University of Alberta

NUMERICAL ALGORITHMS FOR DISCRETE MODELS OF IMAGE DENOISING

by

HANQING ZHAO

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

MATHEMATICS

Department of Mathematical and Statistical Sciences

FALL, 2010
Edmonton, Alberta

Examining Committee

Rong-Qing Jia, Mathematics

Bin Han, Mathematics

Michael Li, Mathematics

Mrinal Mandal, Electrical and Computer Engineering

Charles Chui, Mathematics, University of Missouri - St. Louis

To my parents,

who always have their faith in me.

Thank you for raising me

to be the person I am today.

Abstract

In this thesis, we develop some new models and efficient algorithms for image denoising. The total variation model of Rudin, Osher, and Fatemi(ROF) for image denoising is considered to be one of the most successful deterministic denoising models. It exploits the non-smooth total variation (TV) semi-norm to preserve discontinuities and to keep the edges of smooth regions sharp. Despite its simple form, the TV semi-norm results in a strongly nonlinear Euler-Lagrange equation and poses computational challenge in solving the model efficiently. Moreover, this model produces so-called staircase effect. In this thesis, we propose several new algorithms and models to solve these problems. We study the discretized ROF model and propose a new algorithm which does not involve partial differential equations. Convergence of the algorithm is analyzed. Numerical results show that this algorithm is efficient and stable. We then introduce a denoising model which utilizes high-order difference to approximate piece-wise smooth functions. This model eliminates undesirable staircases, and improves both visual quality and signal-to-noise ratio. Our algorithm is generalized to solve the high-order models. A relaxation technique is proposed for the iteration scheme, aiming to accelerate our solution process. Finally, we propose a method combining total variation and wavelet packets to improve performance on texture-rich images. The ROF model is utilized to eliminate noise, and a wavelet packet transform is used to enhance textures. The numerical results show that the combinational method exploits the advantages of both total variation and wavelet packets.

Acknowledgements

I would like to express my gratitude to my supervisor, Professor Rong-Qing Jia, for his wise guidance and generous support throughout my graduate studies at the Department of Mathematical and Statistical Sciences, University of Alberta.

I would like to thank Professor Bin Han, Professor Feng Dai and Professor Alexander Litvak for their great help, encouragement and valuable discussions on my studies and research.

I am grateful to my friends Wei, Xiaosheng, Zhiyong, Vishaal for all the helpful discussions. I am also thankful to all of the people who work in the Department of Math. and Stat. Sciences for providing such a wonderful research environment.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 An Overview of Image Denoising

Digital images captured by digital cameras or medical devices are generally contaminated by noise. Thus, denoising and reconstructing a degraded image are an important step before we can analyze it.

In reality, the noise is determined by capturing instruments, data transmission or quantization. Though the model of noise is greatly dependent on environment, most noisy models assume that additive Gaussian noise is applied, i.e. the noisy image is formulated as

$$f = u + \epsilon \ ,$$

where u is the noise-free image, ϵ is the Gaussian white noise with standard deviation σ, and f is the observed noisy image which is to be processed.

Most existing denoising methods assume that the noise level σ is known. In the following discussion, we always assume that σ is given.

There are two basic approaches to image denoising, spatial domain methods and transform(frequency) domain methods. The main difference between these two categories is that a transform domain method decomposes the image by a chosen basis before further processing while a spatial domain method processes the observed image data directly.

Transform domain methods have developed rapidly since Donoho's soft thresholding technique [16] was introduced in 1995. The noise is considered high-frequency component in the transform domain for both FFT and DWT and hence thresholding or truncating eliminates noise. The advantage of transform domain methods is that images often have sparse representations in transform domain. Thus dealing with the transform domain is very efficient. However, thresholding also weakens the texture of the image, which is also contained in high-frequency component. Moreover, in transform domain the geometric features of the image is often lost. Since the visual quality of an image is highly relevant to the geometric features, especially edges of objects, sharp edge is a critical criterion for judging the performance of a denoising model. Some frequency-based methods do not work well on preserving edges of objects in an image. They usually cause broken or blurred edges and make the denoised image look less pleasant. To remedy this, some adaptive thresholding (see [11]) and statistical models in transform domain were built to improve visual quality of denoised image and some of which were quite successful in generating high-quality images.

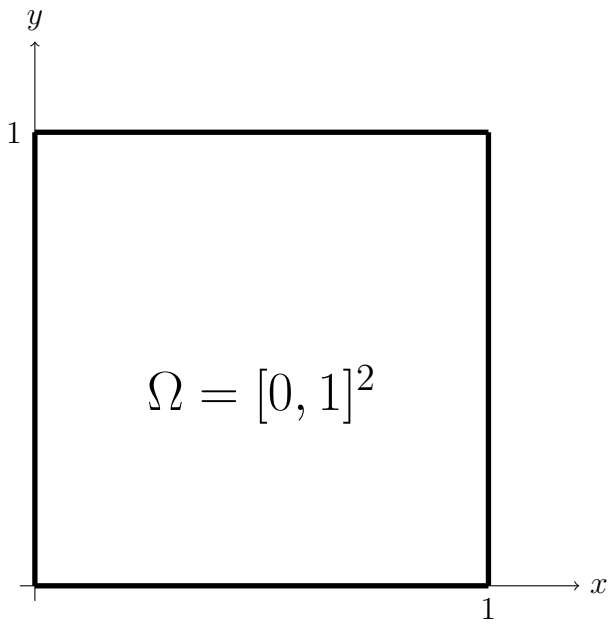
On the other hand, the spatial domain methods focus on the image itself. The PDE-based methods which use differential equations to describe both the noise model and denoising process, and methods using filters are easy to imple-

ment and are usually very efficient.(see [9, 12]) These methods take advantage of existing techniques in numerical PDEs, approximation and optimization, and have attracted more and more attention in recent years. In 1992, Rudin, Osher and Fatemi proposed the celebrated the ROF model in [30], which was extremely simple and of good performance, yet was hard to solve. Our main results are based on the ROF model and its descendants.

1.2 Definitions and Notation

We will discuss image processing in both continuous and discrete contexts. Therefore, it is important to clarify the notations to avoid ambiguity. We will define the notations and the mathematical model for denoising for the continuous case. The discretization will be introduced in the next chapter.

It is convenient to describe an image as a continuous function for theoretical analysis. In most cases, an image, either a wall picture or a digital photo, is fitted in a rectangular region. For simplicity, in this thesis we assume that an image is square-shaped, but all the conclusions and algorithms can be naturally extended to adapt to the general rectangular shape. We also assume that the length of each side of a square-shaped image is 1 unit.
Choosing the lower left corner of the square to be the origin, we construct a coordinate system in the following way:



We further simplify the mathematical model by considering only grayscale images. To avoid becoming lost in various color channels, we assume that the color at any point of an image can be characterized simply by one real number which represents the darkness of that point, ranging from 0 to 255. Usually 0 represents the black and 255 represents the white. Hence, an image can be characterized as a function $u : \Omega \to \mathbb{R}$. Since an image is 2-dimensional, we do not consider the higher-order space $\mathbb{R}^n$ with $n > 2$ in this thesis.

We will denote by u the image through out this thesis. We also denote by $f = u + \varepsilon$ the noisy image, or observed image, in both continuous and discrete cases. ε is the noise we want to remove. In the continuous case, we usually choose $u, f \in L_2(\Omega)$.

We clarify some notations here. Some of them will be explained later in details.

- $\Omega = [0,1]^2$ is the domain.
- $\|\cdot\|_1$ and $\|\cdot\|_2$ are the normal L_1 and L_2 norms on $L_1(\Omega)$ and $L_2(\Omega)$

respectively.

- $W^{k,p}$ is the Sobolev space equipped with the norm

$$\|f\|_{k,p} = \Big(\sum_{i=0}^{k} \|f^{(i)}\|_p^p \Big)^{1/p}.$$

  When $k = p = 1$, $W^{1,1}$ is the space of absolutely continuous functions.

- $f : \Omega \to \mathbb{R}$ is the observed image contaminated with noise.

- $u : \Omega \to \mathbb{R}$ is the clean image or the processed result.

- $\varepsilon : \Omega \to \mathbb{R}$ is the noise, usually considered the Gaussian white noise.

- $\sigma > 0$ is the standard deviation of the noise ε.

- $|\cdot|$ is either the absolute value of an real number or the Euclidean norm on $\mathbb{R}^2$, i.e.

$$\begin{cases} |a| \text{ is the absolute value of } a \in \mathbb{R} \\ |(a,b)| = \sqrt{a^2 + b^2},\ b \in \mathbb{R} \end{cases}$$

- ∇ is the gradient operator : $\nabla u = (u_x, u_y)$

- *div* is the divergence operator which is also written as $(\nabla \cdot)$.

$$div(F) = \nabla \cdot F = div(F_1, F_2) = \frac{\partial F_1}{\partial x} + \frac{\partial F_2}{\partial y} \text{ for } F = (F_1, F_2) : \mathbb{R}^2 \to \mathbb{R}^2$$

With these notations, we are in a position to introduce the *total variation* of a function u:

$$TV(u) := \int_\Omega |\nabla u| = \int_\Omega \sqrt{u_x^2 + u_y^2} \tag{1.1}$$

which is more formally defined as

$$TV(u) := \sup \Big\{ \int_\Omega u \; div(w) \;\Big|\; w = (w_1, w_2) \in C_c^1(\Omega, \mathbb{R}^2), |w| \leq 1 \Big\} . \qquad (1.2)$$

The latter one is more general since it does not require u to be differentiable.

The following minimization problem is a main topic in this thesis:

$$u^* = \arg\min_u \int_\Omega |\nabla u| \quad \text{subject to } \|u - f\|^2 = \sigma^2 . \qquad (1.3)$$

which is called the *the ROF model*. To make it well-defined, we further assume that $TV(u) < \infty$, i.e. u is in a *bounded variation space* which is defined as

$$BV(\Omega) := \big\{ u \in L_1(\Omega) \;\big|\; TV(u) < \infty \big\} .$$

It is known that $BV(\Omega)$ is a Banach space with the norm $\|u\|_{BV} = TV(u) + \|u\|_1$. Moreover, it can be shown that $W^{1,1}(\Omega) \subseteq BV(\Omega) \subseteq L_1(\Omega)$ and the definitions (1.1) and (1.2) are equivalent when $u \in W^{1,1}(\Omega)$. Therefore, we will use $W^{1,1}(\Omega)$ to approximate $BV(\Omega)$. The problem (1.3) becomes

$$u^* = \arg\min_{u \in W^{1,1}(\Omega) \cap L_2(\Omega)} \int_\Omega |\nabla u| \quad \text{subject to } \|u - f\|^2 = \sigma^2 . \qquad (1.4)$$

We will always assume $u \in W^{1,1}(\Omega) \cap L_2(\Omega)$ without further instruction. We refer interested readers to [1], [6] for more information about total variation and BV space.

1.3 Image Denoising based on Total Variation

Suppose u and f are defined on $\Omega = [0,1]^2 \subset \mathbb{R}^2$, satisfying $f = u + \varepsilon$, where ε is the error with standard deviation σ. Let $\nabla u := (u_x, u_y)$ be the gradient operator and $|\cdot|$ be the Euclidean norm on Ω. We have $|\nabla u| = \sqrt{u_x^2 + u_y^2}$. The Total Variation of u is defined by

$$TV(u) = \int_\Omega |\nabla u| = \int_\Omega \sqrt{u_x^2 + u_y^2}\,.$$

In 1992, Rudin, Osher and Fatemi proposed the ROF model in [30]. This model became a popular approach to image denoising very soon. And later, the model was rapidly modified and applied to other topics in image processing such as deconvolution and inpainting.

The ROF model is the following constrained minimization problem

$$u^* = \arg\min_u \int_\Omega |\nabla u| \quad \text{subject to } \|u - f\|^2 = \sigma^2\,, \tag{1.5}$$

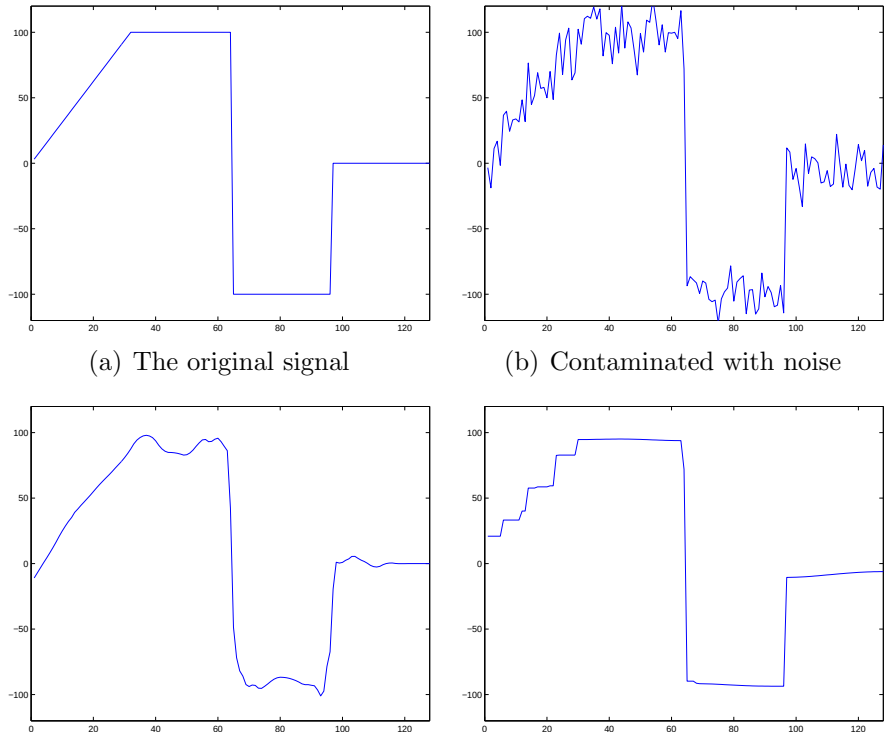
or its unconstrained variation

$$u^* = \arg\min_u \left\{ \int_\Omega |\nabla u| \; + \frac{\mu}{2}\|u - f\|_2^2 \right\}, \tag{1.6}$$

where μ is some properly chosen penalty parameter. The penalty approach (1.6) is standard in the inverse problems community, and is commonly referred to as Tikhonov regularization. In this thesis, (1.6) will be discussed extensively. Since the term $\int_\Omega |\nabla u|$ represents the total variation of u, models involving this term are called total variation based models, or simply TV-based models.

1.4 The Staircase Effect and Improved Variational Models

The ROF model has some favorable properties compared with the existing deterministic denoising models. Here is a comparison of the denoising results between wavelet thresholding methods and the ROF model (1.6).



(a) The original signal

(b) Contaminated with noise

(c) Denoised by the wavelet thresholding

(d) Denoised by the ROF model

Figure 1.1: Comparison on denoising effects between the wavelet shrinkage and the ROF model.

It is clear that the ROF model preserves the discontinuities, or jumps, while wavelet shrinkage causes Gibb's effect and produces artifacts. However, the ROF model does not always work as it is supposed to. With presence of heavy noise, the ROF model often misinterprets some noise as jumps or edges

and produces false edges. Thus it results in piecewise constant which is clearly seen in Figure 1.1(d). This phenomenon is known as "staircase artifact" and is analyzed in [26, 29]. In a two-dimensional case, it produces undesirable blocky images. In spite of this shortcoming, the ROF model remains one of the most successful deterministic denoising models and is still a popular area of research.

To improve visual quality, several methods were proposed to find a trade-off between smoothness and sharpness by modifying the TV term. In 1997, Blomgren, Mulet, Chan and Wong [2] proposed the modified model:

$$u^* = \arg\min_u \left\{ \int_\Omega |\nabla u|^{Q(|\nabla u|)} + \frac{\mu}{2}\|u-f\|_2^2 \right\}. \tag{1.7}$$

where $Q : \mathbb{R} \to \mathbb{R}$ decreases monotonically from $Q(0) = 2$ to $Q(\infty) = 1$, which is more convex when the gradient is small and behaves like the standard ROF model near discontinuities. Hence the goals of reducing staircase and preserving edges are achieved simultaneously.

A better approach was proposed by Lysaker, Lundervold and Tai [22, 23] by replacing the gradient in TV term with a higher-order differential. Their model, known as the LLT model, was

$$\begin{aligned} u^* &= \arg\min_u \left\{ \int_\Omega |\nabla^2 u| + \frac{\mu}{2}\|u-f\|_2^2 \right\} \\ &= \arg\min_u \left\{ \int_\Omega \sqrt{|u_{xx}|^2 + |u_{xy}|^2 + |u_{yx}|^2 + |u_{yy}|^2} + \frac{\mu}{2}\|u-f\|_2^2 \right\}. \end{aligned} \tag{1.8}$$

They also proposed the anisotropic version

$$u^* = \arg\min_u \left\{ \int_\Omega (|u_{xx}| + |u_{yy}|) \ + \frac{\mu}{2}\|u - f\|_2^2 \right\} . \tag{1.9}$$

It is known that high-order PDEs can recover smoother surfaces. Thus this model performs better than the ROF model in the smooth region of an image. Chang, Tai and Xing adopted this idea and proposed the combination model in [7] as follows:

$$u^* = \arg\min_u \ \left\{ (1-g) \int_\Omega |\nabla^2 u| + g \int_\Omega |\nabla u| \ + \frac{\mu}{2}\|u - f\|_2^2 \right\} . \tag{1.10}$$

where g is a properly-chosen weighting function. The combination model was proved to perform well on both edges and smooth regions. However, the existing numerical algorithms for solving those models rely on gradient-descent methods and Euler-Lagrange equations, and are inefficient and time-consuming.

We will propose a fast algorithm solving a model close to (1.10). Instead of replacing ∇u by $\nabla^2 u$ which was done in (1.8), we keep both ∇u and $\nabla^2 u$. Recall the ROF model

$$u^* = \arg\min_u \ \int_\Omega |\nabla u| \ + \frac{\mu}{2}\|u - f\|_2^2 .$$

which can be written as

$$u^* = \arg\min_u \ \frac{1}{\mu} \int_\Omega |\nabla u| \ + \frac{1}{2}\|u - f\|_2^2 .$$

By adding an additional term, we have the model

$$u^* = \arg\min_u \frac{1}{\mu}\int_\Omega |\nabla u| + \frac{1}{\nu}\int_{u\in\Omega} |\nabla^2 u| + \frac{1}{2}\|u-f\|_2^2 . \tag{1.11}$$

where $|\nabla^2 u| = \sqrt{u_{xx}^2 + u_{xy}^2 + u_{yx}^2 + u_{yy}^2}$, following the notation in [23]. We point out that equation (1.11) is not the original LLT model, since LLT model does not contain the first term.

In Chapter 3, we will propose an algorithm to solve this problem.

1.5 The ROF Model and Partial Differential Equations

In the past few decades, the PDE based image denoising models became an active research area. This is because PDE approaches take the advantages of effective treatments from PDE theory and produces high accuracy and stable computation. In 1990, Perona and Malik proposed an anisotropic-diffusion PDE model for image processing [28], which could be deemed as an interpretation of the ROF model from the point of view of diffusion.

The model is as follows:

$$\begin{cases} \dfrac{\partial u}{\partial t} = \nabla \cdot (\, c(|\nabla u|)\, \nabla u), \quad (x,y) \in \Omega, \\ u(x,y,0) = u_0(x,y) \\ \dfrac{\partial u}{\partial \vec{n}} = 0, \quad (x,y) \in \partial\Omega \end{cases} \tag{1.12}$$

where ∇ is the gradient operator, $\nabla\cdot$ is the divergence operator, $c(\lambda)$ is a decreasing function with $\lim_{\lambda\to\infty} c(\lambda) = 0$, $\partial\Omega$ is the boundary of Ω and $\vec{n}$ is

the normal direction of $\partial\Omega$. Let

$$\begin{cases} \tau = \left(\dfrac{u_x}{|\nabla u|}, \dfrac{u_y}{|\nabla u|} \right) = \dfrac{\nabla u}{|\nabla u|} \\ \gamma = \left(-\dfrac{u_y}{|\nabla u|}, \dfrac{u_x}{|\nabla u|} \right) \end{cases}$$

be the normal and tangent unit vectors of the level sets of u. Then the equation (1.12) can be formulated as

$$\frac{\partial u}{\partial t} = \left[\, c(\lambda) + \lambda \, c'(\lambda) \, \right] \frac{\partial^2 u}{\partial \tau^2} + c(\lambda) \frac{\partial^2 u}{\partial \gamma^2}, \quad \text{where } \lambda = |\nabla u|.$$

It is easy to see that we can choose function $c(\lambda)$ wisely such that $c(\lambda) + \lambda \, c'(\lambda)$ is small or negative, therefore making u diffuse less or diffuse backwards perpendicular to edges. Hence the edges do not suffer from blurry effects. As the time t develops, u approaches the clean image with the preserved, or even sharper edges.

Let us concentrate on a special case when $c(\lambda) = 1/\lambda$. The model (1.12) becomes

$$\frac{\partial u}{\partial t} = \nabla \cdot \left(\frac{\nabla u}{|\nabla u|} \right), \tag{1.13}$$

which is the steepest decent method (1.15) on page 13 for solving the ROF model (1.5).

1.6 Early-Stage Solutions to the ROF Model

Despite its simple form, the solution of these minimization problems suffers from serious non-linearity and non-differentiability introduced by the TV term $\int_\Omega |\nabla u|$. During a long period of time, the traditional way of solution is to

solve the Euler-Lagrange equation of an differentiable approximation of 1.5:

$$u^* = \arg\min_u \int_\Omega \sqrt{\nabla u + \beta} \ + \frac{\mu}{2}\|u - f\|_2^2 \text{ . for some } \beta > 0. \tag{1.14}$$

In their original paper [30], Rudin, et al. introduced artificial time marching and solved the following equation with homogeneous Neumann boundary condition

$$\frac{\partial u}{\partial t} = \nabla \cdot \left(\frac{\nabla u}{\sqrt{|\nabla u|^2 + \beta^2}} \right) \tag{1.15}$$

with the stopping criterion $\|u - f\| \leq \sigma$, or

$$\frac{\partial u}{\partial t} = -\mu(u - f) + \nabla \cdot \left(\frac{\nabla u}{\sqrt{|\nabla u|^2 + \beta^2}} \right) \tag{1.16}$$

with $t \to \infty$.

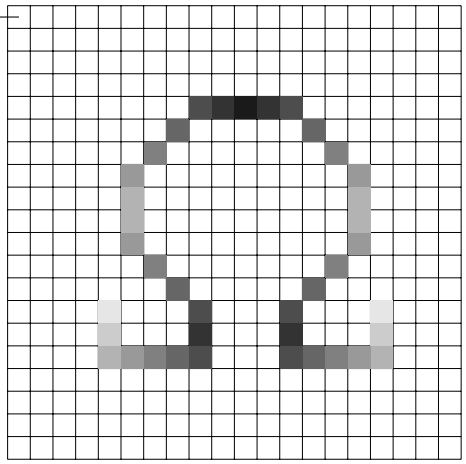
While numerical implementation is straightforward, the non-linearity and poor conditioning of the problem make the convergence very slow. Efforts had been made after the model being proposed. Vogel and Oman [32] proposed a lagged-diffusive fixed-point iteration method. For each $k = 1, 2, \ldots,$, solve the following equations iteratively:

$$\mu(u^{k+1} - f) - \nabla \cdot \left(\frac{\nabla u^{k+1}}{\sqrt{|\nabla u^k|^2 + \beta^2}} \right) = 0 \tag{1.17}$$

Each linearized equation was much more easier to solve and hence the entire calculation was made more efficient.

1.7 Discretization

When we consider discretized models, all the restrictions on u, such as bounded variation and differentiability, become irrelevant. We will exploit this advantage. And hence most of our analysis and proof will be based on discrete case. Note we still use u and f to denote the image in the discrete case. Since the context will show clearly whether it is continuous or discrete, ambiguity is not likely to occur.



A square-shaped image is discretized into an $n \times n$ grid. Each cell in the grid is called a pixel and is filled with one color. Recall that we only consider grayscale images and the color is uniquely determined by a real number. Therefore, an image is modeled as a real-valued function on the discrete domain

$$u : \{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\} \to \mathbb{R},$$

or a real-valued $n \times n$ matrix:

$$u \in \mathbb{R}^{\{1,2,\ldots,n\} \times \{1,2,\ldots,n\}}.$$

We use notation $u_{i,j}$ to represent the value of u at ith row and jth column.

The l_p norm and inner product are defined similarly to the ordinary case:

- $\|u\|_p := \left(\sum_{1\le i,j\le n} |u_{i,j}|^p\right)^{1/p}$
- $\|u\|_\infty := \max_{1\le i,j\le n} |u_{i,j}|$
- $\langle u, v\rangle = \sum_{1\le i,j\le n} u_{i,j}\ v_{i,j}$

We also have to discretize the *total variation* $TV(u) = \int_\Omega |\nabla u|$. The gradient $\nabla u = (u_x, u_y)$ is approximated by the difference operators ∇_x and ∇_y, which are defined as

$$(\nabla_x u)_{i,j} = \begin{cases} 0 & \text{if } i = 1 \\ u_{i,j} - u_{i-1,j} & \text{if } i > 1 \end{cases}$$

and

$$(\nabla_y u)_{i,j} = \begin{cases} 0 & \text{if } j = 1 \\ u_{i,j} - u_{i,j-1} & \text{if } j > 1 \end{cases}$$

It is clear that $\nabla_x u, \nabla_y u$ are the difference operators resembling u_x, u_y. With these notations, the gradient of u is $(\nabla_x u, \nabla_y u)$. Noting $|(\nabla_x u, \nabla_y u)| = \sqrt{(\nabla_x u)^2_{i,j} + (\nabla_y u)^2_{i,j}}$, the total variation $TV(u) = \int_\Omega |\nabla u|$ can be discretized as

$$\begin{aligned} TV(u) &= \sum_{1\le i,j\le n} \sqrt{(\nabla_x u)^2_{i,j} + (\nabla_y u)^2_{i,j}} \\ &= \left\| \sqrt{(\nabla_x u)^2 + (\nabla_y u)^2} \right\|_1 \end{aligned} \tag{1.18}$$

We have another discretization of the TV function. If we let $|(a, b)| =$

$|a| + |b|$ instead of $\sqrt{a^2 + b^2}$, we have

$$\begin{aligned} TV_a(u) &= \sum_{1 \le i,j \le n} |(\nabla_x u)_{i,j}| + |(\nabla_y u)_{i,j}| \\ &= \|\nabla_x u\|_1 + \|\nabla_y u\|_1 \end{aligned} \tag{1.19}$$

which is the *anisotropic* discrete total variation. Correspondingly, (1.18) is the *isotropic* discrete total variation.

We will analyze the high-order models (1.11). So we have to define the following notations. Let Δ_x and Δ_y be the difference operators defined by

$$(\Delta_x u)_{i,j} = \begin{cases} u_{1,j} - u_{2,j} & \text{if } i = 1 \\ 2u_{i,j} - u_{i-1,j} - u_{i+1,j} & \text{if } 1 < i < n \\ u_{n,j} - u_{n-1,j} & \text{if } i = n \end{cases}$$

and

$$(\Delta_y u)_{i,j} = \begin{cases} u_{i,1} - u_{i,2} & \text{if } j = 1 \\ 2u_{i,j} - u_{i,j-1} - u_{i,j+1} & \text{if } 1 < j < n \\ u_{i,n} - u_{i,n-1} & \text{if } j = n \ . \end{cases}$$

It is easy to see that they are discrete approximations to u_{xx} and u_{yy}.

To make the problem more easily described in the language of matrix algebra, we stretch the $n \times n$ matrix u row-by-row into an $n^2 \times 1$ column vector:

$$u = (u_{1,1}, u_{1,2}, \ldots, u_{1,n},\ u_{2,1}, u_{2,2}, \ldots, u_{n,n})^T,$$

where T stands for the transpose of a matrix. And the index is rearranged

from 1 to n^2 like what is usually done in numerical PDE:

$$u = (u_1, u_2, \ldots, u_{n^2}) \ .$$

We also rewrite the difference operators $\nabla_x, \nabla_y : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n \times \mathbb{R}^n$ into $\nabla_x, \nabla_y : \mathbb{R}^{n^2} \to \mathbb{R}^{n^2}$ defined by

$$(\nabla_x u)_i := \begin{cases} 0 & \text{if } i \leq n \\ u_i - u_{i-n} & \text{if } i > n \end{cases} \tag{1.20}$$

$$(\nabla_y u)_i := \begin{cases} 0 & \text{if } i = 1 \bmod n \\ u_i - u_{i-1} & \text{otherwise} \end{cases} \tag{1.21}$$

Hence the ROF model (1.6) has the *isotropic* discretization

$$u^* = \arg\min_{u \in \mathbb{R}^{n^2}} \left\| \sqrt{(\nabla_x u)^2 + (\nabla_y u)^2} \right\|_1 + \frac{\mu}{2} \|u - f\|_2^2 \ . \tag{1.22}$$

And the *anisotropic* model is

$$u^* = \arg\min_{u \in \mathbb{R}^{n^2}} \|\nabla_x u\|_1 + \|\nabla_y u\|_1 + \frac{\mu}{2} \|u - f\|_2^2 \ . \tag{1.23}$$

We point out that the *isotropic* model and *anisotropic* one yield very close solutions. But it is obvious that the *anisotropic* model is much easier to code and solve.

Similarly, Δ_x, Δ_y are replaced by Δ_x and Δ_y:

$$(\Delta_x u)_i := \begin{cases} u_i - u_{i+n} & \text{if } i \le n \\ u_i - u_{i-n} & \text{if } i > n^2 - n \\ 2u_i - u_{i-n} - u_{i+n} & \text{if } n < i \le n^2 - n \end{cases} \tag{1.24}$$

$$(\Delta_y u)_i := \begin{cases} u_i - u_{i+1} & \text{if } j = 1 \bmod n \\ u_i - u_{i-1} & \text{if } j = 0 \bmod n \\ 2u_i - u_{i-1} - u_{i+1} & \text{otherwise} \end{cases} \tag{1.25}$$

Clearly, $\Delta_x = \nabla_x^T \nabla_x$ and $\Delta_y = \nabla_y^T \nabla_y$. In this thesis, we always assume that u and f are one-dimensional vectors and use $\nabla_x, \nabla_y, \Delta_x, \Delta_y$ as the difference operators.

In [7, 23], isotropic discretization was applied. However, our numerical experiment shows that isotropic and anisotropic discretizations differ very little. Therefore we use anisotropic discretization to avoid complex modelling and computation and gain fast computational speed. Hence, the discretized model is

$$u^* = \arg\min_{u \in \mathbb{R}^{n^2}} \frac{1}{\mu}\big(\|\nabla_x u\|_1 + \|\nabla_y u\|_1\big) + \frac{1}{\nu}\big(\|\Delta_x u\|_1 + \|\Delta_y u\|_1\big) + \frac{1}{2}\|u - f\|_2^2 . \tag{1.26}$$

1.8 Outlines of the Thesis

In this thesis, we try to develop some efficient algorithms to solve the total variation based image denoising models. The paper is organized as follows:

Chapter 1 gives an overview of image denoising and the mathematical background of imaging models. The ROF model is introduced. Its advantages and disadvantages are discussed. We then introduce the high-order models to overcome the disadvantages of the ROF model. At the end of this chapter, we introduce some early-stage algorithms for the ROF model, and discretize the model into isotropic and anisotrpic forms.

Chapter 2 investigates some numerical methods based on the Bregman iteration for solving the ROF model, and then proposes a fast algorithm. We start with the Bregman Iteration, and introduce Goldstein and Osher's Split Bregman algorithm for the ROF model. A fast algorithm is proposed, and its convergence is analyzed. We end this chapter by reporting the numerical results of our algorithm. In numerical experiments, a relaxation technique is proposed to accelerate the iteration.

Chapter 3 discusses solutions to a high-order variational model. Our fast algorithm is generalized and adapted to the high-order model. Its convergence is proved. The relaxation technique is extended to the high-order model to accelerate the iteration.

Chapter 4 proposes a combinational algorithm for texture-rich images. We analyze the performance of the ROF model on textures.A brief introduction of wavelets and wavelet packets is given. Then we propose the algorithm combining variational model and wavelets. The numerical experiments show that the new algorithm has good performance on texture-rich images.

Chapter 5 ends the paper with some final conclusions.

Chapter 2

Fast Algorithms for the Total Variation Model

In this chapter, we will give a comprehensive survey of solutions to the ROF model. It is interesting that some of the current hard-to-solve problems of imaging have elegant solution with some methods developed by the optimization community in 1960s and 1970s. Recently the solution to the ROF model was boosted by the idea of solving problem (1.6) directly, instead of solving its Euler-Lagrange equation. Several swift iteration schemes emerge and they are more efficient and more accurate than classical approaches. But most of them were proved to be descendants of some old and inconspicuous papers, such as [3] by Bregman, 1967.

We start with introducing the Bregman iteration. Then we analyze Goldstein and Osher's Split Bregman Method, on which our algorithm is based. Our fast algorithm will be introduced with a partial proof. The proof of a general form of our algorithm will be given in the next chapter. In the last section of this chapter, we present the numerical results of our algorithm and

the Split Bregman method. A relaxation technique is proposed to accelerate the computation.

2.1 The Bregman Iteration

The Bregman iteration was first introduced to imaging science in [27] by Osher et al. in 2005. It is based on the Bregman distance which was defined in [3] as a measure of the distance between the value of two vectors under certain energy function. These concepts are important for understanding our algorithm, so we will discuss them in details. We first introduce the concept *subgradient*.

We consider a vector space $\mathbb{R}^n$. Let E be a continuous and convex function on $\mathbb{R}^n$. Then the subgradient of E at point u_0 is a vector $p \in \mathbb{R}^n$ such that

$$E(u) - E(u_0) \geq \langle p, u - u_0 \rangle \ , \forall u \in \mathbb{R}^n.$$

The *subdifferential* $\partial E(u_0)$ is the set of subgradients of E at u_0. Clearly, p is the gradient of E if E is differentiable at u_0. Therefore, we also use $\partial E(u)$ to denote the gradient.

The *Bregman distance* associated with E between u and u_0 is

$$D_E^p(u, u_0) = E(u) - E(u_0) - \langle p, u - u_0 \rangle \ .$$

where p is a subgradient of E at u_0. If E is not differentiable at u_0, there may be multiple choice of p. Figure 2.1 shows an intuitive 1-D example of the Bregman distance.

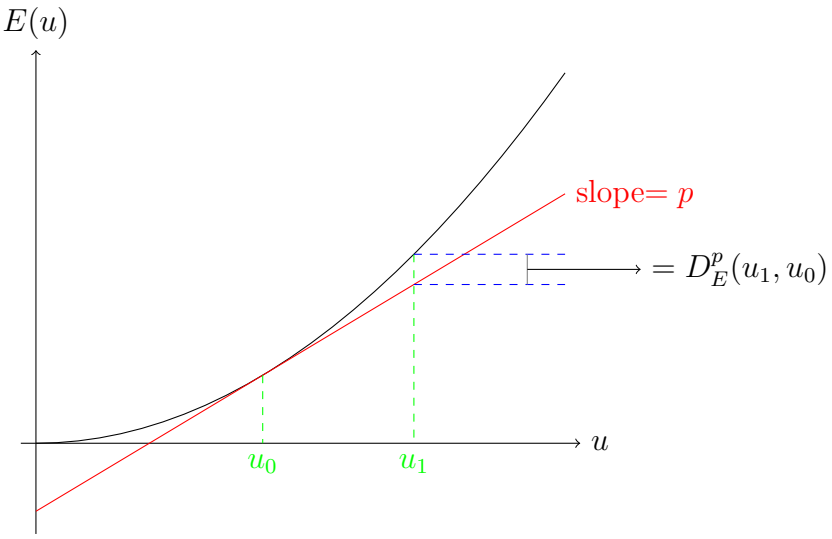


Figure 2.1: The Bregman Distance.

It is clear that $D_E^p(u, u_0) \geq 0$ as long as E is convex. Moreover, if v lies on the line segment connecting u and u_0, then $D_E^p(u, u_0) \geq D_E^p(v, u_0)$. Therefore, the Bregman distance is a measure of closeness between u and u_0. On the other hand, it is obvious that in general Bregman distance is not symmetric: $D_E^p(u, v) \neq D_E^q(v, u)$ where $p \in \partial E(v)$ and $q \in \partial E(u)$.

We are ready to introduce the Bregman iteration. Suppose E is a convex function and H is a function such that $\min_u H(u) = 0$. We want to solve the following optimization problem:

$$\min_u E(u) \text{ subject to } H(u) = 0 \, . \tag{2.1}$$

The usual solution is to solve the related unconstrained problem

$$\min_u E(u) + \lambda H(u) \, . \tag{2.2}$$

When $\lambda \to \infty$, (2.2) gives the same solution as (2.1). But as λ is getting larger, the problem usually becomes ill-posed and harder to solve.

The *Bregman iteration* was proposed to solve the dilemma of solving an well-posed problem and increasing λ to get an accurate solution. In his paper [3], Bregman suggested that we fix λ, and the problem is solved by doing the following iteration scheme

$$\begin{aligned} u^{k+1} &= \arg\min_u D_E^p(u, u^k) + \lambda H(u) \\ &= \arg\min_u E(u) - E(u^k) - \langle p^k, u - u^k \rangle + \lambda H(u) \\ &= \arg\min_u E(u) - \langle p^k, u - u^k \rangle + \lambda H(u) \end{aligned}$$

where $p^k \in \partial E(u^k)$. He also gave the explicit formula for computing p^k:

$$p^{k+1} = p^k - \partial H(u^{k+1})$$

where ∂H is the gradient of H. Since H is differentiable, the iteration is well-defined. Let us restate the scheme for convenience:

$$\begin{cases} u^{k+1} = \arg\min_u D_E^p(u, u^k) + \lambda H(u) \\ p^{k+1} = p^k - \partial H(u^{k+1}) \end{cases} \tag{2.3}$$

Bregman claimed that under some mild conditions, this iteration will converge to the solution of (2.1). In this way, we do not have to increase λ and thus avoid the problems brought by large λ.

In [27], the Bregman iteration was applied to image denoising. However, this powerful tool was not used to accelerate the computation. The paper

claimed that the Bregman iteration increased the image quality. Recall that the ROF model (1.6) is

$$u^* = \arg\min_u \ TV(u) \ + \frac{\mu}{2}\|u - f\|_2^2 \ .$$

Comparing it with (2.2), and recalling that the effect of Bregman iteration is equivalent to letting $\lambda \to \infty$, they argue that if λ was chosen smaller than $\mu/2$, then after some iterations the Bregman iteration scheme will generate some solution close to u^*. Here follows the details.

Let $E(u) = TV(u)$, which obviously is convex, and let $H(u) = \frac{\lambda}{2}\|u - f\|_2^2$. It is easy to get $\partial H(u) = \lambda(u - f)$. The Bregman iteration becomes

$$\begin{cases} u^{k+1} = \arg\min_u D_E^p(u, u^k) + \dfrac{\lambda}{2}\|u - f\|_2^2 \\ p^{k+1} = p^k + \lambda(f - u^{k+1}) \end{cases}$$

This seemingly complex iteration can be simplified. Let $b^k = \frac{1}{\lambda}p^k$. Then the first equation can be simplified as follows:

$$\begin{aligned} u^{k+1} &= \arg\min_u D_E^p(u, u^k) + \frac{\lambda}{2}\|u - f\|_2^2 \\ &= \arg\min_u E(u) - E(u^k) - \langle p^k, u - u^k\rangle + \frac{\lambda}{2}\|u - f\|_2^2 \\ &= \arg\min_u E(u) - \lambda\langle b^k, u - u^k\rangle + \frac{\lambda}{2}\langle u - f, u - f\rangle \\ &= \arg\min_u E(u) - \lambda\langle b^k, u - f\rangle + \frac{\lambda}{2}\langle u - f, u - f\rangle \qquad (2.4) \\ &= \arg\min_u E(u) + \frac{1}{\lambda}\langle b^k, b^k\rangle - \lambda\langle b^k, u - f\rangle + \frac{\lambda}{2}\langle u - f, u - f\rangle \\ &= \arg\min_u E(u) + \frac{\lambda}{2}\langle u - f - b^k, u - f - b^k\rangle \\ &= \arg\min_u E(u) + \frac{\lambda}{2}\|u - f - b^k\|_2^2 \ . \end{aligned}$$

Noting $E(u) = TV(u)$, eventually we have

$$\begin{cases} u^{k+1} = \arg\min_u TV(u) + \dfrac{\lambda}{2}\|u - f - v^k\|_2^2 \\ b^{k+1} = b^k + f - u^{k+1} \ . \end{cases}$$

We point out that the first equation coincides with the ROF model with exceptions of v^k and λ. Thus it is as hard to solve as the ROF model itself. Therefore, this iteration scheme runs slower than solving the ROF model. The objective of this scheme is, as they claimed, different visual quality than a single-step ROF model.

The convergence was also studied in their paper. The following theorem was proposed in [27].

Theorem 2.1. *For an initial vector $p^0 \in \mathbb{R}$, let u^{k+1} and p^{k+1} be given by the Bregman iteration (2.3). Then $p^k \in \partial E(u^k)$ and $H(u^{k+1}) \leq H(u^k)$ for $k = 1, 2, \ldots$. Moreover, if $\min_{u\in\mathbb{R}} H(u) = 0$, then*

$$\lim_{k\to\infty} H(u^k) = 0.$$

In [21], convergence of the Bregman iteration was further discussed. Consider the minimization problem

$$\min_u E(u) \quad \text{subject to } Au = b \tag{2.5}$$

where $u, b \in \mathbb{R}^n$ and A is an $n \times n$ matrix. The Bregman iteration for this

problem is

$$\begin{cases} u^{k+1} = \arg\min_u \{D_E^{p^k}(u, u^k) + \frac{\lambda}{2}\|Au - b\|_2^2 \\ p^{k+1} = p^k - \lambda A^T(Au^{k+1} - b). \end{cases} \tag{2.6}$$

Using the same technique as (2.4), this scheme can be simplified as

$$\begin{cases} u^{k+1} = \arg\min_u \{\{E(u) + \frac{\lambda}{2}\|Au - b^k\|_2^2\} \\ b^{k+1} = b^k + b - Au^{k+1}, \end{cases} \tag{2.7}$$

with $p^k = \lambda A^T(b^k - b)$. We propose the following basic criterion for the convergence of (2.7).

Theorem 2.2. *Suppose problem (2.5) has a unique solution $\tilde{u}$. For $k = 0, 1, \ldots,$ let u^{k+1} and b^{k+1} be given by (2.7). Then $\lim_{k\to\infty} u^k = \tilde{u}$, provided the following three conditions are satisfied:*

1. $\lim_{k\to\infty}(u^{k+1} - u^k) = 0$,
2. $(u^k)_{k=1,2,\ldots}$ *is a bounded sequence in* $\mathbb{R}^n$,
3. $(b^k)_{k=1,2,\ldots}$ *is a bounded sequence in* $\mathbb{R}^n$.

Proof. Since $(u^k)_k, (b^k)_k$ are bounded, they have convergent sub-sequences. Hence, we can find $(k_j)_{j=1,2,\ldots} \subset \mathbb{Z}^+$ s.t. $\lim_{j\to\infty} u^{k_j} = u^*$ and $\lim_{j\to\infty} p^{k_j} = p^*$ exist. By the first condition we also have $\lim_{j\to\infty} u^{k_j+1} = u^*$ By Theorem 2.1, $\lim_{j\to\infty} \|Au^{k_j} - b\|_2^2 = 0$. Hence, $Au^* = b$. It follows from (2.6) that

$$\begin{aligned} &E(u^{k_j+1}) - \langle p^{k_j}, u^{k_j+1} - u^{k_j}\rangle + \frac{\lambda}{2}\|Au^{k_j+1} - b\|_2^2 \\ \leq &E(u) - \langle p^{k_j}, u - u^{k_j}\rangle + \frac{\lambda}{2}\|Au - b\|_2^2 \end{aligned}$$

for all $u \in \mathbb{R}^n$. Choosing $u = \tilde{u}$ in the above inequality and letting $j \to \infty$, we obtain

$$E(u^*) \leq E(\tilde{u}) - \langle p^*, \tilde{u} - u^* \rangle.$$

We can estimate $\langle p^*, \tilde{u} - u^* \rangle$ as follows. Since $p^k = \lambda A^T(b^k - b))$, p^{k_j} lies in the range of $A^T, \forall j = 1, 2, \ldots$. Hence, p^* also lies in the range of A^T, i.e. $p^* = A^T w$ for some $w \in \mathbb{R}^n$. Therefore,

$$\langle p^*, \tilde{u} - u^* \rangle = \langle A^T w, \tilde{u} - u^* \rangle = \langle w, A(\tilde{u} - u^*) \rangle = 0.$$

Consequently, we have $E(u^*) \leq E(\tilde{u})$. Combining this conclusion and $Au^* = b$, we have $u^* = \tilde{u}$.

Since $\lim_{j \to \infty} u^{k_j} = u^*$ for any convergent sub-sequence $(u^{k_j})_j$, we conclude that $(u^k)_{k=1,2,\ldots}$ itself converges to $\tilde{u}$. □

2.2 The Split Bregman Method

The revolution of fast algorithms for solving the ROF model was made by Goldstein and Osher in 2008. In their paper [17], a splitting technique was proposed and was extremely successful in solving ℓ_1 regularized problems like total variation based denoising and compressed sensing. Combining with splitting technique, the Bregman iteration became the *Split Bregman iteration* and soon attracted more and more attention.

Before we introduce the Split Bregman, recall the ROF model (1.6)

$$
\begin{aligned}
u^* &= \arg\min_u \int_\Omega |\nabla u| \ + \frac{\mu}{2}\|u-f\|_2^2 \\
&= \begin{cases} \arg\min_u \int_\Omega \sqrt{u_x^2+u_y^2} \ + \frac{\mu}{2}\|u-f\|_2^2 \text{ which is isotrpic} \\ \arg\min_u \int_\Omega (|u_x|+|u_y|) \ + \frac{\mu}{2}\|u-f\|_2^2 \text{ which is anisotrpic} \end{cases}
\end{aligned}
$$

For simplicity, we consider the one dimensional case. Assume $\Omega = [0,1] \subset \mathbb{R}$. The 1-D ROF model is

$$
u^* = \arg\min_{u\in\Omega} \{\|u_x\|_1 + \frac{\mu}{2}\|u-f\|_2^2\} \ , \tag{2.8}
$$

and its discretized form is

$$
u^* = \arg\min_{u\in\mathbb{R}^n} \{\|\nabla u\|_1 + \frac{\mu}{2}\|u-f\|_2^2\} \ , \tag{2.9}
$$

where $\nabla : \mathbb{R}^n \to \mathbb{R}^n$ is the difference operator defined by

$$
(\nabla u)_i = \begin{cases} 0 & \text{if } i = 1 \\ u_i - u_{i-1} & \text{if } i > 1 \ . \end{cases}
$$

Note that it is the one dimensional version of ∇_x and ∇_y and can be written as a matrix

$$
\nabla = \begin{bmatrix} 0 & 0 & \cdots & & 0 \\ -1 & 1 & & \cdots & 0 \\ 0 & -1 & 1 & & \cdots \\ \vdots & & \ddots & \ddots & \\ 0 & \cdots & & -1 & 1 \end{bmatrix} .
$$

The difficulty of solving the ROF model is how to deal with differential ∇u and the indifferentiable ℓ_1 norm $\|\cdot\|_1$ simultaneously. The splitting technique solves this dilemma wisely by splitting these two difficulties. Let $v = \nabla u$, then the problem becomes

$$u^* = \arg\min_{u\in\mathbb{R}^n} \|v\|_1 + \frac{\mu}{2}\|u - f\|_2^2 \text{ subject to } v = \nabla u \,. \tag{2.10}$$

Recall that the Bregman iteration was designed to solve problems of the form

$$\min_u E(u), \quad \text{subject to } H(u) = 0 \,.$$

Comparing it with (2.10), we can apply the Bregman iteration to this problem easily. Since one of the advantages of Bregman iteration is that E does not have to be differentiable, we exploit this good property by doing the Bregman iteration with respect to v rather than u. We temporarily fix u, and let $E(v) = \|v\|_1 + \frac{\mu}{2}\|u - f\|_2^2$ and $H(v) = \frac{\lambda}{2}\|v - \nabla u\|_2^2$. Then we come up with the following iteration:

$$\begin{cases} v^{k+1} = \arg\min_v D_E^p(v, v^k) + \dfrac{\lambda}{2}\|v - \nabla u\|_2^2 \\ p^{k+1} = p^k - \lambda(v - \nabla u) \,. \end{cases} \tag{2.11}$$

This iteration can be simplified with the same technique as (2.4), into

$$\begin{cases} v^{k+1} = \arg\min_v \|v\|_1 + \dfrac{\mu}{2}\|u - f\|_2^2 + \dfrac{\lambda}{2}\|v - \nabla u - b^k\|_2^2 \\ b^{k+1} = b^k + \nabla u - v^{k+1} \,. \end{cases} \tag{2.12}$$

Taking u into account, in the first equation, we solve the minimization problem w.r.t. both u and v. Hence the iteration scheme becomes

$$\begin{cases} (u^{k+1}, v^{k+1}) = \arg\min_{u,v} \|v\|_1 + \frac{\mu}{2}\|u - f\|_2^2 + \frac{\lambda}{2}\|v - \nabla u - b^k\|_2^2 \\ b^{k+1} = b^k + \nabla u^{k+1} - v^{k+1} \ . \end{cases} \tag{2.13}$$

which is called the *Split Bregman iteration*. If (2.8) has a solution, this iteration will converge to the solution of (2.10). Hence u^k will converge to the solution of (2.8). The convergence of this iteration scheme is studied in [21].

2.3 The Alternating Bregman Method

The Split Bregman Iteration (2.13) is not a fast algorithm yet. The solution to the first equation of (2.13) is essential for efficiency of the Split Bregman method. Instead of solving (u^{k+1}, v^{k+1}) together, we can split the problem

$$(u^{k+1}, v^{k+1}) = \arg\min_{u,v} \|v\|_1 + \frac{\mu}{2}\|u - f\|_2^2 + \frac{\lambda}{2}\|v - \nabla u - b^k\|_2^2$$

into two sub-problems

$$\begin{cases} \tilde{u} = \arg\min_{u} \|v^k\|_1 + \frac{\mu}{2}\|u - f\|_2^2 + \frac{\lambda}{2}\|v^k - \nabla u - b^k\|_2^2 \\ \tilde{v} = \arg\min_{v} \|v\|_1 + \frac{\mu}{2}\|u^k - f\|_2^2 + \frac{\lambda}{2}\|v - \nabla u^k - b^k\|_2^2 \end{cases} \tag{2.14}$$

and solve them respectively. They can be simplified as

$$\begin{cases} \tilde{u} = \arg\min_{u} \frac{\mu}{2}\|u - f\|_2^2 + \frac{\lambda}{2}\|v^k - \nabla u - b^k\|_2^2 \\ \tilde{v} = \arg\min_{v} \|v\|_1 + \frac{\lambda}{2}\|v - \nabla u^k - b^k\|_2^2 \ . \end{cases} \tag{2.15}$$

Clearly, in general $(\tilde{u}, \tilde{v}) \neq (u^{k+1}, v^{k+1})$. To get a better approximation, one need to iterate (2.15) several times while updating both u and v on each iteration. In real computation, we let $\tilde{u}^0 = u^k$ and $\tilde{v}^0 = v^k$ and use the following iteration to approximate the first equation of (2.13):

1: $\tilde{u}^0 = u^k$ and $\tilde{v}^0 = v^k$
2: Choose a proper number of iteration M
3: **for** $m = 1$ to M **do**
4: $\quad \tilde{u}^m = \arg\min_u \|\tilde{v}^{m-1}\|_1 + \frac{\mu}{2}\|u - f\|_2^2 + \frac{\lambda}{2}\|\tilde{v}^{m-1} - \nabla u - b^k\|_2^2$
5: $\quad \tilde{v}^m = \arg\min_v \|v\|_1 + \frac{\mu}{2}\|\tilde{u}^m - f\|_2^2 + \frac{\lambda}{2}\|v - \nabla\tilde{u}^m - b^k\|_2^2$
6: **end for**
7: $u^{k+1} = \tilde{u}^M$ and $v^{k+1} = \tilde{v}^M$

Since the solutions of problems of image processing are rounded into integers, we do not need accurate approximation. Goldstein and Osher in their paper [17] pointed out that only one iteration is enough, i.e. $(\tilde{u}^1, \tilde{v}^1)$ is a good approximation to (u^{k+1}, v^{k+1}).

Letting $M = 1$ and plugging this alternating minimization scheme into (2.13), we have the following scheme:

$$\begin{cases} u^{k+1} = \arg\min_u \ \dfrac{\mu}{2}\|u - f\|_2^2 + \dfrac{\lambda}{2}\|v^k - \nabla u - b^k\|_2^2 \\ v^{k+1} = \arg\min_v \ \|v\|_1 + \dfrac{\lambda}{2}\|v - \nabla u^{k+1} - b^k\|_2^2 \\ b^{k+1} = b^k + \nabla u^{k+1} - v^{k+1} \ . \end{cases} \tag{2.16}$$

In (2.16), we still have to solve two minimization problems in every iteration. Since the first sub-problem of (2.16) contains only $\|\cdot\|_2^2$, it is differentiable

and has the optimality condition

$$\begin{aligned} &\mu(u-f)+\lambda\nabla^T(\nabla u+b^k-v^k)=0 \\ \Rightarrow\ &(\mu I+\lambda\nabla^T\nabla)u=\mu f+\lambda\nabla^T(v^k-b^k) \\ \Rightarrow\ &u^{k+1}=(\mu I+\lambda\nabla^T\nabla)^{-1}\ \left[\mu f+\lambda\nabla^T(v^k-b^k)\right]. \end{aligned} \tag{2.17}$$

The second sub-problem of (2.16) has the form

$$v=\arg\min_{v\in\mathbb{R}^n}\ \|v\|_1+\frac{\lambda}{2}\|v-b\|_2^2$$

for some $b\in\mathbb{R}^n$. Let us consider a simple case when $n=1$. Suppose $a\in\mathbb{R}$ and $\lambda>0$, it is easy to verify that the problem

$$x=\arg\min_{x\in\mathbb{R}}\ |x|+\frac{\lambda}{2}(x-a)^2$$

has the solution

$$x=shrink(a,\frac{1}{\lambda}):=\begin{cases} -(a-\frac{1}{\lambda}), & \text{if } a<-\frac{1}{\lambda} \\ 0, & \text{if } -\frac{1}{\lambda}\le a\le\frac{1}{\lambda} \\ a-\frac{1}{\lambda}, & \text{if } a>\frac{1}{\lambda} \end{cases}$$

which is equivalent to soft-thresholding.

Now we consider $v\in\mathbb{R}^n$. Denoting by v_i the ith element of the vector v, we

have

$$\begin{aligned}
&\min_{v\in\mathbb{R}^n} \ \|v\|_1 + \frac{\lambda}{2}\|v-b\|_2^2 \\
=&\min_{v\in\mathbb{R}^n} \ \sum_{i=1}^{n} v_i + \frac{\lambda}{2}\sum_{i=1}^{n}(v-b)^2 \\
=&\min_{v\in\mathbb{R}^n} \ \sum_{i=1}^{n}\left[v_i + \frac{\lambda}{2}(v_i-b_i)^2\right] \\
=&\sum_{i=1}^{n}\min_{v_i\in\mathbb{R}} \ \left[v_i + \frac{\lambda}{2}(v_i-b_i)^2\right]
\end{aligned}$$

Thus the minimization problem is separable. Therefore, to minimize $v = \arg\min_{v\in\mathbb{R}^n} \ \|v\|_1 + \frac{\lambda}{2}\|v-b\|_2^2$, we extend the *shrink* function to $\mathbb{R}^n$ and have the solution

$$v = shrink(b, \frac{1}{\lambda})$$

in the sense of

$$v_i = shrink(b_i, \frac{1}{\lambda}) \ .$$

Applying the discussion above to the second subproblem of (2.16), we have an explicit solution

$$\begin{aligned}
v^{k+1} &= \arg\min_{v\in\mathbb{R}^n} \ \|v\|_1 + \frac{\lambda}{2}\|v-\nabla u^{k+1}-b^k\|_2^2 \\
&= shrink(\nabla u^{k+1} + b^k, \frac{1}{\lambda}) \ .
\end{aligned} \tag{2.18}$$

On the basis of the discussion above, we have the 1-D *Alternating Split Bregman Method*

Algorithm 2.1 One dimensional Alternating Split Bregman Method

$u^0 \leftarrow f$, $v^0 \leftarrow 0$, and $b^0 \leftarrow 0$
while $\|v - \nabla u\|_2 <$ some tolerance **do**
  $u^{k+1} = (\mu I + \lambda \nabla^T \nabla)^{-1} \left[\mu f + \lambda \nabla^T (v^k - b^k)\right]$
  $v^{k+1} = shrink(\nabla u^{k+1} + b^k, \frac{1}{\lambda})$
  $b^{k+1} = b^k + \nabla u^{k+1} - v^{k+1}$
end while

The name Alternating Split Bregman Method was proposed in [31].

We define $cut(u, \frac{1}{\lambda}) = u - shrink(u, \frac{1}{\lambda})$. Then the last step can be rewritten as

$$b^{k+1} = cut(b^k + \nabla u^{k+1}, \frac{1}{\lambda}) \ .$$

which coincide with our JZ Algorithm 2.4 on page 38. This method inspired us to propose our fast algorithm.

2.4 The Algorithm of Goldstein and Osher

In Algorithm 2.1 we still have to solve a linear system to obtain u^{k+1} and hence in theory no explicit algorithm can be derived. However, the linear system $u^{k+1} = (\mu I + \lambda \nabla^T \nabla)^{-1} \left[\mu f + \lambda \nabla^T (v^k - b^k)\right]$ is well-posed and it does not need to be solved precisely. In fact, Goldstein and Osher in [17] pointed out that u^{k+1} solved by Gauss-Seidel method with only one iteration is precise enough for the entire algorithm to converge efficiently.

Recall that the anisotropic ROF model (1.23) is

$$u^* = \arg\min_{u \in \mathbb{R}^{n^2}} \|\nabla_x u\|_1 + \|\nabla_y u\|_1 + \frac{\mu}{2} \|u - f\|_2^2 \ .$$

Note that we have two terms $\nabla_x u$ and $\nabla_y u$ to approximate. The split form is

$$\begin{aligned} u^* = \underset{u\in\mathbb{R}^{n^2}}{\arg\min}\ \|v_x\|_1 + \|v_y\|_1\ + \frac{\mu}{2}\|u-f\|_2^2\ . \\ \text{subject to } v_x = \nabla_x u\ \text{ and }\ v_y = \nabla_y u\ . \end{aligned} \tag{2.19}$$

Following the same process of the 1-dimensional case, the Alternating Split Bregman Method for this 2-dimensional model is

Algorithm 2.2 Alternating Split Bregman Method for the Anisotropic ROF model

$u^0 \leftarrow f, v_x^0 \leftarrow 0, v_y^0 \leftarrow 0, b_x^0 \leftarrow 0$ and $b_y^0 \leftarrow 0$
while $\|v_x - \nabla_x u\|_2 + \|v_y - \nabla_y u\|_2 <$ some tolerance **do**
$u^{k+1} = (\mu I + \lambda\nabla_x^T\nabla_x + \lambda\nabla_y^T\nabla_y)^{-1}\ \left[\mu f + \lambda\nabla_x^T(v_x^k - b_x^k) + \lambda\nabla_y^T(v_y^k - b_y^k)\right]$

$v_x^{k+1} = shrink(\nabla_x u^{k+1} + b_x^k, \frac{1}{\lambda})$
$v_y^{k+1} = shrink(\nabla_y u^{k+1} + b_y^k, \frac{1}{\lambda})$
$b_x^{k+1} = b_x^k + \nabla_x u^{k+1} - v_x^{k+1}$
$b_y^{k+1} = b_y^k + \nabla_y u^{k+1} - v_y^{k+1}$
end while

The Gauss-Seidel solution to u^{k+1} can be written component-wise as $u_{i,j}^{k+1} = G_{i,j}^k$ where

$$\begin{aligned} G_{i,j}^k =& \frac{\lambda}{\mu+4\lambda}(u_{i+1,j}^k + u_{i-1,j}^k + u_{i,j+1}^k + u_{i,j-1}^k \\ &+ v_{x,i-1,j}^k - v_{x,i,j}^k + v_{y,i,j-1}^k - v_{y,i,j}^k - b_{x,i-1,j}^k + b_{x,i,j}^k - b_{y,i,j-1}^k + b_{y,i,j}^k) \\ &+ \frac{\mu}{\mu+4\lambda} f_{i,j}\ . \end{aligned}$$

Eventually, Goldstein and Osher's algorithm is

Algorithm 2.3 GO Algorithm for the Anisotropic ROF model

$u^0 \leftarrow f, v_x^0 \leftarrow 0, v_y^0 \leftarrow 0, b_x^0 \leftarrow 0$ and $b_y^0 \leftarrow 0$
while $\|v_x - \nabla_x u\|_2 + \|v_y - \nabla_y u\|_2 <$ some tolerance **do**
  $u^{k+1} = G^k$
  $v_x^{k+1} = shrink(\nabla_x u^{k+1} + b_x^k, \frac{1}{\lambda})$
  $v_y^{k+1} = shrink(\nabla_y u^{k+1} + b_y^k, \frac{1}{\lambda})$
  $b_x^{k+1} = b_x^k + \nabla_x u^{k+1} - v_x^{k+1}$
  $b_y^{k+1} = b_y^k + \nabla_y u^{k+1} - v_y^{k+1}$
end while

We call this algorithm the GO algorithm throughout this thesis. This algorithm is very efficient. For a 512×512 image with noise level $\sigma = 25$, 30 iterations (i.e. stop at $k = 30$) is enough to attain optimal denoising result for most of images. On our computer with Intel Core 2 6400 2.13GHz and Windows Vista 32, the CPU time is 0.3 seconds.

The algorithm for isotropic case is also discussed in [17]. Recall the isotropic discrete ROF model (1.22)

$$\begin{aligned} u^* &= \arg\min_{u\in\mathbb{R}^{n^2}} \left\| \sqrt{(\nabla_x u)^2 + (\nabla_y u)^2} \right\|_1 + \frac{\mu}{2}\|u - f\|_2^2 \\ &= \arg\min_{u\in\mathbb{R}^{n^2}} \sum_{1\le i,j\le n} \sqrt{(\nabla_x u)_{i,j}^2 + (\nabla_y u)_{i,j}^2} + \frac{\mu}{2}\|u - f\|_2^2 \,. \end{aligned}$$

The split formulation becomes

$$\begin{aligned} u^* &= \arg\min_{u\in\mathbb{R}^{n^2}} \ \|\sqrt{v_x + v_y}\|_1 + \frac{\mu}{2}\|u - f\|_2^2 \\ \text{subject to} \quad & v_x = \nabla_x u \ \text{ and } \ v_y = \nabla_y u \,. \end{aligned} \tag{2.20}$$

It is easy to see that v_x and v_y do not decouple, and the minimization problem

$$(v_x^{k+1}, v_y^{k+1}) = \arg\min_{v_x, v_y} \|\sqrt{v_x + v_y}\|_1 + \frac{\lambda}{2}\|v_x - \nabla_x u - b_x\|_2^2 + \frac{\lambda}{2}\|v_y - \nabla_y u - b_y\|_2^2$$

is not separated as the anisotropic case. This problem is solved using a generalized shrinkage formula proposed in [34]:

$$v_x^{k+1} = shrink(s^k, \frac{1}{\lambda})\frac{\nabla_x u + b_x^k}{s^k}$$
$$v_y^{k+1} = shrink(s^k, \frac{1}{\lambda})\frac{\nabla_y u + b_y^k}{s^k}$$

where

$$s^k = \sqrt{(\nabla_x u + b_x^k)^2 + (\nabla_y u + b_y^k)^2}\ .$$

Besides this difference, u^{k+1}, b_x^{k+1} and b_y^{k+1} are solved in the same way as the anisotropic model. In numerical experiment, the isotropic GO algorithm cost twice as much time as the anisotropic one in a single iteration.

2.5 Our Algorithms

We propose an explicit iterative algorithm in [20] to solve the anisotropic discrete ROF model (1.23). Suppose f is the noisy image and μ is a properly chosen parameter depending on the noise level. The anisotropic discrete ROF model is the following optimization problem:

$$u^* = \arg\min_{u \in \mathbb{R}^{n^2}} \|\nabla_x u\|_1 + \|\nabla_y u\|_1 + \frac{\mu}{2}\|u - f\|_2^2\ .$$

Picking an arbitrary constant $0 < \gamma < 1/8$ and let $\lambda = \mu\gamma$, we have the following iteration scheme.

Algorithm 2.4 JZ Algorithm

1: $b_x^1 \leftarrow 0, b_y^1 \leftarrow 0, \lambda \leftarrow \gamma\mu$ and $u^1 = f$
2: **for** $k = 1$ to M **do**
3: $\quad b_x^{k+1} = cut(\nabla_x u^k + b_x^k, \frac{1}{\lambda})$
4: $\quad b_y^{k+1} = cut(\nabla_y u^k + b_y^k, \frac{1}{\lambda})$
5: $\quad u^{k+1} = f - \gamma\nabla_x^T b_x^{k+1} - \gamma\nabla_y^T b_y^{k+1}$
6: **end for**

This algorithm is very simple and turned out to be extremely efficient. We will prove that u^k converges to the solution of the ROF model. We point out that for fast convergence, γ should be as large as possible, and $\gamma < 1/8$ is a sufficient but not necessary condition for the iteration scheme to converge. In numerical experiment, γ could be as large as $1/4$ to make the algorithm converge faster.

This scheme can also be used to solve the isotropic model (1.22), i.e. the minimization problem

$$u^* = \arg\min_{u\in\mathbb{R}^{n^2}} \left\| \sqrt{(\nabla_x u)^2 + (\nabla_y u)^2} \right\|_1 + \frac{\mu}{2}\|u - f\|_2^2.$$

The corresponding algorithm is

Algorithm 2.5 JZ Algorithm for The Isotropic Model

1: $b_x^1 \leftarrow 0, b_y^1 \leftarrow 0, \lambda \leftarrow \gamma\mu$ and $u^1 = f$
2: **for** $k = 1$ to M **do**
3: $\quad t_x^{k+1} = \nabla_x u^k + b_x^k, \quad t_y^{k+1} = \nabla_y u^k + b_y^k$
4: $\quad s^{k+1} = \sqrt{(t_x^{k+1})^2 + (t_y^{k+1})^2}$
5: $\quad b_x^{k+1} = cut(s^{k+1}, \frac{1}{\lambda})\frac{t_x^{k+1}}{s^{k+1}}, \quad b_y^{k+1} = cut(s^{k+1}, \frac{1}{\lambda})\frac{t_y^{k+1}}{s^{k+1}}$
6: $\quad u^{k+1} = f - \gamma\nabla_x^T b_x^{k+1} - \gamma\nabla_y^T b_y^{k+1}$
7: **end for**

This scheme is also very efficient and produces almost the same images as the anisotropic one.

We will investigate Algorithm 2.5 for the isotropic model in the next section. Then we compare numerical performance of our Algorithm 2.4 (JZ) with Goldstein and Osher's Algorithm 2.3 (GO). The convergence of Algorithm 2.4 for the anisotropic model will be discussed in the next chapter.

2.6 Convergence Analysis for the Isotropic Model

In this section, we give a proof of convergence of the JZ Algorithm for the isotropic ROF model. We will skip the anisotropic case since it is covered by the general form of JZ Algorithm which is proved in Chapter 3.

Recall the isotropic ROF model (1.22):

$$u^* = \arg\min_{u\in\mathbb{R}^{n^2}} \left\| \sqrt{(\nabla_x u)^2 + (\nabla_y u)^2} \right\|_1 + \frac{\mu}{2}\|u - f\|_2^2 \,.$$

The JZ Algorithm for this minimization problem is

$$\begin{cases} t_x^{k+1} = \nabla_x u^k + b_x^k, \quad t_y^{k+1} = \nabla_y u^k + b_y^k \\ s^{k+1} = \sqrt{(t_x^{k+1})^2 + (t_y^{k+1})^2} \\ b_x^{k+1} = cut(s^{k+1}, \frac{1}{\lambda}) \frac{t_x^{k+1}}{s^{k+1}} \\ b_y^{k+1} = cut(s^{k+1}, \frac{1}{\lambda}) \frac{t_y^{k+1}}{s^{k+1}} \\ u^{k+1} = f - \gamma \nabla_x^T b_x^{k+1} - \gamma \nabla_y^T b_y^{k+1} \,. \end{cases} \tag{2.21}$$

The main result of this section is the following theorem.

Theorem 2.3. *For $k = 0, 1, \ldots,$ let b_x^k, b_y^k, u^k be given by the iteration scheme (2.21). If $0 < \gamma < 1/8$, then $\lim_{k\to\infty} u^k = u^*$.*

We extend the proof of convergence of the anisotropic model in [20] to prove Theorem 2.3. The difficulty for the isotropic scheme is that ∇_x and ∇_y do not decouple. Therefore, we must deal with them simultaneously. To prove this theorem, we introduce some notations. Recall u^k, f, b_x^k, b_y^k are $n^2 \times 1$ column vectors, and ∇_x, ∇_y are linear operators, i.e. $n^2 \times n^2$ matrices. We combine b_x^k and b_y^k into a $2n^2 \times 2n^2$ column vector as follows:

$$b^k = \begin{bmatrix} b_x^k \\ b_y^k \end{bmatrix}.$$

We also combine ∇_x and ∇_y into $\nabla : \mathbb{R}^{n^2} \to \mathbb{R}^{2n^2}$ defined by

$$\nabla = \begin{bmatrix} \nabla_x \\ \nabla_y \end{bmatrix}.$$

Let t^k be combination of t_x^k and t_y^k:

$$t^k = \begin{bmatrix} t_x^k \\ t_y^k \end{bmatrix}.$$

Note that the notations $\nabla_x, \nabla_y, t_x^k, t_y^k$ will still be used. We also stretch s^k into a $2n^2 \times 1$ vector by duplicating itself into $\left[\begin{smallmatrix} s^k \\ s^k \end{smallmatrix}\right]$:

$$\begin{aligned} s_i^{k+1} = s_{i+n^2}^{k+1} &= \sqrt{(t_{x,i}^{k+1})^2 + (t_{y,i}^{k+1})^2} \\ &= \sqrt{(t_i^{k+1})^2 + (t_{i+n^2}^{k+1})^2}. \end{aligned}$$

With these notations, (2.21) can be rewritten as

$$\begin{cases} t^{k+1} = \nabla u^k + b^k \\ s_i^{k+1} = s_{i+n^2}^{k+1} = \sqrt{(t_i^{k+1})^2 + (t_{i+n^2}^{k+1})^2} \\ b^{k+1} = cut(s^{k+1}, \frac{1}{\lambda}) \frac{t^{k+1}}{s^{k+1}} \\ u^{k+1} = f - \gamma \nabla^T b^{k+1} \ . \end{cases} \tag{2.22}$$

To prove Theorem (2.3), we show that (2.22) can be interpreted as another form:

$$\begin{cases} t^{k+1} = \nabla u^k + b^k \\ s_i^{k+1} = s_{i+n^2}^{k+1} = \sqrt{(t_i^{k+1})^2 + (t_{i+n^2}^{k+1})^2} \\ v^{k+1} = \begin{bmatrix} v_x^{k+1} \\ v_y^{k+1} \end{bmatrix} = \arg\min_v \left\{ \left\| \sqrt{(v_x)^2 + (v_y)^2} \right\|_1 + \frac{\lambda}{2} \|v - t^{k+1}\|_2^2 \right\} \\ b^{k+1} = t^{k+1} - v^{k+1} = \nabla u^k + b^k - v^{k+1} \\ u^{k+1} = \arg\min_u \{ \frac{1}{2} \|B(u-f)\|_2^2 - \langle B^2(u^k - f), u - u^k \rangle + \frac{\gamma}{2} \|v^{k+1} - \nabla u\|_2^2 \} \end{cases} \tag{2.23}$$

where B is an s.p.d matrix such that $B^2 = I - \gamma \nabla^T \nabla$. Existence of B is guaranteed by $\lambda < 1/8$. We first show that v^{k+1} defined by the third equation of (2.23) is equivalent to

$$v^{k+1} = shrink(s^{k+1}, \frac{1}{\lambda}) \frac{t^{k+1}}{s^{k+1}}.$$

Let us investigate the minimization problem

$$\min_{x,y \in \mathbb{R}} \sqrt{x^2 + y^2} + \frac{\lambda}{2} |x - c_x|^2 + \frac{\lambda}{2} |y - c_y|^2,$$

where $c_x, c_y \in \mathbb{R}$ and $\lambda > 0$ are given. For $(x, y) \in \mathbb{R}^2$, let $J(x, y) := \sqrt{x^2 + y^2}$, $F(x, y) := (\lambda/2)|x - c_x|^2 + (\lambda/2)|y - c_y|^2$, and $E(x, y) := J(x, y) + F(x, y)$. It is easily seen that

$$\partial F(0, 0) = (\lambda c_x, \lambda c_y)$$

and

$$\partial G(0, 0) = (x, y) \in \mathbb{R}^2 \ : \ x^2 + y^2 < 1.$$

Consequently, we have

- Case 1. $c_x^2 + c_y^2 \leq \frac{1}{\lambda^2}$. In this case $\arg\min_{x,y} E(x, y) = (0, 0)$.

- Case 2. $c_x^2 + c_y^2 \geq \frac{1}{\lambda^2}$. In this case we have

$$\frac{x}{\sqrt{x^2 + y^2}} + \lambda(x - c_x) \text{ and } \frac{y}{\sqrt{x^2 + y^2}} + \lambda(y - c_y).$$

  It follows that $x/c_x = y/c_y$. Hence, there exists a real number t s.t. $x = tc_x$ and $y = tc_y$. Consequently,

$$E(x, y) = |t|\sqrt{c_x^2 + c_y^2} + \frac{\lambda}{2}(t - 1)^2(c_x^2 + c_y^2).$$

  Taking t as the variable, we can see that $E(x, y)$ achieves the minimum if and only if

$$t = shrink\Big(1, \frac{1}{\lambda\sqrt{c_x^2 + c_y^2}}\Big),$$

  which implies

$$x = shrink(s, 1/\lambda)\frac{c_x}{s} \text{ and } y = shrink(s, 1/\lambda)\frac{c_y}{s},$$

where $s = \sqrt{c_x^2 + c_y^2}$.

Therefore, we conclude that

$$v^{k+1} = \arg\min_v \left\{ \left\| \sqrt{(v_x)^2 + (v_y)^2} \right\|_1 + \frac{\lambda}{2} \|v - t^{k+1}\|_2^2 \right\}$$

is true, if and only if

$$v^{k+1} = shrink(s^{k+1}, \frac{1}{\lambda}) \frac{t^{k+1}}{s^{k+1}}.$$

Consequently, it is clear that

$$b^{k+1} = t^{k+1} - v^{k+1} = cut(s^{k+1}, \frac{1}{\lambda}) \frac{t^{k+1}}{s^{k+1}}.$$

For $(u^k)_{k=1,2,\ldots}$, we consider the last equation of (2.23). Note $B^2 = I - \gamma \nabla^T \nabla$. By differentiating the last equation of (2.23), we have

$$\begin{aligned} & B^2(u^{k+1} - f) - B^2(u^k - f) + \gamma \nabla^T (\nabla u^{k+1} - v^{k+1}) = 0 \\ \Rightarrow \quad & B^2(u^{k+1} - u^k) = \gamma \nabla^T (v^{k+1} - \nabla u^{k+1}) \\ \Rightarrow \quad & (I - \gamma \nabla^T \nabla)(u^{k+1} - u^k) = \gamma \nabla^T (v^{k+1} - \nabla u^{k+1}) \\ \Rightarrow \quad & u^{k+1} - u^k = \gamma \nabla^T (v^{k+1} - \nabla u^k) \end{aligned} \tag{2.24}$$

The fourth equation in (2.23) implies

$$v^{k+1} - \nabla u^k = b^k - b^{k+1} \ .$$

Plugging it into (2.24), we have

$$u^{j+1} - u^j = \gamma \nabla^T (b^j - b^{j+1}), \quad j = 1, 2, \ldots, k.$$

Adding up from $j = 1$ to k, we get

$$\begin{aligned} & \textstyle\sum_{j=1}^{k}(u^{j+1} - u^j) = \sum_{j=1}^{k} \gamma \nabla^T (b^j - b^{j+1}) \\ \Rightarrow \quad & u^{k+1} - u^1 = \textstyle\sum_{j=1}^{k} \gamma \nabla^T (b^j - b^{j+1}) \\ \Rightarrow \quad & u^{k+1} - f = -\gamma \nabla^T b^{k+1} \\ \Rightarrow \quad & u^{k+1} = f - \gamma \nabla^T b^{k+1} \end{aligned}$$

which equates the last equation of (2.22). Hence, we conclude that (2.22) and (2.23) are equivalent.

We introduce some notations:

- $J(v) = \| \sqrt{v_x^2 + v_y^2} \|$
- ∂J is the subdifferential of J
- $D_J^p(v, v_0) = J(v) - J(v_0) - \langle p, v - v_0 \rangle$ be the Bregman distance between v and v_0 associated with J, where $p \in \partial J(v)$
- $\Upsilon^k = \gamma \| v^{k+1} - \nabla u^k \|_2^2$

With these notations, we have

$$v^{k+1} = \arg\min_v \{ J(v) + \frac{\lambda}{2} \| v - \nabla u^k - b^k \| \}$$

which implies

$$\begin{aligned} & 0 \in \partial J(v^{k+1}) + \lambda (v^{k+1} - \nabla u^k - b^k). \\ \Rightarrow \quad & \lambda (b^k + \nabla u^k - v^{k+1}) \in \partial J(v^{k+1}) \\ \Rightarrow \quad & \lambda b^{k+1} \in \partial J(v^{k+1}) \\ \Rightarrow \quad & \lambda b^k \in \partial J(v^k) \end{aligned}$$

Let $p^k = \lambda b^k$, then $p^k \in \partial J(v^k)$. By the definition of Bregman distance, we deduced a useful relation

$$D_J^{p^k}(v, v^k) = \|v\| - \|v^k\| - \langle p^k, v - v^k \rangle \geq 0, \ v \in \mathbb{R}^n$$

and especially when $v = v^{k+1}$,

$$D_J^{p^k}(v^{k+1}, v^k) = \|v^{k+1}\| - \|v^k\| - \langle p^k, v^{k+1} - v^k \rangle \geq 0.$$

Lemma 2.4. *$(u^k), (v^k)_k, (b^k)_k, (p^k)_k$ are all bounded sequences.*

Proof. By the definition of *cut* function, it is clear that $\|b^k\|_\infty = \sup_k b^k \leq \frac{1}{\lambda}$. Therefore, $(b^k)_k$ is a bounded sequence. And since $p^k = \lambda b^k$ we conclude that $(p^k)_k$ are also bounded.
The fourth equation in (2.23) implies

$$v^{k+1} = b^k + \nabla u^k - b^{k+1}.$$

Hence $(v^k)_k$ is bounded.

Since $u^{k+1} = f - \gamma \nabla^T b^k$, $(u^k)_k$ is also bounded.

□

Lemma 2.5. $\lim_{k\to\infty} \|u^{k+1} - u^k\|_2^2 = 0$. *Moreover,* $\Upsilon^k \geq \Upsilon^{k+1}$ *for all* $k \in \mathbb{Z}$.

Proof. We first show $\|v^{k+1} - \nabla u^k\|_2^2 \leq \|v^k - \nabla u^k\|_2^2$.

By the definition of v^{k+1}, we have

$$\begin{aligned} v^{k+1} &= \arg\min_v \{\|v\| + \tfrac{\lambda}{2}\|v - \nabla u^k - b^k\|_2^2\} \\ &= \arg\min_v \{\|v\| + \tfrac{\lambda}{2}\|v - \nabla u^k\|_2^2 + \tfrac{\lambda}{2}\|b^k\|_2^2 - \lambda\langle v - \nabla u^k, b^k\rangle\} \\ &= \arg\min_v \{\|v\| + \tfrac{\lambda}{2}\|v - \nabla u^k\|_2^2 - \langle \lambda b^k, v - v^k\rangle\} \\ &= \arg\min_v \{\|v\| + \tfrac{\lambda}{2}\|v - \nabla u^k\|_2^2 - \langle p^k, v - v^k\rangle\} \ . \end{aligned}$$

Therefore, v^{k+1} is the minimizer of $\|v\| + \frac{\lambda}{2}\|v - \nabla u^k\|_2^2 - \langle p^k, v - v^k\rangle$. It follows that

$$\begin{aligned} & \|v^{k+1}\| + \tfrac{\lambda}{2}\|v^{k+1} - \nabla u^k\|_2^2 - \langle p^k, v^k - v^k\rangle \\ \leq\ & \|v^k\| + \frac{\lambda}{2}\|v^k - \nabla u^k\|_2^2 \\ \Rightarrow\quad & 0 \leq D_J^{p^k} \leq \frac{\lambda}{2}\|v^k - \nabla u^k\|_2^2 - \frac{\lambda}{2}\|v^{k+1} - \nabla u^k\|_2^2 \\ \Rightarrow\quad & \|v^{k+1} - \nabla u^k\|_2^2 \leq \|v^k - \nabla u^k\|_2^2. \end{aligned} \tag{2.25}$$

Then we show $\Upsilon^{k+1} \leq \Upsilon^k$. Recall the last step of the iteration (2.23)

$$\begin{aligned} u^{k+1} = \ & \arg\min_u \ \{\ \tfrac{1}{2}\|B(u - f)\|_2^2 - \langle B^2(u^k - f), u - u^k\rangle \\ & + \tfrac{\gamma}{2}\|v^{k+1} - \nabla u\|_2^2\ \}\ . \end{aligned}$$

Since u^{k+1} is the minimizer, it is smaller to substitute $u = u^{k+1}$ than $u = u^k$:

$$\begin{aligned}
& \tfrac{1}{2}\|B(u^{k+1} - f)\|_2^2 - \langle D^2(u^k - f), u^{k+1} - u^k\rangle + \tfrac{\gamma}{2}\|v^{k+1} - \nabla u^{k+1}\|_2^2 \\
& \leq \tfrac{1}{2}\|B(u^k - f)\|_2^2 + \tfrac{\gamma}{2}\|v^{k+1} - \nabla u^k\|_2^2 \\
\Rightarrow \quad & \tfrac{1}{2}\|B(u^{k+1} - f)\|_2^2 - \langle D^2(u^k - f), u^{k+1} - u^k\rangle - \tfrac{1}{2}\|B(u^k - f)\|_2^2 \\
& \leq \tfrac{\gamma}{2}\|v^{k+1} - \nabla u^k\|_2^2 - \tfrac{\gamma}{2}\|v^{k+1} - \nabla u^{k+1}\|_2^2 \\
\Rightarrow \quad & \tfrac{1}{2}\|B(u^{k+1} - u^k)\|_2^2 \\
& \leq \tfrac{\gamma}{2}\|v^{k+1} - \nabla u^k\|_2^2 - \tfrac{\gamma}{2}\|v^{k+1} - \nabla u^{k+1}\|_2^2 \,.
\end{aligned}$$

Applying (3.5), we have

$$\begin{aligned}
& \tfrac{1}{2}\|B(u^{k+1} - u^k)\|_2^2 \\
& \leq \tfrac{\gamma}{2}(\|v^{k+1} - \nabla u^k\|_2^2 - \|v^{k+1} - \nabla u^{k+1}\|_2^2) \\
& \leq \tfrac{\gamma}{2}(\|v^{k+1} - \nabla u^k\|_2^2 - \|v^{k+2} - \nabla u^{k+1}\|_2^2) \\
& = \Upsilon^k - \Upsilon^{k+1} \\
\Rightarrow \quad & \Upsilon^{k+1} \leq \Upsilon^k - \tfrac{1}{2}\|B(u^{k+1} - u^k)\|_2^2 \leq \Upsilon^k
\end{aligned}$$

Therefore $\Upsilon^{k+1} \leq \Upsilon^k$. On the other hand, it is clear that $\Upsilon^k \geq 0$. Hence $\lim_{k\to\infty} \Upsilon^k$ exists. It follows that

$$\lim_{k\to\infty} \frac{1}{2}\|B(u^{k+1} - u^k)\|_2^2 = \lim_{k\to\infty} \Upsilon^k - \Upsilon^{k+1} = 0.$$

Hence, we deduce that

$$\lim_{k\to\infty} \|u^{k+1} - u^k\|_2^2 = 0 \,.$$

□

Lemma 2.6 ($v^* = \nabla u^*$)**.** $\lim_{k\to\infty} \Upsilon^k = 0$. *It follows that* $\lim_{k\to\infty} v^k - \nabla u^k = 0$.

Proof. Noting $v^{k+1} - \nabla u^k = b^k - b^{k+1}$, we consider $p^k - p^{k+1}$ firstly.

For any $v \in \mathbb{R}^n$, we have

$$\begin{aligned} & D_J^{p^{j+1}}(v, v^{j+1}) - D_J^{p^j}(v, v^j) + D_J^{p^j}(v^{j+1}, v^j) \\ = \; & J(v) - J(v^{j+1}) - \langle p^{j+1}, v - v^{j+1} \rangle - J(v) + J(v^j) + \langle p^j, v - v^j \rangle \\ & + J(v^{j+1}) - J(v^j) - \langle p^j, v^{j+1} - v^j \rangle \\ = \; & \langle p^j - p^{j+1}, v - v^{j+1} \rangle \; . \end{aligned}$$

This implies

$$D_J^{p^{j+1}}(v, v^{j+1}) - D_J^{p^j}(v, v^j) \leq \langle p^j - p^{j+1}, v - v^{j+1} \rangle \; .$$

On the other hand, consider the function

$$M(v) = \frac{\lambda}{2} \| v - \nabla u^{j+1} \|_2^2$$

we have

$$\begin{aligned} & p^j - p^{j+1} = \lambda(b^j - b^{j+1}) = \lambda(v^{j+1} - \nabla u^j) = \partial M(v^{j+1}) \\ \Rightarrow \; & D_M^{p^j - p^{j+1}}(v) = M(v) - M(v^{j+1}) - \langle p^j - p^{j+1}, v - v^{j+1} \rangle > 0 \\ \Rightarrow \; & M(v) - M(v^{j+1}) \geq \langle p^j - p^{j+1}, v - v^{j+1} \rangle \geq D_J^{p^{j+1}}(v, v^{j+1}) - D_J^{p^j}(v, v^j) \\ \Rightarrow \; & \tfrac{\lambda}{2} \| v - \nabla u^j \|_2^2 - \tfrac{\lambda}{2} \| v^{j+1} - \nabla u^j \|_2^2 \\ & \geq \| v^j \| - \| v^{j+1} \| + \langle p^j, v - v^j \rangle - \langle p^{j+1}, v - v^{j+1} \rangle \end{aligned}$$

Choosing $v = \nabla u^j$, we come up with

$$\begin{aligned} & \tfrac{\lambda}{2} \| v^{j+1} - \nabla u^j \|_2^2 \\ \leq \; & \| v^{j+1} \| - \| v^j \| + \langle p^{j+1}, \nabla u^j - v^{j+1} \rangle - \langle p^j, \nabla u^j - v^j \rangle \\ = \; & (\| v^{j+1} \| - \| v^j \|) + (\langle p^{j+1}, \nabla u^j - v^{j+1} \rangle - \langle p^j, \nabla u^{j-1} - v^j \rangle) - \langle p^j, \nabla u^j - \nabla u^{j-1} \rangle \; . \end{aligned}$$

Summing j from m to $k-1$, we obtain

$$\begin{aligned} & \tfrac{\lambda}{2}\textstyle\sum_{j=m}^{k-1}\|v^{j+1}-\nabla u^j\|_2^2 \\ \leq\ & \|v^k\|-\|v^m\|+\langle p^k,\nabla u^{k-1}-v^k\rangle-\langle p^m,\nabla u^{m-1}-v^m\rangle]-\textstyle\sum_{j=m}^{k-1}\langle p^j,\nabla u^j-\nabla u^{j-1}\rangle\ . \end{aligned}$$

Since the sequences $(u^j)_j,(v^j)_j,(p^j)_j$ are bounded, there exists positive constants $C_1 l$ and C_2 independent of k and m such that

$$\frac{\lambda}{2}\sum_{j=m}^{k-1}\|v^{j+1}-\nabla u^j\|_2^2\leq C_{l,1}+C_{l,2}(k-m)\eta_m$$

where $\eta_m=\sup_{j\geq m}\|u^{j+1}-u^j\|_2\overset{m\to\infty}{\longrightarrow}0$. Hence

$$\sum_{j=m}^{k-1}\Upsilon^j\leq C_1+C_2(k-m)\eta_m.$$

Since (Υ^j) is a decreasing sequence, we have

$$\begin{aligned} & (k-m)\Upsilon^k\leq C_1+C_2(k-m)\eta_m \\ \Rightarrow\ & \Upsilon^k\leq\tfrac{C_1}{k-m}+C_2\eta_m\ \text{ for any } 1<m<k-1 \\ \Rightarrow\ & \Upsilon^k\leq\tfrac{C_1}{m}+C_2\eta_m\ \text{ for } m=\lfloor k/2\rfloor \end{aligned}$$

Therefore, $\lim_{k\to\infty}\Upsilon^k=0$. It follows that $\lim_{k\to\infty}v^k-\nabla u^k=0$. □

We are in a position to prove Theorem 2.3. Assume

$$u^*=\underset{u\in\mathbb{R}^{n^2}}{\arg\min}\ \{\big\|\sqrt{(\nabla_x u)^2+(\nabla_y u)^2}\big\|_1+\frac{\mu}{2}\|u-f\|_2^2\}$$

is the unique solution. We'll show $\lim_{k\to\infty}u^k=u^*$.

Proof. Let $F(u) = \frac{1}{2}\|u - f\|_2^2$. Then $\partial F(u^{k+1}) = u^{k+1} - f$. $\forall w \in \mathbb{R}^{n^2}$, we have

$$F(u^{k+1} + w) - F(u^{k+1}) + \langle u^{k+1} - f, w\rangle = D_F^{u^{k+1}-f}(u^{k+1} + w, u^{k+1}) \geq 0.$$

By the last equation of (2.22) we know $u^{k+1} - f = -\gamma\nabla^T b^k$. Hence,

$$F(u^{k+1} + w) - F(u^{k+1}) + \langle \gamma b^k, \nabla w\rangle \geq 0. \tag{2.26}$$

On the other hand, recall $p^k = \lambda b^k \in \partial J(v^k)$. It follows that

$$\begin{aligned} & J(v^k + \nabla w) - J(v^k) - \langle p^k, \nabla w\rangle \geq 0 \\ \Rightarrow\quad & \|v^k + \nabla w\| - \|v^k\| - \langle \lambda b^k, \nabla w\rangle \geq 0 \\ \Rightarrow\quad & \langle \gamma b^k, \nabla w\rangle \leq \tfrac{1}{\mu}\|v^k + \nabla w\| - \tfrac{1}{\mu}\|v^k\|. \end{aligned}$$

The two inequalities above yield

$$\begin{aligned} & \|v^k\| + \mu F(u^{k+1}) \\ \leq\quad & \|v^k + \nabla w\| + \mu F(u^{k+1} + w), \quad \forall w \in \mathbb{R}^{n^2}. \end{aligned} \tag{2.27}$$

(u^k) is a bounded sequence. Thus it has convergent subsequences. Picking up arbitrary convergent subsequent $(u^{k_j})_{j=1,2\ldots}$, we denote $\tilde{u} = \lim_{j\to\infty} u^{k_j}$. Since $\lim_{k\to\infty}(u^{k+1} - u^k) = 0$, we also have $\lim_{j\to\infty} u^{k_j+1} = \tilde{u}$.

By Lemma 2.6, $\lim_{j\to\infty} v^{k_j} = \nabla\tilde{u}$. Therefore, replacing u^k in (2.27) by u^{k_j} and letting $j \to \infty$, we obtain

$$\begin{aligned} & \|\nabla\tilde{u}\| + \mu F(\tilde{u}) \\ \leq\quad & \|\nabla(\tilde{u} + w)\| + \mu F(\tilde{u} + w), \ \forall w \in \mathbb{R}^{n^2}. \end{aligned} \tag{2.28}$$

Hence, $\tilde{u} = \arg\min_u \{\|\nabla u\| + \mu F(u)\} = u^*$. Since $\lim_{j\to\infty} u^{k_j} = \tilde{u} = u^*$ is true for any convergent subsequence u^{k_j}, we conclude that

$$\lim_{k\to\infty} u^k = u^* \ .$$

□

2.7 Numerical Performance

The rate of convergence of JZ Algorithm 2.4 is limited by the restriction $\gamma < 1/8$. To accelerate the calculation, we apply a relaxation technique to the iteration. Recall the iteration scheme of JZ algorithm:

$$\begin{cases} b_x^{k+1} = cut(\nabla_x u^k + b_x^k, \dfrac{1}{\lambda}) \\ b_y^{k+1} = cut(\nabla_y u^k + b_y^k, \dfrac{1}{\lambda}) \\ u^{k+1} = f - \gamma \nabla_x^T b_x^{k+1} - \gamma \nabla_y^T b_y^{k+1} \ . \end{cases}$$

Choosing proper $0 < t < 1$, we rewrite the third equation as

$$u^{k+1} = (1-t)u^k + t\big(f - \gamma \nabla_x^T b_x^{k+1} - \gamma \nabla_y^T b_y^{k+1}\big) \ .$$

With the coefficient t, the restriction $\gamma < 1/8$ is relaxed. Algorithm 2.4 becomes

Algorithm 2.6 JZ algorithm with relaxation for the anisotropic ROF model

$b_x^1 \leftarrow 0, b_y^1 \leftarrow 0, \lambda \leftarrow \gamma\mu$ and $u^1 = f$
for $k = 1$ to M **do**
  $b_x^{k+1} = cut(\nabla_x u^k + b_x^k, \frac{1}{\lambda})$
  $b_y^{k+1} = cut(\nabla_y u^k + b_y^k, \frac{1}{\lambda})$
  $u^{k+1} = (1-t)u^k + t\big(f - \gamma\nabla_x^T b_x^{k+1} - \gamma\nabla_y^T b_y^{k+1}\big)$
end for

And the algorithm for isotropic case is

Algorithm 2.7 JZ algorithm with relaxation for the isotropic ROF model

$b_x^1 \leftarrow 0, b_y^1 \leftarrow 0, \lambda \leftarrow \gamma\mu$ and $u^1 = f$
for $k = 1$ to M **do**
  $w_x^{k+1} = \nabla_x u^k + b_x^k, w_y^{k+1} = \nabla_y u^k + b_y^k$
  $s^{k+1} = \sqrt{(w_x^{k+1})^2 + (w_y^{k+1})^2}$
  $b_x^{k+1} = cut(s^{k+1}, \frac{1}{\lambda})\frac{w_x^{k+1}}{s^{k+1}}, b_y^{k+1} = cut(s^{k+1}, \frac{1}{\lambda})\frac{w_y^{k+1}}{s^{k+1}}$
  $u^{k+1} = (1-t)u^k + t\big(f - \gamma\nabla_x^T b_x^{k+1} - \gamma\nabla_y^T b_y^{k+1}\big)$
end for

With relaxation technique, we can choose proper t such that the algorithm converges without the restriction $\gamma < 1/8$. In numerical experiments, we found out that $\gamma = 0.5$ is optimal for rate of convergence.

Before reporting the performance of our model and algorithm, we introduce two concepts to describe the criterion of the quality of the resulting image. Recall these notations:

- $\Omega = \{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\}$ is the domain.
- $u_0 : \Omega \to \mathbb{R}$ is the original clean image.
- $f : \Omega \to \mathbb{R}$ is the image with noise.
- $u : \Omega \to \mathbb{R}$ is the image after processing. This is the image to be evaluated.

The *Mean Squared Error*(MSE) between u and f is defined by

$$\text{MSE} := \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}(u_{i,j} - u_{0,i,j})^2 = \frac{1}{n^2}\|u - u_0\|_2^2.$$

And the *Peak Signal-to-Noise Ratio*(PSNR) is defined by

$$PSNR := 20 \cdot \log_{10}\left(\frac{\max(u_0)}{\sqrt{\text{MSE}}}\right)$$

where $\max(u_0)$ is the maximum possible pixel value of the image u_0. In our experiment, this is 255.

The clean image u_0 is considered to have the best visual quality. Therefore, we want the resulting image u to be as close to u_0 as possible. Clearly, MSE is inversely related to the quality of the resulting image. On the contrary, PSNR is positively related to it.

We report computational experiments for four images. All the programs are C++ codes running on an desktop computer with Intel Core 2 6400 2.3GHz and 2G memory. The compiler is MinGW/GCC 3.4.5 for windows. And the operating system is Windows Vista Business x32.

Figure 2.2 shows the original clean images. The sizes of our testing images *Peppers, Lena, Boat, Man* are $256 \times 256, 512 \times 512, 512 \times 512, 1024 \times 1024$, respectively. The noisy images are generated by adding Gaussian noise to the clean images using the MATLAB function **randn**, with pseudo-random seed set by **randn('state',0)**.

We tested our *JZ Algorithm*(JZ), *JZ with relaxation*(JZr) and Goldstein and Osher's *GO Algorithm*(GO). Since these algorithms converge to the same

(a) Peppers, 256×256

(b) Lena, 512×512

(c) Boat, 512×512

(d) Man, 1024×1024

Figure 2.2: The clean images for our test problems.

solution, they generate the same clean images. The comparison is focused on the efficiency of these schemes. The stopping criterion is $\|u^k - u^{k-1}\|_2 \leq 10^{-3}n^2$, where n is the size of the image. We report the number of iterations and the CPU time.

The parameter μ is adjusted for each image and each noise level to get optimal quality. There is no algorithm reported so far for an automatic estimation

of the optimal μ. A rough estimation can be made by

$$\mu = 2.14/\sigma - 0.02$$

for most of images, where σ is assumed to be given. In [33], Wang and Shang proposed a method to estimate σ by

$$\sigma = 1.0482 \, \underset{i,j}{median}(|(\nabla u)_{i,j}|).$$

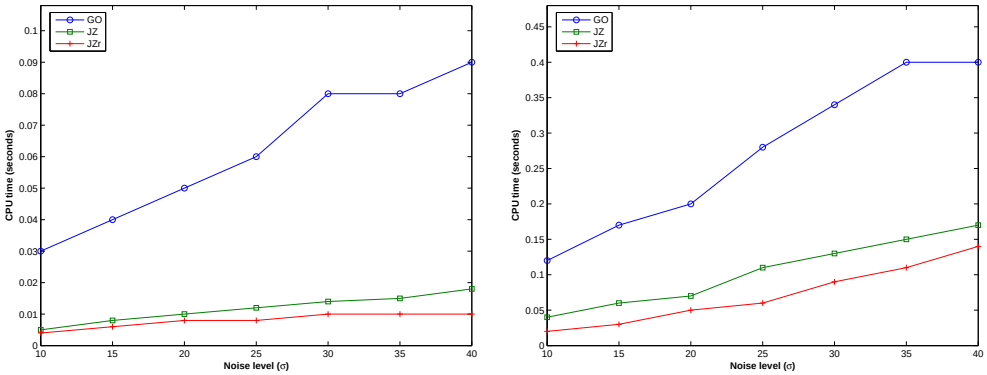
This estimation is accurate for most natural images.

Figure 2.3 shows the numerical performance of the *GO Algorithm* and *JZ algorithm* on these images. We tested each image with different noise levels. Each algorithm iterates until the optimal PSNR is reached in the sense that

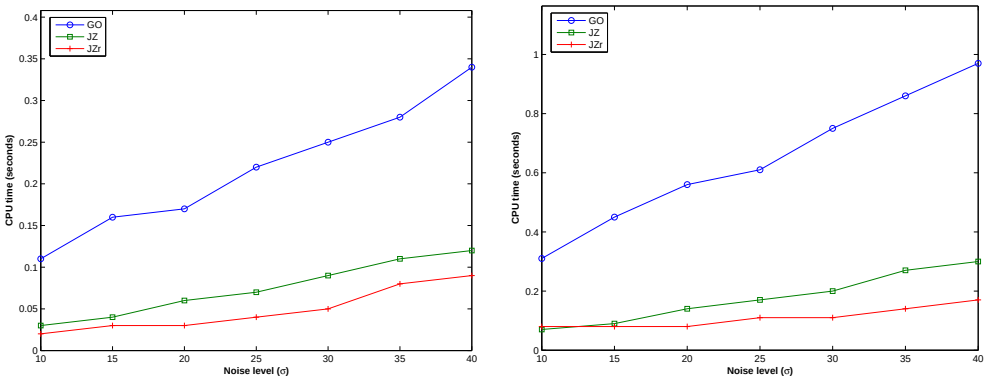
$$PSNR_{optimal} - PSNR(u^k) < 0.02 \; .$$

The x-axis represents the noise level ranging from $\sigma = 10$ to 40. And the y-axis represents the CPU time. It is clear that our algorithms are much faster than the GO Algorithm in each noise level.

Table 2.1 to Table 2.4 show the details of performance on each image. σ is the noise level. The parameter μ is inversely related to σ. Since all the algorithms solve the same model, they should produce images with the same PSNR. The number of iterations and the CPU time of each algorithm increase as the noise level grows.



(a) Peppers (b) Lena



(c) Boat (d) Man

Figure 2.3: Comparison results on CPU time of algorithms.

σ		10	15	20	25	30	35	40
μ		0.18	0.115	0.082	0.064	0.052	0.044	0.04
PSNR		33.47	31.26	29.79	28.59	27.60	26.80	26.16
GO	Iter	8	11	14	16	18	20	22
	time	0.03	0.04	0.05	0.06	0.08	0.08	0.09
JZ	Iter	3	5	6	7	8	10	11
	time	0.005	0.008	0.01	0.012	0.014	0.015	0.018
JZr	Iter	2	3	4	4	5	6	6
	time	0.004	0.006	0.008	0.008	0.01	0.01	0.01

Table 2.1: Comparison results on number of iterations and CPU time of image *Peppers*, 256×256.

σ		10	15	20	25	30	35	40
μ		0.18	0.11	0.075	0.057	0.047	0.04	0.034
PSNR		34.36	32.46	31.17	30.15	29.45	28.82	28.20
GO	Iter	9	12	15	20	25	30	30
	time	0.12	0.17	0.2	0.28	0.34	0.4	0.4
JZ	Iter	6	10	12	16	19	21	23
	time	0.04	0.06	0.08	0.13	0.15	0.17	0.19
JZr	Iter	3	5	6	9	12	14	16
	time	0.02	0.03	0.05	0.06	0.09	0.11	0.14

Table 2.2: Comparison results on number of iterations and CPU time of image *Lena*, 512×512.

σ		10	15	20	25	30	35	40
μ		0.21	0.13	0.087	0.067	0.053	0.045	0.038
PSNR		32.47	30.53	29.14	28.14	27.33	26.69	26.13
GO	Iter	7	10	13	16	18	21	25
	time	0.11	0.16	0.17	0.22	0.25	0.28	0.34
JZ	Iter	4	6	8	12	14	16	18
	time	0.03	0.04	0.06	0.07	0.09	0.11	0.12
JZr	Iter	3	4	4	5	7	10	12
	time	0.02	0.03	0.03	0.04	0.05	0.08	0.09

Table 2.3: Comparison results on number of iterations and CPU time of image *Boat*, 512×512.

σ		10	15	20	25	30	35	40
μ		0.22	0.13	0.09	0.07	0.052	0.045	0.038
PSNR		32.83	30.80	29.40	28.28	27.37	26.62	25.90
GO	Iter	6	9	11	12	15	17	19
	time	0.31	0.45	0.56	0.61	0.75	0.86	0.97
JZ	Iter	3	4	6	7	8	11	12
	time	0.07	0.09	0.14	0.17	0.2	0.27	0.30
JZr	Iter	3	3	3	4	4	5	6
	time	0.08	0.08	0.08	0.11	0.11	0.14	0.17

Table 2.4: Comparison results on number of iterations and CPU time of image *Man*, 1024×1024.

Chapter 3

Denoising Models based on High-Order Difference Schemes

3.1 High-order Difference Schemes in Image Denoising

As we stated in Chapter 1, sometimes the ROF model does not have ideal performance in visual quality since it produces staircase effect. One good way to solve this problem is to use higher-order differential terms instead of, or in addition to, the total variation term $\int_{u\in\Omega} |\nabla u|$. The resulting model was originally proposed by Lysaker, Lundervold and Tai [22, 23], which is called the *LLT Model*.

The numerical computation has developed fast for solving this model. The JZ Algorithm can be naturally adopted for this kind of problems.

In this chapter, we consider the discretized model (1.26):

$$u^* = \arg\min_{u\in\mathbb{R}^{n^2}} \frac{1}{\mu}\big(\|\nabla_x u\|_1 + \|\nabla_y u\|_1\big) + \frac{1}{\nu}\big(\|\Delta_x u\|_1 + \|\Delta_y u\|_1\big) + \frac{1}{2}\|u-f\|_2^2 .$$

We will extend our fast algorithm to solve the discretized model. Moreover, we apply the relaxing technique to boost the iteration scheme, making it extremely efficient compared to other algorithms.

3.2 An Extension of Our Algorithm to High-order Difference

We can easily extend the JZ Algorithm and adapt it to the high-order model (1.26). For convenience we restate it here:

$$u^* = \arg\min_{u\in\mathbb{R}^{n^2}} \frac{1}{\mu}\big(\|\nabla_x u\|_1 + \|\nabla_y u\|_1\big) + \frac{1}{\nu}\big(\|\Delta_x u\|_1 + \|\Delta_y u\|_1\big) + \frac{1}{2}\|u-f\|_2^2$$

By introducing new variables c_x and c_y, we can split $\Delta_x u$ and $\Delta_y u$ and take them out of $\|\cdot\|_1$. Hence we have Algorithm 3.1.

Algorithm 3.1 JZ Algorithm Based on High-Order Scheme

Choose γ_1, γ_2 s.t. $8\gamma_1 + 32\gamma_2 < 1$
$\lambda_1 \leftarrow \gamma_1\mu, \lambda_2 \leftarrow \gamma_2\nu$
$b_x^1 \leftarrow 0, b_y^1 \leftarrow 0, c_x^0 \leftarrow 0, c_y^0 \leftarrow 0$ and $u^1 = f$
while $\|u^{k+1} - u^k\| <$ some tolerance **do**
  $b_x^{k+1} = cut(\nabla_x u^k + b_x^k, \frac{1}{\lambda_1})$
  $b_y^{k+1} = cut(\nabla_y u^k + b_y^k, \frac{1}{\lambda_1})$
  $c_x^{k+1} = cut(\Delta_x u^k + c_x^k, \frac{1}{\lambda_2})$
  $c_y^{k+1} = cut(\Delta_y u^k + c_y^k, \frac{1}{\lambda_2})$
  $u^{k+1} = f - \gamma_1(\nabla_x^T b_x^{k+1} + \nabla_y^T b_y^{k+1}) - \gamma_2(\Delta_x^T c_x^{k+1} + \Delta_y^T c_y^{k+1})$
end while

which solves the discrete model (1.26).

In the next two sections, we prove the convergence of the JZ Algorithm 2.4 and 3.1. We will achieve this by proving a more general conclusion. From (1.23, page 17) and (1.26, page 18), the total variation-based anisotropic denoising models can be generalized into the following form

$$u^* = \arg\min_{u\in\mathbb{R}^{n^2}} \frac{1}{2}\|u-f\|_2^2 + \sum_{l=1}^{L}\frac{1}{\mu_l}\|A_l u\|_1 \tag{3.1}$$

where $A_l : \mathbb{R}^n \to \mathbb{R}^n$ are various difference operators such as $\nabla_x, \nabla_y, \Delta_x, \Delta_y$.

Choosing proper γ_l for each A_l and letting $\lambda_l = \lambda_l$, the JZ Algorithm for (3.1) is

$$\begin{cases} b_l^{k+1} = cut(A_l u^k + b_l^k, \dfrac{1}{\lambda_l}) \quad k = 1, \dots L \\ u^{k+1} = f - \displaystyle\sum_{l=1}^{L} \gamma_l A_l^T b_l^{k+1} \end{cases} \tag{3.2}$$

The convergence of the algorithm depends on A_l and γ_l. Suppose $\rho(A_l^T A_l)$ is the spectral radius of $A_l^T A_l$. We have the following conclusion about the convergence of (3.2):

Theorem 3.1. *Suppose*

1. *u^* is the unique solution to (3.1)*
2. *$\sum_{l=1}^{L} \gamma_l \, \rho(A_l^T A_l) < 1$*

Then the iteration scheme (3.2) yields $\lim_{k\to\infty} u^k = u^*$.

Note that the condition $8\gamma_1 + 32\gamma_2 < 1$ of **Algorithm 3.1** is the direct

corollary of assumption 2.

We will prove this theorem in the following three sections.

3.3 Preliminary Results

To investigate the convergence of (3.2), we will show that it could be formulated into another form:

$$\begin{cases} v_l^{k+1} = \arg\min\limits_{v} \ \|v\|_1 + \dfrac{\lambda_l}{2}\|v - A_l u^k - b_l^k\|_2^2, \quad k = 1, 2 \ldots, L \\ b_l^{k+1} = b_l^k + A_l u^k - v_1^{k+1}, \quad l = 1, 2 \ldots, L \\ u^{k+1} = \arg\min\limits_{u} \ \{\dfrac{1}{2}\|B(u - f)\|_2^2 - \langle B^2(u^k - f), u - u^k\rangle \\ \qquad\qquad + \sum\limits_{l=1}^{L} \dfrac{\gamma_l}{2}\|v_l^{k+1} - A_l u\|_2^2\} \end{cases} \tag{3.3}$$

where B is a s.p.d matrix such that $B^2 = I - \sum_{l=1}^{L} \gamma_l A_l^T A_l$. From the second assumption of Theorem 3.1 we know that $\rho(\sum_{l=1}^{L} \gamma_l A_l^T A_l) < 1$ and hence $I - \sum_{l=1}^{L} \gamma_l A_l^T A_l$ is s.p.d. Therefore, there exists a unique s.p.d. matrix B s.t. $B^2 = I - \sum_{l=1}^{L} \gamma_l A_l^T A_l$, i.e. B is well-defined.

While all the algorithms are well-defined, we show that the algorithm (3.3) coincides with (3.2).

Proposition 3.2. *Given the same initial data, then (3.3) and (3.2) generate the same sequences $b_l^k, l = 1, \ldots, L$ and u^k.*

Proof. It is obvious that the first equation of (3.3) is equivalent to

$$v_l^{k+1} = shrink(A_l u^k + b_l^k, \frac{1}{\lambda_l}) \ .$$

Therefore, the second equation in (3.3) yields

$$\begin{aligned} b_l^{k+1} &= (A_l u^k + b_l^k) - shrink(A_l u^k + b_l^k, \frac{1}{\lambda_l}) \\ &= cut(A_l u^k + b_l^k, \frac{1}{\lambda_l}) \ . \end{aligned}$$

Hence $(b_l^k)_k, l = 1, 2, \ldots, L$ are equivalent in the two algorithms.

For $(u^k)_{k=1,2,\ldots}$, we consider the last equation of (3.3). Noting $B^2 = I - \sum_{l=1}^{L} \gamma_l \ A_l^T A_l$, we differentiate the last equation in (3.3) which is

$$\begin{aligned} u^{k+1} = \arg\min_u \ \{ &\frac{1}{2}\|B(u-f)\|_2^2 - \langle B^2(u^k - f), u - u^k \rangle \\ &+ \sum_{l=1}^{L} \ \frac{\lambda_l}{2} \|v_l^{k+1} - A_l u\|_2^2 \} \end{aligned}$$

and obtain

$$\begin{aligned} & B^2(u^{k+1} - f) - B^2(u^k - f) + \textstyle\sum_{l=1}^{L} \gamma_l A_l^T (A_l u^{k+1} - v_l^{k+1}) = 0 \\ \Rightarrow \quad & B^2(u^{k+1} - u^k) = \textstyle\sum_{l=1}^{L} \gamma_l A_l^T (v_l^{k+1} - A_l u^{k+1}) \\ \Rightarrow \quad & (I - \textstyle\sum_{l=1}^{L} \gamma_l A_l^T A_l)(u^{k+1} - u^k) = \textstyle\sum_{l=1}^{L} \gamma_l A_l^T (v_l^{k+1} - A_l u^{k+1}) \\ \Rightarrow \quad & u^{k+1} - u^k = \textstyle\sum_{l=1}^{L} \gamma_l A_l^T (v_l^{k+1} - A_l u^k) \end{aligned} \tag{3.4}$$

The second equation in (3.3) implies

$$v_l^{k+1} - A_l u^k = b_l^k - b_l^{k+1} \ .$$

Plugging it into (3.4), we have

$$u^{j+1} - u^j = \sum_{l=1}^{L} \gamma_l A_l^T (b_l^j - b_l^{j+1}), \quad j = 1, 2, \ldots, k.$$

Adding up from $j = 1$ to k, we get

$$\begin{aligned}
& \textstyle\sum_{j=1}^{k}(u^{j+1} - u^j) = \sum_{j=1}^{k} \sum_{l=1}^{L} \gamma_l \, A_l^T (b_l^j - b_l^{j+1}) \\
\Rightarrow \quad & u^{k+1} - u^1 = \textstyle\sum_{l=1}^{L} \sum_{j=1}^{k} \gamma_l \, A_l^T (b_l^j - b_l^{j+1}) \\
\Rightarrow \quad & u^{k+1} - f = -\textstyle\sum_{l=1}^{L} \gamma_l A_l^T b_l^{k+1} \\
\Rightarrow \quad & u^{k+1} = f - \textstyle\sum_{l=1}^{L} \gamma_l A_l^T b_l^{k+1}
\end{aligned}$$

which equates the last equation of (3.2). Hence, we conclude that (3.2) and (3.3) generate the same result. □

In the next section we will prove the convergence of (3.3). By Proposition 3.2, the convergence of (3.2) follows.

To prove the theorem, we introduce some notations:

- $J(v) = \|v\|_l$
- ∂J is the subdifferential of J
- $D_J^p(u, v) = J(u) - J(v) - \langle p, u - v \rangle$ be the Bregman distance between u and v associated with J, where $p \in \partial J(v)$
- $\Upsilon^k = \sum_{l=1}^{L} \gamma_l \|v_l^{k+1} - A_l u^k\|_2^2$

With these notations, the first equation of (3.3) becomes

$$v_l^{k+1} = \arg\min_v \{J(v) + \frac{\lambda_l}{2} \|v - A_l u^k - b_l^k\|\}$$

which implies

$$\begin{aligned} & 0 \in \partial J(v_l^{k+1}) + \lambda_l(v_l^{k+1} - A_l u^k - b_l^k). \\ \Rightarrow \quad & \lambda_l(b_l^k + A_l u^k - v_l^{k+1}) \in \partial J(v_l^{k+1}) \\ \Rightarrow \quad & \lambda_l b_l^{k+1} \in \partial J(v_l^{k+1}) \\ \Rightarrow \quad & \lambda_l b_l^k \in \partial J(v_l^k) \end{aligned}$$

Let $p_l^k = \lambda_l b_l^k$, then $p_l^k \in \partial J(v_l^k)$. By the definition of Bregman distance, we deduced a useful relation

$$D_J^{p_l^k}(v, v_l^k) = \|v\| - \|v_l^k\| - \langle p_l^k, v - v_l^k \rangle \geq 0, \ v \in \mathbb{R}^n$$

and especially when $v = v_l^{k+1}$,

$$D_J^{p_l^k}(v^{k+1}, v_l^k) = \|v^{k+1}\| - \|v_l^k\| - \langle p_l^k, v^{k+1} - v_l^k \rangle \geq 0.$$

This is true for all $l = 1, 2, \ldots, L$

Lemma 3.3 (Boundedness)**.** *$(u^k), (v_l^k)_k, (b_l^k)_k, (p_l^k)_k, l = 1, 2, \cdots, L$ are all bounded sequences.*

Proof. By the definition of *cut* function, it is clear that $\|b_l^k\|_\infty = \sup_k b_l^k \leq \frac{1}{\lambda_l}, \quad \forall\ l = 1, 2, \ldots, L.$ Therefore, for every l, $(b_l^k)_k$ is a bounded sequence. And since $p_l^k = \lambda_l b_l^k$ we conclude that $(p_l^k)_k$ are also bounded.
The second equation in (3.3) implies

$$v_l^{k+1} = b_l^k + A_l u^k - b_l^{k+1}.$$

Thus $(v_l^k)_k$ is bounded for each l.

Since $u^{k+1} = f - \sum_{l=1}^{L} \gamma_l A_l^T b_l^k$, $(u^k)_k$ is also bounded.

□

Lemma 3.4. $\lim_{k\to\infty} \|u^{k+1} - u^k\|_2^2 = 0$. *Moreover,* $\Upsilon^k \geq \Upsilon^{k+1}$ *for all* $k \in \ltimes$.

Proof. We first show $\|v_l^{k+1} - A_l u^k\|_2^2 \leq \|v_l^k - A_l u^k\|_2^2$.

By the first equation of (3.3), we have

$$\begin{aligned} v_l^{k+1} &= \arg\min_v \{\|v\|_l + \tfrac{\lambda_l}{2}\|v - A_l u^k - b_l^k\|_2^2\} \\ &= \arg\min_v \{\|v\|_l + \tfrac{\lambda_l}{2}\|v - A_l u^k\|_2^2 \ + \tfrac{\lambda_l}{2}\|b_l^k\|_2^2 - \lambda_l\langle v - A_l u^k, b_l^k\rangle\} \\ &= \arg\min_v \{\|v\|_l + \tfrac{\lambda_l}{2}\|v - A_l u^k\|_2^2 \ - \langle\lambda_l b_l^k, v - v_l^k\rangle\} \\ &= \arg\min_v \{\|v\|_l + \tfrac{\lambda_l}{2}\|v - A_l u^k\|_2^2 \ - \langle p_l^k, v - v_l^k\rangle\} \ . \end{aligned}$$

Therefore, v_l^{k+1} is the minimizer of $\|v\|_l + \frac{\lambda_l}{2}\|v - A_l u^k\|_2^2 - \langle p_l^k, v - v_l^k\rangle$. It follows that it is smaller substituting $v = v_l^{k+1}$ than substituting $v = v_l^k$:

$$\begin{aligned} & \|v_l^{k+1}\|_l + \tfrac{\lambda_l}{2}\|v_l^{k+1} - A_l u^k\|_2^2 - \langle p_l^k, v_l^k - v_l^k\rangle \\ & \leq \|v_l^k\|_l + \frac{\lambda_l}{2}\|v_l^k - A_l u^k\|_2^2 \\ \Rightarrow \quad & 0 \leq D_J^{p_l^k} \leq \frac{\lambda_l}{2}\|v_l^k - A_l u^k\|_2^2 - \frac{\lambda_l}{2}\|v_l^{k+1} - A_l u^k\|_2^2 \\ \Rightarrow \quad & \|v_l^{k+1} - A_l u^k\|_2^2 \leq \|v_l^k - A_l u^k\|_2^2, \ l = 1, 2, \ldots, L \ . \end{aligned} \tag{3.5}$$

Then we show $\Upsilon^{k+1} \leq \Upsilon^k$. Recall the last step of the iteration (3.3)

$$\begin{aligned} u^{k+1} = \quad & \arg\min_u \ \left\{ \ \tfrac{1}{2}\|B(u-f)\|_2^2 - \langle B^2(u^k - f), u - u^k\rangle \right. \\ & \left. + \textstyle\sum_{l=1}^L \ \frac{\gamma_l}{2}\|v_l^{k+1} - A_l u\|_2^2 \ \right\} . \end{aligned}$$

Since u^{k+1} is the minimizer, it is smaller to substitute $u = u^{k+1}$ than $u = u^k$:

$$\begin{aligned} & \tfrac{1}{2}\|B(u^{k+1}-f)\|_2^2 - \langle D^2(u^k - f), u^{k+1} - u^k\rangle + \textstyle\sum_{l=1}^L \frac{\gamma_l}{2}\|v_l^{k+1} - A_l u^{k+1}\|_2^2 \\ & \leq \tfrac{1}{2}\|B(u^k-f)\|_2^2 + \textstyle\sum_{l=1}^L \frac{\gamma_l}{2}\|v_l^{k+1} - A_l u^k\|_2^2 \\ \Rightarrow \quad & \tfrac{1}{2}\|B(u^{k+1}-f)\|_2^2 - \langle D^2(u^k - f), u^{k+1} - u^k\rangle - \tfrac{1}{2}\|B(u^k-f)\|_2^2 \\ & \leq \textstyle\sum_{l=1}^L \frac{\gamma_l}{2}\|v_l^{k+1} - A_l u^k\|_2^2 - \sum_{l=1}^L \frac{\gamma_l}{2}\|v_l^{k+1} - A_l u^{k+1}\|_2^2 \\ \Rightarrow \quad & \tfrac{1}{2}\|B(u^{k+1}-u^k)\|_2^2 \\ & \leq \textstyle\sum_{l=1}^L \frac{\gamma_l}{2}\|v_l^{k+1} - A_l u^k\|_2^2 - \sum_{l=1}^L \frac{\gamma_l}{2}\|v_l^{k+1} - A_l u^{k+1}\|_2^2 . \end{aligned}$$

Applying (3.5), we have

$$\begin{aligned} & \tfrac{1}{2}\|B(u^{k+1}-u^k)\|_2^2 \\ & \leq \textstyle\sum_{l=1}^L \frac{\gamma_l}{2}(\|v_l^{k+1} - A_l u^k\|_2^2 - \|v_l^{k+1} - A_l u^{k+1}\|_2^2) \\ & \leq \textstyle\sum_{l=1}^L \frac{\gamma_l}{2}(\|v_l^{k+1} - A_l u^k\|_2^2 - \|v_l^{k+2} - A_l u^{k+1}\|_2^2) \\ & = \Upsilon^k - \Upsilon^{k+1} \\ \Rightarrow \quad & \Upsilon^{k+1} \leq \Upsilon^k - \tfrac{1}{2}\|B(u^{k+1}-u^k)\|_2^2 \leq \Upsilon^k \end{aligned}$$

Therefore $\Upsilon^{k+1} \leq \Upsilon^k$. On the other hand, it is clear that $\Upsilon^k \geq 0$. Hence $\lim_{k\to\infty} \Upsilon^k$ exists. It follows that

$$\lim_{k\to\infty} \frac{1}{2}\|B(u^{k+1}-u^k)\|_2^2 = \lim_{k\to\infty} \Upsilon^k - \Upsilon^{k+1} = 0.$$

Hence, we deduce that

$$\lim_{k\to\infty} \|u^{k+1} - u^k\|_2^2 = 0 \ .$$

□

Lemma 3.5 ($v^* = \nabla u^*$)**.** $\lim_{k\to\infty} \Upsilon^k = 0$. *It follows that* $\lim_{k\to\infty} v_l^k - A_l u^k = 0$, *for each* $l = 1, 2, \ldots, L$.

Proof. Noting $v_l^{k+1} - A_l u_l^k = b_l^k - b_l^{k+1}$, we consider $p_l^k - p_l^{k+1}$ firstly.

For any $v \in \mathbb{R}^n$, we have

$$\begin{aligned} & D_J^{p_l^{j+1}}(v, v_l^{j+1}) - D_J^{p_l^j}(v, v_l^j) + D_J^{p_l^j}(v_l^{j+1}, v_l^j) \\ = \ & J(v) - J(v_l^{j+1}) - \langle p_l^{j+1}, v - v_l^{j+1}\rangle - J(v) + J(v_l^j) + \langle p_l^j, v - v_l^j\rangle \\ & + J(v_l^{j+1}) - J(v_l^j) - \langle p_l^j, v_l^{j+1} - v_l^j\rangle \\ = \ & \langle p_l^j - p_l^{j+1}, v - v_l^{j+1}\rangle \ . \end{aligned}$$

This implies

$$D_J^{p_l^{j+1}}(v, v_l^{j+1}) - D_J^{p_l^j}(v, v_l^j) \le \langle p_l^j - p_l^{j+1}, v - v_l^{j+1}\rangle \ .$$

On the other hand, consider the function

$$M(v) = \frac{\lambda_l}{2}\|v - A_l u^{j+1}\|_2^2$$

we have

$$
\begin{aligned}
& p_l^j - p_l^{j+1} = \lambda_l(b_l^j - b_l^{j+1}) = \lambda_l(v_l^{j+1} - A_l u^j) = \partial M(v_l^{j+1}) \\
\Rightarrow\ & D_M^{p_l^j - p_l^{j+1}}(v) = M(v) - M(v_l^{j+1}) - \langle p_l^j - p_l^{j+1}, v - v_l^{j+1}\rangle > 0 \\
\Rightarrow\ & M(v) - M(v_l^{j+1}) \geq \langle p_l^j - p_l^{j+1}, v - v_l^{j+1}\rangle \geq D_J^{p_l^{j+1}}(v, v_l^{j+1}) - D_J^{p_l^j}(v, v_l^j) \\
\Rightarrow\ & \tfrac{\lambda_l}{2}\|v - A_l u^j\|_2^2 - \tfrac{\lambda_l}{2}\|v_l^{j+1} - A_l u^j\|_2^2 \\
& \geq \|v_l^j\|_l - \|v_l^{j+1}\|_l + \langle p_l^j, v - v_l^j\rangle - \langle p_l^{j+1}, v - v_l^{j+1}\rangle
\end{aligned}
$$

Choosing $v = A_l u^j$, we come up with

$$
\begin{aligned}
& \tfrac{\lambda_l}{2}\|v_l^{j+1} - A_l u^j\|_2^2 \\
\leq\ & \|v_l^{j+1}\|_l - \|v_l^j\|_l + \langle p_l^{j+1}, A_l u^j - v_l^{j+1}\rangle - \langle p_l^j, A_l u^j - v_l^j\rangle \\
=\ & (\|v_l^{j+1}\|_l - \|v_l^j\|_l) + (\langle p_l^{j+1}, A_l u^j - v_l^{j+1}\rangle - \langle p_l^j, A_l u^{j-1} - v_l^j\rangle) - \langle p_l^j, A_l u^j - A_l u^{j-1}\rangle\ .
\end{aligned}
$$

Summing j from m to $k-1$, we obtain

$$
\begin{aligned}
& \tfrac{\lambda_l}{2}\textstyle\sum_{j=m}^{k-1}\|v_l^{j+1} - A_l u^j\|_2^2 \\
\leq\ & \|v_l^k\|_l - \|v_l^m\|_l + \langle p_l^k, A_l u^{k-1} - v_l^k\rangle - \langle p_l^m, A_l u^{m-1} - v_l^m\rangle] - \textstyle\sum_{j=m}^{k-1}\langle p_l^j, A_l u^j - A_l u^{j-1}\rangle\ .
\end{aligned}
$$

Since the sequences $(u^j)_j, (v_l^j)_j, (p_l^j)_j$ are bounded, there exists positive constants $C_{l,1}l$ and $C_{l,2}$ independent of k and m such that

$$
\frac{\lambda_l}{2}\sum_{j=m}^{k-1}\|v_l^{j+1} - A_l u^j\|_2^2 \leq C_{l,1} + C_{l,2}(k-m)\eta_m
$$

where $\eta_m = \sup_{j\geq m}\|u^{j+1} - u^j\|_2 \overset{m\to\infty}{\longrightarrow} 0$. Which is true for all $l = 1, 2, \ldots, L$. Adding all the inequalities up for $l = 1, 2, \ldots, L$ yields

$$
\sum_{j=m}^{k-1}\Upsilon^j \leq C_1 + C_2(k-m)\eta_m
$$

for some constants C_1 and C_2.

By lemma 3.4, we know that (Υ^j) is a decreasing sequence. Hence, we have

$$\begin{aligned} & (k-m)\Upsilon^k \leq C_1 + C_2(k-m)\eta_m \\ \Rightarrow \quad & \Upsilon^k \leq \tfrac{C_1}{k-m} + C_2\eta_m \ \text{ for any } 1 < m < k-1 \\ \Rightarrow \quad & \Upsilon^k \leq \tfrac{C_1}{m} + C_2\eta_m \ \text{ for } m = \lfloor k/2 \rfloor \end{aligned}$$

Therefore, $\lim_{k\to\infty} \Upsilon^k = 0$. It follows that $\lim_{k\to\infty} v_l^k - A_l u^k = 0$. □

3.4 Convergence Analysis of the Algorithm

Here we prove the main theorem. Assume $u^* = \arg\min_{u\in\mathbb{R}^{n^2}} \frac{1}{2}\|u-f\|_2^2 + \sum_{l=1}^L \frac{1}{\mu_l}\|A_l u\|_1$ is the unique solution. We'll show $\lim_{k\to\infty} u^k = u^*$.

Proof. Let $F(u) = \frac{1}{2}\|u-f\|_2^2$. Then $\partial F(u^{k+1}) = u^{k+1} - f$. $\forall w \in \mathbb{R}^n$, we have

$$F(u^{k+1}+w) - F(u^{k+1}) + \langle u^{k+1} - f, w\rangle = D_F^{u^{k+1}-f}(u^{k+1}+w, u^{k+1}) \geq 0.$$

By the last equation of (3.2) we know $u^{k+1} - f = -\sum_{l=1}^L \gamma_l A_l^T b_l^k$. Hence,

$$F(u^{k+1}+w) - F(u^{k+1}) + \sum_{l=1}^L \langle \gamma_l b_l^k, A_l w\rangle \geq 0. \tag{3.6}$$

On the other hand, recall $p_l^k = \lambda_l b_l^k \in \partial J(v_l^k)$. It follows that

$$\begin{aligned} & J(v_l^k + A_l w) - J(v_l^k) - \langle p_l^k, A_l w\rangle \geq 0 \\ \Rightarrow \quad & \|v_l^k + A_l w\|_l - \|v_l^k\|_l - \langle \lambda_l b_l^k, A_l w\rangle \geq 0 \\ \Rightarrow \quad & \langle \gamma_l b_l^k, A_l w\rangle \leq \tfrac{1}{\mu_l}\|v_l^k + A_l w\|_l - \tfrac{1}{\mu_l}\|v_l^k\|_l, \end{aligned}$$

which is true for all $l = 1, 2, \ldots, L$.

Substituting these inequalities into (3.6), we obtain

$$\begin{aligned} & \sum_{l=1}^{L} \frac{1}{\mu_l} \|v_l^k\|_l + F(u^{k+1}) \\ \leq \; & \sum_{l=1}^{L} \frac{1}{\mu_l} \|v_l^k + A_l w\|_l + F(u^{k+1} + w), \quad \forall w \in \mathbb{R}^n. \end{aligned} \tag{3.7}$$

(u^k) is a bounded sequence, thus it has convergent subsequences. Picking up arbitrary convergent subsequent $(u^{k_j})_{j=1,2\ldots}$, we denote $\tilde{u} = \lim_{j\to\infty} u^{k_j}$. Since $\lim_{k\to\infty}(u^{k+1} - u^k) = 0$, we also have $\lim_{j\to\infty} u^{k_j+1} = \tilde{u}$.

By Lemma 3.5, $\lim_{j\to\infty} v_l^{k_j} = A_l\tilde{u}, \quad l = 1, 2, \ldots, L$. Therefore, replacing u^k in (3.7) by u^{k_j} and letting $j \to \infty$, we obtain

$$\begin{aligned} & \sum_{l=1}^{L} \frac{1}{\mu_l} \|A_l\tilde{u}\|_l + F(\tilde{u}) \\ \leq \; & \sum_{l=1}^{L} \frac{1}{\mu_l} \|A_l(\tilde{u} + w)\|_l + F(\tilde{u} + w), \; \forall w \in \mathbb{R}^n. \end{aligned} \tag{3.8}$$

Hence, $\tilde{u} = \arg\min_u \{\sum_{l=1}^{L} \frac{1}{\mu_l} \|A_l u\|_l + F(u)\} = u^*$.
Since $\lim_{j\to\infty} u^{k_j} = \tilde{u} = u^*$ is true for any convergent subsequence u^{k_j}, we conclude that

$$\lim_{k\to\infty} u^k = u^* \; .$$

□

3.5 Relaxation Technique and Numerical Results

The restriction $8\gamma_1 + 32\gamma_2 < 1$ makes the Algorithm 3.1 converge very slow. In numerical experiments, 200 or more iterations are needed for satisfactory convergence, resulting in 5 seconds or more CPU time on an average computer.

To accelerate the calculation, we apply the relaxation technique as we did in Chapter 2. Recall the general form of JZ iteration scheme (3.2):

$$\begin{cases} b_l^{k+1} = cut(A_l u^k + b_l^k, \dfrac{1}{\lambda_l}) \quad k = 1, \dots L \\ u^{k+1} = f - \displaystyle\sum_{l=1}^{L} \gamma_l A_l^T b_l^{k+1}. \end{cases}$$

Choosing proper $0 < t < 1$, we rewrite the second equation as

$$u^{k+1} = (1-t)u^k + t\big(f - \sum_{l=1}^{L} \gamma_l A_l^T b_l^{k+1}\big).$$

With the coefficient t, the restriction $\sum_{l=1}^{L} \gamma_l \rho(A_l^T A_l) < 1$ is relaxed. Algorithm 3.1 then becomes

Algorithm 3.2 The general form of JZ algorithm with Relaxation

Choose $\gamma_1 = 0.5, \gamma_2 = 0.5$, and $t = 0.1$
$\lambda_1 \leftarrow \gamma_1 \mu, \lambda_2 \leftarrow \gamma_2 \nu$
$b_x^1 \leftarrow 0, b_y^1 \leftarrow 0, c_x^0 \leftarrow 0, c_y^0 \leftarrow 0$ and $u^1 = f$
while $\|u^{k+1} - u^k\| <$ some tolerance **do**
  $b_x^{k+1} = cut(\nabla_x u^k + b_x^k, \frac{1}{\lambda_1})$
  $b_y^{k+1} = cut(\nabla_y u^k + b_y^k, \frac{1}{\lambda_1})$
  $c_x^{k+1} = cut(\Delta_x u^k + c_x^k, \frac{1}{\lambda_2})$
  $c_y^{k+1} = cut(\Delta_y u^k + c_y^k, \frac{1}{\lambda_2})$
  $u^{k+1} = (1-t)u^k + t\big[f - \gamma_1(\nabla_x^T b_x^{k+1} + \nabla_y^T b_y^{k+1}) - \gamma_2(\Delta_x^T c_x^{k+1} + \Delta_y^T c_y^{k+1})\big]$
end while

We will report the details of relaxation at the end of this section.

We report the results of computational experiments. All the programs are C++ codes running on an desktop computer with Intel Core 2 6400 2.13GHz and 2G memory. The compiler is MinGW/GCC 3.4.5 for windows. And the operating system is Windows Vista Business x32. We use the same images

and the same noise levels as the experiments in the previous Chapter.

We first report the quality of the images generated by our model. Figure 3.1 shows the comparison of visual quality between the ROF model and the higher-order model. We focus on the face region of image *Lena*. It is easy to see that the ROF model produces false edges on smooth face. On the other hand, the high-order model reduces this staircase effect significantly and makes the face of *Lena* look natural.

Figure 3.2 shows the comparison of PSNR between the ROF model and the higher-order model. The x-axis represents the noise level, and the y-axis represents the PSNR. Our model performs consistently better than the ROF model on all images.

The detailed data is shown in Table 3.1. In all levels of noise, our model improves the PSNR by 0.6 to 0.7. This is equivalent to 1/6 less MSE than the ROF model.

σ		10	15	20	25	30	35	40
Peppers	ROF	33.47	31.26	29.79	28.59	27.60	26.80	26.16
	Higher-order	34.15	32.04	30.48	29.29	28.30	27.51	26.86
Lena	ROF	34.36	32.46	31.17	30.15	29.45	28.82	28.20
	Higher-order	35.05	33.21	31.92	30.90	30.13	29.50	28.85
Boat	ROF	32.47	30.53	29.14	28.14	27.33	26.69	26.13
	Higher-order	32.88	30.98	29.61	28.58	27.78	27.12	26.53
Peppers	ROF	32.83	30.80	29.40	28.28	27.37	26.62	25.90
	Higher-order	33.35	31.37	29.97	28.83	27.89	27.10	26.35

Table 3.1: Comparison results on PSNR of the images denoised by the ROF model and the high-order model.

The numerical experiments also reveal the importance of relaxation. Without relaxation, convergence of Algorithm 3.1 is only guaranteed when $8\gamma_1 + 32\gamma_2 < 1$. In numerical experiments, this limitation can be slightly eased and we usually choose $\gamma_1 = \gamma_2 = 0.05$. Table 3.2 reports the number of iterations

(a) Denoised by the ROF model

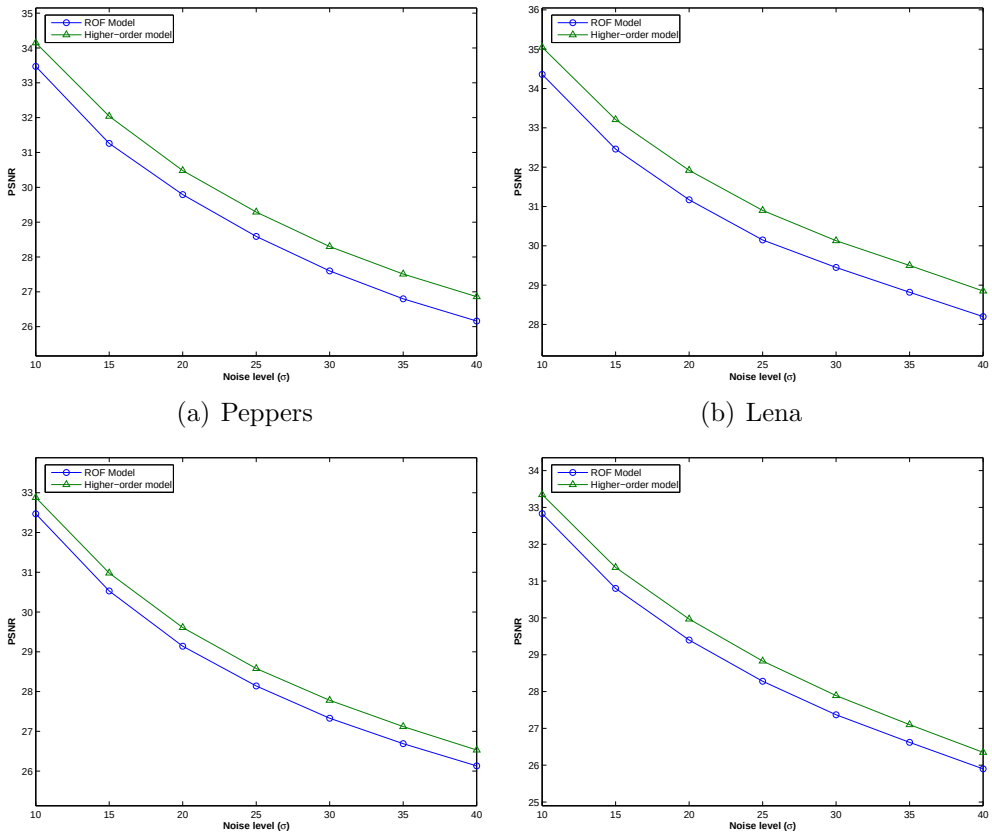
(b) Denoised by the High-order model

(c) The ROF model:face

(d) The High-order model:face

Figure 3.1: Comparison results between the higher-order model and the ROF model on visual quality of *Lena*.

and CPU time needed for three algorithms. ROF stands for the data of JZ algorithm for the anisotropic ROF model, which was previously reported in Chapter 2. Without relaxation, the algorithm needs up to 250 iterations, taking 7.8 seconds. However, with relaxation, we choose $t = 0.18$ and increase those parameters to $\gamma_1 = \gamma_2 = 0.5$. Table 3.2 shows that the number of iterations is reduced significantly with relaxation.



(a) Peppers (b) Lena

(c) Boat (d) Man

Figure 3.2: Comparison results between our higher-order model and the ROF model on PSNR.

σ		10	15	20	25	30	35	40
ROF	Iter	6	10	12	16	19	21	23
	time	0.04	0.06	0.08	0.13	0.15	0.17	0.19
Higher-order	Iter	80	130	160	180	200	220	250
	time	2.6	4.0	5.1	5.7	6.2	6.8	7.8
Higher-order relax	Iter	20	25	30	30	30	35	35
	time	0.7	0.8	1.0	1.0	1.0	1.2	1.2

Table 3.2: Comparison results on number of iterations and CPU time(seconds) of proposed algorithms.

Chapter 4

Combination of Wavelets with Variational Techniques

4.1 Motivation

It is known that the ROF model and the similar total variation based image denoising models are good at recovering piecewise-smooth functions. These models eliminate noise by wiping out oscillating component, or the high frequency component, while preserving the smooth/low frequency component of a function/image. These models work very well for images like *Lena* or *Peppers* (Figure 4.1), as we discussed in the previous chapters. It is clear that these images consist of large pieces of smooth region.

However, these variational models do not fit to the sort of images which contains lots of texture, such as the popular *Barbara* image (Figure 4.2). A texture-rich image u can be decomposed into $u = c + t$, where c stands for the smooth/cartoon parts and t stands for the oscillation/texture parts. Figure 4.3 shows such a decomposition of *Barbara*.

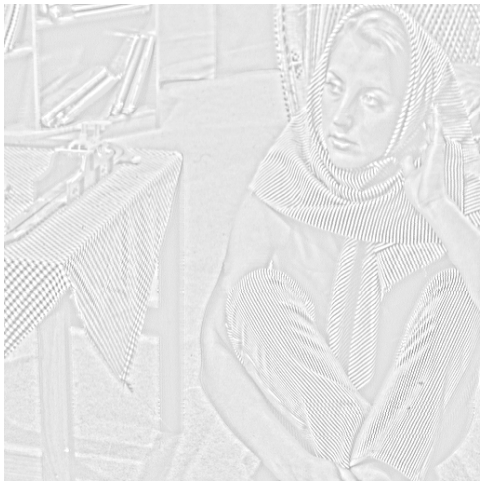
Figure 4.1: The smooth images *Lena* and *Peppers*.

Figure 4.2: The texture-rich image *Barbara*.

Unfortunately, the noise is also highly oscillatory and is also contained in G. Therefore $r = f - u$ contains both noise and texture and they are not

(a) Barbara

(b) Cartoon

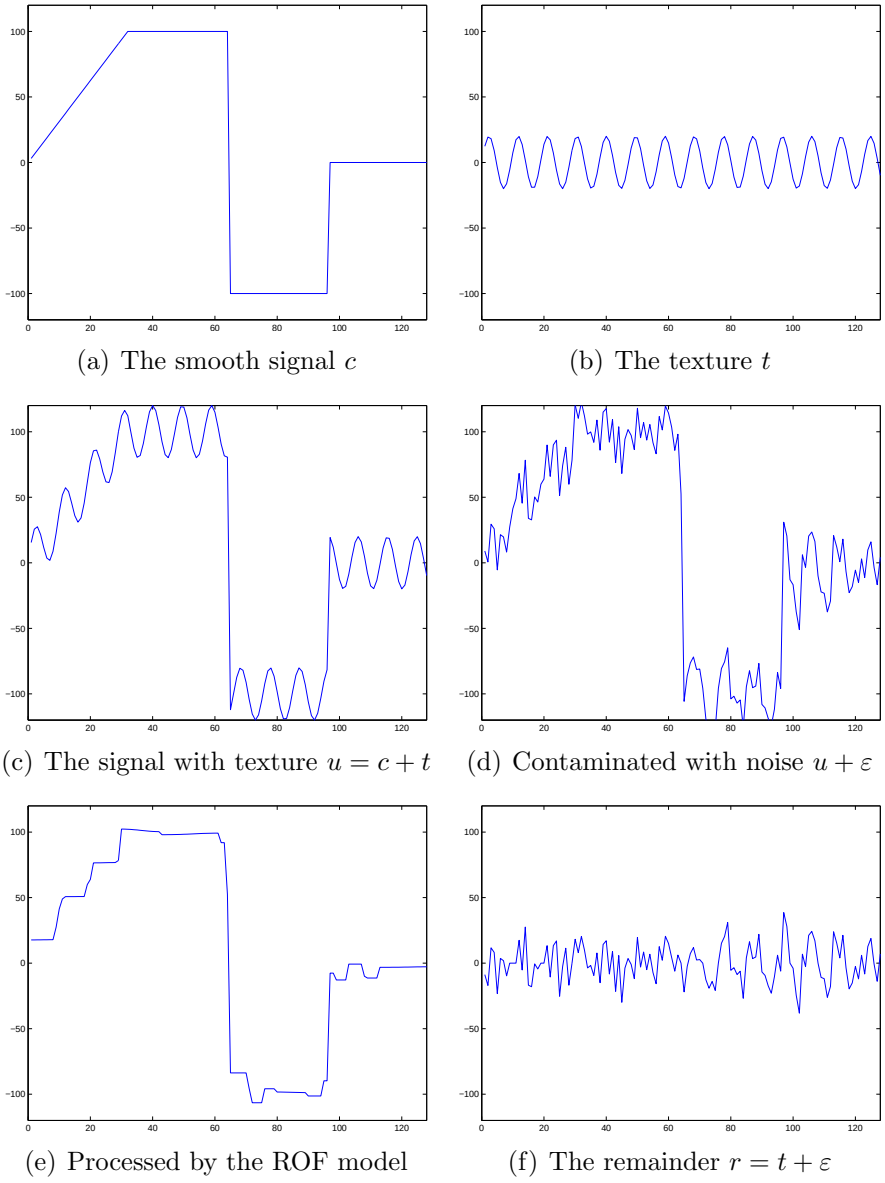


(c) Texture

Figure 4.3: $image = cartoon + texture$.

separated by the ROF model according to the discussion above. The ROF model and its descendants do not work well if $u = c+t$ contains heavy texture t, resulting in an over-smoothed solution $u = c$ and a remainder $r = t+\varepsilon$ with undesirable texture t in it. Generally speaking, we want to recover a textured image $u = c + t$ and leave only the noise in the remainder $r = \varepsilon$, but t and ε usually mix together and can not be separated by the ROF model. This is

shown in Figure 4.4.



(a) The smooth signal c (b) The texture t

(c) The signal with texture $u = c + t$ (d) Contaminated with noise $u + \varepsilon$

(e) Processed by the ROF model (f) The remainder $r = t + \varepsilon$

Figure 4.4: The ROF model does not separate texture from noise.

It is clear that the ROF model does not separate texture and noise, leaving all the oscillations in the remainder (Figure 4.4(f)). Hence, we have to process it further to recover the desired texture. This motivates us to combine the ROF

model and wavelet shrinkage together to improve the quality of denoising for the particular family of images with textures.

4.2 Multiresolution Analysis and Wavelets

We intoduce wavelet and wavelet packet transforms.

As usual, for $1 \leq p < \infty$, $L_p(\mathbb{R})$ denotes the Banach space of all measurable functions f on $\mathbb{R}$ such that $||f||_p < \infty$, where

$$||f||_p := \left(\int_{\mathbb{R}} |f(x)|^p dx \right)^{1/p} \text{ for } 1 \leq p < \infty.$$

Throughout this chapter, we only consider $p = 2$. In this case $L_2(\mathbb{R})$ is a Hilbert space.

A countable set $\{\phi_k\}_{k\in\mathbb{Z}}$ in $L_2(\mathbb{R})$ is said to be a **Riesz sequence** if there exist two positive constants A and B such that the inequalities

$$A \left(\sum_{k\in\mathbb{Z}} |c_k|^2 \right)^{1/2} \leq \left\| \sum_{k\in\mathbb{Z}} c_k \phi_k \right\| \leq B \left(\sum_{k\in\mathbb{Z}} |c_k|^2 \right)^{1/2}$$

hold true for every sequence $\{c_k\}_{k\in\mathbb{Z}}$ in ℓ_2, where A and B are called **Riesz bounds**. Moreover, if the linear span of $\{\phi_k\}_{k\in\mathbb{Z}}$ is dense in $L_2(\mathbb{R})$, then it is a **Riesz Basis** of $L_2(\mathbb{R})$.

We define the wavelets and multiresolution analysis. For a comprehensive introduction to wavelets, we refer readers to [8],[15] and [25].

- Scaling functions and Wavelets

- **Scaling functions**: $\phi \in L_2(\mathbb{R})$ and $\phi_{n,j} := 2^{n/2}\phi(2^n \cdot -j),\ j \in \mathbb{Z}$.
- **Wavelets**: $\psi \in L_2(\mathbb{R})$ and $\psi_{n,j} := 2^{n/2}\psi(2^n \cdot -j),\ j \in \mathbb{Z}$.
- Denote $V_n := span\{\phi_{n,j}\}_{j\in\mathbb{Z}}$ and $W_n := span\{\psi_{n,j}\}_{j\in\mathbb{Z}}$.
- ϕ satisfies the **refinement equation**: $\phi = \sum_{j\in\mathbb{Z}} h(j)\phi(2 \cdot -j)$ for some $h \in \ell_2(\mathbb{Z})$, where h is called the **refinement mask**. Therefore, $V_{n-1} \subset V_n$ and we say that V_n is fine and V_{n-1} is coarse in comparison.
- For ψ, we also have $\psi = \sum_{j\in\mathbb{Z}} g(j)\phi(2 \cdot -j)$ for some $g \in \ell_2(\mathbb{Z})$. That suggests $W_{n-1} \subset V_n$. Some times g is called the wavelet mask.

- Scaling functions and wavelets must satisfy the following conditions to form a **Multiresolution Analysis**
  - $\{\phi_{n,j}\}_{j\in\mathbb{Z}}$ is a Riesz basis of V_n and $\{\psi_{n,j}\}_{j\in\mathbb{Z}}$ is a Riesz basis of W_n.
  - $\bigcup_{n\in\mathbb{Z}} V_n$ is dense in $L_2(\mathbb{R})$ and $\bigcap_{n\in\mathbb{Z}} V_n = \{0\}$.
    In this way, $\{V_n\}_{n\in\mathbb{Z}}$ form a **multiresolution analysis**.
  - In addition to $V_n \supset V_{n-1}$ and $V_n \supset W_{n-1}$, we have $V_n = V_{n-1} + W_{n-1}$.
  - Therefore, $L_2(\mathbb{R}) = V_1 + W_1 + W_2 + W_3 + \ldots$. and $\{\psi_{n,j}\}_{n,j\in\mathbb{Z}}$ form a Riesz basis of $L_2(\mathbb{R})$.

- Orthogonal, Biorthogonal and Semi-orthogonal wavelets:
  If $\{\psi_{n,j}\}_{n,j\in\mathbb{Z}}$ form an orthonormal basis instead of a Riesz basis, then we have *orthogonal wavelets* (see [15]). Otherwise, depending on the orthogonality between adjacent levels we have *biorthogonal wavelets* (see [13]) or *semi-orthogonal wavelets* (see [10]).

Given a function $f = \sum_{j\in\mathbb{Z}} s_{n,j}\ \phi_{n,j} \in V_n$, since $V_n = V_{n-1} + W_{n-1}$, we can rewrite f as

$$f = \sum_{j\in\mathbb{Z}} s_{n-1,j}\ \phi_{n-1,j} + \sum_{j\in\mathbb{Z}} t_{n-1,j}\ \psi_{n-1,j}\ .$$

In this way, the original representation of f in $\{s_{n,j}\}_{j\in\mathbb{Z}}$ is decomposed into the representation in $\{s_{n-1,j}\}_{j\in\mathbb{Z}}$ and $\{t_{n-1,j}\}_{j\in\mathbb{Z}}$ of a lower level. The algorithm we use to derive the coefficients $\{s_{n,j}\}_{j\in\mathbb{Z}}$ and $\{t_{n,j}\}_{j\in\mathbb{Z}}$ is called the *Discrete Wavelet Transform (DWT)*. If we continue to perform DWT on $s_{n-1}, s_{n-2}, \ldots$ recursively while keep $t_{n-1}, t_{n-2}, \ldots$, we get

$$f = \sum_{j\in\mathbb{Z}} s_{1,j}\ \phi_{1,j} + \sum_{1\leqslant k\leqslant n-1}\ \sum_{j\in\mathbb{Z}} t_{k,j}\ \psi_{k,j}\ .$$

In practice, the original data is usually a vector of length 2^n which could be sampled from an image. We denote this original set of data by $\{s_j\}, j =$

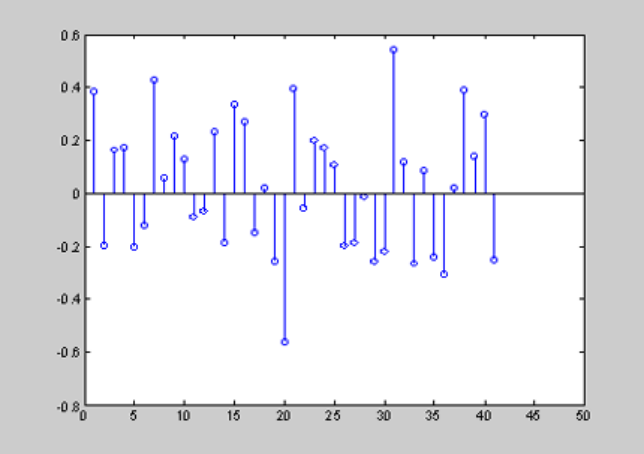


Figure 4.5: A sample of signal.

$1, 2, \ldots, 2^n$. Clearly, it can be represented by a function $f_n = \sum_{j=1}^{n} s_j\ \phi_{n,j} \in V_n$. Then we can apply the discrete wavelet transform to decompose it into coefficients in lower levels. The algorithm of this procedure can be regarded

as applying the filters $\bar{h}$ and high pass filter $\bar{g}$ on $\{s_{n,j}\}$ as follows:

$$s_{m,j} = \sum_k \bar{h}(j-2k)s_{m+1,j},$$
$$t_{m,j} = \sum_k \bar{g}(j-2k)s_{m+1,j},$$

where $\bar{h}, \bar{g}$ are *dual* filters of h, g. This process is described in Figure 4.6.

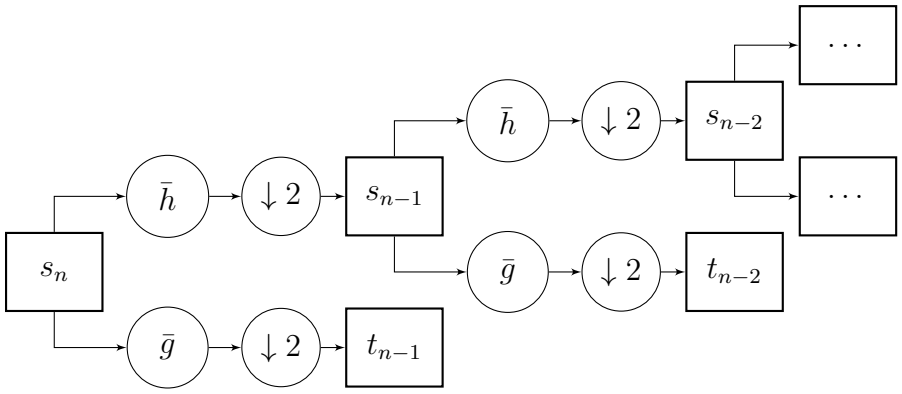


Figure 4.6: The Process of discrete wavelet transform.

This fast algorithm was proposed by Mallat[24]. So far, the most successful wavelet in application, especially in image compression, is the so-called 9/7 wavelet which is biorthogonal (see [13]). The reason it's called 9/7 is that the length of the masks/filters h and g are 9 and 7 respectively.

The wavelet transform is only performed on $s_m, m \in \mathbb{Z}$. If we also decompose t_m with dual filters $\bar{h}$ and $\bar{g}$, we have the *wavelet packet transform* (Figure 4.7). For details of wavelet packets, we refer reader to [25].

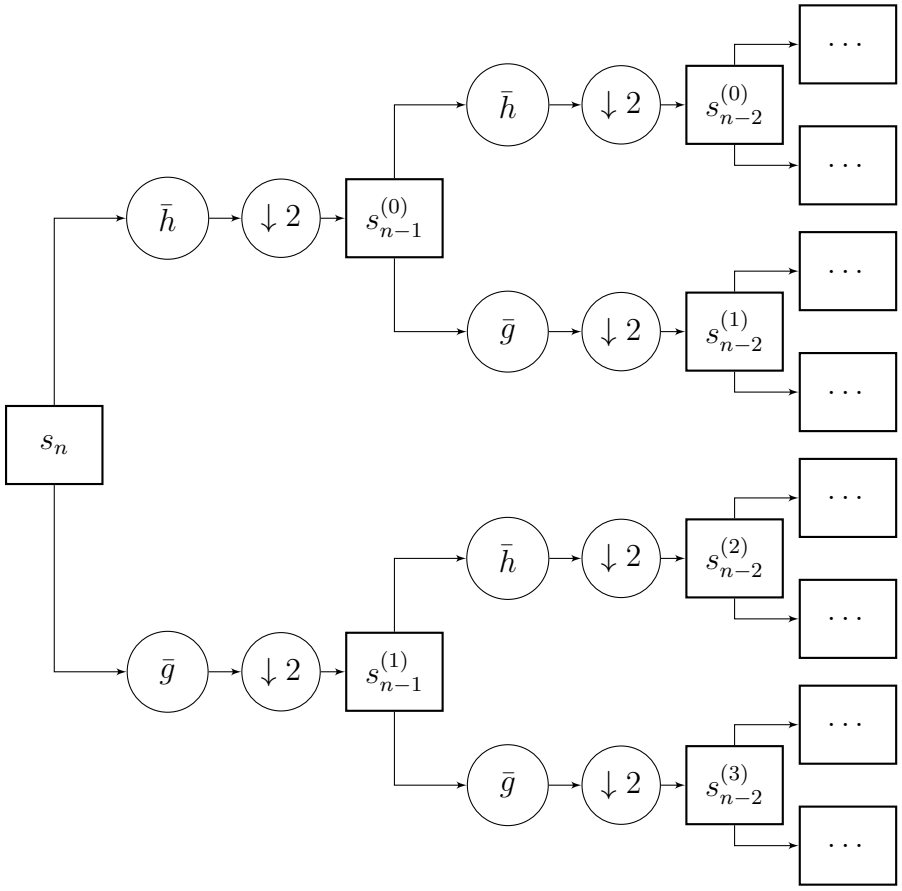


Figure 4.7: The Process of discrete wavelet packet transform (DWPT).

4.3 Discrete Wavelets on Intervals

In application, the vanishing moment of dual wavelets is an important issue. A wavelet function ψ is said to have vanishing moments of order n, if

$$\int_{\mathbb{R}} x^p \psi(x) dx = 0,\ p = 0, 1, 2, \ldots, n-1.$$

And, a filter/mask $g = [g_1, g_2, \ldots, g_l]$ have vanishing moments of order n if

$$\sum_{k=1}^{l} k^p g_k = 0,\ p = 0, 1, 2, \ldots, n-1.$$

A relatively high order of vanishing moments of dual wavelets is desirable for the quality of an image.

The popular 9/7 wavelets has vanishing moments of order 4, which is good enough for most applications. However, when processing data with finite length, we have to truncate the filter or to extend the data. On the boundary of a sequence, the vanishing moment of a wavelet is reduced, as shown in Figure 4.1.

	Order of vanishing moments	Vanishing moments on the boundary
D4	2	1
B2-4	2	1
D6	3	1
9/7	4	1

Table 4.1: List of orders of vanishing moments of different wavelets.

To remedy this, we propose boundary filters for higher order of vanishing moments by constructing wavelets on discrete domain. To avoid ambiguous notation, we denote by v the discrete scaling function and by w the corresponding discrete wavelet. Suppose E is a finite interval of $\mathbb{Z}$. Then we follow these steps to construct discrete wavelets on the interval.

1. Fix a finest level M and let $v_{M,k} = e_k,\ \forall k \in \mathbb{Z}$ where $\{e_k\}_{k\in\mathbb{Z}}$ is the canonical basis of ℓ_2. Namely, $V_{M,k}(j) = \delta_{kj}\ \forall k, j \in \mathbb{Z}$.

2. Find a proper mask h and g and define

$$v_{m-1,k} := \sum_{j\in\mathbb{Z}} h_j v_{m,j} \quad \text{and} \quad w_{m-1,k} = \sum_{j\in\mathbb{Z}} g_j v_{m,j}$$

such that $\{w_{m,k}\}_{k\in\mathbb{Z}}$ is a Riesz sequence in ℓ_2 with the Riesz bounds

being independent of m.

3. Adapt $\{v_{m,k}\}$ and $\{w_{m,k}\}$ to E by constructing boundary elements.

In step 1 and 2, we choose the masks 10/2 and 6/2 which correspond to biorthogonal spline wavelets. The detailed construction can be found in [15]. Our purpose of such construction is that it is easier to construct boundary filters in discrete case. We construct a family of discrete wavelets which fall into two categories: one with the mask of 6 coefficients and the other one with the mask of 10 coefficients. We name them JZ6 and JZ10 respectively.

Recall that

$$v_{m-1,k} = \sum_{j\in\mathbb{Z}} h_j v_{m,j} \text{ and } w_{m-1,k} = \sum_{j\in\mathbb{Z}} g_j v_{m,j} \ .$$

We have the following masks:

- Inner masks JZ6:

  h: $[-1, 1, 8, 8, 1, -1]/8$,

  g: $[1, -1]$.

- Inner masks of JZ10:

  h: $[3, -3, -22, 22, 128, 128, 22, -22, -3, 3]/128$,

  g: $[1, -1]$.

The boundary elements distinguish the discrete wavelets from continuous ones in algorithm. For simplicity we assume that the interval E is $\{1, 2, \ldots, 2^M\}$ where M stands for the finest level. In our construction, only the scaling functions $v_{m,k}$ need boundary elements. The wavelets $w_{m,k}$, however, always keep

the $[1, -1]$ scheme.

JZ6 wavelets are described as follows:

- The scaling functions satisfy the refinement equation $v_{m-1,k} = \sum_{j\in\mathbb{Z}} h_j v_{m,j}$ where the mask $\{h_j\}$ is given by $[-1,\ 1,\ 8,\ 8,\ 1,\ -1]/8$

- There are 2 boundary masks on each side:

  $h^1 :\ [10, 6, 1, -1]/8,$

  $g^2 :\ [-2, 2, 8, 8, 1, -1]/8.$

- For w we have $g = [1, -1]$. Therefore,
  $w_{m-1,k} := v_{m,2k-1} - v_{m,2k-2},\ k = 1, 2, \ldots, 2^{m-1}.$

With the wavelets we constructed, the decomposition algorithm is obtained. Suppose $f = \sum_{k=1}^{2^m} s_{m,k} v_{m,k}$, we have

$$f = \sum_{k=1}^{2^{m-1}} s_{m-1,k} v_{m-1,k} + \sum_{k=1}^{2^{m-1}} t_{m-1,k} w_{m-1,k},$$

where

$$\begin{pmatrix} s_{m-1,1} \\ s_{m-1,2} \\ \vdots \\ s_{m-1,2^{m-1}} \\ \hline t_{m-1,1} \\ t_{m-1,2} \\ \vdots \\ t_{m-1,2^{m-1}} \end{pmatrix} = \frac{1}{16} \left(\begin{array}{cccccccccc} 8 & 8 & & & & & & & & \\ & & 8 & 8 & & & & & & \\ & & & & \cdots & \cdots & & & & \\ & & & & & & & & 8 & 8 \\ \hline 6 & -10 & 2 & 2 & & & & & & \\ -1 & -1 & 8 & -8 & 1 & 1 & & & & \\ & & -1 & -1 & 8 & -8 & 1 & 1 & & \\ & & & & \cdots & \cdots & & & & \\ & & & & & & -2 & -2 & 10 & -6 \end{array} \right) \begin{pmatrix} s_{m,1} \\ s_{m,2} \\ \vdots \\ s_{m,2^m} \end{pmatrix}.$$

The construction of JZ10 wavelets is similar:

- The mask of the inner scaling function is

  $[\frac{3}{128},\ -\frac{3}{128},\ -\frac{11}{64},\ \frac{11}{64},\ 1, 1,\ \frac{11}{64},\ -\frac{11}{64},\ -\frac{3}{128},\ \frac{3}{128}]$.

- We have 4 boundary masks on each side. They are:

  $h^1:\ [106, 96, 16, -16, -3, 3]/128,$

  $h^2:\ [-32, 32, 128, 128, 22, -22, -3, 3]/128,$

  $h^3:\ [-16, 16, 128, 128, 22, -22, -3, 3]/128,$

  $h^4:\ [-22, 22, 128, 128, 22, -22, -3, 3]/128.$

- The wavelets are defined in the same way as JZ6.

  $g = [1, -1]$ and $w_{m-1,k} := v_{m,2k-1} - v_{m,2k-2},\ k = 1, 2, \ldots, 2^{m-1}$.

Suppose $f = \sum_{k=1}^{2^m} s_{m,k} v_{m,k}$, we have the following decomposition:

$$f = \sum_{k=1}^{2^{m-1}} s_{m-1,k} v_{m-1,k} + \sum_{k=1}^{2^{m-1}} t_{m-1,k} w_{m-1,k}.$$

The algorithm can be described as

$$\begin{pmatrix} s_{m-1,1} \\ s_{m-1,2} \\ \vdots \\ s_{m-1,2^{m-1}} \\ \hline t_{m-1,1} \\ t_{m-1,2} \\ \vdots \\ t_{m-1,2^{m-1}} \end{pmatrix} = \frac{1}{256} A \begin{pmatrix} s_{m,1} \\ s_{m,2} \\ \vdots \\ s_{m,2^m} \end{pmatrix},$$

where

$$A = \left(\begin{array}{cccccccccccccc} 128 & 128 & & & & & & & & & & & & \\ & & 128 & 128 & & & & & & & & & & \\ & & & & \cdots & \cdots & & & & & & & & \\ & & & & & & & & & & & & 128 & 128 \\ \hline 96 & -160 & 32 & 32 & & & & & & & & & & \\ -16 & -16 & 128 & -128 & 16 & 16 & & & & & & & & \\ 3 & 3 & -22 & -22 & 128 & -128 & 22 & 22 & -3 & -3 & & & & \\ & & 3 & 3 & -22 & -22 & 128 & -128 & 22 & 22 & -3 & -3 & & \\ & & & & & \cdots & \cdots & & & & & & & \\ & & & 3 & 3 & -22 & -22 & 128 & -128 & 22 & 22 & -3 & -3 & \\ & & & & & & & -16 & -16 & 128 & -128 & 16 & 16 & \\ & & & & & & & & & -32 & -32 & 160 & -96 & \end{array}\right).$$

We also constructed another version of JZ6 and JZ10 wavelets which are called JZ6-C and JZ10-C respectively. The difference is the boundary elements, while

the inner masks are the same as JZ6 and JZ10. The boundary masks of them are:

- JZ6-C:

  $h^1 : [11, 5, 1, -1]/8,$

  $h^2 : [-4, 4, 8, 8, 1, -1]/8,$

  $h^3 : [1, -1, -1, 1, 8, 8, 1, -1]/8.$

- JZ10-C:

  $h^1 : [186, 70, 10, -10, -3, 3]/128,$

  $h^2 : [-94, 94, 146, 110, 22, -22, -3, 3]/128,$

  $h^3 : [46, -46, -34, 34, 128, 128, 22, -22, -3, 3]/128,$

  $h^4 : [-10, 10, 6, -6, -22, 22, 128, 128, 22, -22, -3, 3]/128.$

A fundamental difference between JZ6/JZ10 and JZ6-C/JZ10-C is that the latter ones have their corresponding continuous wavelets. The inner and boundary masks of JZ6-C/JZ10-C can be used to construct biorthogonal continuous wavelets on the interval. Since the continuous wavelets have the same mask and dual mask as the discrete ones, they also have the same decomposition algorithm. JZ6-C and JZ10-C provide higher vanishing moments on the boundary than JZ6 and JZ10 but give worse numerical performance. This is because the Riesz bounds of JZ6-C and JZ10-C are affected by the boundary wavelets, and the Riesz bounds result in bigger condition numbers of transformation matrices.

An advantage of discrete wavelets is their flexibility on the boundary. We have full control of the boundary elements and we can give them some special

properties, such as high order of vanishing moments. Most successful wavelet algorithms implement symmetric extensions to deal with the boundary, but symmetric extension only grants vanishing moment of order 1.

	Order of vanishing moments	Vanishing moments on the boundary
D4	2	1
B2-4	2	1
D6	3	1
9/7	4	1
JZ6	3	2
JZ6-C	3	3
JZ10	5	2
JZ10-C	5	4

Table 4.2: Comparison on vanishing moments of different wavelet bases on the boundary.

The wavelet packet transforms in the next section is performed by those filters.

4.4 Combination of Wavelet Packets with the ROF model

Since wavelet thresholding and wavelet packet thresholding introduce artifacts at the edges of cartoon components (see [6]), a natural incentive is using these techniques only within a smooth region. The motivation of our idea is inspired by image separation. The variational models perform well in separation of cartoon component, while the wavelet packets are widely used for extraction of textures. We can exploit the advantages of both approaches by combining them together.

Recall our notations:

- f is the original image contaminated with noise.
- c is the cartoon, or smooth component.
- t is the texture component.
- ε is the noise.
- $r = f - c = t + \varepsilon$
- $u = c + t$ is the clean image we want to obtain.

The idea of our approach is that we use variational models, e.g. the ROF model, to separate the cartoon components c and the remainder r. Since the cartoon components and thus its edges are separated from r, r contains only texture and noise and is appropriate to be processed by wavelet packets. Therefore, the following circuit is proposed.

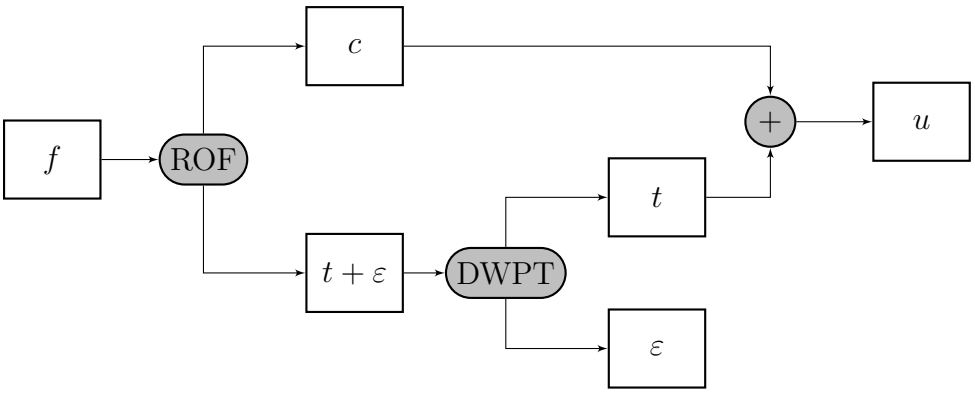


Figure 4.8: Our algorithm: Combining the ROF model with wavelet packets.

We interpret the algorithm as follows.

1. For the observed noisy image f, use a variational model to obtain its smooth component c. In practice, we use the JZ Algorithm to solve the ROF model

$$c = \arg\min_{c \in BV(\Omega)} \{TV(c) + \frac{\mu}{2}\|c - f\|_2^2\}.$$

2. DWPT stands for discrete wavelet packet thresholding. It consists of three steps:

   (a) Perform wavelet packet decomposition on $r := f - c$. Suppose $\tilde{r}$ is the transformed data.

   (b) Process $\tilde{r}$ to get $\tilde{t}$ and $\tilde{\varepsilon}$. Soft thresholding/shrinkage is good enough for most images.

$$\tilde{t} = shrink(\tilde{r}) \text{ and } \tilde{\varepsilon} = \tilde{r} - \tilde{t}.$$

   (c) Apply the inverse wavelet packet transform to $\tilde{t}, \tilde{\varepsilon}$ and get t, ε, where t is the texture and ε is the noise.

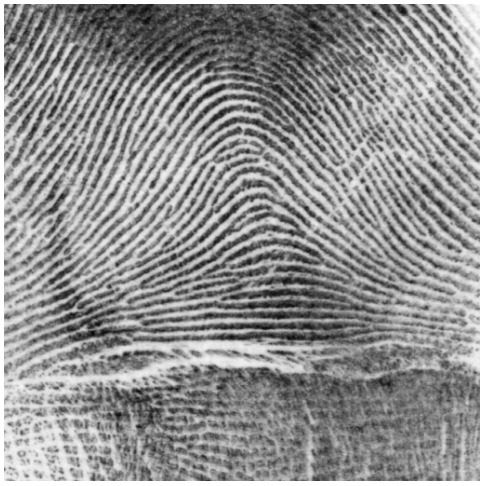
3. Add up the cartoon and the texture to obtain the clean image: $u = c + t$.

4.5 Numerical Experiments

We test the algorithm on three representative texture-rich images: *Barbara*, *Fingerprint*, and *Dollar*.

In the numerical experiment, the noise level is chosen as $\sigma = 25$. We compare our results with the pure thresholding/shrinkage method via wavelet decomposition and wavelet packet decomposition, and the ROF model. Table 4.3 lists the performance of each method for each image. For all images,

(a) Barbara



(b) Fingerprint

(c) Dollar

Figure 4.9: The images with texture for our test problems.

our algorithm outperforms the stand-alone shrinkage or the ROF model.

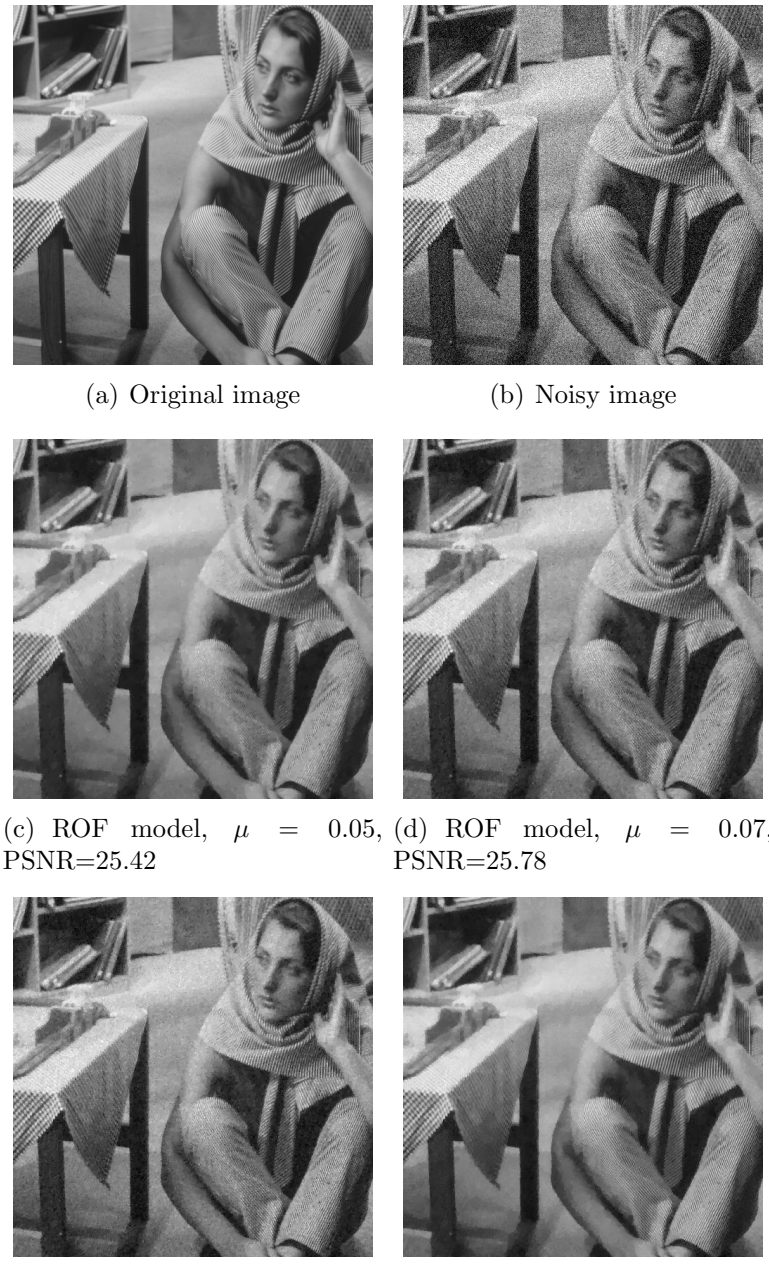
Image	Wavelet Shrinkage	Wavelet Packet	ROF model	Our method
Barbara	25.37	25.00	25.78	26.66
Fingerprint	22.86	22.89	23.35	23.84
Dollar	23.31	22.36	23.67	24.14

Table 4.3: Comparison results between our combinational algorithm and other algorithms on PSNR of different images.

The improvement made by our algorithm is most visible on *Barbara*. This

is demonstrated in Figure 4.10. The ROF model is solved three times with $\mu = 0.05, 0.07$ and 0.07 respectively. For $\mu = 0.05$, the image is a little bit over-smoothed. In this case we have better denoising effect, but the texture is weaken. For $\mu = 0.08$, the denoising is not completely performed, leaving large amount of noise on the image. For $\mu = 0.07$, the ROF model gives the best PSNR. In this case the resulting image has the best balance between noise and texture, but some noise is still left on the image. Our method is obviously optimal among these methods. It gives the highest PSNR, which is much greater than the result of the ROF model.

Figure 4.11 enlarges some details of *Barbara*. Our algorithm has the advantage of both removing noises effectively and retaining texture details. The comparison for *Fingerprint* and *Dollar* is shown in Figure 4.12.



(a) Original image

(b) Noisy image

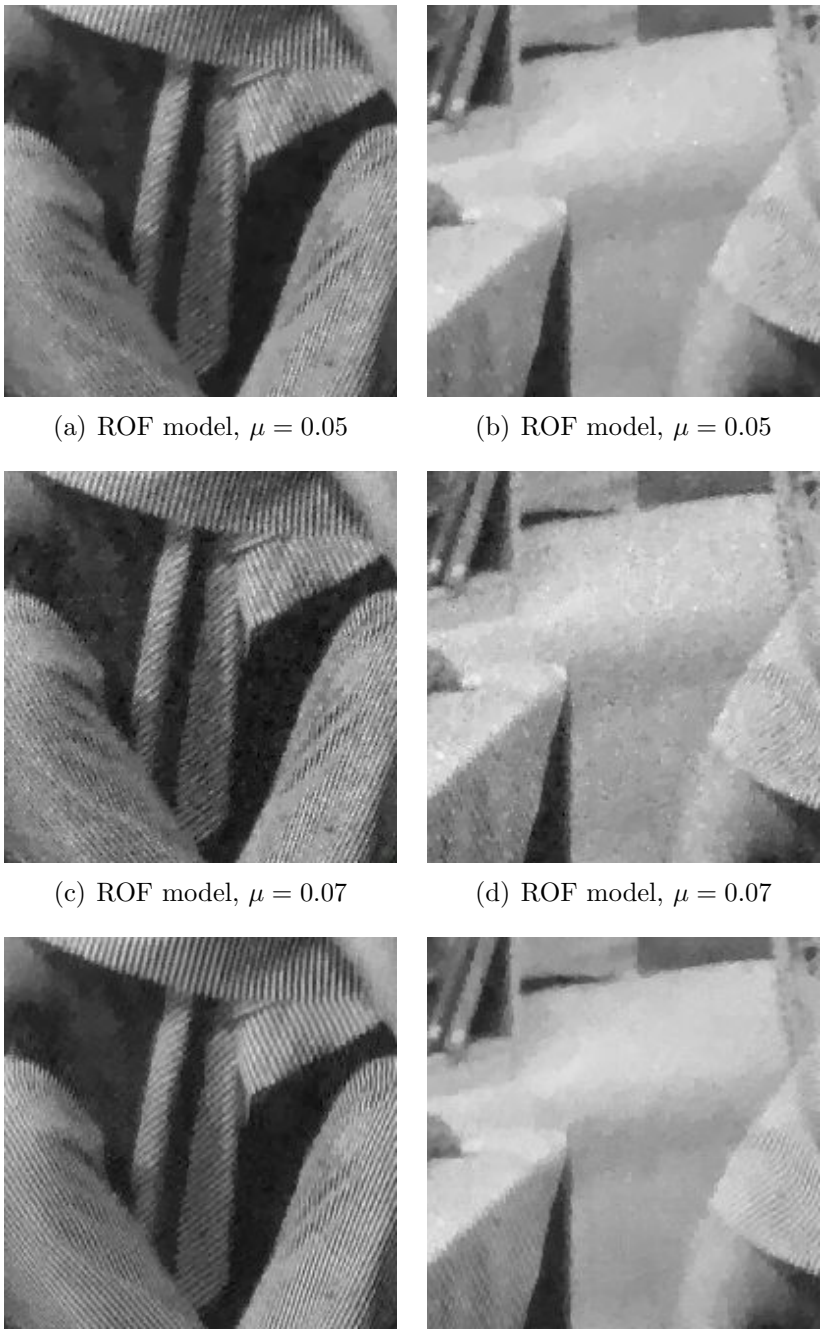
(c) ROF model, $\mu = 0.05$, PSNR=25.42

(d) ROF model, $\mu = 0.07$, PSNR=25.78

(e) ROF model, $\mu = 0.08$, PSNR=25.61

(f) Our method, PSNR=26.66

Figure 4.10: Comparison on PSNR of different models on *Barbara*.

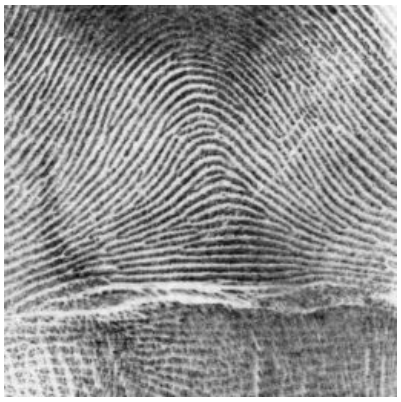


(a) ROF model, $\mu = 0.05$ (b) ROF model, $\mu = 0.05$

(c) ROF model, $\mu = 0.07$ (d) ROF model, $\mu = 0.07$

(e) Our method (f) Our method

Figure 4.11: Comparison results on details of *Barbara*.



(a) Original fingerprint image

(b) Original dollar image

(c) ROF, $PSNR = 23.35$

(d) ROF, $PSNR = 23.67$

(e) Our model,$PSNR = 23.84$

(f) Our model,$PSNR = 24.14$

Figure 4.12: Comparison results on *Fingerprint* and *Dollar*.

Chapter 5

Conclusions and Future Work

This thesis presents some new algorithms and new models for solving image denoising problems. All of them are very efficient and competitive to the existing popular methods. The algorithms are based on the splitting technique and are related to the Bregman iteration. Therefore, they do not have limitations of solution to the Euler-Lagrange equations. Moreover, they are explicit schemes without the requirement of solving any inverse problem. This property discriminates them from the Split Bregman method which solves a linear system in the iteration, and makes our algorithms extremely efficient compared with existing solutions to total-variation-based models. To further boost these algorithms, a relaxation technique is proposed to offset the restrictions.

We also propose some new ideas to improve the ROF model itself. For relatively smooth images, we extend the discrete ROF model with difference operators of order two. With differences of order two, the high-order model successfully reduces staircase effects and achieves better image quality for smooth images. For a particular family of texture-rich images which contain periodic textures, the ROF model is used to separate smooth regions from

high-frequency components. A combinational method utilizes wavelet packet shrinkage to deal with textures, and achieves big improvement in denoising results of a family of texture-rich images.

While these algorithms and methods worked well, there are a few areas where the theory and implementation could be improved.

We proposed relaxation technique and designed Algorithm 2.6, 2.7, and 3.2. In numerical experiments, relaxed algorithms have weakened constrains on parameter λ, and converge faster than the original algorithms. However, convergence of the algorithms with relaxation has not been proved in this thesis. A more general proof of convergence of algorithms, covering relaxation, is needed for completion of theory. Moreover, the rate of convergence has not been studied. In the future, the rate of convergence may also be estimated for better understanding to the schemes.

The implementation of our algorithms requires that the fidelity parameter μ be chosen by human intervention. A rough estimation $\mu = 2.14/\sigma - 0.02$ is proposed, but it only works for those natural images with average smoothness and textures, and has to be revised according to the characteristics of each image. In the past, most researches on the ROF model assume that μ is given or found. In application, human intervention is often not practical, and an automatic algorithm is necessary for a method of image denoising. We are trying to design an automatic method to find the optimal μ for every image at different noise levels. Since noise level can be estimated by $\sigma = 1.05 \underset{i,j}{median}(|(\nabla_x u)_{i,j}| + |(\nabla_y u)_{i,j}|)/2$, a successful estimation of μ will make the entire denoising process automatic for every image without human intervention.

The high-order model provides a convenient way to improve image quality

of total-variation-based denoising methods. In our research, the fidelity coefficients μ_1 and μ_2 are constants at all pixels of an image. Numerical experiments show that low-order difference works well at edges of smooth regions, while high-order difference reduces staircase inside smooth regions. Therefore, this model may be improved by replacing constant coefficients by some location-dependent parameters which balance the difference operators of order one and order two in different areas of an image.

Bibliography

[1] G. Aubert, P. Kornprobst, Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations, Springer-Verlag, New York, 2006.

[2] P. Blomgren, T. F. Chan, P. Mulet, C. K. Wong, Total variation image restoration: numerical methods and extensions, in: Proceedings of the 1997 International Conference on Image Processing (ICIP'97), vol. 3, IEEE Computer Society, 1997, p. 384.

[3] L. M. Bregman, A relaxation method of finding a common point of convex sets and its application to the solution of problems in convex programming, USSR Computational Mathematics and Mathematical Physics 7 (1967) 200–217.

[4] J. F. Cai, S. Osher, Z. W. Shen, Convergence of the linearized Bregman iteration for ℓ_1-norm minimization, Mathematics of Computation 78 (2009) 2127–2136.

[5] J. F. Cai, S. Osher, Z. W. Shen, Linearized Bregman iterations for compressed sensing, Mathematics of Computation 78 (2009) 1515–1536.

[6] T. F. Chan, J. Shen, Image Processing and Analysis, Society for Industrial and Applied Mathematics, Philadelphia, 2005.

[7] Q. Chang, X. C. Tai, L. Xing, A compound algorithm of denoising using second-Order and fourth-Order partial differential equations, Numerical Mathematics: A Journal of Chinese UniversitiesEnglish Series 2.

[8] C. K. Chui, An Introduction to Wavelets, Academic Press, San Diego, 1992.

[9] C. K. Chui, R. Garnett, T. Huegerich, W. He, A universal noise removal algorithm with an impulse detector, IEEE Transactions on Image Processing 14 (2005) 1747–1754.

[10] C. K. Chui, J. Z. Wang, On compactly supported spline wavelets and a duality principle, Transactions of the American Mathematical Society 330 (1992) 903–915.

[11] C. K. Chui, J. Z. Wang, Wavelet-based minimal-energy approach to image restoration, Applied and Computational Harmonic Analysis 23 (2007) 114–130.

[12] C. K. Chui, J. Z. Wang, PDE models associated with the bilateral filter, Advances in Computational Mathematics 31 (2009) 131–156.

[13] A. Cohen, I. Daubechies, J. C. Feauveau, Biorthogonal bases of compactly supported wavelets, Communications on Pure and Applied Mathematics 45 (1992) 485–650.

[14] A. Cohen, I. Daubechies, P. Vial, Wavelets on the interval and fast wavelet transforms, Applied and Computational Harmonic Analysis 1 (1993) 54–81.

[15] I. Daubechies, Ten Lectures on Wavelets, Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[16] D. L. Donoho, De-noising by soft-thresholding, IEEE Transactions on Information Theory 41 (1995) 613–627.

[17] T. Goldstein, S. Osher, The Split Bregman method for L1-regularized problems, SIAM Journal on Imaging Sciences 2 (2009) 323–343.

[18] B. Han, R. Q. Jia, Characterization of Riesz bases of wavelets generated from multiresolution analysis, Applied and Computational Harmonic Analysis 23 (2007) 321–345.

[19] R. Q. Jia, Spline wavelets on the interval with homogeneous boundary conditions, Advances in Computational Mathematics 30 (2009) 177–200.

[20] R. Q. Jia, H. Q. Zhao, A fast algorithm for the total variation model of image denoising, Advances in Computational Mathematics (2009) DOI 10.1007/s10444–009–9128–5.

[21] R. Q. Jia, H. Q. Zhao, W. Zhao, Convergence analysis of the Bregman method for the variational model of image denoising, Applied and Computational Harmonic Analysis 27 (2009) 367–379.

[22] M. Lysaker, A. Lundervold, X. C. Tai, Noise removal using fourth-order partial differential equation with applications to medical magnetic reso-

nance images in space and time, IEEE Transactions on Image Processing 12 (2003) 1579–1590.

[23] M. Lysaker, X. C. Tai, Iterative image restoration combining total variation minimization and a second-order functional, International Journal of Computer Vision 66 (2006) 5–18.

[24] S. G. Mallat, A theory for multiresolution signal decomposition: the wavelet representation, IEEE Transactions on Pattern Analysis and Machine Intelligence 11 (1989) 674–693.

[25] S. G. Mallat, A Wavelet Tour of Signal Processing, Academic Press, San Diego, 1999.

[26] M. Nikolova, Weakly constrained minimization: application to the estimation of images and signals involving constant regions, Journal of Mathematical Imaging and Vision 21 (2004) 155–175.

[27] S. Osher, M. Burger, D. Goldfarb, J. Xu, W. Yin, An iterative regularization method for total variation-based image restoration, Multiscale Modeling and Simulation 4 (2005) 460–489.

[28] P. Perona, J. Malik, Scale-space and edge detection using anisotropic diffusion, IEEE Transactions on Pattern Analysis and Machine Intelligence 12 (1990) 629–639.

[29] W. Ring, Structural properties of solutions to total variation regularization problems, Mathematical Modelling and Numerical Analysis 34 (2000) 799–810.

[30] L. Rudin, S. Osher, E. Fatemi, Nonlinear total variation based noise removal algorithms, Physica D 60 (1992) 259–268.

[31] S. Setzer, Split Bregman algorithm, Douglas-Rachford splitting and frame shrinkage, in: Proceedings of the 2nd International Conference on Scale Space and Variational Methods in Computer Vision, Voss, Norway, vol. 5567, Springer, pp. 464–476.

[32] C. R. Vogel, M. E. Oman, Iterative methods for total variation denoising, SIAM Journal on Scientific Computing 17 (1996) 227–238.

[33] J. Z. Wang, X. Shang, Adaptive smoothing of anisotropic diffusion equations in image denoising, manuscript.

[34] Y. Wang, W. Yin, Y. Zhang, A fast algorithm for image deblurring with total variation regularization, Rice University CAAM Technical Report TR07-10.

[35] W. Yin, S. Osher, D. Goldfarb, J. Darbon, Bregman iterative algorithms for l 1-minimization with applications to compressed sensing, SIAM Journal on Imaging Science 1 (2008) 143–168.