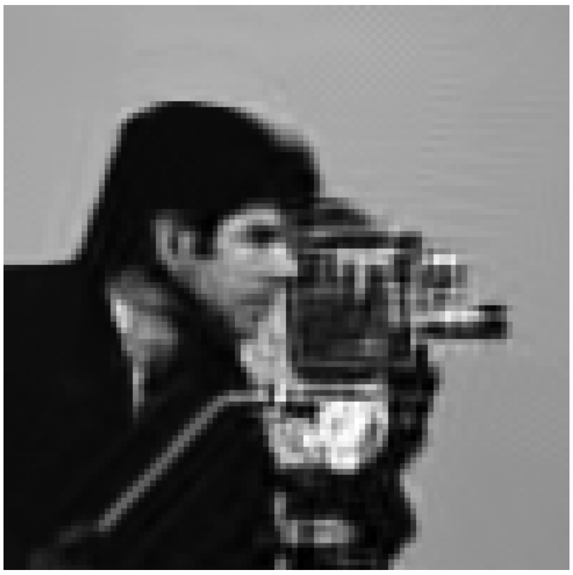

(f) Wiener filter deconvolution using the oracle energy spectrum. PSNR=27.19, SSIM=0.7235.

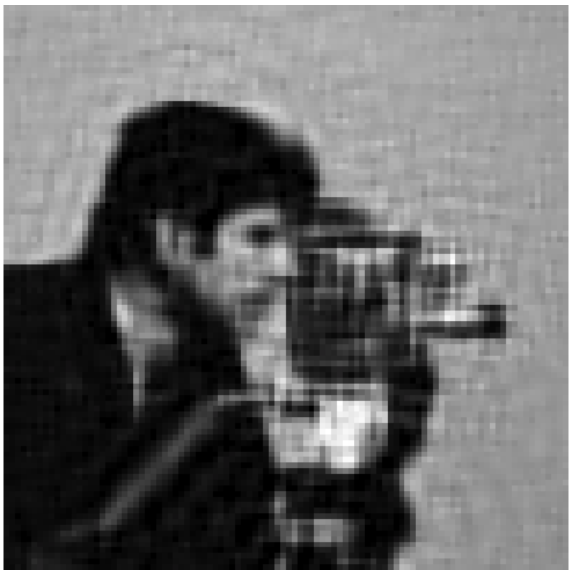
Figure 5.8.: An example of the performance of various deconvolution algorithms on the cameraman image.


(a) Close up of original image.


(b) Close up of noisy blurred image.


(c) Close up of two-step deconvolution using proposed denoising.


(d) Close up of two-step deconvolution using BM3D denoising.


(e) Close up of FISTA deconvolution in the curvelet domain.

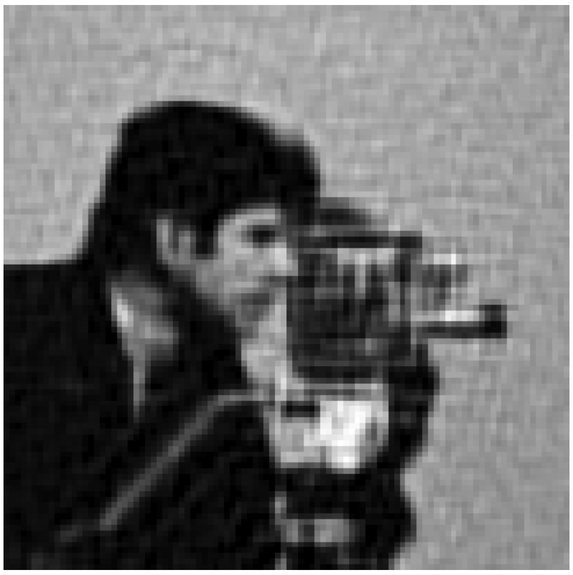
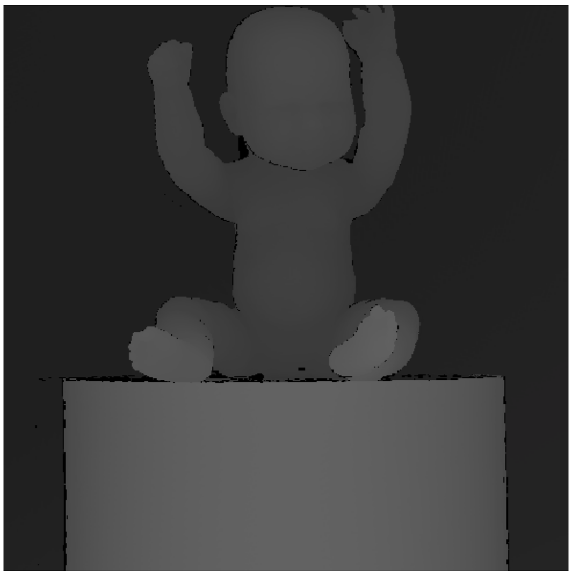
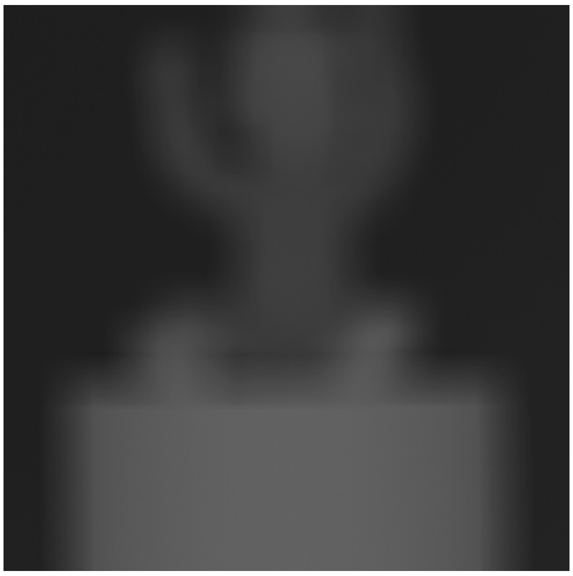
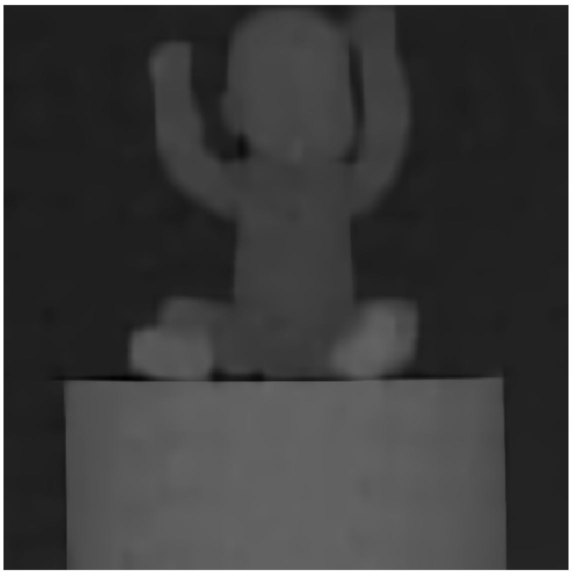
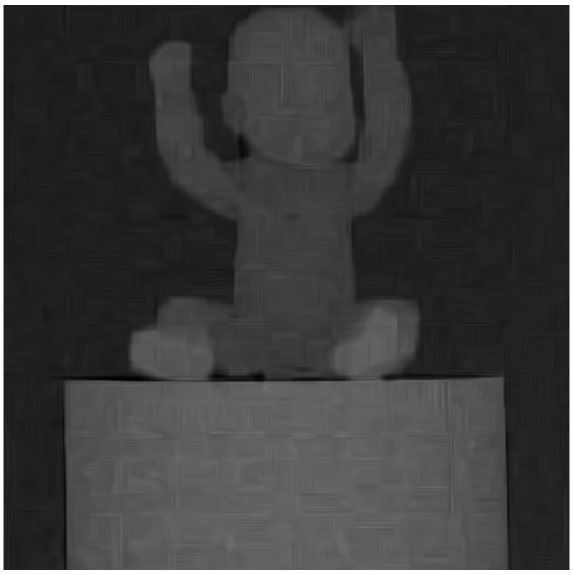

(f) Close up of Wiener filter deconvolution using the oracle energy spectrum.

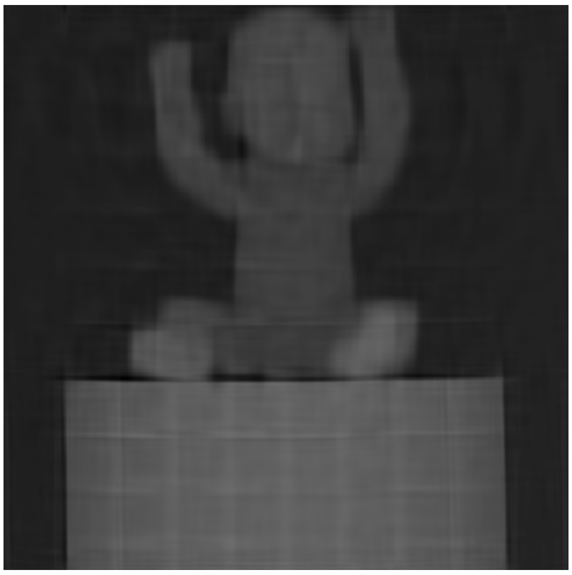
Figure 5.9.: An example of the performance of various deconvolution algorithms on the cameraman image.


(a) Original image.


(b) Noisy blurred image. PSNR=27.76.


(c) Two-step deconvolution using proposed denoising. PSNR=33.15dB, SSIM=0.9510.


(d) Two-step deconvolution using BM3D denoising. PSNR=32.94dB, SSIM=0.8966.


(e) FISTA deconvolution in the curvelet domain. PSNR=32.22dB, SSIM=0.9338.

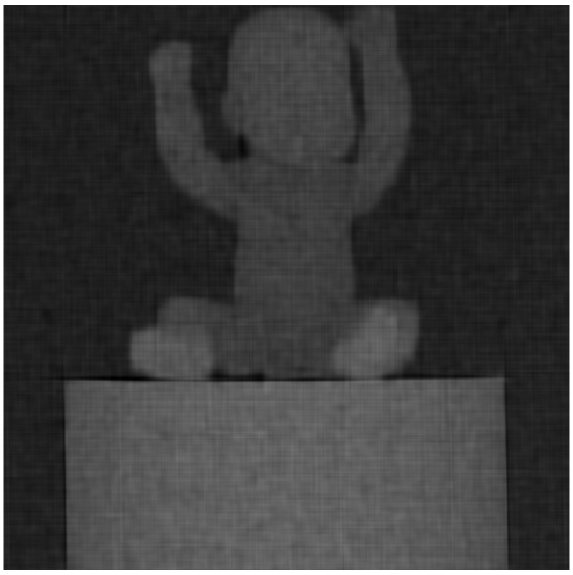

(f) Wiener filter deconvolution using the oracle energy spectrum. PSNR=31.89, SSIM=0.8428.

Figure 5.10.: An example of the performance of various deconvolution algorithms on the cameraman image.

5.3.2. Iterative minimisation of surrogate functionals

In Chapter 2 we introduced MM algorithms and showed how iterative thresholding fits into this framework. We will now use a surrogate function and MM ideas to develop an iterative algorithm to minimise the nonlinear function

$$C(\boldsymbol{\theta}) = \|\boldsymbol{y} - \boldsymbol{H}D(\boldsymbol{\theta})\|_2^2 + \zeta\sigma_z^2 P^\theta(\boldsymbol{\theta}). \tag{5.4}$$

Recall that the MM strategy requires us to find a surrogate function, $C_{sur}(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(\boldsymbol{k})})$, that is a maximiser of $C(\boldsymbol{\theta})$ at the current estimate $\boldsymbol{\theta}^{(\boldsymbol{k})}$; i.e.,

$$C_{sur}(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(\boldsymbol{k})}) \geq C(\boldsymbol{\theta}) \qquad \forall\boldsymbol{\theta} \qquad \text{and} \tag{5.5}$$

$$C_{sur}(\boldsymbol{\theta}^{(\boldsymbol{k})} \mid \boldsymbol{\theta}^{(\boldsymbol{k})}) = C(\boldsymbol{\theta}^{(\boldsymbol{k})}). \tag{5.6}$$

The MM iteration,

$$\boldsymbol{\theta}^{(\boldsymbol{k+1})} = arg \min_{\boldsymbol{\theta}} C_{sur}(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(\boldsymbol{k})}), \tag{5.7}$$

is then guaranteed to be decreasing, since

$$C(\boldsymbol{\theta}^{(\boldsymbol{k+1})}) \leq C_{sur}(\boldsymbol{\theta}^{(\boldsymbol{k+1})} \mid \boldsymbol{\theta}^{(\boldsymbol{k})}) \leq C_{sur}(\boldsymbol{\theta}^{(\boldsymbol{k})} \mid \boldsymbol{\theta}^{(\boldsymbol{k})}) = C(\boldsymbol{\theta}^{(\boldsymbol{k})}),$$

where the first inequality follows directly from (5.5), the second from the minimisation in (5.7) and the equality from (5.6).

In order to use this framework we need to find a maximiser of C that we can minimise. Consider the surrogate function

$$C_{sur}(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(\boldsymbol{k})}) = C(\boldsymbol{\theta}) - \|\boldsymbol{H}D(\boldsymbol{\theta}) - \boldsymbol{H}D(\boldsymbol{\theta}^{(\boldsymbol{k})})\|^2 + \alpha\|D(\boldsymbol{\theta}) - D(\boldsymbol{\theta}^{(\boldsymbol{k})})\|^2, \tag{5.8}$$

which is a maximiser of C provided that $\alpha \geq \|\boldsymbol{H}\|_2^2$.

Substituting this surrogate function into (5.7), expanding and dropping all terms independent of $\boldsymbol{\theta}$ yields

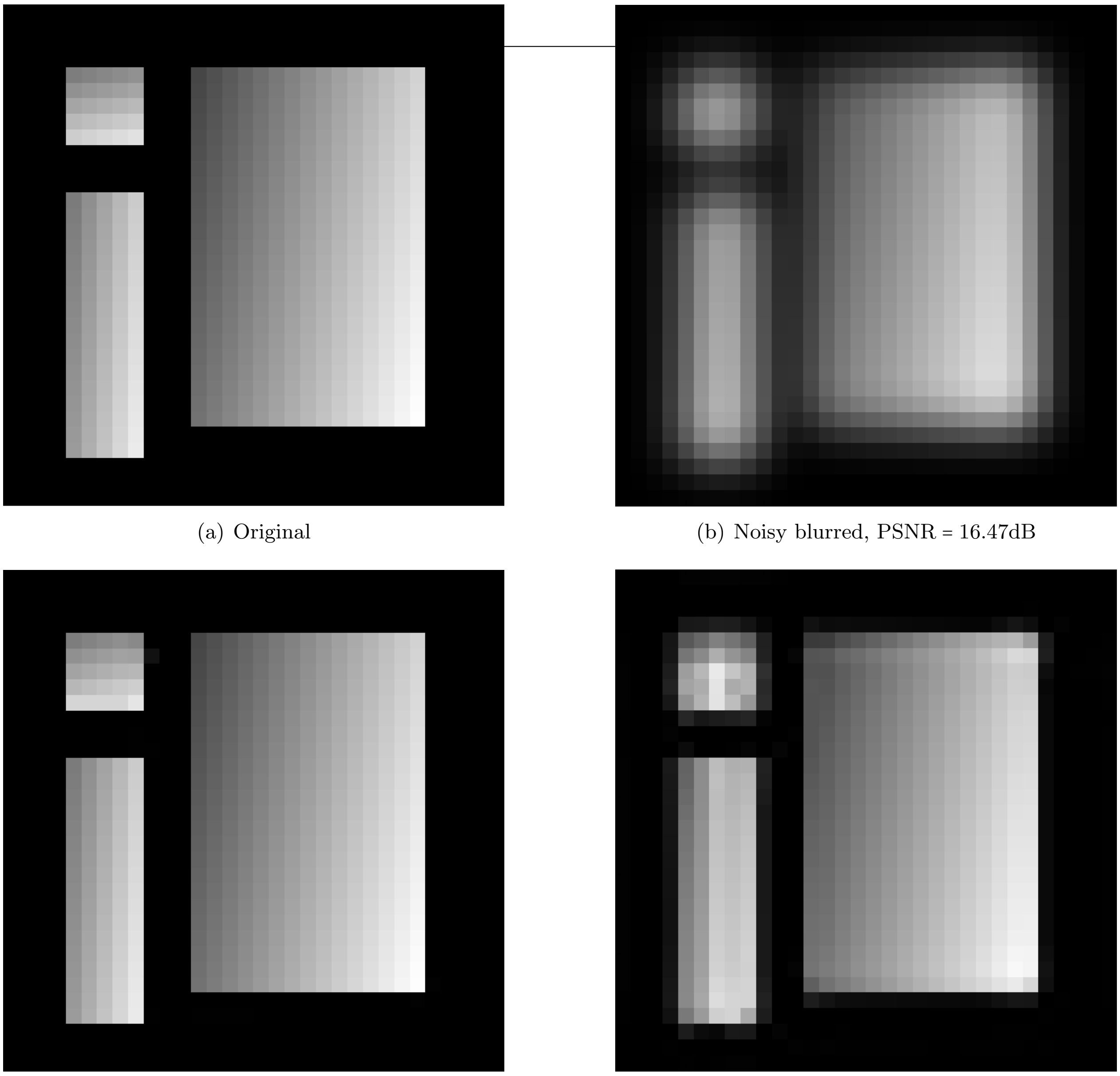
$$\boldsymbol{\theta}^{(\boldsymbol{k+1})} = arg \min_{\boldsymbol{\theta}} \Big[\|\boldsymbol{y} - \boldsymbol{H}D(\boldsymbol{\theta})\|_2^2 + \zeta\sigma_z^2 P^\theta(\boldsymbol{\theta}) - \|\boldsymbol{H}D(\boldsymbol{\theta}) - \boldsymbol{H}D(\boldsymbol{\theta}^{(\boldsymbol{k})})\|^2 + \alpha\|D(\boldsymbol{\theta}) - D(\boldsymbol{\theta}^{(\boldsymbol{k})})\|^2\Big]$$

$$
\begin{aligned}
=&arg\min_{\boldsymbol{\theta}}\Big[-2D(\boldsymbol{\theta})^T\boldsymbol{H}^T\boldsymbol{y}+\|\boldsymbol{H}D(\boldsymbol{\theta})\|_2^2+\zeta\sigma_z^2P^\theta(\boldsymbol{\theta})\\
&-\|\boldsymbol{H}D(\boldsymbol{\theta})\|_2^2+2D(\boldsymbol{\theta})^T\boldsymbol{H}^T\boldsymbol{H}D(\boldsymbol{\theta}^{(\boldsymbol{k})})+\alpha\|D(\boldsymbol{\theta})\|_2^2-2\alpha D(\boldsymbol{\theta})^TD(\boldsymbol{\theta}^{(\boldsymbol{k})})\Big].
\end{aligned}
$$

The cancellation of the $\|\boldsymbol{H}D(\boldsymbol{\theta})\|_2^2$ term is what makes the surrogate function easily minimisable. By rearranging and comparing to (5.2), we see that the surrogate function is minimised by denoising:

$$
\begin{aligned}
\boldsymbol{\theta}^{(\boldsymbol{k+1})}&=arg\min_{\boldsymbol{\theta}}\left[\|D(\boldsymbol{\theta})\|_2^2-2D(\boldsymbol{\theta})^T\left(\frac{\boldsymbol{H}^T}{\alpha}(\boldsymbol{y}-\boldsymbol{H}D(\boldsymbol{\theta}^{(\boldsymbol{k})}))+D(\boldsymbol{\theta}^{(\boldsymbol{k})})\right)+\frac{\zeta\sigma_z^2}{\alpha}P^\theta(\boldsymbol{\theta})\right]\\
&=arg\min_{\boldsymbol{\theta}}\left[\left\|D(\boldsymbol{\theta}^{(\boldsymbol{k})})+\frac{\boldsymbol{H}^T}{\alpha}(\boldsymbol{y}-\boldsymbol{H}D(\boldsymbol{\theta}^{(\boldsymbol{k})}))-D(\boldsymbol{\theta})\right\|_2^2+\frac{\zeta\sigma_z^2}{\alpha}P^\theta(\boldsymbol{\theta})\right]\\
&=Denoise\left(D(\boldsymbol{\theta}^{(\boldsymbol{k})})+\frac{\boldsymbol{H}^T}{\alpha}\left(\boldsymbol{y}-\boldsymbol{H}D(\boldsymbol{\theta}^{(\boldsymbol{k})})\right)\right).
\end{aligned}
\tag{5.9}
$$

Figure 5.11 shows an example of the iterative deconvolution algorithm for a piecewise polynomial image. The PSF was a 7×7 quadratic spline and the standard deviation of the noise was 0.25. In this case, we achieve almost perfect reconstruction, however the convergence is too slow to be practical when the original signal is not so sparsely represented by the model. The algorithm used for comparison is iterated hard thresholding in a wavelet basis which, in this case, is the more effective than the soft thresholding variants.



(a) Original

(b) Noisy blurred, PSNR = 16.47dB

(c) Deconvolved with proposed iterative algorithm. PSNR = 50.22dB.

(d) Deconvolved with iterative hard thresholding in a wavelet basis, PSNR = 27.38dB.

Figure 5.11.: An example of deconvolving a piecewise linear image using iterative algorithms.

5.4. Interpolation

With very minor modifications, the previously described approximation algorithm can be used for interpolation. We model the problem by setting

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{z},$$

where $\boldsymbol{H} \in \mathbb{R}^{N_v \times N}$ is the identity matrix but with the rows corresponding to the unknown pixels removed. Here $\boldsymbol{x} \in \mathbb{R}^N$ is the desired image and $\boldsymbol{y}, \boldsymbol{z} \in \mathbb{R}^{N_v}$ are truncated vectors over just the, N_v, known or visible pixels, representing the measured and noise images respectively. We will assume that $\boldsymbol{z}$ is zero-mean white Gaussian with a small standard deviation, and that we know $\boldsymbol{H}$; i.e., we know the locations of the available samples. The approximation problem in this framework can be posed as follows:

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= arg \min_{\boldsymbol{\theta}} \left[\|\boldsymbol{y} - HD(\boldsymbol{\theta})\|_2^2 + \lambda \tilde{P}^{\theta}(\boldsymbol{\theta}) \right] \\ &= arg \min_{\boldsymbol{\theta}} \left[\|\boldsymbol{y} - D_v(\boldsymbol{\theta})\|_2^2 + \lambda \tilde{P}^{\theta}(\boldsymbol{\theta}) \right],\end{aligned}$$

where D_v is the corresponding truncated quadtree representation over just the visible pixels given the parameters $\boldsymbol{\theta}$. We use a modified penalty, $\tilde{P}^{\theta}(\boldsymbol{\theta})$, that is almost identical to the previous described penalty, $P^{\theta}(\boldsymbol{\theta})$. The only difference is that the cost of a polynomial region is increased by a factor of $\frac{N_i}{N_i^v}$, where N_i and N_i^v are the number of pixels and the number of visible pixels in the i-th region respectively. This modification increases the penalty on regions with fewer known pixels resulting in a sparser model, and obviously when $N_i = N_i^v$ the penalty is as previously defined.

Modifying the penalty in this way allows successful interpolation over a wide range of images and sampling rates with a fixed λ, chosen once experimentally. In the following simulations $\lambda = 50$; however, in some cases we could have obtained more accurate results by optimising λ for the particular experiment.

The truncated quadtree representation can be calculated by putting holes in the polynomial subspace basis functions and we interpolate by reconstructing with the corresponding functions with no holes. In order to generate the correct interpolation we have to modify the coefficients when we switch back to the full subspace functions.

To be more precise let $\boldsymbol{B} = \boldsymbol{QR}$ be the thin QR decomposition for a polynomial region; i.e., $\boldsymbol{B} \in \mathbb{R}^{N \times d}$ is the transform matrix with columns that span the subspace of 2-D polynomials; $\boldsymbol{Q} \in \mathbb{R}^{N \times d}$ and $\boldsymbol{R} \in \mathbb{R}^{d \times d}$ are the orthogonal and upper triangular matrices respectively.

We only measure N_v pixels, so we remove the columns from $\boldsymbol{B}$ that correspond to an unknown pixel and calculate a new, truncated, QR decomposition:

$$\boldsymbol{B_T} = \boldsymbol{Q_T R_T},$$

where $\boldsymbol{B_T}, \boldsymbol{Q_T} \in \mathbb{R}^{N_v \times d}$ and $\boldsymbol{R_T} \in \mathbb{R}^{d \times d}$. Note that, $\boldsymbol{Q_T}$ and $\boldsymbol{R_T}$ are not simply $\boldsymbol{Q}$ and $\boldsymbol{R}$ with columns removed. They are a new truncated QR decomposition, calculated from $\boldsymbol{B_T}$.

Let $\boldsymbol{t_T} \in \mathbb{R}^{N_V}$ be the measured tile. The first stage of the interpolation is to calculate the coefficients

$$\hat{\boldsymbol{\theta}}_{\boldsymbol{Q_T}} = \boldsymbol{Q_T}^T \boldsymbol{t_T}.$$

Here we have used the notation $\hat{\boldsymbol{\theta}}_{\boldsymbol{Q_T}}$ for the coefficients generated from the matrix $\boldsymbol{Q_T}$.

We will convert these coefficients into the coefficients for $\boldsymbol{B_T}$. These coefficients are given by

$$\hat{\boldsymbol{\theta}}_{\boldsymbol{B_T}} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{t_T} - \boldsymbol{B_T}\boldsymbol{\theta}\|_2^2.$$

The minimum can be found by setting the derivative, with respect to $\boldsymbol{\theta}$, equal to zero:

$$\begin{aligned} 0 &= -2\boldsymbol{B_T}^T \boldsymbol{t_T} + 2\boldsymbol{B_T}^T \boldsymbol{B_T} \hat{\boldsymbol{\theta}}_{\boldsymbol{B_T}} \\ \hat{\boldsymbol{\theta}}_{\boldsymbol{B_T}} &= \left(\boldsymbol{B_T}^T \boldsymbol{B_T}\right)^{-1} \boldsymbol{B_T}^T \boldsymbol{t_T}. \end{aligned}$$

Finally inserting $\boldsymbol{B_T} = \boldsymbol{Q_T}\boldsymbol{R_T}$ and $\hat{\boldsymbol{\theta}}_{\boldsymbol{Q_T}} = \boldsymbol{Q_T}^T \boldsymbol{t_T}$ yields

$$\begin{aligned} \hat{\boldsymbol{\theta}}_{\boldsymbol{B_T}} &= (\boldsymbol{R_T}^T \boldsymbol{Q_T}^T \boldsymbol{Q_T} \boldsymbol{R_T})^{-1} \boldsymbol{R_T}^T \boldsymbol{Q_T}^T \boldsymbol{t_T} \\ &= \boldsymbol{R_T}^{-1} (\boldsymbol{R_T}^T)^{-1} \boldsymbol{R_T}^T \hat{\boldsymbol{\theta}}_{\boldsymbol{Q_T}} \\ \hat{\boldsymbol{\theta}}_{\boldsymbol{Q_T}} &= \boldsymbol{R_T} \hat{\boldsymbol{\theta}}_{\boldsymbol{B_T}}. \end{aligned}$$

So we have a simple relationship that relates the coefficients for $\boldsymbol{Q_T}$ and $\boldsymbol{B_T}$. By similar analysis we can show that

$$\hat{\boldsymbol{\theta}}_{\boldsymbol{Q}} = \boldsymbol{R} \hat{\boldsymbol{\theta}}_{\boldsymbol{B}}.$$

In order to interpolate, we require that the reconstructions $\boldsymbol{B}\hat{\boldsymbol{\theta}}_{\boldsymbol{B}}$ and $\boldsymbol{B_T}\hat{\boldsymbol{\theta}}_{\boldsymbol{B_T}}$ are the same at the N_v known pixel values. Since $\boldsymbol{B}$ and $\boldsymbol{B_T}$ are equal at these pixel locations, this can be achieved by setting $\hat{\boldsymbol{\theta}}_{\boldsymbol{B}} = \hat{\boldsymbol{\theta}}_{\boldsymbol{B_T}}$. Therefore, the interpolated reconstruction is

$$\boldsymbol{B}\hat{\boldsymbol{\theta}}_{\boldsymbol{B}} = \boldsymbol{B}\hat{\boldsymbol{\theta}}_{\boldsymbol{B_T}} = \boldsymbol{B}(\boldsymbol{R_T})^{-1} \hat{\boldsymbol{\theta}}_{\boldsymbol{Q_T}},$$

or, if we wish to use orthogonalised subspace functions, the reconstruction is

$$\boldsymbol{Q}\hat{\boldsymbol{\theta}}_{\boldsymbol{Q}} = \boldsymbol{Q}\boldsymbol{R}\hat{\boldsymbol{\theta}}_{\boldsymbol{B}} = \boldsymbol{Q}\boldsymbol{R}(\boldsymbol{R}_{\boldsymbol{T}})^{-1}\hat{\boldsymbol{\theta}}_{\boldsymbol{Q}_{\boldsymbol{T}}};$$

i.e., the modified coefficients are

$$\hat{\boldsymbol{\theta}}_{\boldsymbol{Q}} = \boldsymbol{R}(\boldsymbol{R}_{\boldsymbol{T}})^{-1}\hat{\boldsymbol{\theta}}_{\boldsymbol{Q}_{\boldsymbol{T}}}.$$

Since the deconvolution iteration given in (5.9) is valid for the interpolation degradation model (5.4), it can be used to perform interpolation. In this case, we can set $\alpha = 1$ and, due to the structure of $\boldsymbol{H}$, the update $D(\boldsymbol{\theta}^{(\boldsymbol{k})}) + \boldsymbol{H}^T\left(\boldsymbol{y} - \boldsymbol{H}D(\boldsymbol{\theta}^{(\boldsymbol{k})})\right)$ is equivalent to inserting the known pixels back into the approximation $D(\boldsymbol{\theta}^{(\boldsymbol{k})})$. We will not interpolate using this approach, however this iteration is employed by Li [45]. In this approach BM3D denoising is used with a regularisation parameter that decreases at each iteration. The technique then provides a very effective way to apply the non-local approximation techniques to interpolation. In the following simulations we will compare the proposed interpolation algorithm against this non-local approach, adaptive kernel regression and traditional techniques.

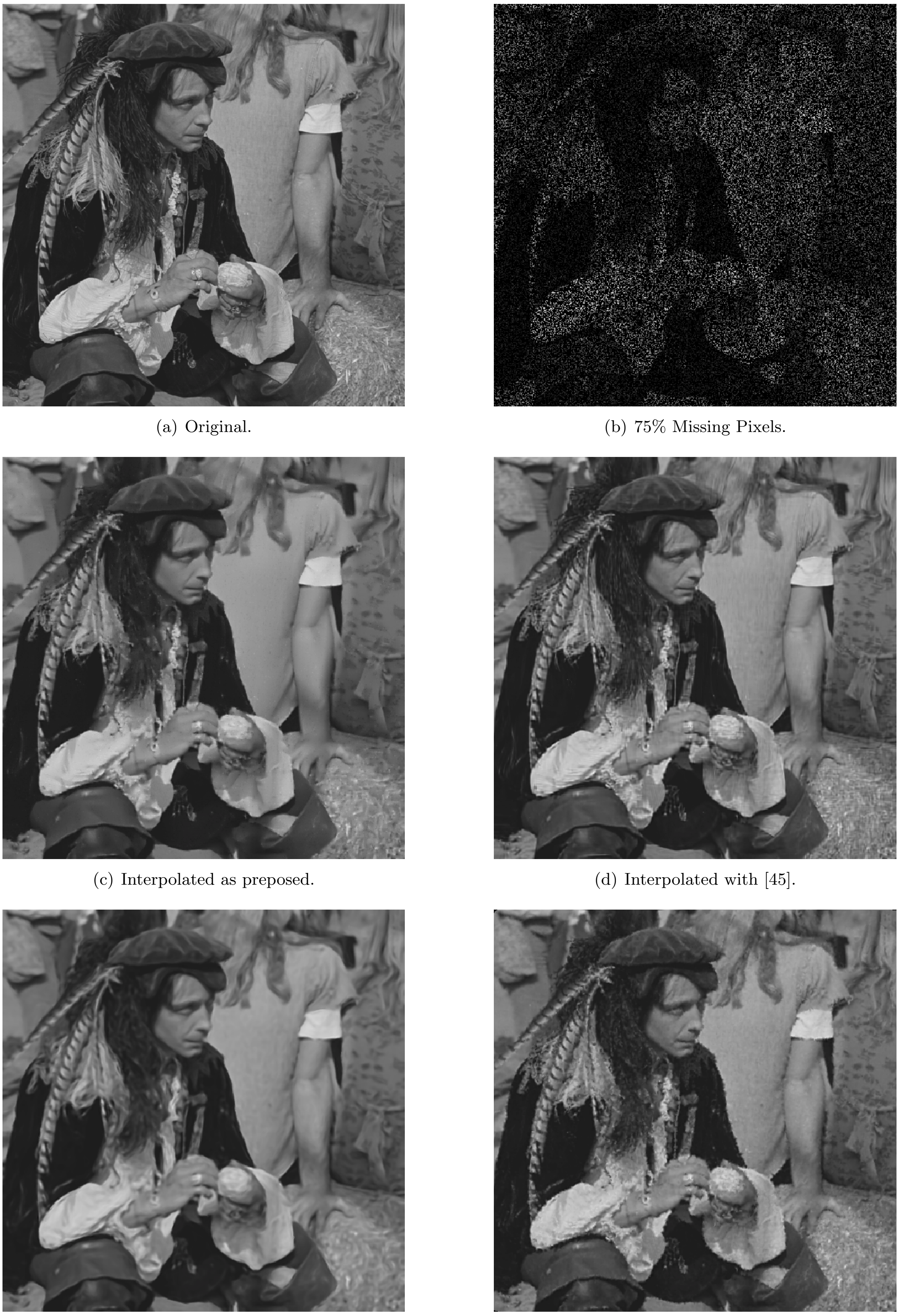
Tables 5.5 - 5.6 and Figs. 5.12 - 5.17 show the interpolation results for natural and depth images.

The proposed method and non local interpolation algorithm are unsupervised, since we use a fixed λ. In order for a fair comparison we used the fixed parameters, given in the kernel regression software, whenever possible. However, in order to produce competitive results in high degradation cases, we tuned the kernel size when 95% of the pixels were removed.

Like denoising, the proposed method is state of the art for depth images and competitive for natural images, particularly in high degradation cases. The piecewise polynomial model, once again, produces smoother results that can lack texture but also prevents distortions in higher degradation cases. In order to optimise the PSNR, all methods retain or insert the known samples into the reconstruction. Since our algorithm produces larger smooth regions, these samples are more evident: see for example the roof in Fig. 5.15(c). This effect explains why the SSIM index performance is not as positive as the denoising case.

		PSNR Interpolation Results						SSIM Index Interpolation Results					
Image	%	NL	Proposed	KR	Bi-Cubic	Bi-Linear	Bi-Constant	NL	Proposed	KR	Bi-Cubic	Bi-Linear	Bi-Constant
Boat (512 × 512)	75	**29.38**	28.67	27.84	27.41	27.28	25.15	**0.8573**	0.8323	0.8153	0.8065	0.8038	0.7338
	80	**28.28**	27.69	27.19	26.57	26.49	24.59	**0.8286**	0.7991	0.7910	0.7762	0.7734	0.7053
	85	**26.79**	26.40	26.04	25.57	25.53	23.77	**0.7878**	0.7534	0.7554	0.7382	0.7352	0.6675
	90	25.14	**25.14**	23.96	24.34	24.40	22.80	**0.7329**	0.6970	0.6908	0.6863	0.6848	0.6185
	95	23.10	**23.39**	21.40	22.52	22.65	21.38	**0.6406**	0.6055	0.5901	0.6067	0.6067	0.5395
Cameraman (256 × 256)	75	**25.33**	24.95	24.73	23.88	23.89	22.11	**0.8677**	0.8459	0.8379	0.8253	0.8238	0.7820
	80	**24.19**	24.16	23.90	23.11	23.19	21.54	**0.8399**	0.8210	0.8146	0.7996	0.7988	0.7562
	85	22.94	**23.15**	22.63	22.05	22.19	20.76	**0.8032**	0.7856	0.7812	0.7639	0.7639	0.7241
	90	21.86	**22.42**	20.95	21.14	21.31	19.91	**0.7599**	0.7476	0.7302	0.7221	0.7226	0.6839
	95	20.13	**20.81**	17.68	19.36	19.56	18.51	**0.6915**	0.6744	0.6430	0.6551	0.6569	0.6199
Hill (512 × 512)	75	**30.44**	29.91	29.44	28.99	28.96	27.13	**0.8487**	0.8181	0.8113	0.8117	0.8080	0.7462
	80	**29.42**	29.08	28.76	28.19	28.20	26.47	**0.8188**	0.7865	0.7869	0.7813	0.7775	0.7147
	85	**28.19**	28.06	27.73	27.31	27.37	25.70	**0.7781**	0.7448	0.7518	0.7424	0.7393	0.6740
	90	26.89	**26.96**	25.82	26.25	26.35	24.80	**0.7253**	0.6886	0.6878	0.6920	0.6892	0.6204
	95	24.99	**25.19**	23.84	24.22	24.36	23.30	**0.6406**	0.6030	0.5987	0.6167	0.6149	0.5387
Lena (512 × 512)	75	**33.18**	32.21	32.10	31.31	30.96	28.19	**0.9102**	0.8945	0.8967	0.8885	0.8862	0.8279
	80	**31.80**	31.07	31.37	30.20	29.95	27.44	**0.8924**	0.8754	0.8834	0.8702	0.8677	0.8058
	85	**30.24**	29.74	30.09	29.00	28.82	26.61	**0.8711**	0.8499	0.8626	0.8472	0.8445	0.7791
	90	**28.36**	28.11	27.13	27.48	27.37	25.35	**0.8373**	0.8117	0.8127	0.8124	0.8096	0.7383
	95	25.67	**26.14**	24.91	24.94	24.95	23.47	**0.7793**	0.7551	0.7587	0.7548	0.7525	0.6722
Man (512 × 512)	75	**29.49**	29.12	28.99	28.49	28.40	26.19	**0.8700**	0.8407	0.8415	0.8445	0.8394	0.7781
	80	**28.47**	28.26	28.24	27.58	27.54	25.54	**0.8424**	0.8103	0.8197	0.8153	0.8104	0.7477
	85	**27.37**	27.23	27.19	26.66	26.66	24.82	**0.8078**	0.7692	0.7886	0.7799	0.7752	0.7104
	90	26.02	**26.03**	25.08	25.46	25.51	23.77	**0.7562**	0.7144	0.7284	0.7288	0.7249	0.6570
	95	24.18	**24.30**	22.50	23.55	23.65	22.24	**0.6699**	0.6305	0.6301	0.6476	0.6454	0.5719
Peppers (256 × 256)	75	**29.37**	27.22	25.78	26.71	26.33	23.31	**0.9091**	0.8926	0.8922	0.8783	0.8751	0.8103
	80	**27.80**	26.07	25.10	25.56	25.27	22.80	**0.8838**	0.8701	0.8751	0.8547	0.8516	0.7843
	85	**26.67**	24.84	24.14	24.50	24.25	21.96	**0.8657**	0.8427	0.8491	0.8278	0.8243	0.7517
	90	**25.18**	23.40	21.93	23.44	23.16	20.78	**0.8312**	0.7980	0.7897	0.7896	0.7862	0.7027
	95	**22.65**	21.60	20.15	21.35	21.14	19.30	**0.7597**	0.7170	0.7081	0.7150	0.7115	0.6140
Average		**26.78**	26.38	25.55	25.57	25.52	23.66	**0.8036**	0.7758	0.7741	0.7693	0.7668	0.7025

Table 5.5.: PSNR and SSIM index comparisons of the proposed interpolation algorithm for natural images. The interpolation algorithms used for comparison are NL, the non-local algorithm presented in [45], KR, adaptive kernel regression [68], and bi-cubic interpolation. The best result is shown in bold and % denotes the percentage of pixels that have been randomly removed.

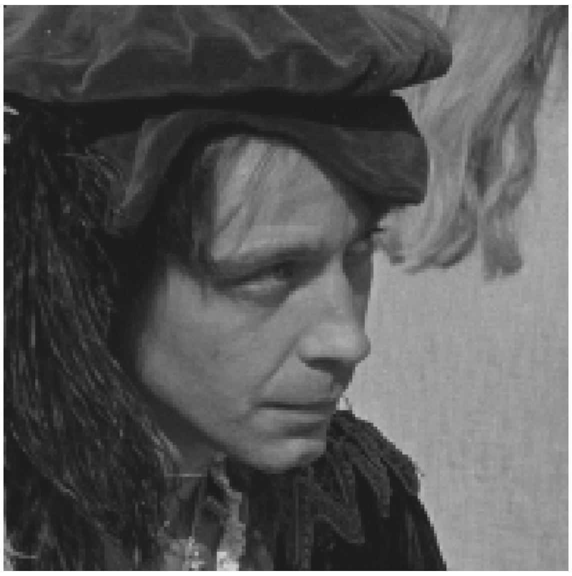


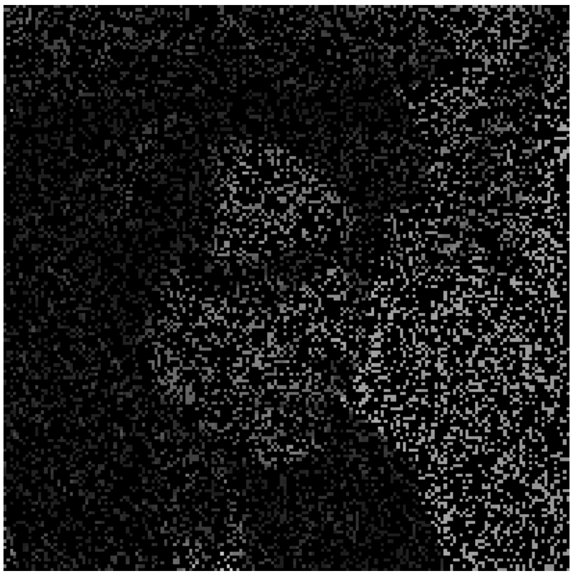
(a) Original. (b) 75% Missing Pixels.

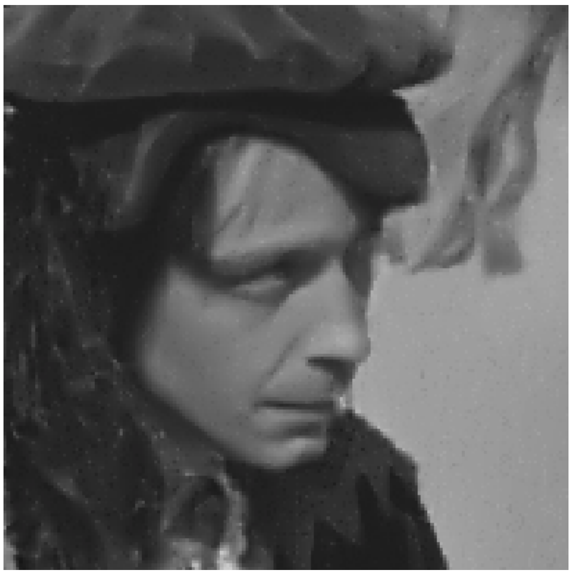
(c) Interpolated as preposed. (d) Interpolated with [45].

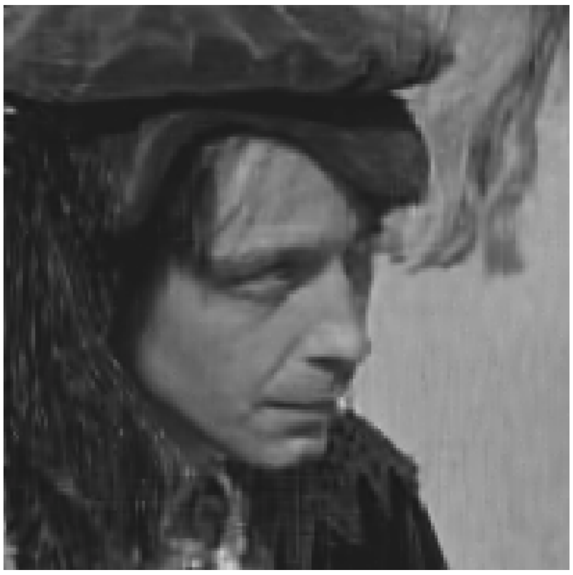
(e) Interpolated with adaptive kernel regression. (f) Bi-cubic interpolation.

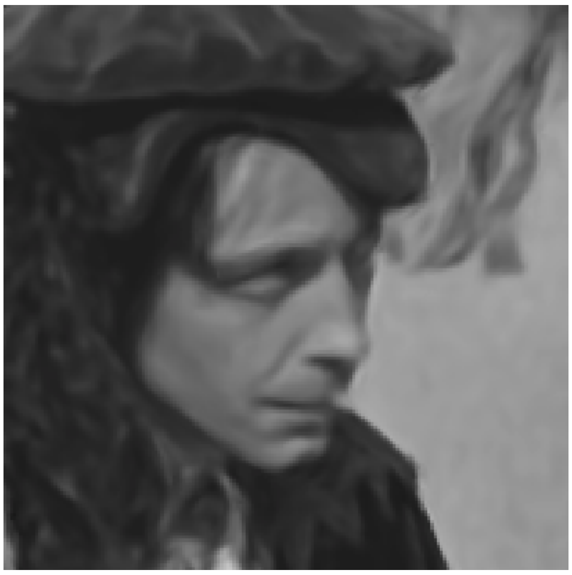
Figure 5.12.: Visual comparison of the four top performing interpolation algorithms for the hill natural image with 75% missing pixels.


(a) Close up of original.


(b) Close up of degraded with 75% Missing Pixels.


(c) Close up of proposed interpolation.


(d) Close up of nonlinear interpolation [45].


(e) Close up of adaptive kernel regression interpolation.

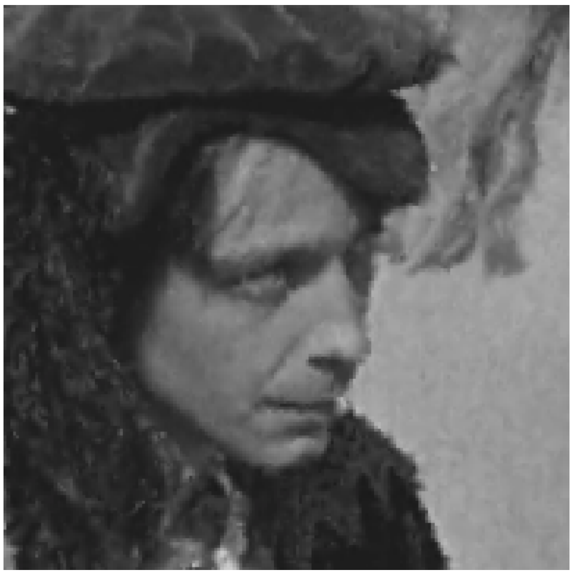
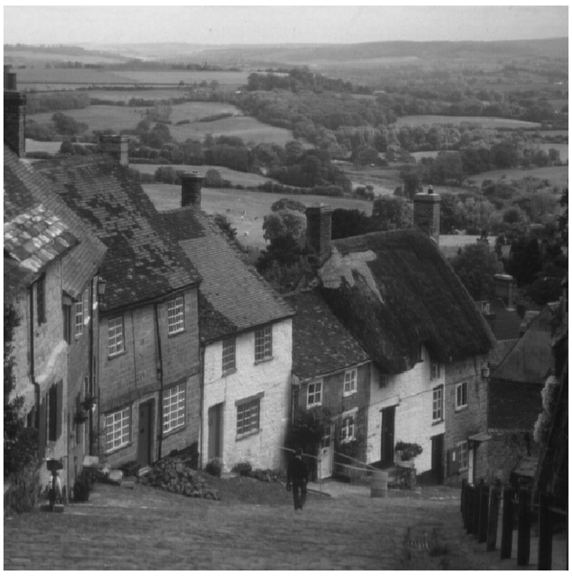
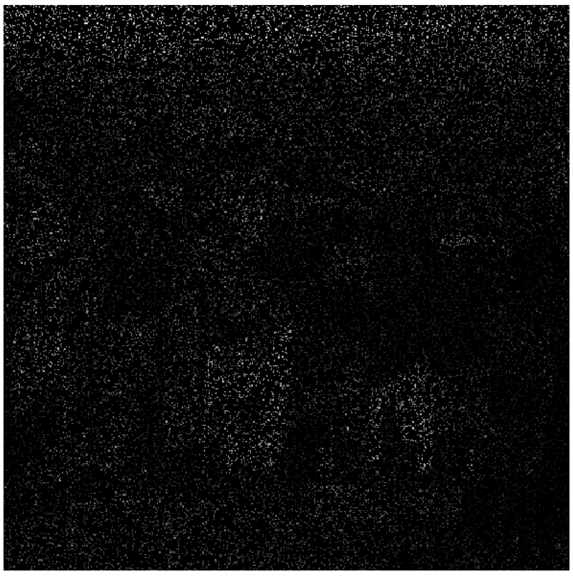
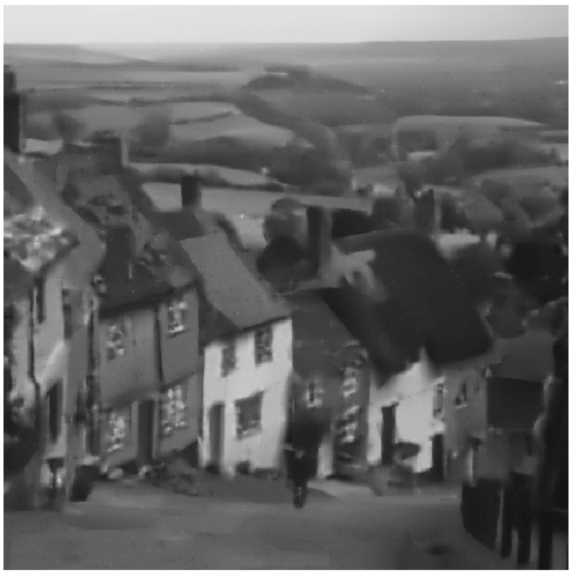
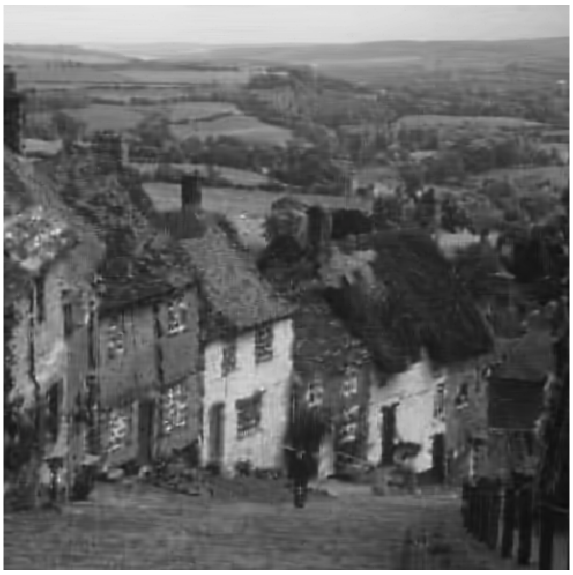

(f) Close up of bi-cubic interpolation.

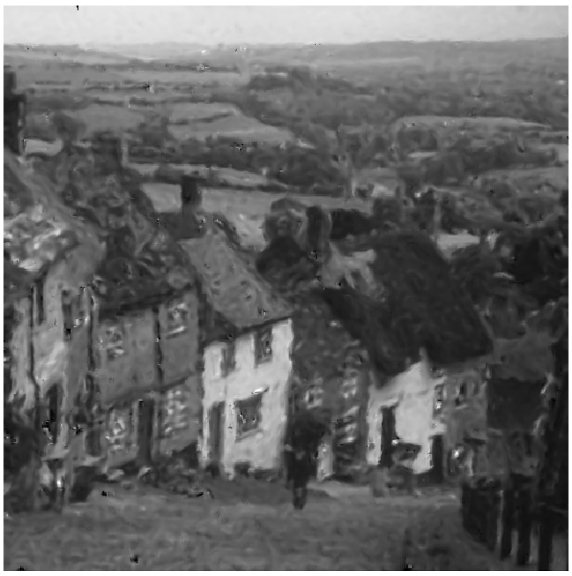
Figure 5.13.: Visual comparison of the four top performing interpolation algorithms for the man natural image with 75% missing pixels.


(a) Original.


(b) 90% Missing Pixels.


(c) Interpolated as preposed.


(d) Interpolated with [45].


(e) Interpolated with adaptive kernel regression.

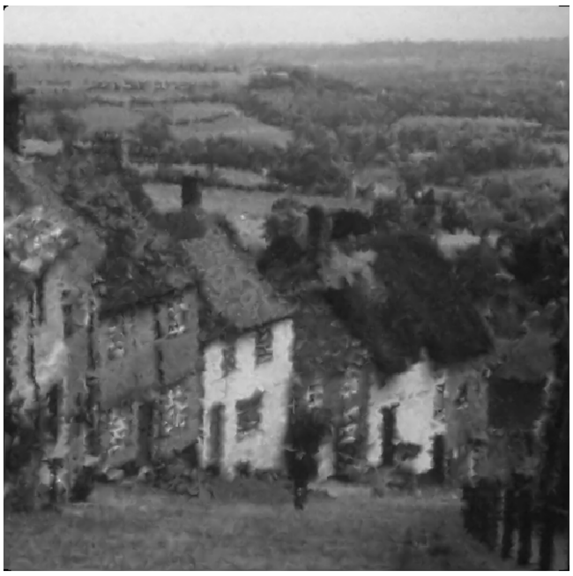
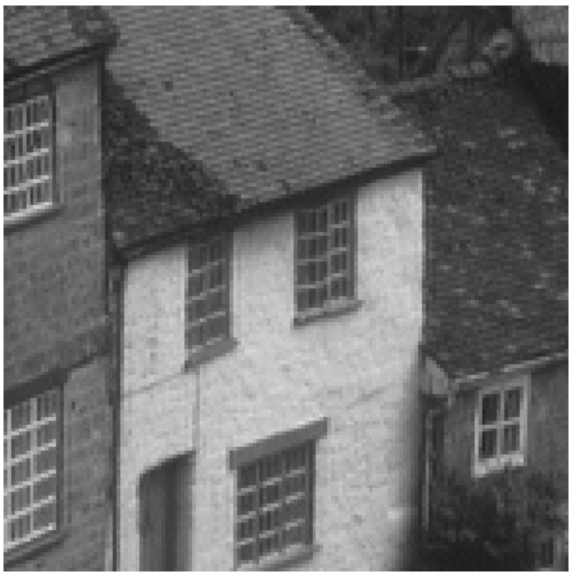
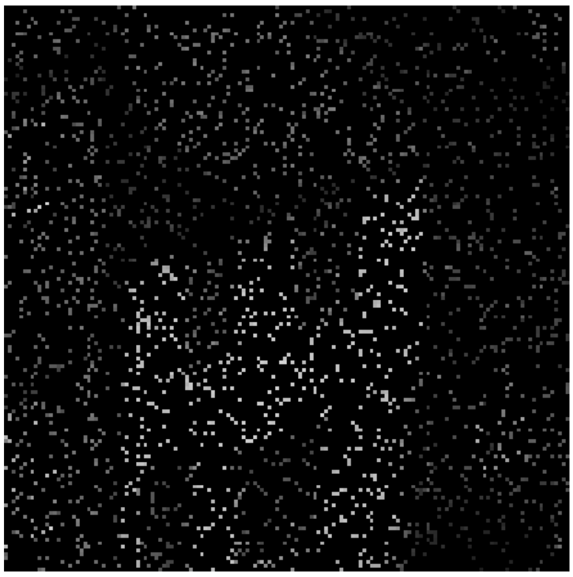
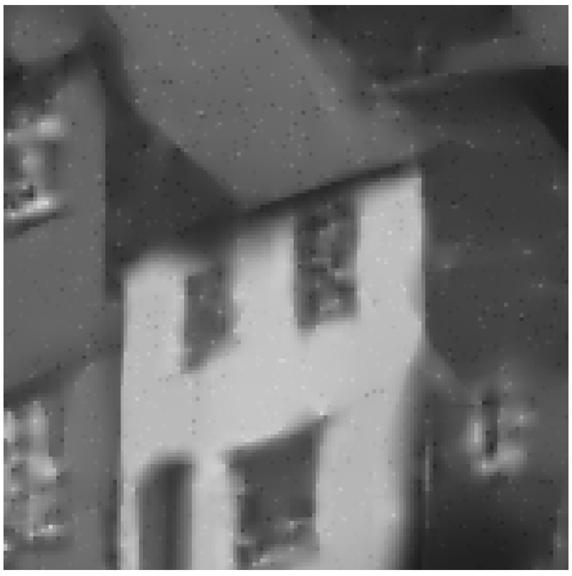
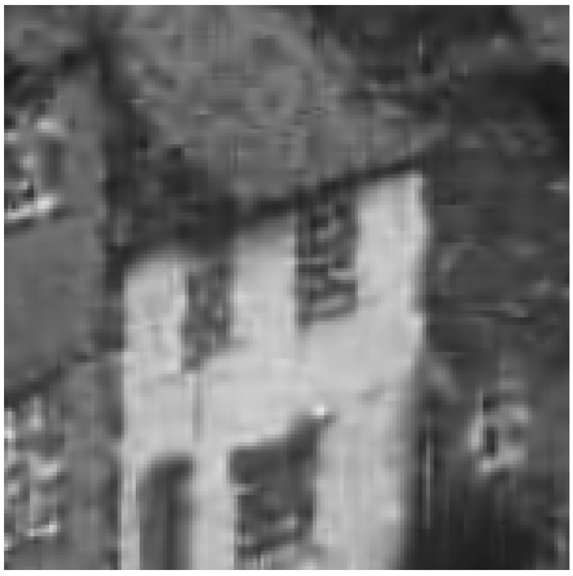

(f) Bi-linear interpolation.

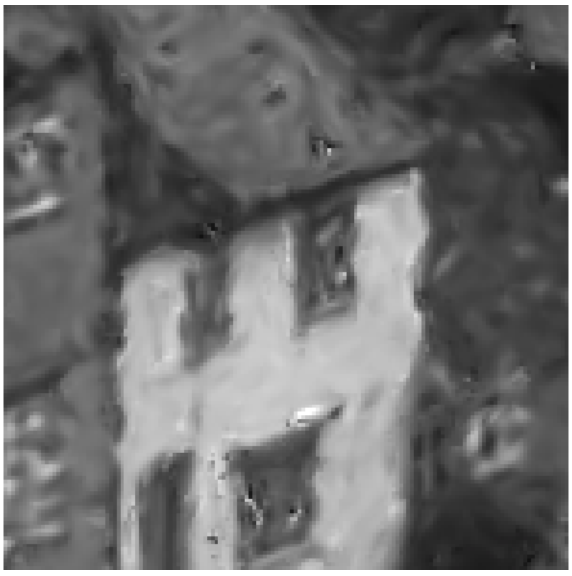
Figure 5.14.: Visual comparison of the four top performing interpolation algorithms for the hill natural image with 90% missing pixels.


(a) Close up of original.


(b) Close up of degraded with 90% Missing Pixels.


(c) Close up of proposed interpolation.


(d) Close up of nonlinear interpolation [45].


(e) Close up of adaptive kernel regression interpolation.

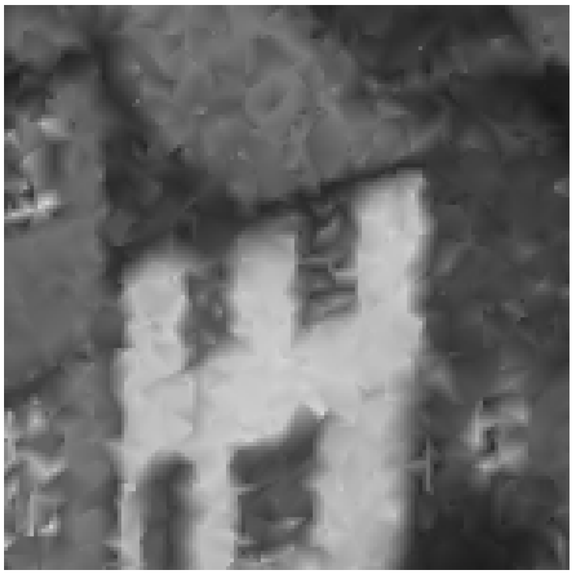
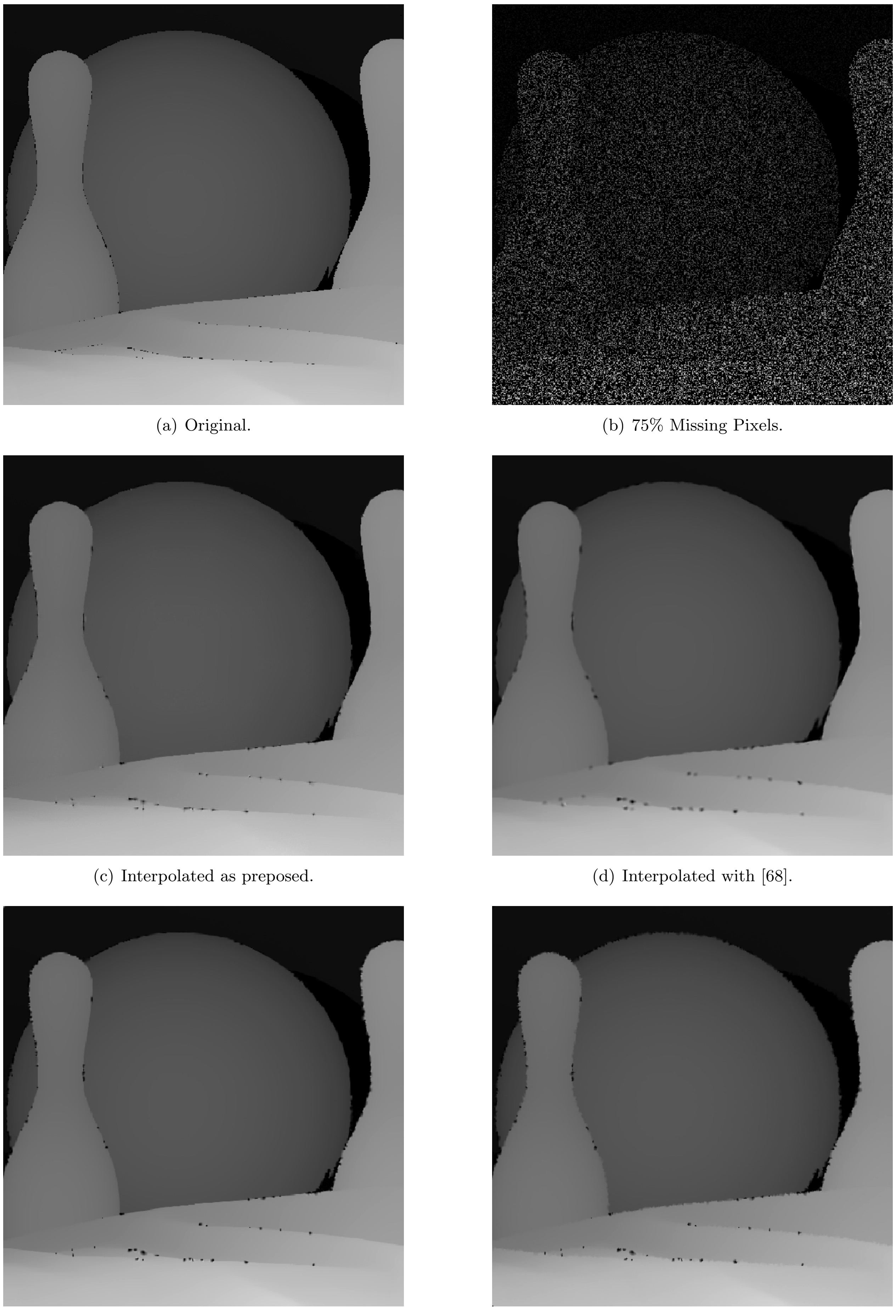

(f) Close up of bi-linear interpolation.

Figure 5.15.: Visual comparison of the four top performing interpolation algorithms for the hill natural image with 90% missing pixels.

		PSNR Interpolation Results						SSIM Index Interpolation Results					
Image	%	Proposed	NL	KR	Bi-Linear	Bi-Cubic	Bi-Constant	Proposed	NL	KR	Bi-Linear	Bi-Cubic	Bi-Constant
Aloe (512 × 512)	75	**31.05**	30.60	30.73	29.50	29.32	27.61	**0.9532**	0.9516	0.9491	0.9316	0.9303	0.9129
	80	**30.42**	29.87	29.89	28.95	28.77	27.19	**0.9454**	0.9426	0.9390	0.9222	0.9202	0.9038
	85	**29.64**	28.82	28.54	28.20	27.97	26.41	**0.9344**	0.9283	0.9243	0.9094	0.9062	0.8899
	90	**28.72**	27.83	26.79	27.39	27.15	25.68	**0.9193**	0.9099	0.8952	0.8929	0.8883	0.8729
	95	**27.21**	26.23	25.61	25.81	25.60	24.51	**0.8871**	0.8767	0.8700	0.8649	0.8584	0.8461
Art (512 × 512)	75	**29.76**	29.03	29.33	28.35	28.13	26.75	**0.9469**	0.9433	0.9411	0.9272	0.9257	0.9084
	80	**29.11**	28.19	28.37	27.67	27.42	26.19	**0.9392**	0.9337	0.9319	0.9169	0.9153	0.8987
	85	**28.42**	27.65	27.48	27.25	27.02	25.70	**0.9286**	0.9241	0.9200	0.9070	0.9054	0.8878
	90	**27.52**	26.76	25.47	26.48	26.21	24.98	**0.9122**	0.9080	0.8951	0.8908	0.8886	0.8718
	95	**26.03**	25.24	24.67	24.65	24.37	23.84	**0.8784**	0.8768	0.8701	0.8629	0.8600	0.8457
Baby (512 × 512)	75	**35.97**	34.82	35.58	34.35	34.19	32.74	**0.9774**	0.9738	0.9759	0.9697	0.9691	0.9627
	80	**35.45**	34.15	34.91	33.82	33.63	32.33	**0.9745**	0.9699	0.9723	0.9658	0.9651	0.9594
	85	**34.50**	33.15	33.62	33.03	32.83	31.61	**0.9700**	0.9633	0.9656	0.9603	0.9594	0.9541
	90	**33.81**	32.29	32.04	32.40	32.16	30.79	**0.9649**	0.9563	0.9557	0.9544	0.9530	0.9466
	95	**32.53**	30.65	31.20	31.24	30.97	29.74	**0.9540**	0.9456	0.9465	0.9441	0.9422	0.9371
Bowling ball (512 × 512)	75	**34.59**	32.94	33.73	32.41	32.14	30.56	**0.9789**	0.9758	0.9767	0.9705	0.9691	0.9629
	80	**33.88**	32.07	32.88	31.60	31.35	29.93	**0.9757**	0.9710	0.9725	0.9658	0.9643	0.9581
	85	**33.36**	31.41	31.54	31.06	30.82	29.36	**0.9720**	0.9665	0.9658	0.9612	0.9592	0.9532
	90	**32.21**	30.43	29.31	30.18	29.93	28.71	**0.9656**	0.9586	0.9542	0.9533	0.9507	0.9464
	95	**30.34**	29.11	29.39	28.02	27.89	27.89	**0.9497**	0.9480	0.9482	0.9437	0.9397	0.9366
Average		**31.23**	30.06	30.05	29.62	29.39	28.13	**0.9464**	0.9412	0.9385	0.9307	0.9285	0.9178

Table 5.6.: PSNR and SSIM index comparisons of the proposed interpolation algorithm for depth images. The interpolation algorithms used for comparison are NL, the non-local algorithm presented in [45]; KR, adaptive kernel regression[68]; and bi-cubic, bi-linear and bi-constant interpolation. The best result is shown in bold and % denotes the percentage of pixels that have been randomly removed.



(a) Original.

(b) 75% Missing Pixels.

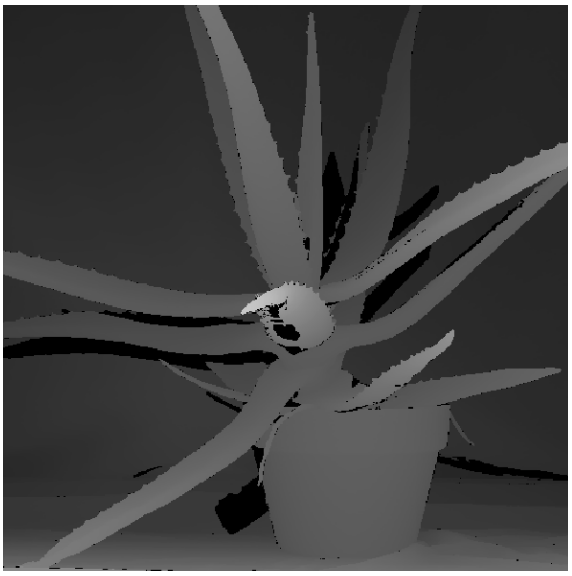
(c) Interpolated as preposed.

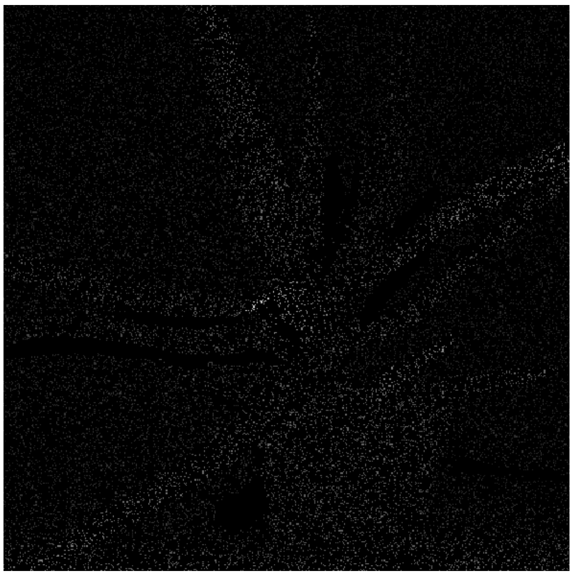
(d) Interpolated with [68].

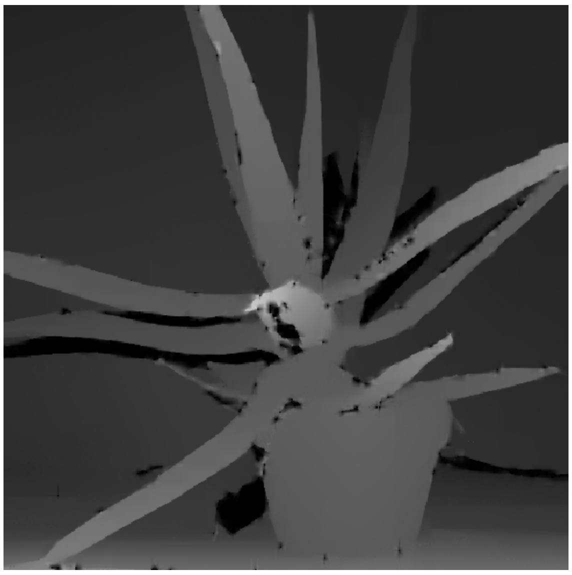
(e) Interpolated with [45].

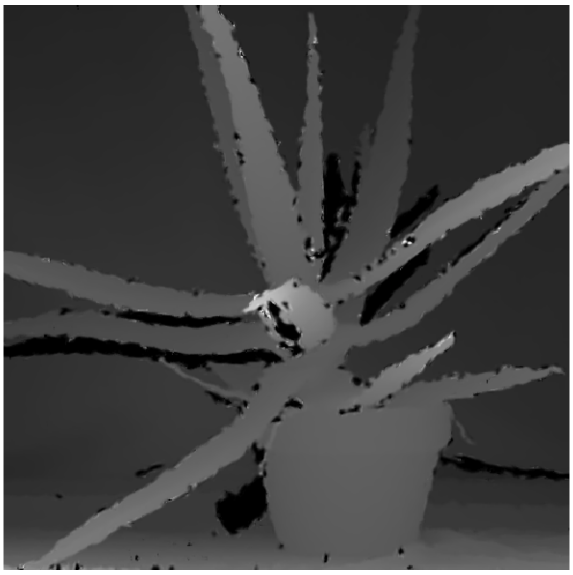
(f) Bi-linear interpolation.

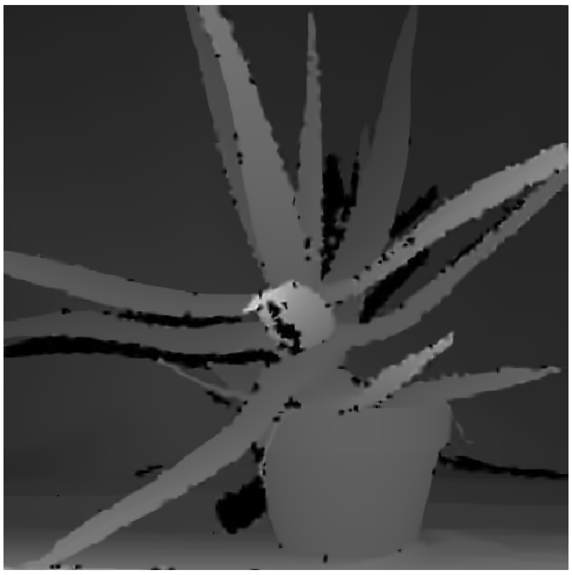
Figure 5.16.: Visual comparison of the top four interpolation algorithms for the bowling ball depth image with 75% missing pixels.


(a) Original.


(b) 90% Missing Pixels.


(c) Interpolated as preposed.


(d) Interpolated with [68].


(e) Interpolated with [45].

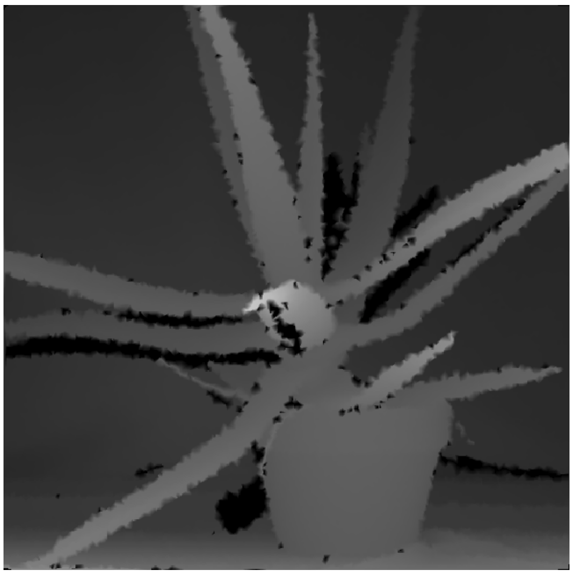
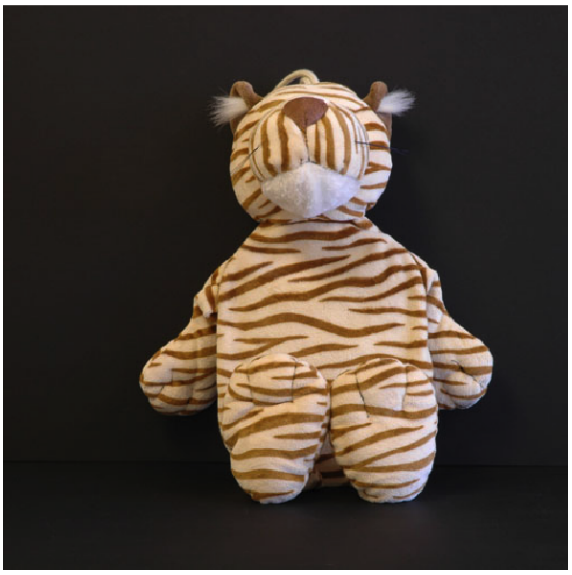

(f) Bi-linear interpolation.

Figure 5.17.: Visual comparison of the top four interpolation algorithms for the aloe depth image with 90% missing pixels.

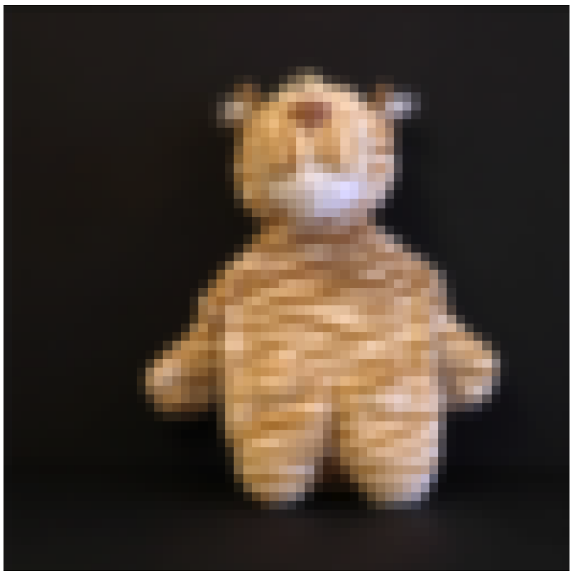
5.5. Super resolution

In Chapter 3 we showed that the problem of multi-view super resolution could be decomposed into three steps: registration, interpolation and deconvolution. We briefly outlined an algorithm, [3], that used FRI theory to accurately register low resolution images. In the same chapter we also introduced bi-cubic interpolation. In this section we will perform super resolution, registering and interpolating using these two approaches. The deconvolution will be performed using our two-step approach and, in order for comparison, other state of the art deconvolution methods. This section can, thus, be thought of as an interesting application that provides further analysis of our deconvolution algorithm. Note that the interpolation in the super resolution process is carried out on blurred data. Since our model is designed for piecewise polynomial images with sharp discontinuities it does not make sense to use our interpolation here. Furthermore, a weakness of bi-cubic interpolation is its inability to deal with sharp edge contours, a problem that is reduced when dealing with blurred data.

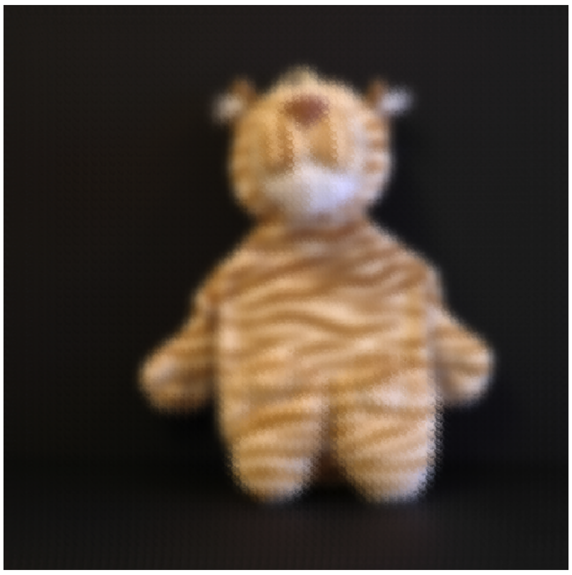
Figures 5.18 - 5.19 show a super resolution simulation experiment. The original image has been artificially sampled from different viewpoints to create 64 low resolution images. The viewpoints are assumed to be unknown and registration, interpolation and deconvolution performed as described. The deconvolution algorithms used are CGLS, FoRWaRD and the two-step approach using either BM3D or the proposed denoising. Interestingly, in this case our algorithm outperforms the BM3D approach. In previous deconvolution experiments we were competitive but suffered from not modelling the noise as well as the BM3D. However, in this situation, where the noise is due to registration and interpolation errors, we perform better. This suggests that in practical applications, where the noise can be harder to model, our approach may be more competitive. The CGLS reconstruction produces the best PSNR but suffers from heavy Gibbs ripples. One can easily argue, using both visual inspection and the SSIM index, that, despite the CGLS's higher PSNR, our result is favourable.



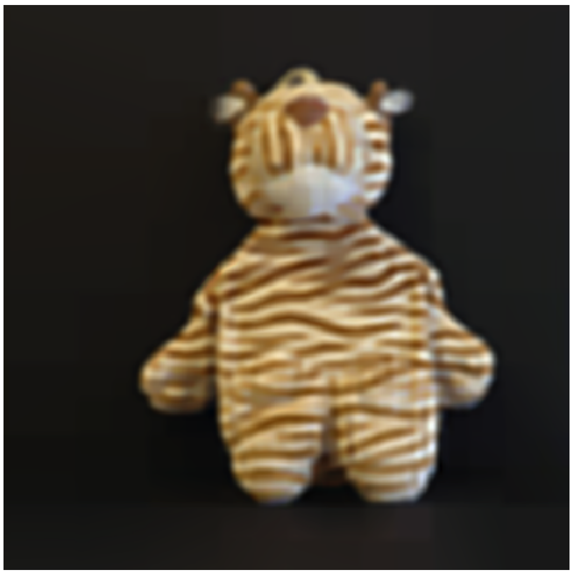
(a) Original (512 × 512).



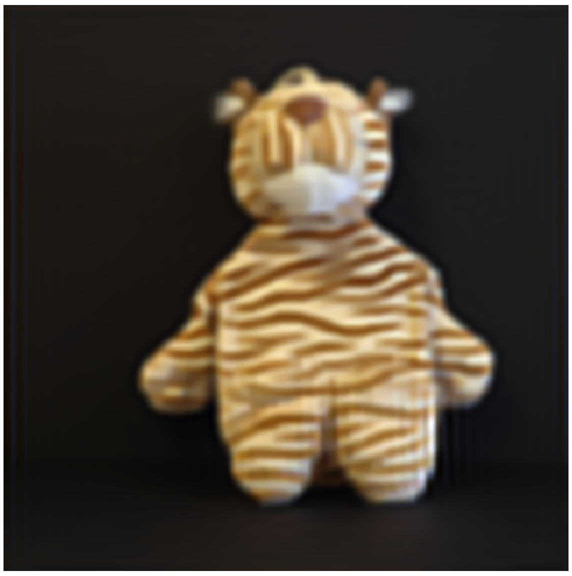
(b) One of 64 low resolution images (64 × 64).



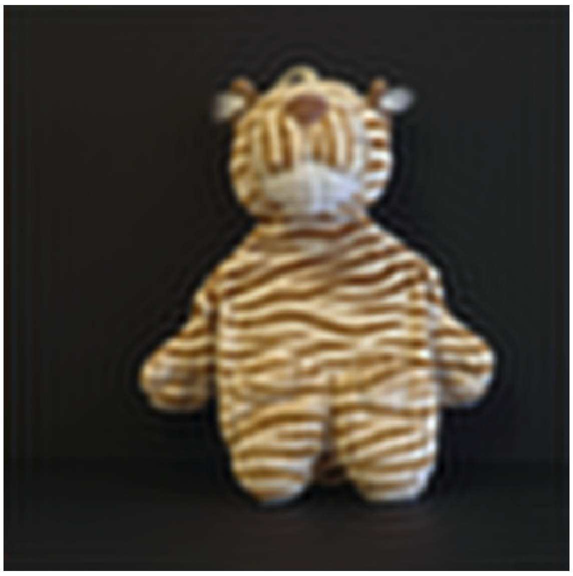
(c) Interpolated reconstruction before restoration. (512 × 512)



(d) Super resolved using proposed restoration (512 × 512). PSNR=24.25, SSIM=0.8430

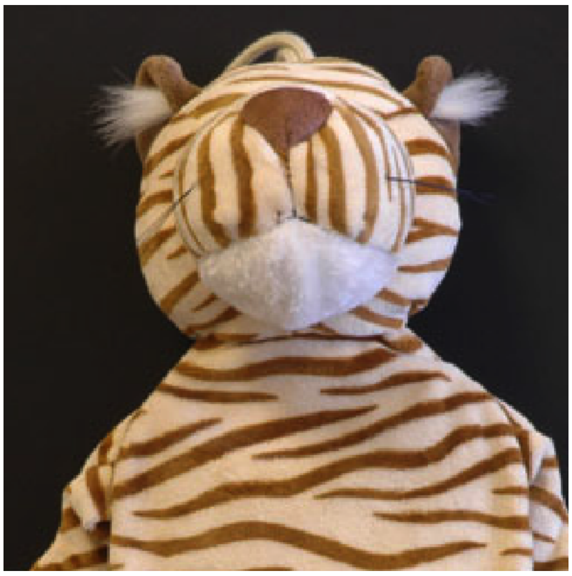


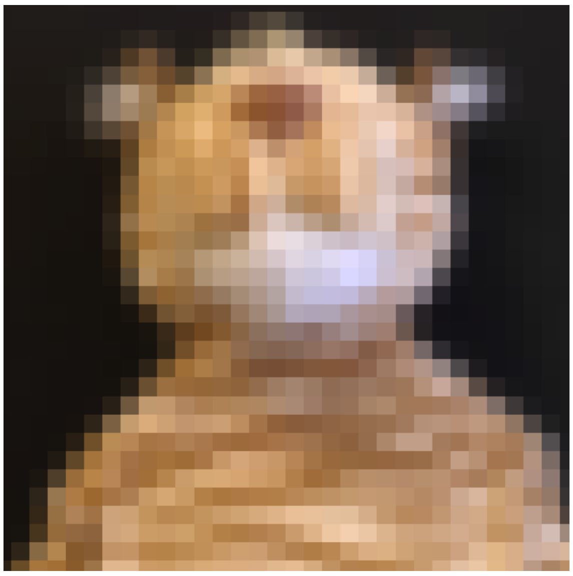
(e) Super resolved using BM3D restoration (512 × 512). PSNR=23.87, SSIM=0.8280

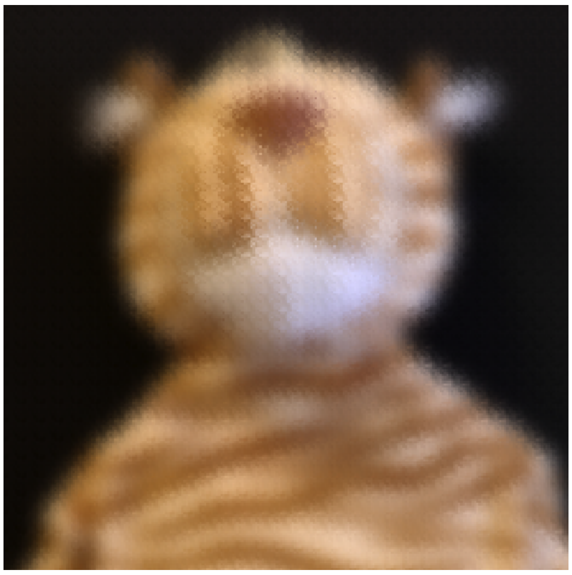


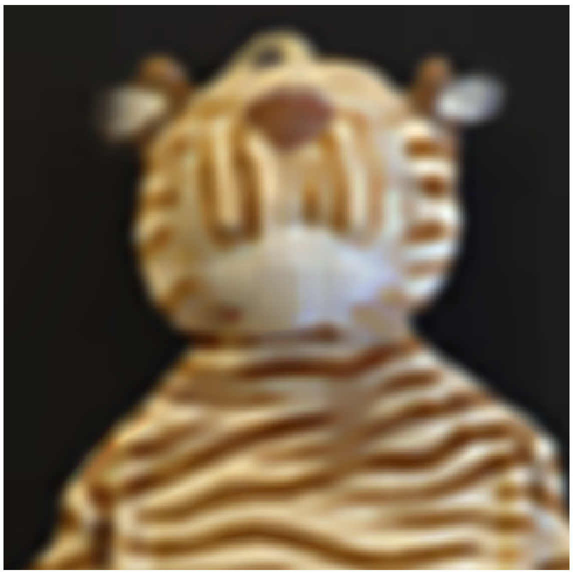
(f) Super resolved using CGLS restoration (512 × 512). PSNR=24.30, SSIM=0.8343

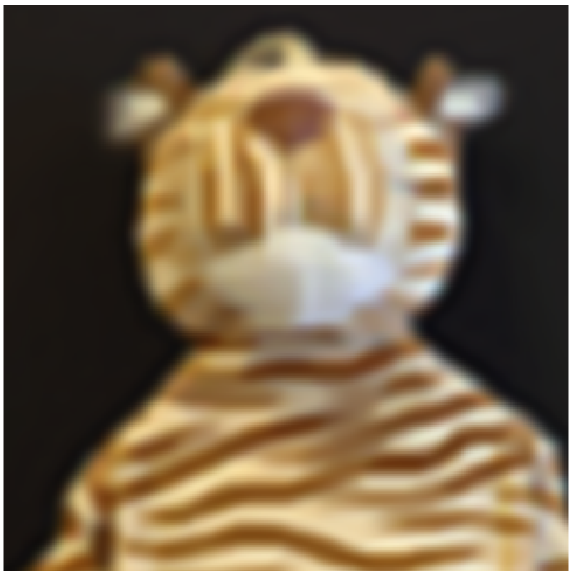
Figure 5.18.: Comparison of various deconvolution algorithms for the problem of image super resolution.


(a) Close up of original


(b) Close up of one of 64 low resolution images


(c) Close up of interpolated reconstruction before restoration.


(d) Close up of super resolved using proposed restoration.


(e) Close up of super resolved using BM3D restoration.

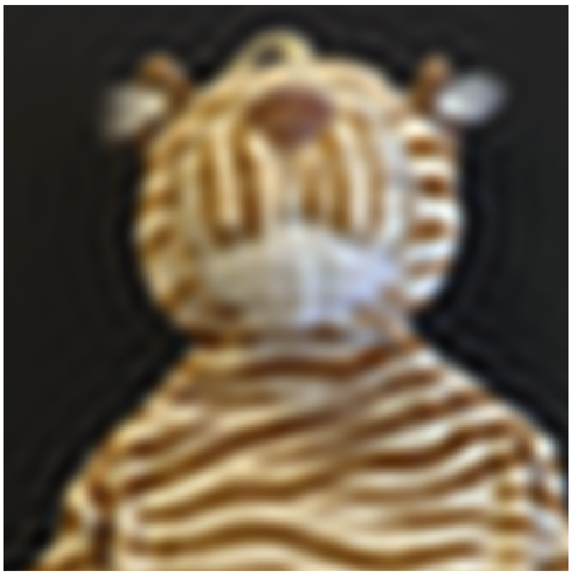

(f) Close up of super resolved using CGLS restoration

Figure 5.19.: A closeup of the comparison of various deconvolution algorithms for the problem of image super resolution.

5.6. Summary

In this chapter we have adapted the previously presented approximation algorithm to restoration and enhancement. Denoising was achieved by using cycle spinning, to exploit the shift variance, and selecting a regularisation parameter proportional to the noise variance.

The same denoising algorithm was used to perform deconvolution in two ways. The first used regularised inverses to sharpen the image, and the proposed denoising to remove the resulting noise. The algorithm was competitive for natural images but was not as good as non-local approaches which, due to the fact that they better distribute the noise into frequency band, can filter coloured noise more effectively. However, we applied the same algorithms to the deconvolution stage of image super resolution and in this case, where the noise is due to registration and interpolation errors and thus harder to model, initial results suggest the proposed method is even better.

Finally we proposed an algorithm that approximated the signal over just the known pixels. Interpolation could then be achieved by reconstructing the approximation using the full polynomial subspace functions. Simulation results for both denoising and interpolation suggest that the proposed algorithms are competitive for natural images, particularly when the degradation is high. Furthermore, state of the art performance is achieved when the original signal is close to the model. Depth images are a class of signals that fit this criterium and have recently being receiving increased research interest. The proposed algorithm for interpolating irregularly sampled depth data could be particularly applicable to depth image acquisition.

CHAPTER 6

Conclusions and Future Work

6.1. Thesis summary

In this thesis, we have seen that sparse promoting priors are a very useful tool for image restoration. Consequently, the quest for sparser approximations of images is an attractive research topic, which has been a key part of this thesis.

Sparse seeking optimisation problems and sparse promoting transforms were introduced in Chapter 2. We saw that the well known and commonly used hard and soft thresholding operators minimise l_0 and l_1 constrained problems when the dictionary is unitary. In the non-unitary case these operators are only the first step of iterative algorithms that converge to local and global minimums of the non-convex and convex problems respectively. These iterative thresholding or shrinkage algorithms have had many applications in image processing and have been presented in many different forms. In this thesis we presented them from the perspective of proximal gradient descent and MM algorithms, which provided both an intuitive understanding and a reasonable formal analysis.

Later in the same chapter, an overview of sparse promoting transforms of 1-D and 2-D signals was given. We saw that wavelets with compact support and vanishing moments sparsely decompose piecewise polynomial signals, particularly in 1-D. In 2-D, wavelets' lack of direction adaptability produces inefficiencies around complex edge structures. This can be improved by using overcomplete and adaptive transforms, such as Ridgelets, Curvelets and Bandlets, with increased direction adaptability. Shukla et al [63] proposed a compression algorithm that could optimally represent piecewise polynomial signals, by using a quadtree decomposition and a more flexible joining to adaptively partition the image. Each adaptive region is approximated by a piecewise polynomial with at most one discontinuity. In Chapter 4, we modified this algorithm and made it more appropriate to restoration and enhancement problems. This included a novel fast way to search for the optimum edge discontinuity for each adaptive region that eased the computational load significantly.

The thesis also presented applications in the following image restoration and enhancement problems: denoising, deconvolution, interpolation and multi-view super resolution. In Chapter 3 it was shown that sparse approximation is a powerful prior that is at the heart of many of the state of the art algorithms for these problems.

Chapter 5 proposed new algorithms using the proposed quadtree structured approximation. Denoising was performed using the previously proposed approximation algorithm with a regularisation parameter proportional to the noise variance. Cycle spinning was also used to exploit the shift variance of the transform. Simulation results suggest that the proposed algorithm is state of the art when the signal is in the model (e.g. depth images) and competitive for natural images when the degradation is high.

We also presented two deconvolution algorithms, one that applies a Fourier regularised inverse followed by denoising and, the other, an iterative approach that extends the iterative shrinkage algorithms to our non-linear model. Simulation results for natural and depth images were given for the regularised inverse approach, with comparisons against the state of the art. Due to the slow convergence of the iterative algorithm, only a result for a piecewise polynomial image was given. A further application for the regularised filter approach was given in the form of multi-view super resolution. In Chapter 3 it was shown that this problem could be solved using multiple steps, including deconvolution. In Chapter 5 simulation results were provided to compare the proposed deconvolution algorithm to the state of the art for this problem. In traditional simulations the proposed deconvolution is not competitive for natural images. However, the super resolution result suggests that the approach may have some merit in practical situations when the noise is harder to model.

Finally, interpolation was achieved by approximating an image, with missing pixels, over just the known pixels and reconstructing with the full polynomial subspace functions. Simulation results provided comparisons for the problem of interpolating from an irregularly sampled grid of data. Like denoising, our algorithm is competitive for natural images and state of the art for depth images, which could have interesting applications in depth image acquisition.

6.2. Future research

To conclude this thesis we discuss some possible directions for future research:

- Depth images provide an interesting class of images for the proposed framework. In many depth sensing setups, a depth measurement is only obtained at certain locations and interpolation is required in order to obtain a full depth image. Additionally, a full colour image is normally also captured, which, with additional research, could be exploited to further enhance any interpolation. Further investigation of the Microsoft Kinect sensor is also needed in order to extract the pure depth sensor measurements, before Microsoft's own interpolation. This would provide a comparison between any proposed interpolation and the current implementation.

- In the reported interpolation simulations, we used a fixed value of λ for all experiments. In many cases, improved performance could be obtained with a slightly different regularisation parameter. This naturally raises the question if a more theoretical formulation for λ can be obtained. An alternative is to investigate the use of search strategies, such as the L-curve method; however, these methods, unfortunately, require multiple approximations to be calculated. This computation could potentially be reduced by combining a regularisation search strategy with cycle spinning.

- The proposed two-step deconvolution algorithm has potential, however it is limited by the need to select an appropriate regularisation parameter for each experiment. This could again be addressed with a search strategy, such as the L-curve method; however, in this case, a more suitable strategy may be to use an algorithm to estimate the standard deviation of the noise.

- Initial super resolution simulations suggest that the proposed deconvolution algorithm could be an effective algorithm for this problem. Further simulations are needed on both artificially generated and real data in order to fully understand the merits of this approach.

APPENDIX A

Proofs

A.1. Proof that soft thresholding solves the 1-D l_1 constrained minimisation problem

We claim that

$$\hat{\theta} = arg\min_{\theta}\left\{(y-\theta)^2 + \lambda\left|\theta\right|\right\} = \begin{cases} y - \frac{\lambda}{2} & \text{if} \quad y > \frac{\lambda}{2} \\ 0 & \text{if} \quad -\frac{\lambda}{2} \leq y \leq \frac{\lambda}{2} \\ y + \frac{\lambda}{2} & \text{if} \quad y < -\frac{\lambda}{2} \end{cases}.$$

Proof. Since $\hat{\theta}$ is the minimum, there exists a subgradient of $(y-\theta)^2 + \lambda\left|\theta\right|$, at $\hat{\theta}$, that is zero:

$$0 \in 2\hat{\theta} - 2y + \lambda\partial\left|\hat{\theta}\right|.$$

Thus, when $\hat{\theta} > 0$

$$0 = 2\hat{\theta} - 2y + \lambda,$$

and

$$\hat{\theta} = y - \frac{\lambda}{2} \quad \text{if} \quad y > \frac{\lambda}{2}. \tag{A.1}$$

Similarly, when $\hat{\theta} < 0$

$$0 = 2\hat{\theta} - 2y - \lambda$$

and

$$\hat{\theta} = y + \frac{\lambda}{2} \quad \text{if} \quad y < -\frac{\lambda}{2}. \tag{A.2}$$

Finally, when $\hat{\theta} = 0$

$$0 \in -2y + \lambda[-1, 1],$$

and

$$\hat{\theta} = 0 \quad \text{if} \quad y \in \left[-\frac{\lambda}{2}, \frac{\lambda}{2}\right]. \tag{A.3}$$

Combining (A.1), (A.2) and (A.3) completes the proof.

□

A.2. Proof that (2.17) is a maximiser of (2.16)

Let $C(\boldsymbol{\theta}) = C_1(\boldsymbol{\theta}) + C_2(\boldsymbol{\theta})$. We claim that if ∇C_1 is Lipschitz continuous with constant L and $t \leq \frac{1}{L}$ then

$$C_{prox}(\boldsymbol{\theta} \mid \boldsymbol{a}) = C_1(\boldsymbol{a}) + \nabla C_1(\boldsymbol{a})^T(\boldsymbol{\theta} - \boldsymbol{a}) + \frac{1}{2t}\|\boldsymbol{\theta} - \boldsymbol{a}\|_2^2 + C_2(\boldsymbol{\theta})$$

is a maximiser of $C(\boldsymbol{\theta})$; i.e.,

$$\begin{aligned} &C_{prox}(\boldsymbol{\theta} \mid \boldsymbol{a}) \geq C(\boldsymbol{\theta}) \quad \text{and} \\ &C_{prox}(\boldsymbol{a} \mid \boldsymbol{a}) = C(\boldsymbol{a}). \end{aligned}$$

Proof. By inspection, we can see that $C_{prox}(\boldsymbol{a} \mid \boldsymbol{a}) = C(\boldsymbol{a})$. To prove the inequality note that, since ∇C_1 is Lipschitz continuous with constant L, $\nabla^2 C_1 \leq L$; i.e., $\boldsymbol{\theta}^T \nabla^2 C_1 \boldsymbol{\theta} \leq L$ for all $\boldsymbol{\theta}$. Therefore

$$\begin{aligned} \boldsymbol{\theta}^T(\nabla^2 C_1 - LI)\boldsymbol{\theta} &\leq 0 \qquad \forall \boldsymbol{\theta} \\ (\boldsymbol{\theta} - \boldsymbol{a})^T(\nabla^2 C_1 - LI)(\boldsymbol{\theta} - \boldsymbol{a}) &\leq 0 \\ (\boldsymbol{\theta} - \boldsymbol{a})^T \nabla^2 C_1 (\boldsymbol{\theta} - \boldsymbol{a}) &\leq L\|\boldsymbol{\theta} - \boldsymbol{a}\|_2^2. \end{aligned} \tag{A.4}$$

Also C_1 can be exactly represented by its quadratic Taylor expansion:

$$C_1(\boldsymbol{\theta}) = C_1(\boldsymbol{a}) + \nabla C_1(\boldsymbol{a})^T(\boldsymbol{\theta} - \boldsymbol{a}) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{a})^T \nabla^2 C_1(\boldsymbol{a})(\boldsymbol{\theta} - \boldsymbol{a}) \tag{A.5}$$

Combining (A.4) and (A.5) yields

$$C_1(\boldsymbol{\theta}) \leq C_1(\boldsymbol{a}) + \nabla C_1(\boldsymbol{a})^T(\boldsymbol{\theta} - \boldsymbol{a}) + \frac{L}{2}\|\boldsymbol{\theta} - \boldsymbol{a}\|_2^2,$$

and finally adding $C_2(\boldsymbol{\theta})$ to both sides gives the desired inequality:

$$\begin{aligned} C(\boldsymbol{\theta}) &\leq C_1(\boldsymbol{a}) + \nabla C_1(\boldsymbol{a})^T(\boldsymbol{\theta} - \boldsymbol{a}) + \frac{L}{2}\|\boldsymbol{\theta} - \boldsymbol{a}\|_2^2 + C_2(\boldsymbol{\theta}) \\ &\leq C_{prox}(\boldsymbol{\theta} \mid \boldsymbol{a}) \quad \text{if} \quad t \leq \frac{1}{L}. \end{aligned} \tag{A.6}$$

Therefore, C_{prox} is a maximiser of C if $t \leq \frac{1}{L}$. □

A.3. Proof of the proximal gradient descent error bound, as given in (2.18)

Let $C(\boldsymbol{\theta}) = C_1(\boldsymbol{\theta}) + C_2(\boldsymbol{\theta})$. We claim that if ∇C_1 is Lipschitz continuous with constant L, C_2 is convex and $t \leq \dfrac{1}{L}$ then the sequence

$$\boldsymbol{\theta}^{(k+1)} = arg\min_{\boldsymbol{\theta}} \left\{ C_1\left(\boldsymbol{\theta}^{(k)}\right) + \nabla C_1\left(\boldsymbol{\theta}^{(k)}\right)^T\left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(k)}\right) + \frac{1}{2t}\left\|\boldsymbol{\theta} - \boldsymbol{\theta}^{(k)}\right\|_2^2 + C_2(\boldsymbol{\theta}) \right\} \tag{A.7}$$

satisfies

$$C\left(\boldsymbol{\theta}^{(k)}\right) - C\left(\boldsymbol{\theta}^*\right) \leq \frac{\left\|\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}^*\right\|_2^2}{2tk}, \tag{A.8}$$

where $\boldsymbol{\theta}^*$ is the minimum of C.

Proof. From the definition of convexity, any convex differentiable function's first order approximation is a global underestimater:

$$\begin{aligned} & f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x}) \qquad \forall \boldsymbol{y} \\ \Leftrightarrow & f(\boldsymbol{x}) \leq f(\boldsymbol{y}) + \nabla f(\boldsymbol{x})^T(\boldsymbol{x} - \boldsymbol{y}) \qquad \forall \boldsymbol{y}. \end{aligned}$$

Therefore

$$C_1\left(\boldsymbol{\theta}^{(k)}\right) \leq C_1\left(\boldsymbol{\theta}\right) + \nabla C_1\left(\boldsymbol{\theta}^{(k)}\right)^T\left(\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}\right) \quad \forall \boldsymbol{\theta} \tag{A.9}$$

We would like a similar expression for the not necessarily differentiable C_2. Recall,

$$\begin{aligned}\boldsymbol{\theta}^{(k+1)} &= arg\min_{\boldsymbol{\theta}}\left\{C_1\left(\boldsymbol{\theta}^{(\boldsymbol{k})}\right) + \nabla C_1\left(\boldsymbol{\theta}^{(\boldsymbol{k})}\right)^T\left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(\boldsymbol{k})}\right) + \frac{1}{2t}\left\|\boldsymbol{\theta} - \boldsymbol{\theta}^{(\boldsymbol{k})}\right\|_2^2 + C_2(\boldsymbol{\theta})\right\}\\ &= arg\min_{\boldsymbol{\theta}}\left\{2t\nabla C_1\left(\boldsymbol{\theta}^{(\boldsymbol{k})}\right)^T\boldsymbol{\theta} + \boldsymbol{\theta}^T\boldsymbol{\theta} - 2\boldsymbol{\theta}^T\boldsymbol{\theta}^{(\boldsymbol{k})} + 2tC_2(\boldsymbol{\theta})\right\}.\end{aligned}$$

Therefore there exists a subgradient, with respect to $\boldsymbol{\theta}$, at $\boldsymbol{\theta}^{(k+1)}$ which is zero:

$$\mathbf{0} \in 2t\nabla C_1\left(\boldsymbol{\theta}^{(\boldsymbol{k})}\right) + 2\boldsymbol{\theta}^{(\boldsymbol{k+1})} - 2\boldsymbol{\theta}^{(\boldsymbol{k})} + 2t\partial C_2\left(\boldsymbol{\theta}^{(\boldsymbol{k+1})}\right).$$

Rearranging gives a valid subgradient at the point $\boldsymbol{\theta}^{(\boldsymbol{k+1})}$:

$$\frac{\boldsymbol{\theta}^{(\boldsymbol{k})} - \boldsymbol{\theta}^{(\boldsymbol{k+1})}}{t} - \nabla C_1\left(\boldsymbol{\theta}^{(\boldsymbol{k})}\right) \in \partial C_2\left(\boldsymbol{\theta}^{(\boldsymbol{k+1})}\right).$$

The definition of a subgradient states that a vector $\boldsymbol{g}$ is a subgradient of a function f at the point $\boldsymbol{x}$, i.e. $\boldsymbol{g} \in \partial f(x)$, if

$$\begin{aligned} f(\boldsymbol{y}) &\geq f(\boldsymbol{x}) + \boldsymbol{g}^T(\boldsymbol{y} - \boldsymbol{x}) \qquad \forall \boldsymbol{y}\\ \Leftrightarrow f(\boldsymbol{x}) &\leq f(\boldsymbol{y}) + \boldsymbol{g}^T(\boldsymbol{x} - \boldsymbol{y}) \qquad \forall \boldsymbol{y}.\end{aligned}$$

Therefore

$$C_2\left(\boldsymbol{\theta}^{(\boldsymbol{k+1})}\right) \leq C_2(\boldsymbol{\theta}) + \left(\frac{\boldsymbol{\theta}^{(\boldsymbol{k})} - \boldsymbol{\theta}^{(\boldsymbol{k+1})}}{t} - \nabla C_1\left(\boldsymbol{\theta}^{(\boldsymbol{k})}\right)\right)^T\left(\boldsymbol{\theta}^{(\boldsymbol{k+1})} - \boldsymbol{\theta}\right) \qquad \forall \boldsymbol{\theta}. \tag{A.10}$$

Equation (A.6), from the previous proof, can be written as

$$C\left(\boldsymbol{\theta}^{(\boldsymbol{k+1})}\right) \leq C_1\left(\boldsymbol{\theta}^{(\boldsymbol{k})}\right) + \nabla C_1\left(\boldsymbol{\theta}^{(\boldsymbol{k})}\right)^T\left(\boldsymbol{\theta}^{(\boldsymbol{k+1})} - \boldsymbol{\theta}^{(\boldsymbol{k})}\right) + \frac{L}{2}\left\|\boldsymbol{\theta}^{(\boldsymbol{k+1})} - \boldsymbol{\theta}^{(\boldsymbol{k})}\right\|_2^2 + C_2\left(\boldsymbol{\theta}^{(\boldsymbol{k+1})}\right). \tag{A.11}$$

Inserting (A.9) and (A.10) into the right hand side of (A.11), yields

$$\begin{aligned} C\left(\boldsymbol{\theta}^{(\boldsymbol{k+1})}\right) \leq& C_1\left(\boldsymbol{\theta}\right) + \nabla C_1\left(\boldsymbol{\theta}^{(\boldsymbol{k})}\right)^T\left(\boldsymbol{\theta}^{(\boldsymbol{k})} - \boldsymbol{\theta}\right) + \nabla C_1\left(\boldsymbol{\theta}^{(\boldsymbol{k})}\right)^T(\boldsymbol{\theta}^{(\boldsymbol{k+1})} - \boldsymbol{\theta}^{(\boldsymbol{k})}) + \frac{L}{2}\left\|\boldsymbol{\theta}^{(\boldsymbol{k+1})} - \boldsymbol{\theta}^{(\boldsymbol{k})}\right\|_2^2\\ &+ C_2(\boldsymbol{\theta}) + \left(\frac{\boldsymbol{\theta}^{(\boldsymbol{k})} - \boldsymbol{\theta}^{(\boldsymbol{k+1})}}{t} - \nabla C_1\left(\boldsymbol{\theta}^{(\boldsymbol{k})}\right)\right)^T\left(\boldsymbol{\theta}^{(\boldsymbol{k+1})} - \boldsymbol{\theta}\right) \qquad \forall \boldsymbol{\theta}\\ =& C\left(\boldsymbol{\theta}\right) + \frac{L}{2}\left\|\boldsymbol{\theta}^{(\boldsymbol{k+1})} - \boldsymbol{\theta}^{(\boldsymbol{k})}\right\|_2^2 + \left(\frac{\boldsymbol{\theta}^{(\boldsymbol{k})} - \boldsymbol{\theta}^{(\boldsymbol{k+1})}}{t}\right)^T\left(\boldsymbol{\theta}^{(\boldsymbol{k+1})} - \boldsymbol{\theta}\right) \qquad \forall \boldsymbol{\theta}.\end{aligned}$$

The requirement $t \le \frac{1}{L}$ implies that $\frac{L}{2} \le \frac{1}{2t}$, therefore

$$\begin{aligned}
C\left(\boldsymbol{\theta}^{(k+1)}\right) - C(\boldsymbol{\theta}) \le& \frac{1}{2t}\left(\left\|\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}\right\|_2^2 + 2\left(\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^{(k+1)}\right)^T\left(\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}\right)\right) \qquad \forall\boldsymbol{\theta} \\
=& \frac{1}{2t}\left(\boldsymbol{\theta}^{(k+1)^T}\boldsymbol{\theta}^{(k+1)} - 2\boldsymbol{\theta}^{(k+1)^T}\boldsymbol{\theta}^{(k)} + \boldsymbol{\theta}^{(k)^T}\boldsymbol{\theta}^{(k)} + 2\boldsymbol{\theta}^{(k)^T}\boldsymbol{\theta}^{(k+1)}\right. \\
& \left. -2\boldsymbol{\theta}^{(k)^T}\boldsymbol{\theta} - 2\boldsymbol{\theta}^{(k+1)^T}\boldsymbol{\theta}^{(k+1)} + 2\boldsymbol{\theta}^{(k+1)^T}\boldsymbol{\theta}\right) \qquad \forall\boldsymbol{\theta} \\
=& \frac{1}{2t}\left(\boldsymbol{\theta}^{(k)^T}\boldsymbol{\theta}^{(k)} - 2\boldsymbol{\theta}^{(k)^T}\boldsymbol{\theta} - \boldsymbol{\theta}^{(k+1)^T}\boldsymbol{\theta}^{(k+1)} + 2\boldsymbol{\theta}^{(k+1)^T}\boldsymbol{\theta}\right) \qquad \forall\boldsymbol{\theta} \\
=& \frac{1}{2t}\left(\left\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}\right\|_2^2 - \left\|\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}\right\|_2^2\right) \qquad \forall\boldsymbol{\theta}.
\end{aligned}$$

Let $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ so that

$$C\left(\boldsymbol{\theta}^{(k+1)}\right) - C(\boldsymbol{\theta}^*) \le \frac{1}{2t}\left(\left\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^*\right\|_2^2 - \left\|\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^*\right\|_2^2\right).$$

Then, summing both sides from the 0-th iteration up to the k-th yields

$$\sum_{i=0}^{k-1}\left\{C\left(\boldsymbol{\theta}^{(i+1)}\right) - C(\boldsymbol{\theta}^*)\right\} \le \frac{1}{2t}\sum_{i=0}^{k-1}\left\{\left\|\boldsymbol{\theta}^{(i)} - \boldsymbol{\theta}^*\right\|_2^2 - \left\|\boldsymbol{\theta}^{(i+1)} - \boldsymbol{\theta}^*\right\|_2^2\right\}. \tag{A.12}$$

From the previous proof we know the sequence is monotonically decreasing. Therefore, the last difference in the sum on the left hand side of (A.12) is the smallest, so

$$\sum_{i=0}^{k-1}\left\{C\left(\boldsymbol{\theta}^{(i+1)}\right) - C(\boldsymbol{\theta}^*)\right\} \ge k\left(C\left(\boldsymbol{\theta}^{(k)}\right) - C(\boldsymbol{\theta}^*)\right). \tag{A.13}$$

Also most of the terms in the sum on the right hand side of (A.12) cancel:

$$\begin{aligned}
\sum_{i=0}^{k-1}\left\{\left\|\boldsymbol{\theta}^{(i)} - \boldsymbol{\theta}^*\right\|_2^2 - \left\|\boldsymbol{\theta}^{(i+1)} - \boldsymbol{\theta}^*\right\|_2^2\right\} &= \left\|\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}^*\right\|_2^2 - \left\|\boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^*\right\|_2^2 \\
&\le \left\|\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}^*\right\|_2^2.
\end{aligned} \tag{A.14}$$

Combining (A.12), (A.13) and (A.14) completes the proof:

$$C\left(\boldsymbol{\theta}^{(k)}\right) - C(\boldsymbol{\theta}^*) \le \frac{\left\|\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}^*\right\|_2^2}{2tk}.$$

□

Bibliography

[1] M. Aharon, M. Elad, and A. Bruckstein, “K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation,” *Signal Processing, IEEE Transactions on*, vol. 54, no. 11, pp. 4311–4322, 2006.

[2] B. Alpert, “A class of bases in L^2 for the sparse representation of integral operators,” *SIAM Journal on Mathematical Analysis*, vol. 24, no. 1, pp. 246–262, Jan. 1993.

[3] L. Baboulaz and P. L. Dragotti, “Exact feature extraction using finite rate of innovation principles with an application to image super-resolution,” *Image Processing, IEEE Transactions on*, vol. 18, no. 2, pp. 281–298, 2009.

[4] A. Beck and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[5] Å. Björck, *Numerical Methods For Least Squares Problems.* SIAM, 1996.

[6] T. Blumensath, “Accelerated iterative hard thresholding,” *Signal Processing*, vol. 92, no. 3, pp. 752–756, 2011.

[7] T. Blumensath and M. Davies, “Iterative thresholding for sparse approximations,” *Journal of Fourier Analysis and Applications*, vol. 14, no. 5-6, pp. 629–654654–, 2008.

[8] ——, “Normalized iterative hard thresholding: guaranteed stability and performance,” *Selected Topics in Signal Processing, IEEE Journal of*, vol. 4, no. 2, pp. 298–309, 2010.

[9] A. Buades, B. Coll, and J. Morel, “A non-local algorithm for image denoising,” in *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2005, 60–65 vol. 2.

[10] E. Candes and D. Donoho, “Ridgelets: a key to higher-dimensional intermittency?,” *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 357, no. 1760, pp. 2495–2509, 1999.

[11] ——, “Curvelets: a surprisingly effective nonadaptive representation of objects with edges,” in *Curve and Surface Fitting: Saint-Malo*, University Press, 2000, pp. 0–82 651 357.

[12] ——, “New tight frames of curvelets and optimal representations of objects with piecewise C2 singularities,” *Comm. Pure Appl. Math.*, vol. 57, no. 2, pp. 219–266, 2003.

[13] E. J. Candes, L. Demanet, D. Donoho, and L. Ying, “Fast discrete curvelet transforms,” *Multiscale Modeling & Simulation*, vol. 5, no. 3, pp. 861–899, Jan. 2006.

[14] P. Chatterjee and P. Milanfar, “Patch-based near-optimal image denoising,” *Image Processing, IEEE Transactions on*, vol. 21, no. 4, pp. 1635–1649, 2012.

[15] R. Coifman and D. Donoho, “Translation-invariant de-noising,” Department of Statistics, Tech. Rep., 1995.

[16] P. Combettes and V. Wajs, “Signal recovery by proximal forward-backward splitting,” *Multiscale Modeling & Simulation*, vol. 4, no. 4, pp. 1168–1200, 2006.

[17] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, “Image denoising by sparse 3-D transform-domain collaborative filtering,” *Image Processing, IEEE Transactions on*, vol. 16, no. 8, pp. 2080–2095, 2007.

[18] ——, “Image restoration by sparse 3D transform-domain collaborative filtering,” *SPIE Electronic Imaging*, 2008.

[19] ——, “BM3D image denoising with shape-adaptive principal component analysis,” *Proc. Workshop on Signal Processing with Adaptive Sparse Structured Representations*, 2009.

[20] J. Daniel, W. Gragg, L. Kaufman, and G. Stewart, “Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization,” *Mathematics of Computation*, vol. 30, no. 136, pp. 772–795, 1976.

[21] I. Daubechies, *Ten lectures on wavelets.* Society for Industrial and Applied Mathematics, Jan. 1992.

[22] I. Daubechies, M. Defrise, and C. De Mol, “An iterative thresholding algorithm for linear inverse problems with a sparsity constraint,” *Comm. Pure Appl. Math.*, vol. 57, no. 11, pp. 1413–1457, Jan. 2004.

[23] M. Do and M. Vetterli, “The contourlet transform: an efficient directional multiresolution image representation,” *Image Processing, IEEE Transactions on*, vol. 14, no. 12, pp. 2091–2106, 2005.

[24] N. Dodgson, “Image resampling,” 1992.

[25] ——, “Quadratic interpolation for image resampling,” *Image Processing, IEEE Transactions on*, vol. 6, no. 9, pp. 1322–1326, 1997.

[26] D. Donoho, “Wedgelets: nearly minimax estimation of edges,” *Ann. Statist.*, vol. 27, no. 3, pp. 859–897, 1999.

[27] P. L. Dragotti and M. Vetterli, “Wavelet footprints: theory, algorithms, and applications,” *Signal Processing, IEEE Transactions on*, vol. 51, no. 5, pp. 1306–1323, 2003.

[28] M. Elad, “Why simple shrinkage is still relevant for redundant representations?,” *Information Theory, IEEE Transactions on*, vol. 52, no. 12, pp. 5559–5569, 2006.

[29] M. Elad and M. Aharon, “Image denoising via sparse and redundant representations over learned dictionaries,” *Image Processing, IEEE Transactions on*, vol. 15, no. 12, pp. 3736–3745, 2006.

[30] M. Elad, B. Matalon, J. Shtok, and M. Zibulevsky, “A wide-angle view at iterated shrinkage algorithms,” *Proc. SPIE 6701, Wavelets XII*, p. 670 102, Sep. 2007.

[31] M. Figueiredo, J. Bioucas-Dias, and R. Nowak, “Majorization-minimization algorithms for wavelet-based image restoration,” *Image Processing, IEEE Transactions on*, vol. 16, no. 12, pp. 2980–2991, 2007.

[32] M. Figueiredo and R. Nowak, “An EM algorithm for wavelet-based image restoration,” *Image Processing, IEEE Transactions on*, vol. 12, no. 8, pp. 906–916, 2003.

[33] ——, “A bound optimization approach to wavelet-based image deconvolution,” in *Image Processing, IEEE International Conference on*, 2005, pp. II–7825–.

[34] M. Figueiredo, R. Nowak, and S. Wright, “Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems,” *Selected Topics in Signal Processing, IEEE Journal of*, vol. 1, no. 4, pp. 586–597, 2007.

[35] A. Foi, K. Dabov, V. Katkovnik, and K. Egiazarian, “Shape-adaptive DCT for denoising and image reconstruction,” *Proceedings of SPIE*, vol. 6064, pp. 203–214, 2006.

[36] P. Gill, G. Golub, W. Murray, and M. Saunders, “Methods for modifying matrix factorizations,” *Mathematics of Computation*, vol. 28, no. 126, pp. 505–535, Jun. 2005.

[37] G. Golub and C. Van Loan, *Matrix Computations*, 3rd. The Johns Hopkins University Press, 1996.

[38] G. Gordon and R. Tibshirani, “Accelerated first-order methods,” in *10-725 Optimization*, Carnegie Mellon University, Sep. 2012.

[39] O. Guleryuz, “Weighted averaging for denoising with overcomplete dictionaries,” *Image Processing, IEEE Transactions on*, vol. 16, no. 12, pp. 3020–3034, 2007.

[40] J. Heikkilä, “Pattern matching with affine moment descriptors,” *Pattern recognition*, vol. 37, no. 9, pp. 1825–1834, 2004.

[41] D. Hunter and K. Lange, “A tutorial on MM algorithms,” *Amer. Statist.*, vol. 58, no. 1, pp. 30–37, 2004.

[42] V. Katkovnik, A. Foi, K. Egiazarian, and J. Astola, “From local kernel to nonlocal multiple-model image denoising,” *International journal of computer vision*, vol. 86, no. 1, pp. 1–32, 2010.

[43] K. Lange, D. Hunter, and I. Yang, “Optimization transfer using surrogate objective functions,” *Journal of Computational and Graphical Statistics*, vol. 9, no. 1, pp. 1–20, Jan. 2000.

[44] E. Le Pennec and S. Mallat, “Sparse geometric image representations with bandelets,” *Image Processing, IEEE Transactions on*, vol. 14, no. 4, pp. 423–438, 2005.

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

[46] S. Lloyd, “Least squares quantization in PCM,” *Information Theory, IEEE Transactions on*, vol. 28, no. 2, pp. 129–137, 1982.

[47] 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*, 2009, pp. 2272–2279.

[48] S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, 3rd ed. Academic Press, Jan. 2008.

[49] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 11, no. 7, pp. 674–693, 1989.

[50] ——, "Multiresolution approximations and wavelet orthonormal bases of L2(R)," *Transactions of the American Mathematical Society*, vol. 315, no. 1, pp. 69–87, Sep. 1989.

[51] R. Neelamani, H. Choi, and R. Baraniuk, "ForWaRD: Fourier-wavelet regularized deconvolution for ill-conditioned systems," *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, vol. 52, no. 2, pp. 418–433, 2004.

[52] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k2)$," *Soviet Mathematics Doklady*, 1983.

[53] ——, "Gradient methods for minimizing composite objective function," 2007.

[54] G. Peyré and S. Mallat, "Orthogonal bandelet bases for geometric images approximation," *Comm. Pure Appl. Math.*, vol. 61, no. 9, pp. 1173–1212, 2008.

[55] J. Portilla and L. Mancera, "L0-based sparse approximation: two alternative methods and some applications," in *Proceedings of the SPIE*, San Diego, CA, USA, Aug. 2007, pp. 6701–6772.

[56] D. Scharstein and C. Pal, "Learning conditional random fields for stereo," in *Computer Vision and Pattern Recognition, IEEE Conference on*, 2007, pp. 1–8.

[57] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on IS* -, 2003, I–195I–202 vol.1–.

[58] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International journal of computer vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[59] A. Scholefield and P. L. Dragotti, *Quadtree structured restoration algorithms for piecewise polynomial images*, at Inspire Workshop on Sparsity and its application to large inverse problems, Cambridge, Dec. 2008.

[60] ——, "Image restoration using a sparse quadtree decomposition representation," in *Image Processing, IEEE International Conference on*, 2009, pp. 1473–1476.

[61] ——, "Quadtree structured restoration algorithms for piecewise polynomial images," in *Acoustics, Speech and Signal Processing, IEEE International Conference on*, IEEE, 2009, pp. 705–708.

[62] ——, "Quadtree structured image approximation for denoising and interpolation," *submitted to IEEE Transactions on Image Processing*, May 2013.

[63] R. Shukla, P. L. Dragotti, M. Do, and M. Vetterli, "Rate-distortion optimized tree-structured compression algorithms for piecewise polynomial images," *Image Processing, IEEE Transactions on*, vol. 14, no. 3, pp. 343–359, 2005.

[64] J. Sprinzak and M. Werman, "Affine point matching," *Pattern Recognition Letters*, vol. 15, no. 4, pp. 337–339,

[65] J. L. Starck, E. Candes, and D. Donoho, "The curvelet transform for image denoising," *Image Processing, IEEE Transactions on*, vol. 11, no. 6, pp. 670–684, 2002.

[66] J. L. Starck, F. Murtagh, and J. Fadili, *Sparse Image and Signal Processing*, ser. Wavelets, Curvelets, Morphological Diversity. Cambridge University Press, May 2010.

[67] G. Strang and T. Nguyen, *Wavelets and filter banks.* Wellesley-Cambridge Press, 1997.

[68] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *Image Processing, IEEE Transactions on*, vol. 16, no. 2, pp. 349–366, 2007.

[69] J. Tropp, "Greed is good: algorithmic results for sparse approximation," *Information Theory, IEEE Transactions on*, vol. 50, no. 10, pp. 2231–2242, 2004.

[70] V. Velisavljevic, B. Beferull-Lozano, M. Vetterli, and P. L. Dragotti, "Directionlets: anisotropic multidirectional representation with separable filtering," *Image Processing, IEEE Transactions on*, vol. 15, no. 7, pp. 1916–1933, 2006.

[71] M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding.* Prentice Hall Signal Processing Series, Jan. 1995.

[72] M. Vetterli, P. Marziliano, and T. Blu, "Sampling signals with finite rate of innovation," *Signal Processing, IEEE Transactions on*, vol. 50, no. 6, pp. 1417–1428, 2002.

[73] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *Image Processing, IEEE Transactions on*, vol. 13, no. 4, pp. 600–612, 2004.