Fast Numerical Algorithms for High Order Partial Differential Equations with Applications to Image Restoration Techniques

by

Carlos Francisco Brito Loeza

UNIVERSITY OF LIVERPOOL

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy.

October 2009

Contents

Acknowledgment

I would like to express my gratitude to my supervisor Professor Ke Chen for his guidance and support throughout my doctoral studies. I also would like to thank other members of the mathematical sciences department for their advice and constructive criticism of my work: Professor Vadim Biktashev, Dr. Ozgur Selsil and Dr. David Lewis.

To my colleagues Noppadol Chumchob and Noor Badshah, I want to thank them for all the very interesting discussions we had along this time.

My most sincere and deepest appreciation to my wife Elda and sons David and Daniel who patiently and lovely support me throughout my studies and are the reason of all my efforts. They made my life as graduate student much easier to handle and contributed to enjoy this experience. Life would not be life without them!

Finally, all my gratitude to my parents, Carlos and Fanny, who wisely taught me the true values of life: honesty and hard work and to my sister Wendy for her continuous support.

Last, but by no means least, to the National Council of Sciences and Technology (CONACYT) of Mexico for the financial support provided for my doctoral studies.

Abstract

This thesis deals with the numerical solution of anisotropic nonlinear high order partial differential equations. The differential equations we tackle here arise from the minimization of variational models for image restoration techniques such as denoising and inpainting. Fast and efficient numerical algorithms for these models are in urgent demand but only slow explicit time marching schemes have been reported in the literature.

The main contribution presented here is the development of fast geometrical multigrid methods for these models. Non-standard fixed point methods are developed for each application. When convergent these fixed point methods can be used as standalone algorithms and they are much faster than explicit methods. However, in this thesis we focus more on the use of these methods as non-standard smoothers for nonlinear multigrid methods. Evidence showing the quality of restoration and fast convergence of our multigrid methods will be presented.

An extra bonus is the Chapter 7 where we introduce two new high order models for vectorial (color) image denoising.

List of Figures

List of Tables

Publications

Multigrid method for a modified curvature driven diffusion model for image inpainting. Carlos Brito-Loeza and Ke Chen. *Journal of Computational Mathematics.* 26(6):856–875, 2008.

Fast numerical algorithms for the Euler's elastica digital inpainting model. Carlos Brito-Loeza and Ke Chen. *Journal of Mathematical Imaging and Vision.* Under review, 2008.

Multigrid algorithm for high-order denoising.
Carlos Brito-Loeza and Ke Chen. *SIAM Journal on Imaging Sciences.* Under review, 2009.

High-order denoising models for vector-valued images.
Carlos Brito-Loeza and Ke Chen. *IEEE Transactions on Image Processing.* Provisionally accepted with mandatory minor revisions, 2009.

Presentations

Multigrid algorithms for high-order variational models with applications to digital image denoising and inpainting.
Carlos Brito-Loeza. 23rd Biennial Conference on Numerical Analysis, June 23rd - 26th 2009.

High-order vector-valued models for image restoration.
Carlos Brito-Loeza. SIAM Conference on Imaging Science, April 12-14, 2010, accepted.

Chapter 1

Introduction

1.1 On the numerical solutions of variational models

Variational techniques for image processing tasks have been around already for sometime, some of them as image denoising have been deeply investigated, though there is still room for improvement. Some others like image registration and particularly image inpainting are relatively new and many challenges are still open to research. One of paramount importance is the numerical realization of the existing models, i.e. fast and reliable numerical algorithms. Usually finding the solution of variational models implies minimizing a nonlinear functional which most of the time leads to the numerical solution of nonlinear and anisotropic partial differential equations (PDEs). By studying those which the research community has accepted as the better models, it is possible to identify a trend which suggests that the better the model, the more nonlinear or anisotropic the PDE is. It is well-known that stable numerical solvers for these type of differential equations are difficult to implement. This thesis deals with this problem by proposing fast and efficient numerical algorithms for the solution of a number of nonlinear differential equations.

In some cases variational energies are constructed using intrinsic and frequently invariant geometrical information existent in the image, examples are first order gradients fields, the second order curvature of level sets or the also second order curvature of the image surface. For inpainting, there is real and strong evidence that second order information delivers better results, this can be in the form of reconnection of far apart broken parts of the image features or by recovering smoothly the curvature of missing sections of an object. In denoising, second order contributes to reconstructing smooth or piecewise smooth parts of the image avoiding the so called staircase effect which other models do have and makes images to look blocky.

Thus curvature brings nice properties to restoration models, but introduces new difficulties. The resulting differential equations from curvature-based models are fourth order and even more anisotropic and nonlinear than their second order gradient-based competitors. Numerical approximations of high order derivatives are therefore an important issue, stencils can be extremely large and the numerical treatment of boundary conditions starts being difficult to handle.

Numerical algorithms for fourth order equations are known to be extremely slow, this is the case even for linear differential operators like the fourth order bi-harmonic operator. This brings to life an important, but yet not considered issue: the size of the domain. Image resolution is continuously and rapidly increasing due to technological advances in digital capturing devices, hence currently an image can easily surpass the million pixels in its domain. Therefore, it would be naive to believe that unilevel numerical algorithms can be fast enough to solve difficult high-order anisotropic equations in such a large domain.

In this thesis we take the multilevel approach also known as multigrid approach to tackle this problem. Multigrid methods take advantage on constructing a hierarchy of discretizations where at each level the error equation is partially solved and the new approximation transported to the next coarser level. This process is recursively applied until reaching the coarsest level where and exact, but computationally cheap solution is obtained. Then the process move backwards in the hierarchical structure transporting the more accurate error and updating the approximate solution at each level until reaching the finest level again.

Thus multigrid methods look attractive because only few iterations are used in high resolution levels and many iterations are only applied to obtain the exact solution at the coarsest and lowest resolution level. They are well known to deliver the optimal method for a large class of linear PDEs or nonlinear PDEs with smoothly varying coefficients [139]. For other PDEs, standard application of multigrid methods leads to no convergence. For the nonlinear anisotropic PDEs studied in this thesis, implementing an operating and reliable multigrid algorithm is however not an easy task. Multigrid methods rely on a good numerical algorithm to smooth the error at every level so it can be easily transported to the next coarser level. These numerical algorithms are known as smoothers. Anisotropy makes standard smoothers to fail at the interface due to jumping of coefficients among other things. The problem is that interfaces are represented by edges in an image and common images may have many of them making this a big issue for multigrid algorithms.

The main contribution of this thesis is the development and analysis of good non-standard smoothers for multigrid algorithms for high order anisotropic partial differential equations. In some cases, these smoothers happen to be convergent algorithms so they can be used as standalone algorithms which are much faster than the usual explicit methods, though not than multigrids.

1.2 Thesis outline

Chapter 2

This chapter contains the mathematical tools the reader may need to review while reading the following chapters of the thesis. A short revision in some important mathematical topics ranging from linear spaces, variations of a functional, bounded space of variations and inverse problems to image representation, iterative solutions of linear and nonlinear equations and multigrid algorithms is presented.

Chapter 3

Here we present a short revision of variational models for image reconstruction techniques. We start with energy based anisotropic filters like the total variation model for image denoising explaining its virtues and shortcomings, then we move to illustrate briefly the properties of high order denoising models. A more extensive revision is carried out for image inpainting models, here we cover the Euler's elastica model, the Cahn-Hilliard model and the Mumford-Shah based models. Their virtues and drawbacks are also described. An important non-variational technique for inpainting called patch-based image completion is also explained and examples from this technique are presented.

Chapter 4

A numerical algorithm for solving the third order nonlinear PDE of the curvature driven diffusion (CDD) inpainting model is developed. The differential equation we deal with here degenerates at flat regions inside the inpainting domain and therefore stable numerical algorithms other than slow explicit methods are not at hand. To tackle this problem a slight modification of the original PDE is introduced enabling us to implement a fixed point method for this new differential equation. Although this fixed point method is not convergent we give evidence through local Fourier analysis that it may have good smoothing properties. Using this non-standard fixed point as smoother we develop a nonlinear multigrid method for the original CDD formulation. A short discussion explaining why coarsening is unique for inpainting problems is given and a method to handle this situation proposed. Finally, numerical evidence showing the fast convergence of the algorithm and its quality of reconstruction is presented.

Chapter 5

In this chapter we develop different algorithms for the solution of the Euler's elastica variational inpainting model. Here the challenge involves solving fast and efficiently a fourth order anisotropic and highly nonlinear PDE. We start explaining why standard fixed point methods fail to converge for this equation and then introduce our first algorithms: two novel unconditionally stable semi-implicit time marching (USTM) schemes based on convexity splitting techniques. Using one of these USTM algorithms as foundation, we develop our third algorithm for the elastica model: a non-standard stabilized fixed point (SFP) algorithm. Finally, after showing the smoothing properties of the SFP algorithm we develop an even faster nonlinear multigrid algorithm. Quality restoration and convergence results are presented at the end of the chapter.

Chapter 6

In this chapter we generalize and extend our work of chapter 5 to the curvature-based denoising model. After describing the curvature model and presenting a revision of the challenges to develop efficient multigrid methods for anisotropic nonlinear high order PDEs, we develop a multigrid algorithm using again a non-standard stabilized fixed point method. However, this time the SFP method has the important characteristic of being adaptive, meaning that it applies extra local relaxation only where required. This new feature of the smoother let us to dramatically reduce the number of Gauss-Seidel iterations applied on each grid without decimating the performance of the multigrid algorithm. We also elaborate a more rigorous local Fourier analysis of the smoother and a two-grid analysis to study the convergence properties of the multigrid method. Extensive information about robustness with respect to variations of parameters and effectiveness of the multigrid algorithm are presented.

Chapter 7

This chapter introduces two new high order models for vectorial (color) image denoising. Based on the two observations: (1) that high order models, in particular curvature-based, preserve better the characteristics of the images, and (2) coupling between image channels adapts the level of smoothing allowing to denoise without wiping out any weak channel, we develop two models: a globally coupled and a locally coupled curvature-based color denoising models. Evidence showing that the quality of reconstruction of these high-order-coupled models is better than the one of non-coupled-high-order and coupled-low-order models is presented. The fast numerical solution of these two models is carried out using a multigrid method similar to the one developed in chapter 6. However, coupling brings a new difficulty to the numerical algorithm introducing instability due to the level of regularization changing at every iteration. We illustrate this problem and explain how to adapt the multigrid method to obtain an optimal performance. Numerical and quality results are presented at the end.

Chapter 8

In the this last chapter we propose possible future research directions derived from the work presented in this thesis.

Chapter 2

Mathematical Preliminaries

2.1 Normed linear spaces

On the early days of mathematics a lot of results were obtained on the real line. To facilitate the extension of those results to n-Euclidean spaces (of dimension n), it was required some kind of structure in the n-dimensional space to avoid certain technical difficulties appearing in more general topological spaces. The class of normed linear spaces fits this need admirably.

We start by introducing the required notation we use throughout this thesis and some definitions commonly met in either linear algebra or advanced calculus literature.

The elements of the n-Euclidean space $\mathbb{R}^n$, called either points or vectors, will be represented as the n-tuples $(x_1, \ldots, x_n)$ which for easy of writing we indicate by using a boldface letter. This is,

$$\boldsymbol{x} = (x_1, \ldots, x_n), \quad \boldsymbol{y} = (y_1, \ldots, y_n). \tag{2.1}$$

The inner product of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as

$$\boldsymbol{x} \cdot \boldsymbol{y} = \langle \boldsymbol{x}, \boldsymbol{y} \rangle = x_1 y_1 + \cdots + x_n y_n \tag{2.2}$$

and the length or Euclidean norm of a vector as

$$\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x} \cdot \boldsymbol{x}} = \sqrt{x_1^2 + \cdots + x_n^2}. \tag{2.3}$$

The concept of norm can be generalized with the following definition.

Definition 2.1.1 (**Norm**). *A real valued function $N : L \subseteq \mathbb{R}^n \to \mathbb{R}$ is called a* norm *on L if it satisfies*

$$N(\boldsymbol{x}) \geq 0 \;\; \textit{with equality if and only if}\; \boldsymbol{x} = \boldsymbol{0}, \tag{2.4}$$
$$N(\alpha \boldsymbol{x}) = |\alpha| N(\boldsymbol{x}), \;\; \textit{for a given scalar}\; \alpha \;\textit{and} \tag{2.5}$$
$$N(\boldsymbol{x} + \boldsymbol{y}) \leq N(\boldsymbol{x}) + N(\boldsymbol{y}) \;\; \textit{which is the triangle inequality.} \tag{2.6}$$

The norm of a vector $\boldsymbol{x}$ is usually represented by $\|\boldsymbol{x}\|$.

Example 2.1.2 (p-norm). *Consider $x \in \mathbb{R}^n$, then for any real number $p \geq 1$ the p-norm of x is defined as*

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}, \tag{2.7}$$

where for $p = 2$ we recover the Euclidean norm previously defined in (2.3). Note the p-norm can be extended to vectors having an infinite number of components, yielding the ℓ^p space defined as the set of all infinite sequences of real or complex numbers with finite p-norm.

Example 2.1.3 (L^p-norm). *Consider a function f defined on a domain Ω and $1 \leq p \leq \infty$. Then*

$$\|f\|_p = \left(\int_\Omega |f(x)|^p \, dx \right)^{1/p} \tag{2.8}$$

defines the L^p-norm of f on Ω. Note this is a generalization of the previous example since now f is allowed to have not only countably-infinitely many components but arbitrarily many components. The special case when $p = \infty$ is defined as

$$\|f\|_\infty = \sup_{x \in \Omega} |f(x)|. \tag{2.9}$$

Definition 2.1.4 (Normed linear space). *A space with a norm defined on it is called a* normed linear space.

Definition 2.1.5 (Cauchy sequence). *A Cauchy sequence is a sequence $\{x_i\}$ having the property that for any $\varepsilon > 0$, there exists an integer N such that if $i, j \geq N$, then $\|x_i - x_j\| \leq \varepsilon$.*

Definition 2.1.6 (Banach space). *A normed linear space L is said to be complete if every Cauchy sequence in L converges to an element in L. A complete normed linear space is called a* Banach space.

Definition 2.1.7 (Linear transformation). *A function $T : L \to M$ is called a* linear transformation *if for all $x, y \in L$ and all real α the following two equations are satisfied*

$$T(x + y) = T(x) + T(y). \tag{2.10}$$
$$T(\alpha x) = \alpha T(x). \tag{2.11}$$

Definition 2.1.8 (Convex functions). *A function f defined on an interval S (or a convex subset of some vector space) is called convex if*

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha) f(y) \tag{2.12}$$

for all $x, y \in S$ and $\alpha \in [0, 1]$. It is called strictly convex provided that the inequality is strict for $x \neq y$ and $\alpha \in (0, 1)$.

Example 2.1.9 *Using definition 2.1.8, prove that the total variation of $u : \Omega \subseteq \mathbb{R}^2 \to \mathbb{R}$ defined as*

$$TV(u) = \int_\Omega |\nabla u| \, dxdy$$

is a convex functional.

Proof. *Consider $u_1 \neq u_2$ two arbitrary functions then,*

$$\int_\Omega |\nabla(\alpha u_1 + (1-\alpha)u_2)|\, dxdy = \int_\Omega |\alpha \nabla u_1 + (1-\alpha)\nabla u_2)|\, dxdy$$
$$\leq \alpha \int_\Omega |\nabla u_1|\, dxdy + (1-\alpha)\int_\Omega |\nabla u_2|\, dxdy. \qquad \square$$

Definition 2.1.10 (Lipschitz condition). *A function $f : I \subset \mathbb{R} \to \mathbb{R}$ that satisfies*

$$|f(x) - f(y)| \leq K|x-y|$$

for some K and all $x, y \in I$ is said to satisfy a Lipschitz condition on the interval I and is called a Lipschitz continuous function.

2.2 Derivatives in a normed linear space

Here we give the definitions of derivatives and differentiable functions in normed linear spaces.

Definition 2.2.1 (Fréchet derivative). *Let f be defined on an open set $U \subseteq L$, taking values in a second normed linear space M. Then f is differentiable at $\boldsymbol{x}_0 \in U$ if there is a linear transformation $T : L \to M$ such that, for sufficiently small $\boldsymbol{h} \in L$*

$$T(\boldsymbol{h}) = f(\boldsymbol{x}_0 + \boldsymbol{h}) - f(\boldsymbol{x}_0) - \|\boldsymbol{h}\|\varepsilon(\boldsymbol{x}_0, \boldsymbol{h}), \tag{2.13}$$

where $\varepsilon(\boldsymbol{x}_0, \boldsymbol{h}) \in M$ goes to zero as $\|\boldsymbol{h}\| \to \boldsymbol{0}$. The linear transformation T is called the Fréchet derivative of f and is denoted by $f'(\boldsymbol{x}_0)$.

Definition 2.2.2 (Gâteaux derivative). *Let f be defined on an open set $U \subseteq L$, taking values in a second normed linear space M. Then f is differentiable at $\boldsymbol{x}_0 \in U$ if the two-sided directional derivative $f'(\boldsymbol{x}_0; \boldsymbol{v})$ exists for each $\boldsymbol{v} \in L$; that is, if*

$$\lim_{\varepsilon \to 0} \frac{f(\boldsymbol{x}_0 + \varepsilon \boldsymbol{v}) - f(\boldsymbol{x}_0)}{\varepsilon} = f'(\boldsymbol{x}_0; \boldsymbol{v}). \tag{2.14}$$

Notice that provided f has a Fréchet derivative at $\boldsymbol{x}_0$, f is Gâteaux differentiable at $\boldsymbol{x}_0$ with $f'(\boldsymbol{x}_0)(\boldsymbol{v}) = f'(\boldsymbol{x}_0; \boldsymbol{v})$ because

$$f'(\boldsymbol{x}_0)(\boldsymbol{v}) = \lim_{t \to 0} \frac{f'(\boldsymbol{x}_0)(t\boldsymbol{v}) + \|t\boldsymbol{v}\|\varepsilon(\boldsymbol{x}_0, t\boldsymbol{v})}{t} = \lim_{t \to 0} \frac{f(\boldsymbol{x}_0 + t\boldsymbol{v}) - f(\boldsymbol{x}_0)}{t} = f'(\boldsymbol{x}_0; \boldsymbol{v}).$$

The opposite, this is, Gâteaux differentiability of f at $\boldsymbol{x}_0$ does not, however, guarantee the existence of the Fréchet derivative $f'(\boldsymbol{x}_0)$ without some further conditions on f.

2.3 Calculus of variation

In this section we introduce the basic tools to compute the first variation (also known as the Euler-Lagrange equation) of a functional. Extensive literature in this respect can be found elsewhere. For instance, in the excellent monographies [60, 62, 63].

2.3.1 Variation of a functional

Consider the functional

$$J[u] = \int_{\Omega} F(\boldsymbol{x}, u(\boldsymbol{x}), \nabla u(\boldsymbol{x}))\, d\boldsymbol{x}, \tag{2.15}$$

defined on some normed linear space depending upon the independent variable $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, an unknown function $u(\boldsymbol{x})$ of these variables and its gradient $\nabla u(\boldsymbol{x}) = (u(\boldsymbol{x})_{x_1}, u(\boldsymbol{x})_{x_2}, \ldots, u(\boldsymbol{x})_{x_n})$. Here, for instance, $u(\boldsymbol{x})_{x_1}$ stands for the derivative of $u(\boldsymbol{x})$ with respect to x_1 (similar for others) and $d\boldsymbol{x}$ is the n-differential element defined as $d\boldsymbol{x} = dx_1 dx_2 \cdots dx_n$.

The most important necessary condition to be satisfied by any minimizer of a variational integral like $J(u)$ is the vanishing of its first variation δJ defined as

$$\delta J(u) = \frac{d}{d\varepsilon} J(u + \varepsilon\varphi)\bigg|_{\varepsilon=0}. \tag{2.16}$$

This is, if u is a minimizer of $J(u)$ with respect to variations $\delta u = \varphi$ which do not change boundary values of u, then (2.16) must be satisfied for all φ with compact support [1] in Ω. Then for some $u_0 \in \Omega$, we call $\delta J(u_0)$ the fist variation of J at u_0 in the direction of φ.

To compute the first variation of variational integrals like (2.15) on the direction of a function $\varphi(\boldsymbol{x})$ with compact support in Ω, we define the following transformation:

$$u^*(\boldsymbol{x}) = u(\boldsymbol{x}) + \varepsilon\varphi(\boldsymbol{x}) + O(\varepsilon^2), \tag{2.17}$$

where $\varepsilon \longrightarrow 0$ and $\|\varphi\| \longrightarrow 0$. Then the variation δJ of the functional (2.15) corresponding to the above transformation is defined as the the linear part in ε of the difference $J(u^*) - J(u)$. By using Taylor's theorem is possible to show, see for example [60], that

$$J[u^*] - J[u] = \varepsilon \int_{\Omega} \left(F_u + \sum_{i=1}^{n} F_{u_{x_i}} \varphi_{x_i} \right) d\boldsymbol{x} + O(\varepsilon^2). \tag{2.18}$$

It follows then that the variation of the functional (2.15) is given by

$$\delta J = \int_{\Omega} \left(F_u + \sum_{i=1}^{n} F_{u_{x_i}} \varphi_{x_i} \right) d\boldsymbol{x}. \tag{2.19}$$

[1]The support of a function is the set of points where this function is not zero. Functions with *compact support* in a space X are those for which their support is a compact subset of X.

2.3.2 The Gauss (divergence) theorem

Consider an open and bounded subset $\Omega \subset \mathbb{R}^n$ with piece-wise smooth boundary $\partial\Omega$. Suppose a scalar function $u(\boldsymbol{x})$ is continuously differentiable in $\bar{\Omega}$. Then

$$\int_\Omega u_{x_i} d\boldsymbol{x} = \int_{\partial\Omega} u(\boldsymbol{x})\nu_i ds \tag{2.20}$$

is satisfied for $i = 1, \dots, n$ with $\boldsymbol{\nu} = (\nu_1, \nu_2, \dots, \nu_n)$ the outward unit normal of $\partial\Omega$. Using this result and simple additivity, it is simple to show that for a given vector field $\boldsymbol{F} = \boldsymbol{F}(x)$ the following is true

$$\int_\Omega (\nabla \cdot \boldsymbol{F})\, d\boldsymbol{x} = \int_{\partial\Omega} \boldsymbol{F} \cdot \boldsymbol{\nu}\, ds. \tag{2.21}$$

The latter, is the most common way to present the divergence theorem.

2.3.3 Integration by parts

An immediate consequence of the divergence theorem is the integration by parts formula. This is, for $i = 1, \dots, n$ and for two continuously differentiable functions $u(\boldsymbol{x})$ and $v(\boldsymbol{x})$ in $\bar{\Omega}$

$$\int_\Omega u_{x_i} v(\boldsymbol{x})\, d\boldsymbol{x} = -\int_\Omega u(\boldsymbol{x}) v_{x_i}\, d\boldsymbol{x} + \int_{\partial\Omega} u(\boldsymbol{x})v(\boldsymbol{x})\nu_i ds. \tag{2.22}$$

Or by applying (2.21) to the product of a scalar function g and a vector field $\boldsymbol{F}$ we obtain the vectorial representation

$$\int_\Omega (\boldsymbol{F} \cdot \nabla g + g\nabla \cdot \boldsymbol{F})\, d\boldsymbol{x} = \int_{\partial\Omega} g\boldsymbol{F} \cdot \boldsymbol{\nu}\, ds. \tag{2.23}$$

To finalize this short section we present an example of how to compute the first variation of a functional of our interest.

Example 2.3.1 *Consider the problem of finding the first variation of the functional*

$$F(u) = \int_\Omega |\nabla u|\, dxdy$$

defined on a domain $\Omega \subset \mathbb{R}^2$. To this end, we introduce the small variation $\varepsilon\varphi$ composed by the parameter $\varepsilon \to 0$ and the continuously differentiable function φ with compact support in Ω. Then we compute,

$$\begin{aligned} \frac{d}{d\varepsilon} F(u + \varepsilon\varphi)\Big|_{\varepsilon=0} &= \frac{d}{d\varepsilon}\int_\Omega |\nabla(u + \varepsilon\varphi)|\, dxdy\Big|_{\varepsilon=0} \\ = \int_\Omega \frac{\nabla(u+\varepsilon\varphi)}{|\nabla(u+\varepsilon\varphi)|} \cdot \nabla\varphi\, dxdy\Big|_{\varepsilon=0} &= \int_\Omega \frac{\nabla u}{|\nabla u|} \cdot \nabla\varphi\, dxdy. \end{aligned} \tag{2.24}$$

We can now use integration by parts on (2.24), thus

$$\int_\Omega \frac{\nabla u}{|\nabla u|}\cdot \nabla\varphi \, dxdy = -\int_\Omega \nabla\cdot\left(\frac{\nabla u}{|\nabla u|}\right)\varphi \, dxdy + \int_{\partial\Omega} \varphi \boldsymbol{\nu}\cdot\nabla u \, ds, \tag{2.25}$$

where $\partial\Omega$ is the boundary of Ω, $\boldsymbol{\nu}$ is the unit outward normal vector to $\partial\Omega$ and ds is the length element of integration. If we further require

$$\frac{d}{d\varepsilon}F(u+\varepsilon\varphi)\bigg|_{\varepsilon=0} = 0,$$

then the following PDE known as the Euler-Lagrange equation must be satisfied :

$$\nabla\cdot\left(\frac{\nabla u}{|\nabla u|}\right) = 0 \ \ in \ \Omega \tag{2.26}$$

with Neumann boundary condition $\boldsymbol{\nu}\cdot\nabla u = 0$ on $\partial\Omega$.

2.4 Functions of bounded variation

Let Ω be a bounded open subset of $\mathbb{R}^n$ and let $u \in L^1(\Omega)$. Define the total variation of u as

$$\int_\Omega |Du| = sup\left\{\int_\Omega u \operatorname{div}\varphi \, dx : \ \varphi = (\varphi_1, \varphi_2, \dots, \varphi_n) \in C_0^1(\Omega;\mathbb{R}^n)^n \right. \\ \left. \text{and } \|\varphi_i\|_{L^\infty(\Omega)} \le 1 \text{ for } i = 1,\dots,n\right\}, \tag{2.27}$$

where

$$\operatorname{div}\varphi = \sum_{i=1}^n \frac{\partial\varphi_i}{\partial x_i}(x),$$

dx is the Lebesgue measure[2] and $C_0^1(\Omega)^n$ is the space of continuously differentiable functions with compact support in Ω. Notice that all the components of φ have $L^\infty(\Omega)$-norm less than 1.

As described in [65], a particular and interesting case is when $u \in C^1(\Omega)$, then integration by parts gives

$$\int_\Omega u \operatorname{div}\varphi \, dx = -\int_\Omega \sum_{i=1}^n \frac{\partial u}{\partial x_i}\varphi_i \, dx \tag{2.28}$$

for every $\varphi \in C_0^1(\Omega;\mathbb{R}^n)^n$, so that

$$\int_\Omega |Du| = \int_\Omega |\nabla u| \, dx. \tag{2.29}$$

[2]In Euclidean spaces, the standard way to assign a *measure* (length, area or volume) to a given subset is through the Lebesgue measure. Hence, sets with finite Lebesgue measure are called Lebesgue measurables. In real analysis, this measure is used to define Lebesgue integration.

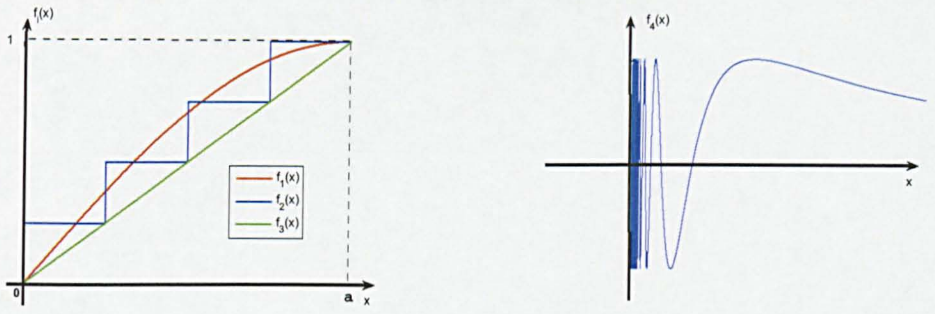


Figure 2.1: On the left, three bounded variation functions with the same total variation. On the right, a function with no bounded variation.

A function $u \in L^1(\Omega)$ is said to have bounded variation in Ω if $\int_\Omega |Du| < \infty$. We define $BV(\Omega)$ as the space of all functions in $L^1(\Omega)$ with bounded variation.

Example 2.4.1 *To better understand bounded variation functions let us briefly analyze some illustrative examples. The following functions $f_1(x), f_2(x)$ and $f_3(x)$ defined below, all belong to $BV(\Omega)$ with $\Omega = [0, \pi/2]$.*

$$f_1(x) = \sin(x), \tag{2.30}$$

$$f_2(x) = \begin{cases} 1/4 & for\, 0 \le x < \pi/8 \\ 1/2 & for\, \pi/8 \le x < \pi/4 \\ 3/4 & for\, \pi/4 \le x < 3\pi/8 \\ 1 & for\, 3\pi/8 \le x < \pi/2 \end{cases}, \tag{2.31}$$

$$f_3(x) = \frac{2x}{\pi}. \tag{2.32}$$

Even more, since f_1, f_2 and f_3 are all monotonic functions[3]*, then all have the same total variation equal to one*[4]*. However, only $f_1(x)$ and $f_3(x)$ are continuous and differentiable functions on Ω.*

Example 2.4.2 *Now consider the function $f_4(x)$ defined as*

$$f_4(x) = \begin{cases} 0 & for\, x = 0 \\ \sin(1/x) & for\, 0 < x \le a \;\; with\;\; a > 0. \end{cases} \tag{2.33}$$

Here $\Omega = [0, a]$ for any $a > 0$. This function is plotted on the right-hand-side in Figure 2.1. We see that as $x \to 0$ the frequency of the oscillations of $f_4(x)$ increases, then the more x approaches zero the more variations need to be added and the value of

[3]In calculus, a function f is called monotonic if for all x and y such that $x \le y$ one has $f(x) \le f(y)$, i.e. f preserves the order.

[4]The total variation of a monotonic real valued function f in an interval (a, b) is given by $|f(b) - f(a)|$, see [113].

the integral (2.29) increases. Therefore, this function has infinite total variation and does not belong to $BV(\Omega)$. Note however that $f_4(x)$ does have bounded variation on $[a,b]$ for any $a > 0$.

Remark 2.4.3 *Under the norm*

$$\|u\|_{BV} = \|u\|_{L^1} + \int_\Omega |Du|, \tag{2.34}$$

$BV(\Omega)$ is a Banach space [65].

An interesting characterization of the functions with bounded variation can also be found in [87].

2.5 The coarea formula

This section introduce a powerful tool for the analysis of BV functions: the coarea formula. This formula gives a natural connection between the total variation of a function u representing an image and the perimeter of its level sets.

Definition 2.5.1 (Borel set). *Given X any topological space, we say that $E \subset X$ is a Borel set if E can be obtained by a countable number of operations, starting from open sets, each operation consisting in taking unions, intersections and complements [115, 16].*

Definition 2.5.2 (Perimeter). *Let E be a Borel set and Ω an open set in $\mathbb{R}^n$. Define the perimeter of E in Ω as*

$$Per(E,\Omega) = \int_\Omega |D\chi_E| = sup\left\{\int_E \operatorname{div}\varphi\, dx : \varphi \in C_0^1(\Omega;\mathbb{R}^n)^n \text{ and } |\varphi(x)| \leq 1 \text{ for } x \in \Omega\right\}, \tag{2.35}$$

where

$$\chi_E = \begin{cases} 1 & \text{if } x \in E \\ 0 & \text{if } x \in \Omega - E \end{cases} \tag{2.36}$$

is the indicator function of E.

Definition 2.5.3 (Coarea formula). *Let $u = u(x)$ and $f = f(x)$ be two scalar functions defined on $\mathbb{R}^n$. Assume that u is Lipschitz continuous and that for almost every $\lambda \in \mathbb{R}$, the level set $L_\lambda = \{x \in \mathbb{R}^n : u(x) = \lambda\}$ is a smooth $(n-1)$-dimensional hyper-surface in $\mathbb{R}^n$. Suppose also that f is continuous and integrable. Then*

$$\int_{\mathbb{R}^n} |\nabla u| f\, dx = \int_{-\infty}^{+\infty} \left(\int_{L_\lambda} f ds\right) d\lambda. \tag{2.37}$$

For the particular case when $f = 1$ and the region of integration is a subset $\Omega \subset \mathbb{R}^n$ we have

$$\int_\Omega |\nabla u|\, dx = \int_{-\infty}^{+\infty} \left(\int_{L_\lambda} ds\right) d\lambda = \int_{-\infty}^{+\infty} Per(L_\lambda, \Omega)\, d\lambda. \tag{2.38}$$

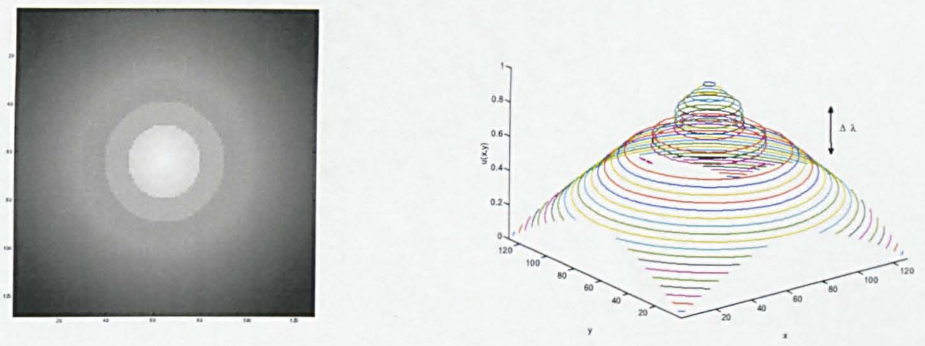


Figure 2.2: On the left a given gray level image $u(\boldsymbol{x})$ and on the right some of its λ-level curves, these are curves where $u(\boldsymbol{x}) = \lambda$ for some $\lambda \in [0,1]$.

Remark 2.5.4 *Let us go back to our definition of an image as a real function $u(\boldsymbol{x})$ taking values on some finite interval, say $[0,1]$ for simplicity. Let us select some slice (level set) of u by setting $u(\boldsymbol{x}) = \lambda$. The slice selected this way represents a λ-level domain defined as $L_\lambda = \{\boldsymbol{x} \in \Omega \subset \mathbb{R}^2 : u(\boldsymbol{x}) = \lambda\}$ for $\lambda \in [0,1]$. We illustrate this with an example in Figure 2.2 where the image in (a) is defined by the function*

$$f(x,y) = \begin{cases} 1 - \sqrt{(x-64)^2 + (y-64)^2}/128 & (x,y) \in \Omega \setminus \Omega_1 \cup \Omega_2 \\ 0.6 & (x,y) \in \Omega_1 \\ 0.7 & (x,y) \in \Omega_2 \end{cases} \tag{2.39}$$

with $\Omega = [0,128]^2$, Ω_1 the ring bounded by the two circles $(x-64)^2 + (y-64)^2 = 14^2$ and $(x-64)^2 + (y-64)^2 = 26^2$ and Ω_2 the ring bounded by the latter and $(x-64)^2 + (y-64)^2 = 38^2$, and the 3-dimensional plot in (b) shows some of its level sets.

Therefore according to (2.35) the perimeter of each slice of u is given by

$$Per(L_\lambda, \Omega) = \int_\Omega |D\chi_{L_\lambda}|. \tag{2.40}$$

And now using the coarea formula we obtain

$$\int_\Omega |\nabla u| \, d\boldsymbol{x} = \int_0^1 Per(L_\lambda, \Omega) \, d\lambda. \tag{2.41}$$

This fantastic result shows that the total variation of a given image u is just the sum of every length of all its λ-level curves. This automatically takes care of all discontinuities of u and therefore the contribution of edges to the total variation integral is enforced.

2.6 Inverse problems and regularization

Inverse problems are commonly encountered in many different branches of science. For instance, water pollution source identification problems [111], hydraulic conductivity identification in steady groundwater flow [57], etc. They are also present in the formulation of many image processing tasks such as denoising, deblurring, inpainting, etc.

The variational approach to these problems usually requires the regularization of the inverse problem to make it well-posed and therefore solvable. In this section, we review the very basic theory of inverse problems and their regularization. We start by introducing the concepts of well- and ill-posed problems.

2.6.1 Well and ill-posed problems

It is usual to refer to a well-posed problem (in the sense of Hadamard) one for which:

1. the solution exists,
2. the solution is unique,
3. the solution depends continuously on the data (stability condition).

If any of these conditions do not hold, then the problem is said to be ill-posed.

2.6.2 Inverse problems

An inverse problem is one where the task is to recover from the known data some parameter values of the physical model.

Inverse problems are typically ill-posed, being the stability condition the most violated condition of well-posedness. In the branch of variational models for image restoration it is not uncommon however to find inverse problems where the uniqueness condition fails as well (inpainting for example).

Example 2.6.1 *The classical example to illustrate an inverse problem (linear in this case) is the Fredholm integral equation of the first kind*

$$f(x) = \int_a^b g(x,y)h(y)\,dy,$$

where the task is to infer $h(y)$ from the data $f(x)$ with smooth $g(x,y)$. Here if the mapping is only injective then the inverse will not be continuous. Hence small errors in the measured data $f(x)$ will introduce large errors in the solution $h(y)$. In this sense the inverse problem is ill-posed.

Example 2.6.2 *We consider now an example from image restoration techniques. A noisy image u^0 defined in $\Omega \subset \mathbb{R}^2$ can easily be constructed by adding certain quantity of Gaussian noise η to a clean image u in such a way that the relation $u^0 = u + \eta$ is satisfied. Now consider the inverse problem of given only u^0 to find u, implying the removal of the noisy part η. This problem can be approached using variational techniques with the extra assumption that the standard deviation σ^2 of the noise is known or at least can be estimated. In this case, the problem is to find u which minimizes*

$$\min_u \left\{ \left| \int_\Omega |u - u^0|^2 \,dxdy - \sigma^2 \right| \right\}. \tag{2.42}$$

Here, we have an inverse problem that is ill-possed due to (2.42) having many possible solutions.

2.6.3 Regularization

Regularization is a technique used to transform an inverse problem into a well-posed problem; Tikhonov *et al.* [138] introduced a popular way to overcome ill-posed minimization problems. The basic idea is to introduce a new constraint to the problem which demands the solution to belong to a specific set of solutions, or to have specific features. For instance, Example 2.6.2 can be *regularized* the following way:

$$\min_{u\in L^2}\left\{\int_\Omega |u-u^0|^2\,dxdy+\alpha\int_\Omega |\nabla u|^2\,dxdy\right\}. \tag{2.43}$$

The first term in (2.43) is the *data or fitting term* and the second is the *regularization term* which requires the solution u to have low gradient values and therefore removing noise. Other examples where Tikhonov regularization has been used can be found in [94, 49] and references therein.

2.6.4 Choosing the regularization parameter

We discuss two different methods for choosing the positive regularization parameter $\alpha > 0$ automatically.

L-curve method

This method proposes to choose an extensive set of parameters $0 < \alpha_1 < \alpha_2 < ... < \alpha_N$ and solve the Tikhonov regularization problem (2.43) for all of them. Then we plot the points $(X(\alpha), Y(\alpha))$ with

$$X(\alpha)\equiv\log\left(\|u_\alpha-u^0\|_2^2\right),\quad Y(\alpha)\equiv\log\left(\|u_\alpha\|_2^2\right)$$

in the plane, resulting in a curve which looks like the letter L. Then we can take the α value corresponding to the corner of the L as the better choice for the regularization parameter. More information about this very useful technique can be found in [71, 72, 70, 144] and references therein.

Morozov's discrepancy principle

Morozov's discrepancy principle is based on the assumption that if we have an estimate on the magnitude of the error in the data (noise in Example 2.6.2) then any solution that yields a measurement with error of same magnitude is acceptable. Morozov's principle gives us a way of choosing the parameter α when $\|u\|_2^2$ is used as regularization instead of any other general functional such as $\|\nabla u\|_2^2$ in Example 2.6.2.

This method involves finding the zero of a nonlinear real function of one variable and is usually computationally quick. References for this method can be found for instance in [120, 144, 15].

2.7 Image representation

In this section we will review some of the most frequent ways to represent images.

2.7.1 Computational representation

Computationally a gray-scale image is a collection of values stored in a 2-dimensional array or matrix $U = [u_{i,j}]_{m \times n}$. Each entry of the array is called a *pixel* of the image and it takes values usually on either $[0, 255]$ or $[0, 1]$ depending upon if the image has been normalized or not and representing the brightness of that specific pixel.

Color or vector-valued images, on the other hand, are multi-dimensional arrays $U = [u_{i,j,k}]_{m \times n \times p}$, where each p-layer (of dimension $m \times n$) may represent color , brightness or a combination of both. In a color image at each (i, j)-location, a *pixel* is now represented by a vector $\boldsymbol{u}_{i,j} = (u_{i,j,1}, u_{i,j,2}, \dots, u_{i,j,p})$. In the popular RGB representation, $p = 3$ and each entry of the vector $\boldsymbol{u}_{i,j}^{RGB} = (u_{i,j,1}, u_{i,j,2}, u_{i,j,3})$ stands for the intensity level of each one of the color channels: red (R), green (G) and blue (B).

2.7.2 Mathematical representation

Mathematically a gray-scale image is characterized by a smooth function

$$u = u(x, y) : \Omega \to \mathbb{R}, \tag{2.44}$$

where $\Omega \subset \mathbb{R}^2$. Based on this, images can be seen as :

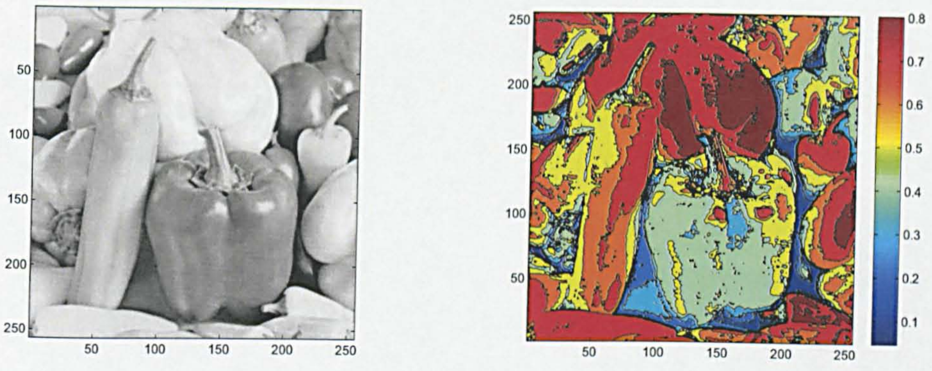
- Parameterized curves.
- The level sets or isophotes of the function u.
- Surfaces where the height is the gray-scale value.

Details of the parameterized-curves representation can be found in [6]. The other two will be reviewed in the following sections. They are important for us since, in this thesis, the restoration models described on the first chapters use the level set representation and the curvature-based denoising model of Chapter 6 adopts the surface representation.

2.7.3 Images as a collection of level sets

Consider a gray level image u defined as in (2.44). Then, for each real value λ define the λ-level set of u as

$$\gamma_\lambda = \{(x, y) \in \Omega : u(x, y) = \lambda\}.$$



(a) (b)

Figure 2.3: (a) Original gray-scale image of size 256×256 taking values on the interval $[0, 1]$. (b) Level sets of the image in a 2-dimensional view.

Then the classical level-set representation of u is the one-parameter family of all the level sets

$$\Gamma_u = \{\gamma_\lambda : \lambda \in \mathbb{R}\}.$$

By adopting this representation, the image domain is partitioned, meaning that $\Omega = \bigcup_{\lambda \in \mathbb{R}} \gamma_\lambda$ with $\gamma_\lambda \bigcap \gamma_\mu = \emptyset$ for $\lambda \neq \mu$. This way an image can be represented as a collection of level sets.

Notice that this representation yields ∇u as the normal vector to each λ-level set and since the curvature κ_{LS} is defined as the rate of change of the unitary normal vector we have that

$$\kappa_{LS} = \nabla \cdot \frac{\nabla u}{|\nabla u|}. \tag{2.45}$$

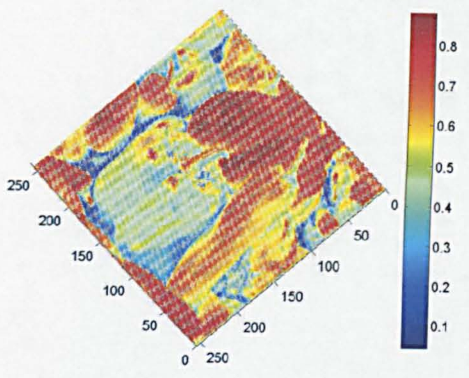
A quite illustrative and detailed derivation of this formula for a general function parameterized by its curvilinear abscissa can be found in [6].

Figure 2.3 shows a gray-scale image and some of its level sets.

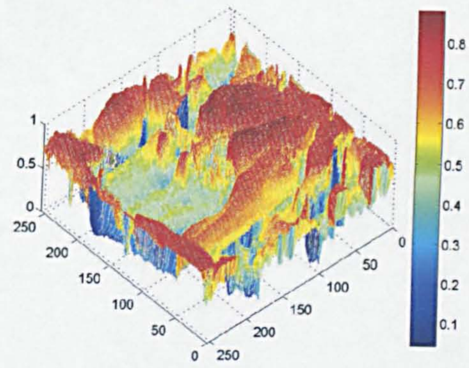
2.7.4 Images as surfaces

A different image representation is obtained from regarding an image as the induced 3-dimensional surface or graph characterized by $z = u(x, y)$ which defines the image surface $(x, y, u(x, y))$. By considering the level set function

$$\phi(x, y, z) = u(x, y) - z,$$



(a)



(b)

Figure 2.4: Two different views of the surface representation of the image in Figure 2.3.

we see that its zero level set $\{(x, y, z) : \phi(x, y, z) = 0\}$ corresponds to the surface $z = u(x, y)$. Therefore, the mean curvature of the surface can be expressed as

$$\kappa_S = \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} \tag{2.46}$$

and since $\nabla \phi = (u_x, u_y, -1)$ and $|\nabla \phi| = \sqrt{u_x^2 + u_y^2 + 1} = \sqrt{|\nabla u|^2 + 1}$ the curvature of the surface can be expressed in terms of u only as

$$\kappa_S = \nabla \cdot \frac{\nabla u}{\sqrt{|\nabla u|^2 + 1}}. \tag{2.47}$$

In Figure 2.4 we show the surface representation of the image presented in Figure 2.3.

2.7.5 Scale-space representation

Scale-space theory is the framework which proposes to represent images as a one-parameter family of smoothed images. Scale-scale representation was initially studied by Witkin [154], Koenderink [82, 83] and Lindeberg [89] and more recently can be found in the works of Weickert [148], Duits *et al.* [51] and references therein.

To obtain such a representation, a smoothing kernel depending upon a parameter t is selected and used to suppress the fine-scale structures of the image. The structures of spatial size smaller than about $\sqrt{t}$ are smoothed away in the scale-space level at scale t. The parameter t is referred to as the scale parameter.

The most known type of scale-space is the Gaussian scale-space which is linear and has the nice property of being possible to derive from it a small set of scale-space axioms [2]. For a given image $u(x, y)$, its linear (Gaussian) scale-space representation is a family of derived images $L(x, y; t)$ defined by the convolution of $u(x, y)$ with the

Gaussian kernel

$$G_t(x,y) = \frac{1}{2\pi t} e^{-(x^2+y^2)/2t} \tag{2.48}$$

such that $L(x,y;t) = (G_t * u)(x,y)$.

Linear scale spaces [154] are also obtained from evolving a parabolic equation of the form

$$\frac{\partial u}{\partial t} = \nabla \cdot (D(x,y)\nabla u)\,, \quad (x,y) \in \Omega, \;\; u(x,y,0) = u^0(x,y), \tag{2.49}$$

which due to $D(x,y)$ being linear, unavoidably smear sharp edges while filtering out noises.

In Figure 2.5 we present some different scales $L(x,y;t)$ obtained through linear (Gaussian) filtering for a model image.

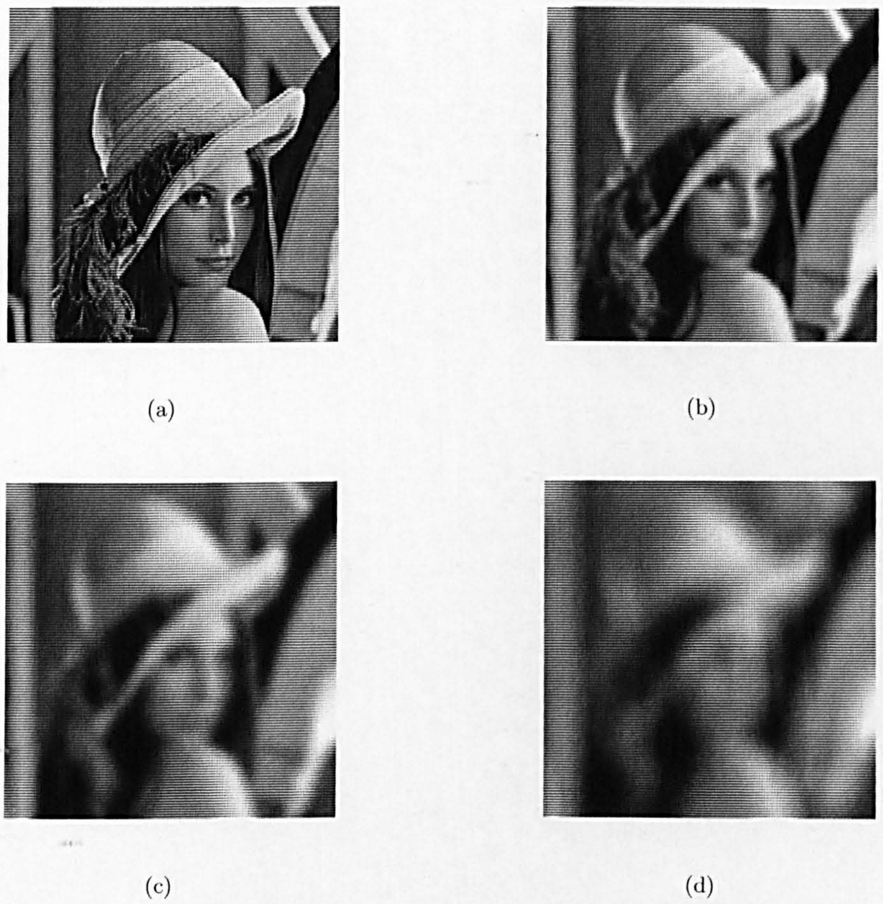


Figure 2.5: (a) Scale-space representation $L(x,y;t)$ at scale $t=0$, corresponding to the original image u. (b) Scale-space representation $L(x,y;t)$ at scale $t=1$. (c) Scale-space representation $L(x,y;t)$ at scale $t=8$. (d) Scale-space representation $L(x,y;t)$ at scale $t=32$.

A remedy for the edge smearing shortcoming from linear filtering was introduced in [108], where the use of different types of nonlinear coefficients $D(u)$ was studied. This type of nonlinear diffusion gives rise to a nonlinear scale-space representation of the image. We will give more details of this method in Chapter 3, where we will study the Perona and Malik's nonlinear diffusion model [108].

2.8 Iterative solution of nonlinear equations

In many applications, a nonlinear system of equations usually arises in the attempt to minimize or find a critical point of a nonlinear functional. This type of systems can be represented as

$$F(\boldsymbol{x}) = \boldsymbol{y}, \tag{2.50}$$

where $F : dom(F) \subset \mathbb{R}^n \to \mathbb{R}^n$ is a nonlinear operator, $\boldsymbol{x} = (x_1, \dots, x_n)$ the vector of unknowns and $\boldsymbol{y} = (y_1, \dots, y_n)$ a fixed vector. It is common practice to absorb the vector $\boldsymbol{y}$ into F and consider only

$$F(\boldsymbol{x}) = \boldsymbol{0} \tag{2.51}$$

with $\boldsymbol{0}$ representing the zero vector.

In this section, we introduce two of the most used iterative nonlinear methods to find solutions for this type of problems: the Newton's method and the Descent method. These two methods are highly used due to two completely different reasons: Newton's method because of its very desirable property of quadratic convergence and Descent method because of its simplicity of implementation. Unfortunately both methods, as we shall show in future chapters, have drawbacks when solving highly nonlinear problems with discontinuous coefficients which are the type of problems we deal with in this thesis.

2.8.1 Newton's method

This method proposes to find the solution of (2.51) by applying the following iterative equation

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - F'(\boldsymbol{x}^k)^{-1} F(\boldsymbol{x}^k), \qquad k = 0, 1, \dots \tag{2.52}$$

In practice, the inverse of $F'(\boldsymbol{x}^k)$ is rarely explicitly computed since may be a difficult task, this happens particularly for problems in which the dimension of the system may be of several thousand. Instead, the system

$$F'(\boldsymbol{x}^k)\boldsymbol{h} = -F(\boldsymbol{x}^k) \tag{2.53}$$

is solved and $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \boldsymbol{h}$ is taken as the new approximation.

For the Newton's iteration (2.52), it can be proved [102] that under certain natural conditions on F', the estimate

$$\|x^{k+1} - x^*\| \leq c\|x^k - x^*\|^p \tag{2.54}$$

with c any positive constant and $p = 2$, holds provided that the iterate x^k is *sufficiently close* to the true solution x^*. Algorithms satisfying (2.54) for $p = 1$ are said to have *linear convergence* and those with $p = 2$ to have *quadratic convergence.* It is here, in (2.54) where the importance of Newton's method rests since it shows that under suitable conditions Newton's method is very quick to converge. Notice however that (2.54) is only satisfied for $\|x^k - x^*\|$ very small. In other words, Newton's method requires a very good initial guess; on the contrary, it is very likely it will get stranded.

Newton's method applied to nonlinear problems with discontinuous coefficients presents very often an erratic behavior. In the following chapters we will show that minimization of total-variation-type functionals as (2.41) in Section 2.5 yields nonlinear systems with highly discontinuous coefficients preventing us to use Newton's method as our iterative solver.

2.8.2 Descent method

Consider a general function $g : \mathbb{R}^n \to \mathbb{R}$. Descent methods propose to minimize g using the following scheme

$$x^{k+1} = x^k - \alpha^k p^k, \qquad k = 0, 1, \ldots, \tag{2.55}$$

where α^k is an scalar allowed to change at each k-iteration and $-p^k$ can be thought as a vector defining the direction along which the new iterate x^{k+1} will be chosen. The parameter α^k is used to determine the *step-length* from x^k to x^{k+1}. It is also common practice to think of the parameter α^k as the *time-step* Δt of a newly introduced time-variable t. In this case, the descent method is referred to as a time-marching method.

The main characteristic of descent methods is that the iterates decrease the function value at each stage, i.e.

$$g(x^{k+1}) \leq g(x^k). \tag{2.56}$$

Gradient descent or steepest descent method

An important class of descent methods is obtained by selecting p^k as the direction of maximal local decrease of g. If g is differentiable at x^k then this direction is given by $\nabla g(x^k)$ resulting in the scheme

$$x^{k+1} = x^k - \gamma \nabla g(x^k), \qquad k = 0, 1, \ldots \tag{2.57}$$

Here γ, which can be fixed or not, must be selected sufficiently small to satisfy condition (2.56). This is the main weakness of the gradient descent method since selecting the optimal γ can be time-consuming and using γ fixed may produce poor convergence. Further, if the curvature of g in different directions is very different, the method will converge very slowly. This situation is very likely to occur in highly nonlinear problems.

2.9 Iterative methods for linear systems of equations

Consider the linear system $Ax = b$ where A is an $m \times n$ matrix, x is the $n \times 1$ vector of unknowns and b is an $m \times 1$ vector.

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \cdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}. \tag{2.58}$$

The above is a system of m linear equations with n unknowns, each one of them (say the i_{th}) of the form

$$\sum_{j=1}^{n} a_{i,j} x_j = b_i. \tag{2.59}$$

In the field of image processing the size of these systems is governed by the quality or resolution of the image is going to be processed. In this new era of high resolution digital cameras, images can easily have millions of pixels of resolution resulting in very large systems of equations; usually of order from 10^4 to 10^6. For this reason, trying to solve these systems with *direct methods* (for instance Gaussian elimination) can be computationally very expensive since a lot of memory is needed [66].

Here is where *iterative methods* like conjugate gradient [116, 66, 102], Jacobi [116, 137], Gauss-Seidel [116, 137] and others result to be very useful. These *iterative methods* compute the solution of the system (up to some accuracy) by starting from a given initial guess and then successively finding new and closer approximations to the true solution of the system. They are computationally cheap and easy to implement.

Before introducing the iterative methods of our interest, first we present some matrix properties which make the linear systems of the form $Ax = b$ suitable to be solved using these iterative methods.

Definition 2.9.1 (Symmetric matrices). *A symmetric matrix A is a square matrix $(m = n)$ that is equal to its transpose A^T. This is, $A = A^T$.*

Definition 2.9.2 (Diagonally dominant matrices). *A matrix A is said to be diagonally dominant if satisfies*

$$|a_{i,i}| \geq \sum_{j \neq i} |a_{i,j}| \quad \textit{for all } i.$$

If the strict inequality is satisfied then we say that A is strictly diagonally dominant.

Definition 2.9.3 (Positive definite matrices). *A real symmetric matrix A is positive definite if $\boldsymbol{x}^T A\boldsymbol{x} > 0$ for all non-zero vectors $\boldsymbol{x}$. This is equivalent to say that all the eigenvalues of the matrix are positive.*

Remark 2.9.4 *A nice consequence of a matrix A being positive definite is that its inverse A^{-1} exists. A matrix being diagonally dominant has very nice properties as well. To start, if A is strictly diagonally dominant then is non-singular meaning the system $A\boldsymbol{x} = \boldsymbol{b}$ has a unique solution. Further, Jacobi and Gauss-Seidel methods for solving a linear system of equations with this type of matrices always converge.*

2.9.1 General iterative method

Iterative methods are heavily used to find the numerical solution of linear systems of equations arising from many and variate physical problems. The general idea of these algorithms is to compute an iterate $\boldsymbol{x}^k$ which gradually approximates the true solution of the linear system. Each iterate is updated through the formula

$$\boldsymbol{x}^{k+1} = B\boldsymbol{x}^k + \boldsymbol{c}, \tag{2.60}$$

where B is known as the iteration matrix and $\boldsymbol{c}$ is a given vector, none of them depending on k. There are many different ways to construct B and $\boldsymbol{c}$ yielding different algorithms.

To express the above equation in algorithm form, we assume that there is an approximation $\boldsymbol{x}^k$ of $\boldsymbol{x}$ and define $\boldsymbol{r}^k = \boldsymbol{b} - A\boldsymbol{x}^k$ as the residual or defect equation for this approximation. We also suppose that it is possible to obtain an approximation $\bar{A}$ of A in such a way that the system $\bar{A}\boldsymbol{e} = \boldsymbol{r}$ is easier to solve than the original system $A\boldsymbol{x} = \boldsymbol{b}$. Hence, we obtain the following iterative algorithm :

Algorithm 1 Iterative Method.
```
1: Select an initial guess $\boldsymbol{x}^0$ and the maximum number of iterations $MAX$
2: for $k = 0$ to $MAX$ do
3:     Compute the residual equation $\boldsymbol{r}^k = \boldsymbol{b} - A\boldsymbol{x}^k$
4:     Solve the system $\bar{A}\boldsymbol{e}^k = \boldsymbol{r}^k$
5:     Update the value of with $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \boldsymbol{e}^k$
6: end for
```

The above is certainly similar to (2.60) with $B = I - (\bar{A})^{-1}A$. The asymptotic speed of convergence of the general iteration (2.60) is characterized by the spectral radius $\rho(B)$ defined as

$$\rho(B) = \max\{|\lambda| : \lambda \text{ eigenvalue of } B\}. \tag{2.61}$$

Asymptotically we have $\|\boldsymbol{x} - \boldsymbol{x}^{k+1}\| \leq \rho(B)\|\boldsymbol{x} - \boldsymbol{x}^k\|$ as $k \to \infty$. Of course, if $\rho(B) < 1$ the iterative scheme (2.60) will converge.

We now proceed to review four of the most popular iterative solvers; some of them we will be using throughout this thesis, they are : Jacobi (JAC), Gauss-Seidel (GS), Successive Over-relaxation (SOR) and Conjugate Gradient (CG) methods.

2.9.2 Jacobi method (JAC)

The Jacobi method results from splitting matrix A as

$$A = D + L + U, \tag{2.62}$$

where D, $-L$ and $-U$ represent the diagonal, lower-triangular, and upper-triangular parts of A, respectively. Then the following iteration is defined

$$x^{k+1} = D^{-1}b + D^{-1}(L + U)x^k. \tag{2.63}$$

Although the above matrix representation helps to easier the understanding of the method, in practice is very seldom implemented this way since for very large systems of equations it would require large amounts of memory. In these cases, the following point-wise formula is better recommended

$$x_i^{k+1} = \left(b_i - \sum_{j \neq i} a_{i,j} x_j^k \right) / a_{i,i}. \tag{2.64}$$

The Jacobi method is also known as the *method of simultaneous displacements* since the updates of x at each entry i can be done at the same time regardless the order. This certainly turns to be and advantage if a parallel implementation can be done. However, it is also true that x^k needs to be kept all the time until the updating of x^{k+1} has finished requiring for example twice the amount of memory used by the Gauss-Seidel method we shall now proceed to review.

2.9.3 Gauss-Seidel method (GS)

The matrix representation of the Gauss-Seidel method is

$$x^{k+1} = (D - L)^{-1}(Ux^k + b), \tag{2.65}$$

with D, $-L$ and $-U$ defined as above. Again, a point-wise formula is also better recommended to save computer memory and this is given by

$$x_i^{k+1} = \left(b_i - \sum_{j<i} a_{i,j} x_j^{k+1} - \sum_{j>i} a_{i,j} x_j^k \right) / a_{i,i}. \tag{2.66}$$

A first look into the above formula reveals that the new iterate in the GS method has a strong dependence upon all the previously evaluated spatial components on the grid. This is, each entry x_i^{k+1} depends upon x_j^{k+1} for all $j < i$. This fact, makes the order on which the equations are evaluated very important. Equation (2.66) describes the lexicographic order for the one dimensional case while the two dimensional case is illustrated on the left hand side of Figure 2.6. A different order of evaluation and very common in practice is the red-black ordering, this is shown on the rigth hand side of Figure 2.6.

21	22	23	24	25
16	17	18	19	20
11	12	13	14	15
6	7	8	9	10
1	2	3	4	5

11	24	12	25	13
21	9	22	10	23
6	19	7	20	8
16	4	17	5	18
1	14	2	15	3

Figure 2.6: Lexicographic and Red-Black ordering

One important advantage of the GS method is that there is no need to keep $x^{(k)}$ in memory until the updating of $x^{(k+1)}$ has finished. On the contrary, we delete every entry $x_i^{(k)}$ as soon as it is no longer needed. This represents an important saving in computer memory, particularly when solving large linear systems.

2.9.4 Successive over-relaxation method (SOR)

This method is nothing but a weighted average between two consecutive iterates of Gauss-Seidel iterations for each entry i, this is

$$x_i^{k+1} = w\bar{x}_i^{k+1} + (1-w)x_i^k, \tag{2.67}$$

where w denotes the given weight and $\bar{x}$ the Gauss-Seidel iterate. The idea is to find the best value of w which will accelerate the rate of convergence to the real solution.

In matrix form the SOR scheme is expressed as

$$x^{k+1} = (D-wL)^{-1}(wU + (1-w)D)x^k + w(D-wL)^{-1}b. \tag{2.68}$$

The value of w in principle must be selected in the interval $(0,2)$. Outside this interval, Kahan [78] proved that SOR does not converge. Sometimes if w is selected from $(0,1]$, it is said that under-relaxation is being applied. Although technically this is correct it is not always used and over-relaxation prevails for $w \in (0,2)$.

Some algorithms have been designed to compute the best value for w, see [158, 66].

2.9.5 Conjugate gradient method (CG)

When A is a symmetric and positive definite matrix, solving the linear system $A\boldsymbol{x} = \boldsymbol{b}$ is equivalent to finding the minimum of the quadratic function

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{b}, \tag{2.69}$$

where $\boldsymbol{x}^T$ indicates the transpose of vector $\boldsymbol{x}$.

The first order condition of (2.69) gives us the answer. This is, stationary points of $f(\boldsymbol{x})$ are attained when $f'(\boldsymbol{x}) = A\boldsymbol{x} - \boldsymbol{b} = \boldsymbol{0}$. Further, since $f(\boldsymbol{x})$ is quadratic, then it has only one stationary point and because of A being positive definite this point is a minimum.

Hence a line search optimization method can be applied to minimize $f(\boldsymbol{x})$. This is, $\boldsymbol{x}$ is updated using the equation

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha\boldsymbol{p}_k, \tag{2.70}$$

where $\boldsymbol{p}_k$ is line search direction and α is a line search parameter chosen to minimize $f(\boldsymbol{x}_k + \alpha\boldsymbol{p}_k)$ along $\boldsymbol{p}_k$. [5]

Due to the special features of $f(\boldsymbol{x})$, there is however no need to perform a line search since α can be chosen analytically. To this end, firstly we observe that the residual equation is simply

$$\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x} = -\nabla f(\boldsymbol{x}), \tag{2.71}$$

and secondly, the minimum over α of $f(\boldsymbol{x}_{k+1})$ occurs precisely when the residual $\boldsymbol{r}_{k+1}$ is orthogonal to the search direction $\boldsymbol{p}_k$. This is,

$$\begin{aligned} \frac{d}{d\alpha}f(\boldsymbol{x}_{k+1}) &= \nabla f(\boldsymbol{x}_{k+1})\frac{d}{d\alpha}(\boldsymbol{x}_{k+1}) \\ &= (A\boldsymbol{x}_{k+1} - \boldsymbol{b})^T\frac{d}{d\alpha}(\boldsymbol{x}_k + \alpha\boldsymbol{p}_k) = -\boldsymbol{r}_{k+1}\boldsymbol{p}_k = 0. \end{aligned} \tag{2.72}$$

Thus by expressing $\boldsymbol{r}_{k+1}$ in terms of the old residual and the search direction,

$$\boldsymbol{r}_{k+1} = \boldsymbol{b} - A\boldsymbol{x}_{k+1} = \boldsymbol{b} - A(\boldsymbol{x}_k + \alpha\boldsymbol{p}_k) = \boldsymbol{r}_k - \alpha A\boldsymbol{p}_k, \tag{2.73}$$

we can solve (2.70) for

$$\alpha = \frac{\boldsymbol{r}_k^T\boldsymbol{p}_k}{\boldsymbol{p}_k^T A\boldsymbol{p}_k}. \tag{2.74}$$

Algorithm 2 express the conjugate gradient method in computational form

[5]Notice, in (2.70) we moved the iteration index from the top to the bottom. That is, in $\boldsymbol{x}_k$, k denotes the k^{th} iterate of $\boldsymbol{x}$. We will use this notation only in this section.

Algorithm 2 Conjugate Gradient Algorithm

```
1: Select an initial guess $\boldsymbol{x}_0$ and the maximum number of iterations $MAX$
2: Compute the initial residual $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$
3: $\boldsymbol{p}_0 = \boldsymbol{r}_0$
4: for $k = 0$ to $MAX$ do
5:     $\alpha_k = \frac{\boldsymbol{r}_k^T \boldsymbol{r}_k}{\boldsymbol{p}_k^T A \boldsymbol{p}_k}$
6:     $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$
7:     $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k A \boldsymbol{p}_k$
8:     $\beta_k = \boldsymbol{r}_{k+1}^T \boldsymbol{r}_{k+1} / \boldsymbol{r}_k^T \boldsymbol{r}_k$
9:     $\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k$
10: end for
```

2.10 Multigrid method

One of the main contributions of this thesis is to show how multigrid methods can be implemented and used to obtain fast solutions of systems of equations (not necessarily linear) arising from the discretization of partial differential equations with particular interest to image restoration problems.

To illustrate the basics of multigrid algorithms we start with an heuristic explanation of how they approach the solution of linear systems of equations. Thus the goal is to solve the linear system

$$L\boldsymbol{u} = \boldsymbol{f}. \tag{2.75}$$

We assume having in our hands a good iterative solver for this linear system. At this moment, the type of this iterative solver is not important although, as we will explain later on, it will show to be of great relevance for the performance of the multigrid algorithm. Actually, we will require from the iterative algorithm to be a good *smoother*[6].

Thus, after applying some iterations of our solver to the system (2.75) we obtain an approximate solution $\bar{\boldsymbol{u}}$ which defines the error

$$\boldsymbol{e} = \boldsymbol{u} - \bar{\boldsymbol{u}}. \tag{2.76}$$

Since L is a linear operator the following relation can be derived and is known as the residual or defect equation:

$$L\boldsymbol{e} = L(\boldsymbol{u} - \bar{\boldsymbol{u}}) = \boldsymbol{f} - L\bar{\boldsymbol{u}} = \boldsymbol{r}. \tag{2.77}$$

From here we see that if we can solve exactly the residual equation then we can

[6]The term *smoother* is commonly used in the theory of multigrid algorithms and is used to describe an algorithm capable of reducing the high frequencies of the error fast and efficiently.

obtain u through

$$u = \bar{u} + e. \tag{2.78}$$

Of course solving the residual equation (2.77) as it is, does not represent any advantage since it can be as difficult to solve as the original system (2.75).

To tackle this problem, recall that the iterative methods we introduced in Section 2.9, approximate L with some $\bar{L}$ of the same size in such a way that $\bar{L}e = r$ is easier to solve.

Multigrid methods on the other hand, use a completely different approach. In multigrid methods the idea is first to obtain a lower resolution version of the residual equation (2.77), then solve exactly for the error and interpolate it to the original resolution. In multigrid methods, the system $Le = r$ might be not easier to solve in its low resolution version, but at least is computationally less expensive since the number of variables has been reduced. Further, this idea can be applied recursively to lower levels obtaining optimal performance.

The idea we just described in the previous paragraph is the foundation of what is known as the *two-grid correction scheme*, where grid here means the discrete domain where the problem has been formulated and defines the level of resolution.

To express the *two-grid correction scheme* in mathematical terms, denote by $\Omega = [0,m] \times [0,n]$ the continuous domain where the problem has been originally formulated an let $\boldsymbol{h} = (h_x, h_y)$ represent a vector of finite mesh sizes. We define the infinite grid G_h as

$$G_h = \{(x,y) : x = x_i = ih_x, \;\; y = y_j = jh_y; \;\; i,j \in \mathbb{Z}\}, \tag{2.79}$$

the discrete grid as $\Omega_h = \Omega \cap G_h$ and

$$u_h = u_h(x,y) = u_h(x_i,y_j) = u_h(ih_x, jh_y) \tag{2.80}$$

the discretized version of any function u defined on Ω_h. We usually refer to u_h and f_h as grid functions.

Now, denote by L_h the representation of L on the grid Ω_h which is commonly called the fine-grid. Suppose now that L can be approximated on a coarser grid Ω_H with $H > h$ and let us represent by L_H this approximation. Assume we have two transfer operators R_h^H and I_H^h capable to transport vectors from Ω_h to Ω_H and viceversa. Then, the *two-grid correction scheme* is defined in Algorithm 3.

There are issues that need to be addressed in the above scheme. One is the approximation of operator L on coarser grids and under what conditions it can be well approximated. Another is the construction of good transfer operators to satisfactorily transport r, $\bar{u}$ and e between consecutive grids. Notice that if these vectors are too oscillatory then high order interpolation operators would be required increasing the

Algorithm 3 Two-Grid Correction Scheme

```
1: Solve approximately $L_h u_h = f_h$ to obtain an approximate solution $\bar{u}_h$ on $\Omega_h$
2: Compute the fine-grid residual $r_h = f_h - L_h \bar{u}_h$,
3: Transport this residual to the coarser grid by $r_H = R_h^H r_h$,
4: Solve exactly $L_H e_H = r_H$ on the coarse grid $\Omega_H$,
5: Transport the error to the fine grid by $e_h = I_H^h e_H$,
6: Correct the fine grid approximation by $\bar{u}_h = \bar{u}_h + e_h$,
7: Solve approximately $L_h u_h = f_h$ with initial guess $\bar{u}_h$.
```

computational cost of the algorithm. There is also the possibility of them not being good enough to obtain a fair representation on the next grid.

The above discussion leads to the idea that it would be nice to have smooth residuals since they are easy to interpolate using simple linear or bilinear operators. This automatically imposes the condition in multigrid algorithms of selecting or designing iterative solvers capable of *smoothing* (eliminating the high frequencies) the residual as much as possible. Not only that, we expect to apply only a few iterations on each level so the *smoothing* process needs to be very efficient.

The smoothing property required for the iterative solvers although it looks as a very restrictive condition, for some problems (mainly linear) it is not. On the contrary, it can be taken as an advantage since some of the iterative algorithms we introduced in Section 2.9 like Jacobi, Gauss-Seidel and SOR methods are good smoothers for linear problems with constant or smooth coefficients. It is well-known [66, 116] that these methods eliminate rapidly the high frequency components of the error, but are very slow to converge as stand-alone solvers since they struggle to reduce the low frequency components. In multigrid algorithms this undesirable property is no longer that important since, as we will show shortly, due to the coarsening applied, only high frequency components are distinguishable on coarser grids.

As we mentioned above JAC, GS and SOR are good smoothers for linear problems. However, difficulties start appearing when the problems to solve have discontinuous coefficients, or they are nonlinear. For them constructing efficient smoothers can be very hard. Usually JAC, GS or SOR are not enough, so fixed point methods need to be used as smoothers. This thesis deals with this type of problems in Chapters 4-7 by introducing novel fixed point algorithms in each case.

From the correction scheme described in Algorithm 3 one obtains the iteration operator for the two-grid (h, H) cycle:

$$M_h^H = S_h^{\nu_1} K_h^H S_h^{\nu_2} \text{ with } K_h^H = I_h - R_H^h L_H^{-1} I_h^H L_h, \tag{2.81}$$

where I_h is the identity operator and S_h^{ν} represents the action of ν steps of the iterative solver over u_h to obtain the approximation $\bar{u}_h$ as in step 1 of Algorithm 3. Sometimes, when convenient, we will also denote this action by

$$\bar{u}_h = SMOOTH^{\nu}(u_h, L_h, f_h). \tag{2.82}$$

2.11 Multigrid components

We already know the different components involved in a multigrid algorithm so it is time to start giving an exact definition of each one of them. There are many ways to select these components and they usually depend on the problem to be solved. Due to the list being long, in this section, we give details of the basic multigrid components only for those we will use in this thesis.

2.11.1 Choices of grids

The first decision one has to make is related to the discretization of the problem and therefore the choice of the structure of the grid.

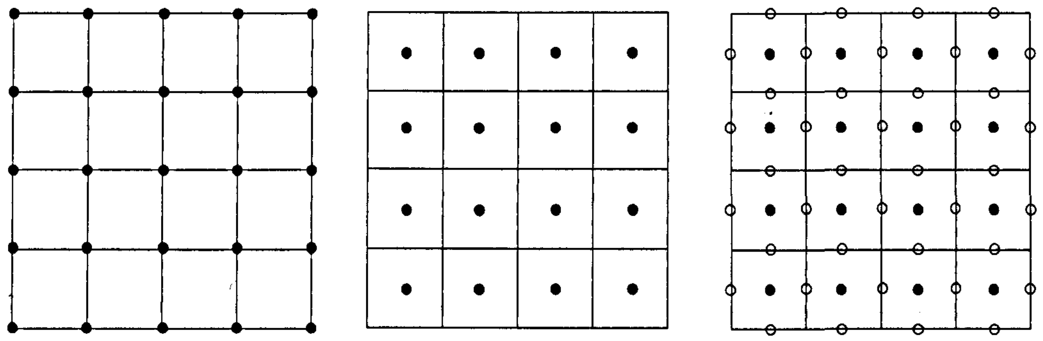


Figure 2.7: Left-hand side: A vertex-centered grid where the unknowns (•)are located at the vertices of the grid. Center: A cell-centered grid with the unknowns (•) defined at the center of the cells. Right-hand side: A staggered grid with two unknowns (• and o); they are defined at different locations within the grid.

There are different ways to construct grids and the final selection has an immediate effect on how the problem will be discretized. Usually the choice of the grid depends on the application, the boundary conditions and many other considerations. For instance, finite difference methods are traditionally used in the context of structured Cartesian grids. For these, there is also the choice of where to locate the unknowns within the grid. In the Figure 2.7 we illustrate the most preferred choices: vertex-centered grid, cell-centered grid and staggered grid.

In vertex-centered grids the unknowns are defined at the vertices of the grid. In cell-centered grids, on the contrary, the location of the unknowns is at the center of each cell. This type of grids is in particular suitable for image processing algorithms since images are already intrinsically arranged in this way. That is, each pixel of the image can be represented as a cell within the cell-centered grid. Finally, staggered grids are useful when different unknowns are located at different locations within the grid. This can be particularly useful for some problems, among them those of high order.

2.11.2 Choices of coarse grids

The simplest and most frequently way to construct a coarse grid Ω_H is by doubling the mesh size h in every direction. This technique is known as *standard coarsening* and is illustrated on the left-hand side of Figure 2.8. In this thesis all the multigrid algorithms we develop use this technique.

In some cases, like when solving anisotropic problems, there is a strong coupling between variables in some direction. In this situation if the coupling between variables is known *a priori* or it can be easily identified then techniques such as semi-coarsening are very helpful. The right-hand side of Figure 2.8 illustrates semi-coarsening in the x-direction.

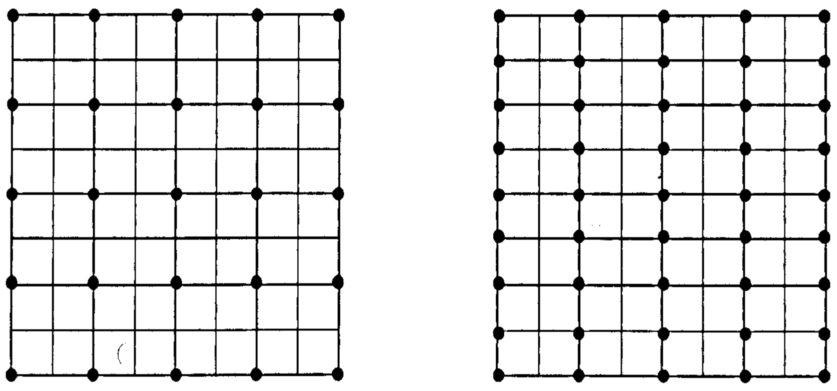


Figure 2.8: Left-hand side: *standard coarsening* where the coarse grid is constructed by doubling the mesh size in the vertical and horizontal directions. Right-hand side: An example of *semi-coarsening* where the coarse grid is obtained by doubling the mesh size only in the x-direction. In both, unknowns are represented by (•).

2.11.3 Transfer operators

The choice of the intergrid transfer operators is closely related to the choice of the coarse grid. There are different types for each case. For instance *injection*, *full-weighting* and *half-weighting* operators for restriction are the most commonly used. The equivalents for interpolation are *linear* and *bilinear* operators, see [139] for details.

In this thesis, we work only with standard coarsening between cell-centered grids and we mainly use the *four-point average* operator for restriction and the *bilinear* operator for interpolation. Thus, in the following, we give the details of their implementation.

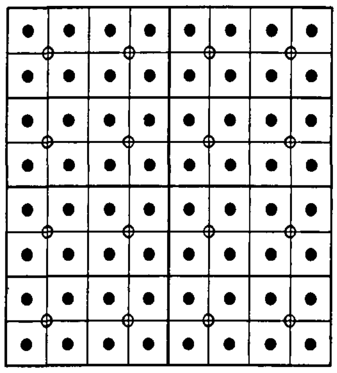
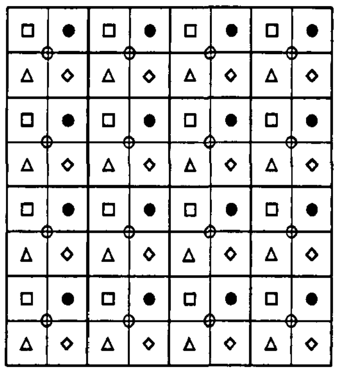



Figure 2.9: Left-hand side: A fine-grid with the symbols for the fine and coarse grid points corresponding to the formula (2.84). Fine-grid points are labeled as (•) and coarse-grid points as (∘). Right-hand side: A fine-grid with a graphical description (through symbols) of the bilinear interpolation used to transfer from the coarse-grid (∘) to the fine-grid (•, ◇, △, □).

Restriction - four point average

Applying this operator to a grid function $(u_h)_{i,j} = u_h(x,y)$ at a coarse grid point $(i,j) = (x,y) \in \Omega_{2h}$ is expressed as

$$I_h^{2h} u_h = u_{2h}, \tag{2.83}$$

and means

$$(u_{2h})_{i,j} = \frac{1}{4}\left[(u_h)_{2i-1,2j-1} + (u_h)_{2i-1,2j} + (u_h)_{2i,2j-1} + (u_h)_{2i,2j}\right]. \tag{2.84}$$

Interpolation - bilinear

Similarly, the *bilinear* interpolation operator for cell-centered discretizations is expressed as

$$I_{2h}^h u_{2h} = u_h \text{ where,} 1 \leq i \leq n/2, \ \ 1 \leq j \leq m/2, \tag{2.85}$$

and defined by

$$\begin{aligned}
(u_h)_{2i,2j} &= \left[9(u_{2h})_{i,j} + 3[(u_{2h})_{i+1,j} + (u_{2h})_{i,j+1}] + (u_{2h})_{i+1,j+1}\right]/16, \text{ for } \bullet \\
(u_h)_{2i-1,2j} &= \left[9(u_{2h})_{i,j} + 3[(u_{2h})_{i-1,j} + (u_{2h})_{i,j+1}] + (u_{2h})_{i-1,j+1}\right]/16, \text{ for } \Box \\
(u_h)_{2i,2j-1} &= \left[9(u_{2h})_{i,j} + 3[(u_{2h})_{i+1,j} + (u_{2h})_{i,j-1}] + (u_{2h})_{i+1,j-1}\right]/16, \text{ for } \Diamond \\
(u_h)_{2i-1,2j-1} &= \left[9(u_{2h})_{i,j} + 3[(u_{2h})_{i-1,j} + (u_{2h})_{i,j-1}] + (u_{2h})_{i-1,j-1}\right]/16, \text{ for } \triangle .
\end{aligned} \tag{2.86}$$

Graphically, both transfer operators are described in the Figure 2.9.

2.11.4 The smoother

Recall from the analysis of the two-grid cycle in Section 2.10, that it would be desirable to have an iterative solver that eliminates the high frequency components of the residual quickly and efficiently. Iterative algorithms with this property are called *smoothers* as we have remarked before. Coarsening also makes enough to have as iterative solver a smoother since low frequencies on one grid are not seen on the next coarser grid.

For linear problems with smooth coefficients, it is well-known that Gauss-Seidel, Jacobi and SOR methods are good smoothers. For nonlinear problems however it is not an easy task to implement a good one.

The smoothing properties of the Gauss-Seidel method will be better understood after reviewing the following example. Similar analysis can be done for the others.

Example 2.11.1 *Consider the one dimensional problem defined by the following discrete equations*

$$\begin{aligned} -u_{i-1} + 2u_i - u_{i+1} = 0, \qquad 1 \le i \le n-1, \\ u_0 = u_n = 0. \end{aligned} \tag{2.87}$$

The exact solution $\boldsymbol{u} = (u_0, \dots, u_n)$ for this problem is $\boldsymbol{u} = \boldsymbol{0}$ (the zero vector), so the error $\boldsymbol{e}$ for this problem is simply $\boldsymbol{e} = -\boldsymbol{v}$ the approximation of $\boldsymbol{u}$ at some stage. Now we will study the performance of lexicographic Gauss-Seidel (GSLEX) method for different selections of the initial guess $\boldsymbol{v}^0 = (v_0^0, \dots, v_n^0)$. In particular, we are interested to see how GSLEX is affected when the initial guess changes its frequency. To this end, we construct an initial guess consisting of the vectors (also called Fourier modes)

$$v_i^0 = \sin\left(\frac{ik\pi}{n}\right), \qquad 0 \le i \le n, \qquad 1 \le k \le n-1. \tag{2.88}$$

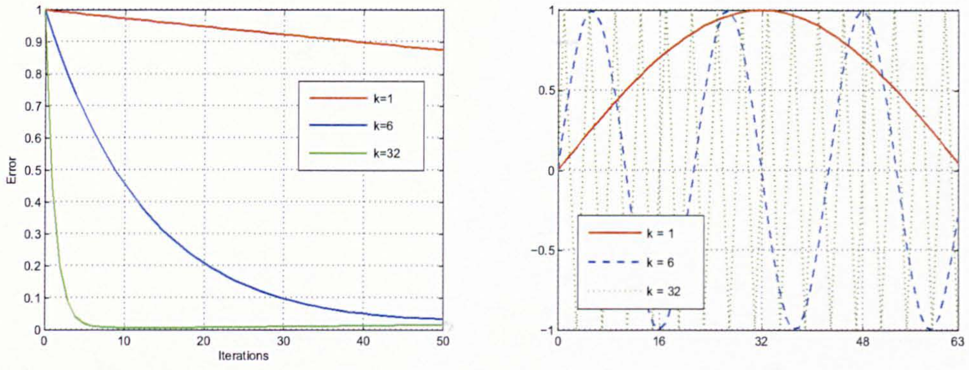
Here, the integer k represents the wavenumber, or frequency of $\boldsymbol{v}^0$. For instance, small k yields a vector $\boldsymbol{v}^0$ with few oscillations along the one-dimensional grid, whilst large k yields a highly oscillatory $\boldsymbol{v}^0$. In the Figure 2.10(b) we display $\boldsymbol{v}^0$ for different values of k.

Thus we implement GSLEX as it was described in Section 2.9.3 with $n = 64$ and apply some relaxation steps to the system (2.87). We display in the Figure 2.10(a) the evolution of the error for $k = 1, 6, 32$.

It is quite clear from Figure 2.10(a) that GSLEX is very quick in reducing the error for small wavenumbers (high frequency), but struggles to decrease it for large wavenumbers (low frequency). For this reason iterative algorithms as GSLEX and JAC and SOR which share similar properties are called smoothers *and are the perfect relaxation schemes for multigrid methods.*

2.12 The multigrid cycle

In Section 2.10 we introduced the two-grid cycle also known as the two-grid correction scheme and explained the principles it is based on. One interesting observation is that



(a) (b)

Figure 2.10: (a) Performance of GSLEX for different wavenumbers $k = 1, 6, 32$. We plot the L^2-norm of the error $\boldsymbol{e}$ against the number of iterations. (b) Here we illustrate the shape of the Fourier mode $\boldsymbol{v}^0$ (size $n = 64$) for the selected values of k. The larger the value of k the more oscillatory $\boldsymbol{v}^0$ is.

by assuming a convergent two-grid cycle there is no need to solve the coarsest grid exactly for it to converge. This immediately and naturally induces the idea of using a different two-grid cycle to solve the coarsest problem. Even more we can apply this method recursively to construct a hierarchy of coarser grids

$$\Omega_{h_\ell}, \Omega_{h_{\ell-1}}, \ldots, \Omega_{h_0}, \tag{2.89}$$

characterized by a sequence of mesh sizes $h_p : p = \ell, \ell - 1, \ldots, 0$ [7]. This way the finest grid $\Omega_h = \Omega_{h_\ell}$ is characterized by $h = h_\ell$ and the coarsest grid Ω_{h_0} by the mesh size h_0. Following [139], to simplify notation we replace the index h_p by h, so now h is the one which varies from ℓ to 0.

Then, assuming there are intergrid transfer operators R_h^{h-1} and I_{h-1}^h as defined before for each Ω_h, we describe in Algorithm 4 the *multigrid cycle* (*MGCY*) scheme to solve (2.75) for a fixed $\ell > 1$.

Depending on the value of γ the structure of the multigrid cycle can take different shapes. In the Figure 2.11, we illustrate with the help of some diagrams different structures of one cycle of a multigrid method. Cycles with a V-shape structure are called V-cycles and are characterized by selecting $\gamma = 1$. Cycles with a W-shape structure are called W-cycles and for them $\gamma = 2$. Those two are the most commonly used although many differents exist. The number γ is called the cycle index.

[7]Within this section, we borrow notation from [139] since we found it very useful to write the equations and algorithms of the multigrid method clearly.

Algorithm 4 $u_h^{k+1} = MGCYC(h, \gamma, u_h^k, L_h, f_h, \nu_1, \nu_2)$

```
(1) Pre-smoothing
Obtain $\bar{u}_h^k$ by applying $\nu_1$ smoothing steps to $u_h^k$. That is,
    $\bar{u}_h^k = SMOOTH^{\nu_1}(u_h^k, L_h, f_h).$    (2.90)
(2) Coarse grid correction
Compute the residual $\bar{r}_h^k = f_h - L_h \bar{u}_h^k$.
Restrict the residual $\bar{r}_{h-1}^k = R_h^{h-1} \bar{r}_h^k$.
Compute an approximate solution $\hat{v}_{h-1}^k$ of the residual equation on $\Omega_{h-1}$.
    $L_{h-1} \hat{v}_{h-1}^k = \bar{r}_{h-1}^k$ by    (2.91)
if $h = 1$ then
    use a direct or fast iterative solver for (2.91).
else if $h > 1$ then
    solve (2.91) approximately by performing $\gamma$ $h$-grid cycles using the zero grid function as first approximation i.e.,
        $\hat{v}_{h-1}^k = MGCYC^{\gamma}(h-1, \gamma, 0, L_{h-1}, \bar{r}_{h-1}^k, \nu_1, \nu_2).$    (2.92)
end if
Interpolate the correction $\hat{v}_h^k = I_{h-1}^h \hat{v}_{h-1}^k$.
Compute the corrected approximation on $\Omega_h$, $u_h^k = \bar{u}_h^k + \hat{v}_h^k$.
(3) Post-smoothing
Compute $u_h^{k+1}$ by applying $\nu_2$ smoothing steps to $u_h^k$.
    $u_h^{k+1} = SMOOTH^{\nu_2}(u_h^k, L_h, f_h).$    (2.93)
```

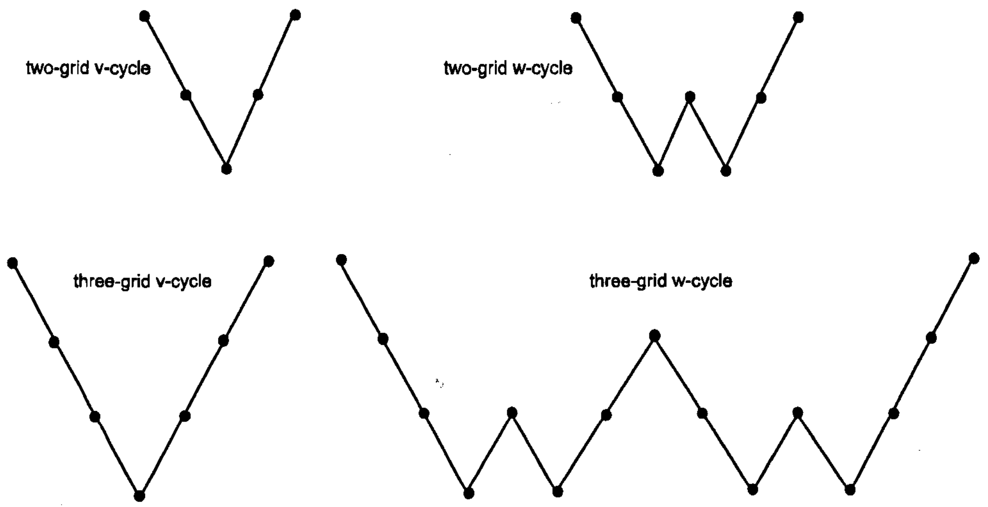


Figure 2.11: Top-row: Illustration of multigrid cycles of two levels with V ($\gamma = 1$) and W ($\gamma = 2$) shape structures. Bottom-row: Illustration of multigrid cycles of three levels, again with V and W shape structures.

2.13 Computational work

In the following, we will estimate the computational work of a V-cycle multigrid method. We restrain ourselves to this structure since all the multigrid algorithms we develop in this thesis are based on it. A more general derivation for different values of γ and therefore different shape structures can be found in [139]. Usually the cost of a cycle is expressed using work units (WU) which may be defined in different ways. Here we define a WU as the computational cost of performing a relaxation sweep on the finest grid. Usually the cost of the intergrid transfer operators is neglected since it amounts to no more than 20% of the cost of the entire cycle.

Thus, for a problem defined in a d-dimensional grid and considering a V-cycle with $\nu_1 = \nu_2 = 1$, each grid Ω^{ph} requires p^{-d} WU. Adding these costs gives

$$\text{V-cycle cost} = 2\left\{1 + 2^{-d} + 2^{-2d} + \cdots + 2^{-nd}\right\} < \frac{2}{1 - 2^{-d}} WU. \tag{2.94}$$

For instance, a single V-cycle has a cost of about $\frac{8}{3}$ WU for 2-dimensional problem, as the ones we work with here.

2.14 Full multigrid

A full multigrid algorithm (FMG) is designed to obtain a very good initial guess for iterative solvers including other multigrid methods. Full multigrid is based on the idea of nested iteration, this is, given the coarse grid Ω_h, one can apply a multigrid cycle (say a V-cycle) to obtain an approximate solution u_h at this h-level, then this u_h is interpolated to the next finer grid Ω_{h+1} to be used as initial guess for another multigrid cycle at the $h+1$-level. This process is carried out until reaching the finest level.

Notice that in FMG is the solution u_h and not the error e_h the one that is interpolated to the next finer level. Usually the operator used to interpolate the solution is denoted by Π_{h-1}^h and is of higher accuracy than the interpolation operators used within the multigrid iteration.

Typically, the FMG scheme is the most efficient multigrid version. In Algorithm 5 we detail this scheme where r is the number of MGCYC cycles to be used.

Algorithm 5 Full Multigrid

```
For $h = 0$,
Solve $L_0 u_0 = f_0$, providing $u_0^{FMG} = u_0$.
for $h = 1$ to $\ell$ do
    $u_h^0 = \Pi_{h-1}^h u_{h-1}^{FMG}$
    $u_h^{FMG} = MGCYC^r(h+1, \gamma, u_h^0, L_h, f_h, \nu_1, \nu_2)$.
end for
```

Here u_h^{FMG} denotes the resulting FMG approximation on grid Ω_h.

In FMG schemes, usually an appropriate interpolation operator Π_{h-1}^h is the Bicubic operator [110] since it provides enough accuracy and is not computationally very expensive.

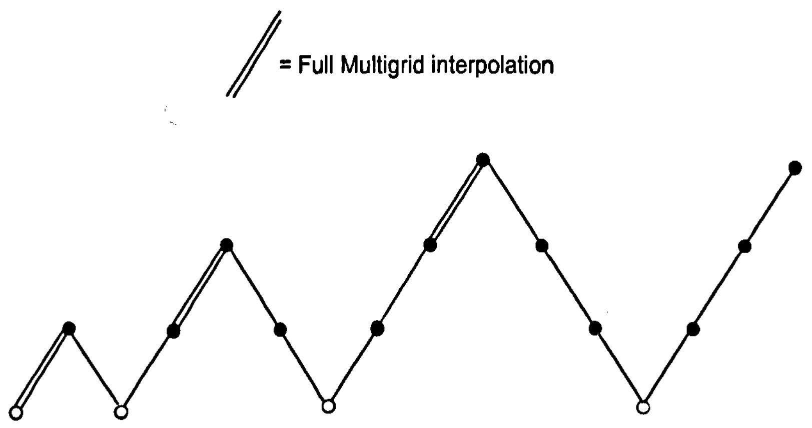


Figure 2.12: The scheme illustrates the typical structure of a full multigrid algorithm.

2.15 Local Fourier analysis

Local Fourier analysis (LFA) is a powerful tool for the analysis of multigrid methods. Due to the use of Fourier functions, this technique can be regarded as restricted to the analysis of only linear problems with constant coefficients defined on an infinite grid. However, as remarked by some authors [139, 152], LFA is still a very useful tool for the analysis of nonlinear with nonconstant coefficients problems. It is the *locality* assumption which precisely let us relax the strong assumptions of the linear coefficients and the infinite grid and extend the applicability of LFA to a more general range of problems. In [139] the local nature of LFA is described as follows:

Under general assumptions any general discrete operator, nonlinear, with nonconstant coefficients can be linearized locally *and can be replaced* locally *(by freezing the coefficients) by an operator with constant coefficients [139].*

The above assumption let us transform a nonlinear problem with nonconstant coefficients into a linear problem with constant coefficients at least locally and therefore applying Fourier analysis to this new linear problem is now valid.

Thus we have tackled the first two difficulties: nonlinearity and nonconstant coefficients. The third difficulty related to the finite grid and its associated boundary conditions can be avoided by assuming that:

Relaxation is a local process and therefore the update of the unknown is carried out using information from nearby neighbors. This allows to neglect the effect of the boundary conditions by replacing the finite domain (or grid) of the problem by and infinite domain [22].

So we have already justified the validity of LFA for both linear problems with constant coefficients and under some assumptions its validity for nonlinear problems with nonconstant coefficients. For the latter type of problems (precisely the ones we are interested in this thesis), the results obtained from LFA must be interpreted as an indication of possible good or poor performance of the multigrid algorithm under analysis. Usually a set of carefully designed experiments need to be carried out to obtain trustable and significative results.

We proceed now to explain with certain amount of detail the foundations of the LFA. Plenty of information in this respect can be found for instance in [139, 22, 152, 20] and references therein.

In the two-dimensional case, the one of our interest in this thesis, LFA studies the actions of linear operators over grid functions characterized by

$$\varphi(\boldsymbol{\theta}, \boldsymbol{x}) = e^{i\boldsymbol{\theta}\boldsymbol{x}/\boldsymbol{h}} = e^{i\theta_1 x_1/h_1} e^{i\theta_2 x_1/h_2}, \tag{2.95}$$

where $\boldsymbol{x} = (x_1, x_2) = (x, y)$, $\boldsymbol{\theta} = (\theta_1, \theta_2)$ and $\boldsymbol{h} = (h_1, h_2) = (h_x, h_y)$ is the vector of mesh size associated to the infinite grid

$$\boldsymbol{G}_h = \{\boldsymbol{x} = \boldsymbol{k}\boldsymbol{h} = (k_1 h_1, k_2 h_2),\ \ \boldsymbol{k} \in \mathbb{Z}^2\}. \tag{2.96}$$

The operator to be analyzed L_h is defined on $\boldsymbol{G}_h$ and is represented as a linear combination (or stencil) of elements of the grid function $\boldsymbol{u}_h$. That is,

$$L_h \boldsymbol{u}_h(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in V} s_{\boldsymbol{k}} \boldsymbol{u}_h(\boldsymbol{x} + \boldsymbol{k}\boldsymbol{h}) \tag{2.97}$$

with constant coefficients $s_k \in \mathbb{R}$ (or $\mathbb{C}$) and V a finite index set. Assuming that $\boldsymbol{\theta}$ varies continuously in $\mathbb{R}^2$, it is not difficult to see that

$$\varphi(\boldsymbol{\theta}, \boldsymbol{x}) = \varphi(\boldsymbol{\theta}', \boldsymbol{x}) \quad \boldsymbol{x} \in \boldsymbol{G}_h \tag{2.98}$$

when the difference between θ_1 and θ_1' and the difference between θ_2 and θ_2' are multiples of 2π. This periodic nature of the grid functions $\varphi(\boldsymbol{\theta}, \boldsymbol{x})$ makes enough to consider

$$\varphi(\boldsymbol{\theta}, \boldsymbol{x}) \text{ with } \boldsymbol{\theta} \in [-\pi, \pi)^2. \tag{2.99}$$

Lemma 2.15.1 *For $-\pi \leq \boldsymbol{\theta} \leq \pi$, all grid functions $\varphi(\boldsymbol{\theta}, \boldsymbol{x})$ are eigenfunctions of any discrete operator which can be described by a difference stencil as in [139]. The relation*

$$L_h\varphi(\boldsymbol{\theta}, \boldsymbol{x}) = \tilde{L}_h(\boldsymbol{\theta})\varphi(\boldsymbol{\theta}, \boldsymbol{x}) \quad \textit{with} \quad \boldsymbol{x} \in \boldsymbol{G}_h \tag{2.100}$$

holds with

$$\tilde{L}_h(\boldsymbol{\theta}) = \sum_{\boldsymbol{k}} s_{\boldsymbol{k}} e^{i\boldsymbol{\theta}\boldsymbol{k}}. \tag{2.101}$$

The proof of this lemma can be found in [139]. $\tilde{L}_h(\boldsymbol{\theta})$ is known as the formal eigenvalue, or the *symbol* of the operator L_h.

Example 2.15.2 *Assuming $h = h_x = h_y$, the symbol for the discrete Laplace operator $L_h = -\Delta_h$ defined by the stencil*

$$L_h = \frac{1}{h^2}\begin{pmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{pmatrix}$$

is computed as follows:

$$\begin{aligned} L_h\varphi(\boldsymbol{\theta}, \boldsymbol{x}) = L_h e^{i\boldsymbol{\theta}\boldsymbol{x}/h} &= \left\{\frac{1}{h^2}(4 - (e^{i\theta_1} + e^{i\theta_2} + e^{-i\theta_1} + e^{-i\theta_2}))\right\} e^{i\boldsymbol{\theta}\boldsymbol{x}/h} \\ &= \tilde{L}_h(\boldsymbol{\theta})e^{i\boldsymbol{\theta}\boldsymbol{x}/h} = \tilde{L}_h(\boldsymbol{\theta})\varphi(\boldsymbol{\theta}, \boldsymbol{x}). \end{aligned} \tag{2.102}$$

Therefore the symbol of the standard discrete Laplace operator is given by

$$\tilde{L}_h(\boldsymbol{\theta}) = \frac{1}{h^2}(4 - (e^{i\theta_1} + e^{i\theta_2} + e^{-i\theta_1} + e^{i\theta_2})) \; = \frac{2}{h^2}(2 - (\cos(\theta_1) + \cos(\theta_2))). \tag{2.103}$$

2.15.1 Smoothing analysis

In the smoothing analysis, one tries to estimate how fast the high frequencies of the error are eliminated by a given iterative solver. When applied in the context of a multigrid algorithm an extra and important factor to consider is the type of coarsening being used by the multigrid method under consideration. This is, how the coarse grid $\boldsymbol{G}_H$ is constructed. The most common and easiest to implement type of coarsening is standard coarsening. In this thesis we only use this method.

Thus, assuming that we apply this standard coarsening to $\boldsymbol{G}_h$, the obtained coarse grid $\boldsymbol{G}_H$ is defined as

$$\boldsymbol{G}_H = \{\boldsymbol{x} = \boldsymbol{k}\boldsymbol{H} \text{ with } \boldsymbol{k} \in \mathbb{Z}^2\}. \tag{2.104}$$

Recall from Sections 2.10 and 2.11.2 that due to coarsening only high frequency components are distinguishable on the coarser grid. In particular, standard coarsening partitions the frequency interval $\Theta = [-\pi, \pi)^2$ in the two sub-intervals

$$\Theta^{low} = \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2 \text{ and } \Theta^{high} = [-\pi, \pi) \setminus \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2. \tag{2.105}$$

That is, $\Theta = \Theta^{low} \cup \Theta^{high}$ as illustrated in Figure 2.13.

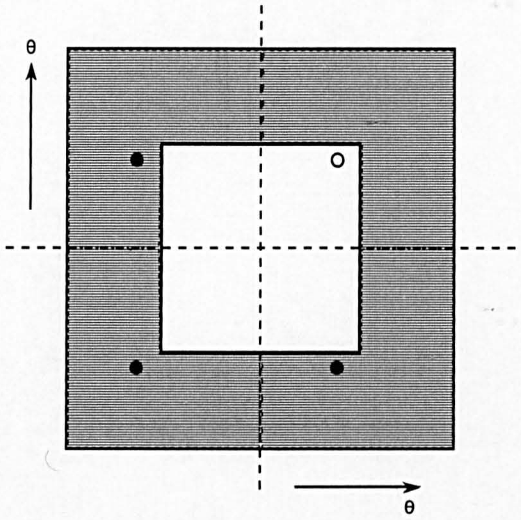


Figure 2.13: Standard coarsening partitions the Θ frequency interval (white plus shaded region) in two sub-intervals: Θ^{low} (white region) and Θ^{high} (shaded region). For a given low frequency $\boldsymbol{\theta}(\circ)$ the three high frequencies $\boldsymbol{\theta}$ for which the corresponding $\varphi(\boldsymbol{\theta}, \boldsymbol{x})$ coincide on $\boldsymbol{G}_h$ are marked by $\bullet$.

For the smoothing and the two-grid analysis this phenomenon is important since only those frequency components

$$\varphi(\boldsymbol{\theta}, \cdot) \text{ with } [-\pi/2, \pi/2)^2 \tag{2.106}$$

are distinguishable on $\boldsymbol{G}_H$. That is, for each $\boldsymbol{\theta}' \in [-\pi/2, \pi/2)^2$, three other frequency components $\varphi(\boldsymbol{\theta}, \cdot)$ with $\boldsymbol{\theta} \in [-\pi, \pi)^2$ coincide on $\boldsymbol{G}_H$ with $\varphi(\boldsymbol{\theta}', \cdot)$ and are not distinguishable on $\boldsymbol{G}_H$.

One thing we must consider is that to be able to apply Fourier smoothing analysis to a given relaxation method when solving the system $L_h \boldsymbol{u}_h = \boldsymbol{f}_h$, we need to assume that this relaxation method can be written locally as

$$L_h^+ \hat{\boldsymbol{u}}_h + L_h^- \bar{\boldsymbol{u}}_h = \boldsymbol{f}_h, \tag{2.107}$$

where $\hat{\boldsymbol{u}}_h$ corresponds to the approximation of $\boldsymbol{u}_h$ before relaxation and $\bar{\boldsymbol{u}}_h$ the approximation after relaxation has been applied. Thus, the relaxation is characterized by

$$L_h = L_h^+ + L_h^- . \tag{2.108}$$

Example 2.15.3 *Using as relaxation method Gauss-Seidel with lexicographic order applied to the Laplace operator $L_h = -\Delta_h$, the splitting reads,*

$$L_h^+ = \frac{1}{h^2}\begin{pmatrix} & 0 & \\ -1 & 4 & 0 \\ & -1 & \end{pmatrix} \text{ and } L_h^- = \frac{1}{h^2}\begin{pmatrix} & -1 & \\ 0 & 0 & -1 \\ & 0 & \end{pmatrix}.$$

Lemma 2.15.4 *Under the assumptions (2.107) and (2.108), all $\varphi(\boldsymbol{\theta},\cdot)$ with $\tilde{L}_h^+(\boldsymbol{\theta}) \neq 0$ are eigenfunctions of S_h*

$$S_h\varphi(\boldsymbol{\theta},\boldsymbol{x}) = \tilde{S}_h(\boldsymbol{\theta})\varphi(\boldsymbol{\theta},\boldsymbol{x}) \tag{2.109}$$

with the amplification factor

$$\tilde{S}_h(\boldsymbol{\theta}) = -\frac{\tilde{L}_h^-(\boldsymbol{\theta})}{\tilde{L}_h^+(\boldsymbol{\theta})}. \tag{2.110}$$

The proof can be found in [139].

Example 2.15.5 *For the Laplace operator using Gauss-Seidel with lexicographic ordering as relaxation method*

$$\tilde{L}_h^+(\boldsymbol{\theta}) = \frac{1}{h^2}\left(4 - e^{-i\theta_1} - e^{-i\theta_2}\right), \quad \tilde{L}_h^-(\boldsymbol{\theta}) = -\frac{1}{h^2}\left(e^{i\theta_1} + e^{i\theta_2}\right),$$
$$\tilde{S}_h(\boldsymbol{\theta}) = -\frac{\tilde{L}_h^-(\boldsymbol{\theta})}{\tilde{L}_h^+(\boldsymbol{\theta})} = \frac{e^{i\theta_1} + e^{i\theta_2}}{4 - e^{-i\theta_1} - e^{i\theta_2}}. \tag{2.111}$$

Based on Lemma 2.15.4 we can now introduce the concept of smoothing factor, firstly proposed by Brandt in [18].

Definition 2.15.6 *The local smoothing factor $\mu_{loc}(S_h)$ of a given relaxation operator S_h is defined as*

$$\mu_{loc} = \mu_{loc}(S_h) = \sup\left\{\left|\tilde{S}_h(\boldsymbol{\theta})\right| ; \boldsymbol{\theta} \in \Theta^{high}\right\}. \tag{2.112}$$

Example 2.15.7 *The local smoothing factor for the discrete Laplace operator $L_h = -\Delta_h$ using as relaxation the lexicographic Gauss-Seidel method is given by*

$$\mu_{loc}(S_h) = \sup\left\{\left|\frac{e^{i\theta_1} + e^{i\theta_2}}{4 - e^{-i\theta_1} - e^{i\theta_2}}\right| ; \boldsymbol{\theta} \in \Theta^{high}\right\}. \tag{2.113}$$

The supremum of (2.113) is attained precisely at $\boldsymbol{\theta} = (\theta_1, \theta_2) = (\pi/2, \cos^{-1}(4/5))$ and therefore the local smoothing factor is

$$\mu_{loc}(S_h) = 0.5 . \tag{2.114}$$

2.15.2 Two-grid analysis

The smoothing analysis described in the previous section is helpful in analyzing the performance of the relaxation method or smoother being used within the multigrid algorithm. However, in order to calculate convergence factors of the two-grid operator

$$M_h^H = S_h^{\nu_2} K_h^H S_h^{\nu_1} \tag{2.115}$$

with K_h^H the standing for the coarse grid operator, we now need to include into the analysis the action of the operators L_h, I_h^H, I_H^h and S_h over the grid function $\varphi(\boldsymbol{\theta}, \boldsymbol{x})$.

In this section, we explain the basics of the two-grid analysis. A detailed explanation can be found in [139, 152]. We represent by L_h and L_{2h} the discrete operators on grids Ω_h and Ω_{2h} of size h and $2h$ respectively and assume that L_{2h}^{-1} exists. Similarly, we represent the smoother operator by S_h . Then the iteration operator for the $(h, 2h)$ two-grid cycle is given by

$$M_h^{2h} = S_h^{\nu_2} K_h^{2h} S_h^{\nu_1} \quad \text{with} \quad K_h^{2h} = I_h - I_{2h}^h L_{2h}^{-1} R_h^{2h} L_h. \tag{2.116}$$

Now, to calculate convergence factors of M_h^{2h}, we use again the fact that quadruples of $\varphi(\boldsymbol{\theta}, \cdot)$ coincide in Ω_{2h} with the respective grid function $\varphi_{2h}(2\boldsymbol{\theta}^{(0,0)}, \cdot)$.

Lemma 2.15.8 *For any low frequency $\boldsymbol{\theta}^{(0,0)} \in \Theta^{low}$, we have*

$$\varphi(\boldsymbol{\theta}^{(0,0)}, \boldsymbol{x}) = \varphi(\boldsymbol{\theta}^{(1,1)}, \boldsymbol{x}) = \varphi(\boldsymbol{\theta}^{(1,0)}, \boldsymbol{x}) = \varphi(\boldsymbol{\theta}^{(0,1)}, \boldsymbol{x}) \quad \boldsymbol{x} \in \boldsymbol{G}_{2h}. \tag{2.117}$$

Each of these four Fourier components

$$\varphi(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \cdot) = \varphi_h(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \cdot) \text{ with } \boldsymbol{\alpha} \in \{(0,0), (1,1), (1,0), (0,1)\}$$

coincides on $\boldsymbol{G}_2 h$ with the respective grid function $\varphi_{2h}(2\boldsymbol{\theta}^{(0,0)}, \cdot)$. This is,

$$\varphi_h(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \boldsymbol{x}) = \varphi_{2h}(2\boldsymbol{\theta}^{(0,0)}, \boldsymbol{x}), \quad \boldsymbol{x} \in \boldsymbol{G}_{2h}. \tag{2.118}$$

Proof of this lemma can be found in [139].

In simply words, the above lemma says that for any low frequency $\boldsymbol{\theta} \in \Theta^{low}$ it is enough to consider the frequencies

$$\boldsymbol{\theta}^{(0,0)} = (\theta_1, \theta_2), \quad \boldsymbol{\theta}^{(1,1)} = (\bar{\theta}_1, \bar{\theta}_2), \quad \boldsymbol{\theta}^{(1,0)} = (\bar{\theta}_1, \theta_2), \quad \boldsymbol{\theta}^{(0,1)} = (\theta_1, \bar{\theta}_2), \tag{2.119}$$

where

$$\bar{\theta}_i = \begin{cases} \theta_i + \pi & \text{if } \theta_i < 0, \\ \theta_i - \pi & \text{if } \theta_i \geq 0. \end{cases} \tag{2.120}$$

The corresponding four $\varphi(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \cdot)$ with $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)$ are called harmonics of each other and for $\boldsymbol{\theta} = \boldsymbol{\theta}^{(0,0)} \in \Theta^{low}$ they generate the four-dimensional space of harmonics

$$E_h^{\boldsymbol{\theta}} = \text{span}[\varphi(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \cdot); \boldsymbol{\alpha} \in \{(0,0), (1,1), (1,0), (0,1)\}]. \tag{2.121}$$

Thus everything reduces to analyze how a function $\psi \in E_h^{\boldsymbol{\theta}}$ is transformed due to the action of the operator M_h^{2h}. We can represent ψ as a linear combination of the grid functions in (2.119). This is,

$$\psi = A^{(0,0)}\varphi(\boldsymbol{\theta}^{(0,0)},\cdot) + A^{(1,1)}\varphi(\boldsymbol{\theta}^{(1,1)},\cdot) + A^{(1,0)}\varphi(\boldsymbol{\theta}^{(1,0)},\cdot) + A^{(0,1)}\varphi(\boldsymbol{\theta}^{(0,1)},\cdot). \quad (2.122)$$

Hence assuming that I_{2h}^h, L_{2h}^{-1}, R_h^{2h} and L_h can be approximated on Ω_h, and Ω_{2h} and $E_h^{\boldsymbol{\theta}}$ remain invariant under S_h, the two-grid operator M_h^{2h} can be represented on $E_h^{\boldsymbol{\theta}}$ for each $\boldsymbol{\theta} \in \Theta^{low}$ by a (4×4)-matrix. The following theorem, which is proved in [139], summarizes everything:

Theorem 2.15.9 *Under the above assumptions, the coarse grid correction operator K_h^{2h} is represented on $E_h^{\boldsymbol{\theta}}$ by the (4×4)-matrix $\hat{K}_h^{2h}(\boldsymbol{\theta})$*

$$\hat{K}_h^{2h}(\boldsymbol{\theta}) = \hat{I}_h - \hat{I}_{2h}^h(\boldsymbol{\theta})\hat{L}_{2h}^{-1}(\boldsymbol{\theta})\hat{R}_h^{2h}(\boldsymbol{\theta})\hat{L}_h(\boldsymbol{\theta}) \quad (2.123)$$

for each $\boldsymbol{\theta} \in \Theta^{low}$. Here,

$$\begin{aligned}
\hat{I}_h &= diag\{1,1,1,1\} \quad \in \mathbb{C}^{4\times4}, \quad \hat{L}_{2h}(\boldsymbol{\theta}) = \tilde{L}_{2h}(2\boldsymbol{\theta}^{(0,0)}) \quad \in \mathbb{C}^{1\times1},\\
\hat{L}_h(\boldsymbol{\theta}) &= diag\{\tilde{L}_h(\boldsymbol{\theta}^{(0,0)}),\ \tilde{L}_h(\boldsymbol{\theta}^{(1,1)}),\ \tilde{L}_h(\boldsymbol{\theta}^{(1,0)}),\ \tilde{L}_h(\boldsymbol{\theta}^{(0,1)})\} \quad \in \mathbb{C}^{4\times4},\\
\hat{R}_h^{2h}(\boldsymbol{\theta}) &= [\tilde{R}_h^{2h}(\boldsymbol{\theta}^{(0,0)})\ \tilde{R}_h^{2h}(\boldsymbol{\theta}^{(1,1)})\ \tilde{R}_h^{2h}(\boldsymbol{\theta}^{(1,0)})\ \tilde{R}_h^{2h}(\boldsymbol{\theta}^{(0,1)})] \quad \in \mathbb{C}^{1\times4},\\
\hat{I}_{2h}^h(\boldsymbol{\theta}) &= \frac{1}{4}[\tilde{I}_{2h}^h(\boldsymbol{\theta}^{(0,0)})\ \tilde{I}_{2h}^h(\boldsymbol{\theta}^{(1,1)})\ \tilde{I}_{2h}^h(\boldsymbol{\theta}^{(1,0)})\ \tilde{I}_{2h}^h(\boldsymbol{\theta}^{(0,1)})]^T \quad \in \mathbb{C}^{4\times1}.
\end{aligned}$$

In other words: if we apply K_h^{2h} to any $\psi \in E_h^{\boldsymbol{\theta}}$, the corresponding coefficients A^{α} in (2.122) are transformed according to

$$\begin{pmatrix} A^{(0,0)} \\ A^{(1,1)} \\ A^{(1,0)} \\ A^{(0,1)} \end{pmatrix} \Longleftarrow \hat{K}_h^{2h}(\boldsymbol{\theta}) \begin{pmatrix} A^{(0,0)} \\ A^{(1,1)} \\ A^{(1,0)} \\ A^{(0,1)} \end{pmatrix}. \quad (2.124)$$

Further, if the spaces $E_h^{\boldsymbol{\theta}}$ are invariant under the smoothing operator S_h, we can also represent M_h^{2h} on $E_h^{\boldsymbol{\theta}}$ by a (4×4)-matrix $\hat{M}_h^{2h}(\boldsymbol{\theta})$ with respect to $E_h^{\boldsymbol{\theta}}$. Here,

$$\hat{M}_h^{2h}(\boldsymbol{\theta}) = \hat{S}_h(\boldsymbol{\theta})^{\nu_2}\hat{K}_h^{2h}(\boldsymbol{\theta})\hat{S}_h(\boldsymbol{\theta})^{\nu_1}, \quad (2.125)$$

with $\hat{K}_h^{2h}(\boldsymbol{\theta})$ from (2.123) and the (4×4)-matrix $\hat{S}_h(\boldsymbol{\theta})$ which represents S_h. This means that M_h^{2h} can be written as

$$M_h^{2h} = B^{(0,0)}\varphi(\boldsymbol{\theta}^{(0,0)},\cdot) + B^{(1,1)}\varphi(\boldsymbol{\theta}^{(1,1)},\cdot) + B^{(1,0)}\varphi(\boldsymbol{\theta}^{(1,0)},\cdot) + B^{(0,1)}\varphi(\boldsymbol{\theta}^{(0,1)},\cdot) \quad (2.126)$$

where

$$\begin{pmatrix} B^{(0,0)} \\ B^{(1,1)} \\ B^{(1,0)} \\ B^{(0,1)} \end{pmatrix} \Longleftarrow \hat{M}_h^{2h}(\boldsymbol{\theta}) \begin{pmatrix} A^{(0,0)} \\ A^{(1,1)} \\ A^{(1,0)} \\ A^{(0,1)} \end{pmatrix}. \quad (2.127)$$

Based on the (4×4)-matrix representation of M_h^{2h} given in (2.125), we can calculate its *asymptotic convergence factor* defined as

$$\rho_{loc}(M_h^{2h}) = \sup\{\rho_{loc}(\hat{M}_h^{2h}) : \boldsymbol{\theta} \in \Theta^{low}, \boldsymbol{\theta} \notin \Lambda\}, \tag{2.128}$$

where

$$\Lambda = \{\boldsymbol{\theta} \in \Theta^{low} : \tilde{L}_h(\boldsymbol{\theta}) = 0 \text{ or } \tilde{L}_{2h}(\boldsymbol{\theta}) = 0\} \tag{2.129}$$

and $\rho_{loc}(\hat{M}_h^{2h})$ is the spectral radius of the (4×4)-matrix $\hat{M}_h^{2h}(\boldsymbol{\theta})$.

The *symbols* [139, 152] in Theorem 2.15.9 depend on the selection of their corresponding operators. For instance, if we select full weighting for the R_h^{2h} operator and bilinear interpolation for the I_{2h}^h operator, their symbols are defined by

$$\begin{aligned} \tilde{R}_h^{2h}(\boldsymbol{\theta}^\alpha) &= \frac{1}{4}(1 + \cos(\theta_1^\alpha)(1 + \cos(\theta_2^\alpha)), \\ \tilde{I}_{2h}^h(\boldsymbol{\theta}^\alpha) &= (1 + \cos(\theta_1^\alpha)(1 + \cos(\theta_2^\alpha)). \end{aligned} \tag{2.130}$$

Formulas for other popular transfer operators can be found in [139, 152]. The symbols for specific L_h and L_{2h} will be computed in Chapter 6.

2.16 Nonlinear multigrid: the full approximation scheme (FAS)

Similar to the linear case, we may want to solve a nonlinear system of discrete equations of the form

$$N_h \boldsymbol{u}_h = \boldsymbol{f}_h, \tag{2.131}$$

where N_h is a nonlinear discrete operator acting on $\boldsymbol{u}_h$. This type of systems of equations usually arises from the discretization (over a discrete domain Ω_h) of nonlinear partial differential equations. For nonlinear problems we still can implement a multigrid method defined recursively on the basis of a two-grid method. There are of course some issues that need to be sorted out before doing that. This multigrid method for nonlinear problems is known as the Full Approximation Scheme (FAS).

In the nonlinear case (2.131) the residual equation on Ω_h is defined by

$$\boldsymbol{r}_h^k = N_h(\bar{\boldsymbol{u}}_h^k - \boldsymbol{v}_h^k) - N_h \bar{\boldsymbol{u}}_h^k. \tag{2.132}$$

Clearly due to the nonlinear operator N_h this equation cannot be reduced as it was done in the linear case (2.77). Therefore, the error $\boldsymbol{e}$ cannot be computed explicitly and transported to the coarser grid Ω_H. Because of this, the whole residual equation (2.132) needs to be approximated on the coarse grid Ω_H and therefore, in contrast to the linear case, not only the residual $\boldsymbol{r}_h^k$ is transported to the coarse grid Ω_H but the relaxed approximation $\bar{u}_h^k$ as well.

On the coarse grid Ω_H, we deal with the problem

$$N_H w_H = f_H, \tag{2.133}$$

where

$$w_H = \bar{u}_H^k + \hat{v}_H \tag{2.134}$$

and the right-hand side f_H is defined as

$$f_H = I_h^H(f_h - N_h\bar{u}_h^k) + N_H\hat{I}_h^H\bar{u}_h^k. \tag{2.135}$$

In Algorithm 6 we describe in detail the FAS multigrid cycle. Here *SMOOTH*

Algorithm 6 $u_h^{k+1} = FASCY(h, \gamma, u^k, N_h, f_h, \nu_1, \nu_2)$

```
(1) Pre-smoothing
Obtain $\bar{u}_h^k$ by applying $\nu_1$ smoothing steps to $u_h^k$. That is,

    $\bar{u}_h^k = SMOOTH^{\nu_1}(u_h^k, N_h, f_h).$        (2.136)

(2) Coarse grid correction
Compute the residual $\bar{r}_h^k = f_h - N_h\bar{u}_h^k$.
Restrict the residual $\bar{r}_{h-1}^k = R_h^{h-1}\bar{r}_h^k$.
Restrict the approximate solution $\bar{u}_h^k$, by $\bar{u}_{h-1}^k = R_h^{h-1}\bar{u}_h^k$.
Compute the right-hand side $f_{h-1} = \bar{r}_{h-1}^k + N_{h-1}\bar{u}_{h-1}^k$
Compute an approximate solution $\hat{w}_{h-1}^k$ of the residual equation on $\Omega_{h-1}$.

    $N_{h-1}\hat{w}_{h-1}^k = f_{h-1}$ by        (2.137)

if $h = 1$ then
  use a direct or fast iterative solver for (2.137).
else if $h > 1$ then
  solve (2.137) approximately by performing $\gamma$ FAS $h$-grid cycles using $\bar{u}_{h-1}^k$ as initial
  approximation

    $\hat{w}_{h-1}^k = FASCYC^{\gamma}(h-1, \gamma, \bar{u}_{h-1}^k, N_{h-1}, f_{h-1}, \nu_1, \nu_2).$        (2.138)

end if
Compute the correction $\hat{v}_{h-1}^k = \hat{w}_{h-1}^k - \hat{u}_{h-1}^k$
Interpolate the correction $\hat{v}_h^k = I_{h-1}^h\hat{v}_{h-1}^k$.
Compute the corrected approximation on $\Omega_h$, $u_h^k = \bar{u}_h^k + \hat{v}_h^k$.

(3) Post-smoothing
Compute $u_h^{k+1}$ by applying $\nu_2$ smoothing steps to $u_h^k$.

    $u_h^{k+1} = SMOOTH^{\nu_2}(u_h^k, N_h, f_h).$        (2.139)
```

means a nonlinear relaxation method with suitable smoothing properties.

Chapter 3

Review of Variational Models for Image Restoration

3.1 Introduction

In this chapter we introduce the two image restoration problems this thesis is concerned with: digital image denoising and digital image inpainting. Both are fundamental problems of image processing since real life images are almost always polluted with some quantity of noise making denoising tools relevant here and occlusions of objects in images occur everywhere being inpainting a highly desired tool to recover (up to some degree of accuracy) the occluded part.

For these two problems, there exits many different approaches. Our interest however centers on variational techniques. They not only have proven to deliver excellent results, but from the mathematical point of view they are very interesting as well. Further a large amount of research topics have been opened from them and many still remain unsolved.

Most the time these two techniques are required at the same time so a good model must be able to cope with this requirement. Of course, one possibility may be to first applying a denoising procedure and then inpainting if needed, but usually the computational time consumed by this two-stage algorithm is much larger than that of a one-stage technique.

We proceed now to explain the way variational techniques are formulated. For easy of explanation we consider the denoising case where the noisy image u^0 is the only available *data*. In variational methods an energy-type functional is defined and then minimized. For instance, one option would be to minimize the L^2-norm of the noise $\eta = u^0 - u$ (assuming we are considering additive noise), this is

$$\min_u \left\{ \frac{1}{2} \int_\Omega |u^0 - u|^2 \, dxdy \right\}. \tag{3.1}$$

However, (3.1) is an ill-posed problem since many solutions (images) could satisfy the above equation. Therefore, a *regularizer* is needed to select the class of solutions we want. That is we would like to solve a problem of the form

$$\min_{u} \left\{ F(u) \equiv \alpha \underbrace{\int_{\Omega} \Phi(u)\, dxdy}_{\textit{Regularization}} + \underbrace{\frac{1}{2}\int_{\Omega} |u - u^0|^2\, dxdy}_{\textit{Fitting}} \right\}, \tag{3.2}$$

where usually $\Phi(u) \geq 0$. The *regularization* term is expected to bring stability to the problem and might guarantee uniqueness of the solution provided the resulting functional is convex. The selection of such a regularizer plays also a very important role since it imposes some properties to the resulting denoised image u.

For the inpainting case the variational model is not very different to (3.2). Since we do not have any information in the occluded part or missing region, only the regularization term is minimized inside this region. We will present with more formality these two problems later on this chapter.

3.2 Denoising

Image denoising is a research topic that has been consistently around within the last decades. Digital images are exposed to noise since the very first moment they are captured, for instance through the use of a digital camera. Noise comes from many possible sources and no imaging procedure is free of noise. It can be visually perceived as variations of random distribution or no particular pattern in the brightness of a displayed image. Thus, a noisy image looks grainy or with snowy appearance. These variations in brightness reduce the quality of the image making particularly difficult to identify small and low contrast objects.

There are many different types of noise and they may be classified as follows:

Gaussian noise

This type of noise is independent at each pixel and independent of the signal intensity. It is caused primarily by thermal noise in the electronic components of digital cameras.

Uniform noise

Here the noise may be signal dependent and has an approximately uniform distribution. It is caused by quantizing the pixels of a sensed image to a number of discrete levels.

Salt and pepper noise

Also called impulsive or spike noise. Visually it can be easily identified since an image containing it will have dark pixels in bright regions and bright pixels in dark regions. This type of noise can be caused by dead pixels in the CCD digital camera, analog-to-digital converter errors, error concealment, etc.

Shot noise

This noise is mainly due to variation in the number of photons sensed at a given exposure level. It has a Poisson distribution, which is usually not very different from Gaussian.

Non-isotropic noise

As its name suggests, non-isotropic noise shows up with a significant orientation in images. Examples are: row noise or column noise commonly found in image sensors and scratches in old films.

In this thesis we consider only images polluted by additive Gaussian noise. By representing a clean image as a two-dimensional function $u = u(x,y) \in \Omega \subset \mathbb{R}^2$, a noisy image $u^0 = u^0(x,y)$ is defined as

$$u^0(x,y) = u(x,y) + \eta(x,y), \tag{3.3}$$

where $\eta = \eta(x,y)$ is the unknown additive noise. The task of removing noise can be accomplished by traditional ways such as linear filters which, though very simple to implement, may cause the restored image to be blurred at edges. A much better technique is to use nonlinear PDE's as anisotropic diffusion filters because they apply different strength of diffusivity to different locations in the image.

3.2.1 PDE-based anisotropic filters

The first class of anisotropic filters we present are the so called PDE-based filters. Here we look for a solution $u(t)$ of a nonlinear partial differential equation that is evolved in time t with the noisy image u^0 as the starting point or initial guess, this is $u(0) = u^0$. After a number of k-iterations the process is stopped and the current iterate $u(k\Delta t)$ is taken as the denoised image.

The Perona-Malik model

The most known anisotropic filter is the Perona-Malik PDE [108] which uses an edge-stopping function $g(|\nabla u|^2)$ as diffusion coefficient. Their model evolves the following problem:

$$\begin{gathered} \frac{\partial u}{\partial t} = \nabla \cdot \left(g(|\nabla u|^2)\nabla u\right) \;\text{ on } \Omega \times (0,\infty), \\ u(x,y,0) = u^0(x,y) \;\text{ on } \Omega \times (0,\infty) \\ \nabla u \cdot \boldsymbol{\nu} = 0 \text{ on } \partial\Omega \times (0,\infty) \text{ with } g = \frac{1}{1+(|\nabla u|/K)^2} \end{gathered} \tag{3.4}$$

and K a fixed constant and $\boldsymbol{\nu}$ the unit outward normal. Unfortunately, this model is regarded as ill-posed since the filter behaves like a backward diffusion across edges. The

Perona-Malik filter might have weak solutions but one should not expect uniqueness nor stability [109]. The problem (3.4) is unstable with respect to perturbations on the initial data as showed in [80] and may have a unique weak solution only for a finite time.

Besides all these theoretical problems the discrete version of (3.4) is most of the time stable and more importantly yields excellent denoising results [149]. On its time this model opened a completely new area of image processing and since its appearance a lot of effort has been carried out to analyze its properties.

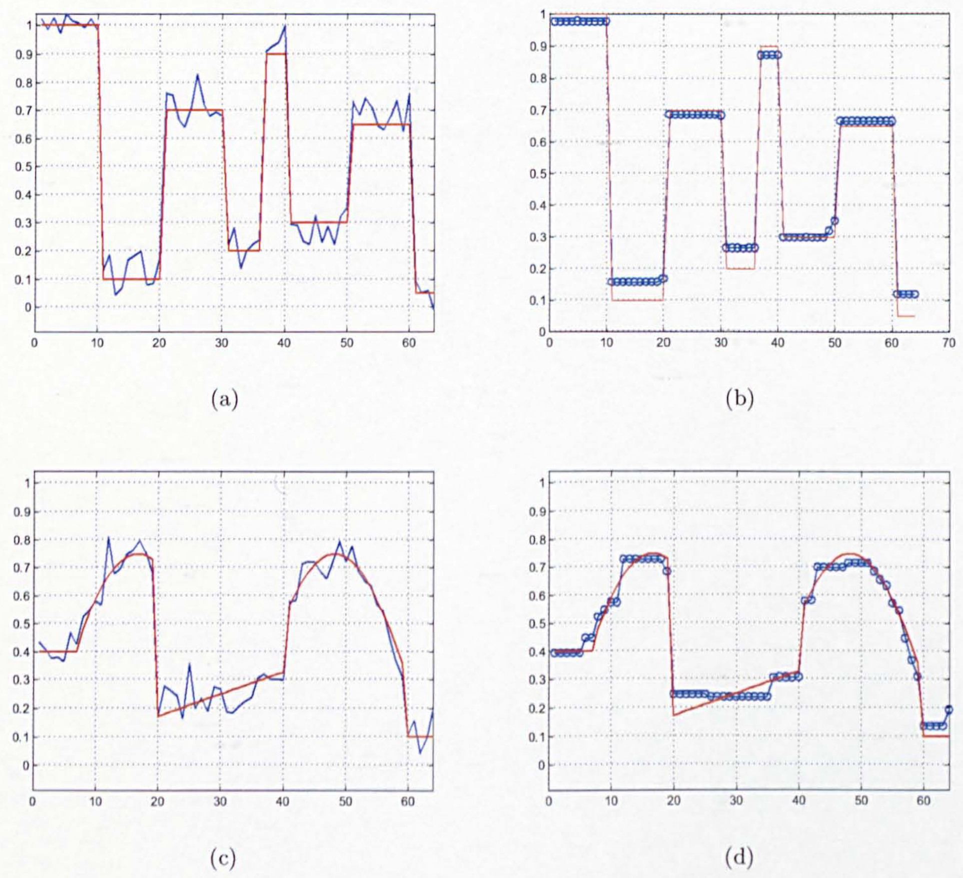


Figure 3.1: (a) Original (red) versus noisy (blue) image in one spatial dimension. Notice the original signal is piece-wise constant. (b) Original versus TV recovered image, reconstruction is very good. (c) Original versus noisy image. This time the original image is a piecewise smooth function. (d) Original versus TV recovered image. The piecewise smooth function was transformed in a piece-wise constant function.

3.2.2 Variational denoising models

The success of PDE-based methods and in particular the one of the Perona-Malik model in removing noise without damaging edges brought the attention to the use of anisotropic filters as an excellent option for image denoising. However, the Perona-Malik model, as we have already mentioned is initial guess dependent, might be unstable and in general its properties are difficult to analyze.

3.2.3 The total variation model

As a variational model, the TV model proposed in [114] not only yields astonishing results by preserving edges and removing the noisy high-frequency components from the image, but its variational setting makes it perfect to analyze its mathematical properties as well. This model proposes to minimize the following functional:

$$\min_{u\in BV}\left\{\int_\Omega |\nabla u|\,dxdy+\frac{\lambda}{2}\int_\Omega |u-u^0|^2\,dxdy\right\}, \tag{3.5}$$

where λ is a positive parameter which selects the quantity of noise to be removed. To begin with, (3.5) is a well-possed problem so existence and uniqueness of its minimizer is guaranteed [31].

Minimization of (3.5) can be done directly as in [33], or by solving the nonlinear second-order PDE

$$-\nabla\cdot\frac{\nabla u}{|\nabla u|}+\lambda(u-u^0)=0 \ \text{ in } \Omega, \tag{3.6}$$

with homogeneous Neumann boundary condition $\nabla u\cdot\nu=0$ on $\partial\Omega$ and with ν the unit outward normal as before. There are interesting connections of this energy minimization model (3.5) with the Perona-Malik model (3.4); see [135]. For the numerical solution of (3.6) there exists very good solvers, for instance see [145, 118, 119, 35] and references therein.

TV denoising model shortcoming

The main problem with the TV model is that it transforms piecewise smooth functions into piecewise constant functions a phenomenon known as staircase effect. Staircase makes denoised images to look blocky. An illustration of this effect over a one-dimensional signal is presented in Figure 3.1.

3.2.4 High-order models

Although some effort has been made (see for instance [95, 118, 46] and the references therein) to numerically reduce the staircase effect in second-order models, some researchers have turned to higher order models trying to avoid this problem; see for example the nice results obtained in Figure 3.2 by using the curvature-based model of [159]. In this direction are for instance the works presented in [157, 92, 91, 159]. However, high-order models involve to solve higher-order PDE's which numerically are more difficult to solve than the low-order ones. This difficulty has prevented in some way the popularization of these models. In Chapter 6, we will study a high-order and

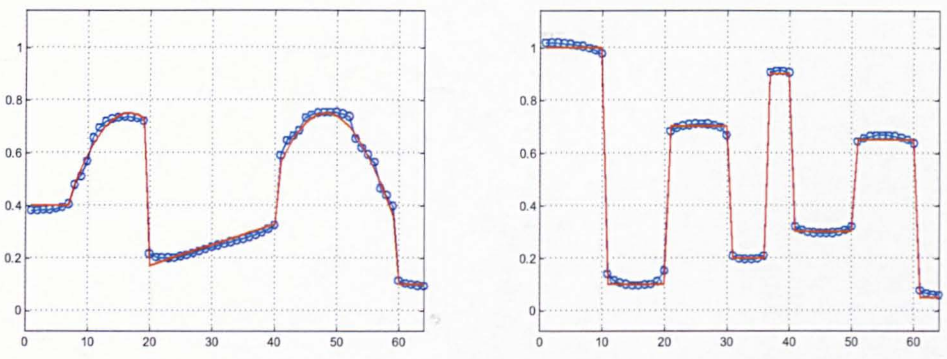


Figure 3.2: (a) Original (red) versus curvature-based (blue) restored image. This result outperforms the one of the TV model since piecewise smooth functions are preserved. (b) Original versus curvature-based recovered image for the piece-wise constant case. Here the curvature-based model delivers a result as good as the one of the TV model.

curvature-based denoising model for which we show how to construct a fast and efficient multigrid method, first of its kind.

3.3 Inpainting

We start by stating what is understood as image inpainting. In words of the authors who recently popularized this topic in the image processing community,

Image inpainting is defined as the process of reconstituting the missing or damaged portions of an image, in order to make it more legible and to restore its unity. The aim of inpainting is then to modify an image in a way that is non-detectable for an observer who does not know the original image [10].

Clearly the above definition leaves the door open to many possible solutions of the inpainting problem. It uses the word *non-detectable* meaning that meanwhile the observer is not able to say with certainty if the inpainted image has been modified or not, we can consider the applied inpainting process as successful. We will see that this non-uniqueness of the solution is perfectly embraced by variational formulations.

There are a variety of reasons why images can have damaged parts, for instance because of some physical degradation like aging, weather or intentional scratching. Not only that, we also would like to recover parts of objects of an image occluded by other objects or to reconstruct parts that have been missing due to digital communication processes.

We can also imagine a number of applications of this technique. Below we mention some of the them:

1. The restoration of old pictures with scratches or missing patches [10, 61].
2. Text removal [121, 10, 125].
3. Digital zooming or superresolution [121, 38].
4. Error concealment [41].
5. Disocclusion in computer vision [97, 136].
6. X-Ray CT artifacts reduction [69].
7. Attacking visible watermarking schemes [76].
8. Inpainting Images on Implicit Surfaces [155].
9. Video inpainting [99, 151, 142, 126, 77, 105, 128, 127, 106].
10. Video stabilization [98].

Considering that the study of this technique is relatively new, it started no more than 15 years ago, the above list is pretty long although by no means exhaustive. Further, many other applications appear every year in many different branches of technology and science.

Thus, inpainting techniques deal with these kinds of problems trying to reconstruct in the best possible way the missing or damaged parts of an image from the available information.

There are basically two main approaches for the image inpainting realization: the *variational* or energy-based approach and the one (non-variational) we call here as the *computational* approach. We start by reviewing the latter, also known as fragment- or patch-based image completion, in the following section.

3.3.1 Patch-based image completion

Patch-based, or Fragment-based image completion is a technique based on the very successful texture synthesis algorithm of Efros and Leung [52]. Texture synthesis algorithms are designed to replicate texture or fill-in large regions of an image with a given sample texture pre-selected by the user. This is, the user provides the algorithm with a *seed* or *patch* of texture and the algorithm automatically fills-in a user-selected region of the image of size much larger than the size of the patch. Information about this kind of algorithms can found in [52, 147, 88, 86] and references therein.

In [47] this idea was adapted to filling-in missing regions of an image not only with texture, but also structure and texture at the same time. The algorithm developed in [47] has, since its appearance, suffered some modifications and adaptations. For instance, extensive work has been done to adapt this algorithm to video inpainting. Some good references on this technique are [156, 50, 134, 84, 124, 107].

Patch-based image completion is very popular among the inpainting community since sometimes it can deliver astonishing results, is very easy to implement and is relatively very fast. This technique is also highly appreciated because it recovers not only the structure or cartoon part of the missing section of the image, but the texture part as well. There are however some drawbacks in this technique; we point out the following:

1. In order to obtain such good results the user must carefully select the size and shape of the patch. If not the result may be completely wrong, see Figure 3.4.

2. It relays on the assumption (not always satisfied) that there is plenty of information's redundancy within the image, meaning that for every missing patch there exists a very similar patch within the available region of the image so that by applying a simple copy and paste procedure one patch at a time we can fill-in satisfactorily the whole missing region.

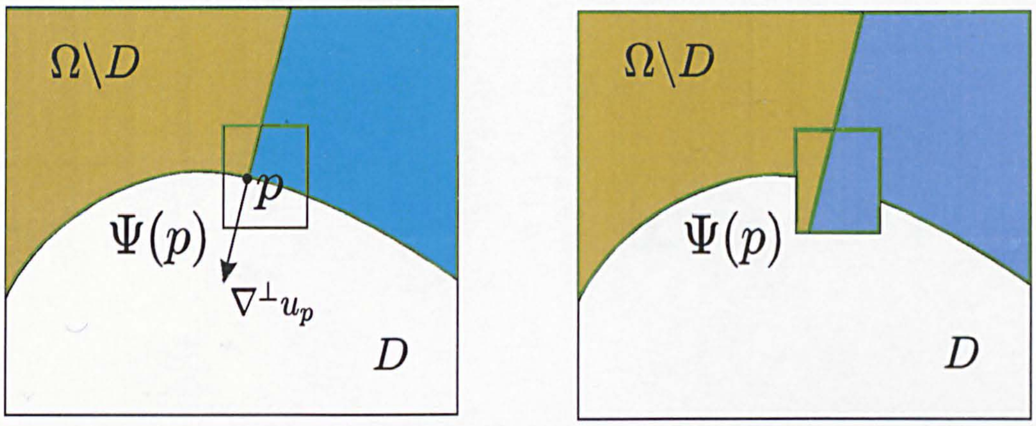
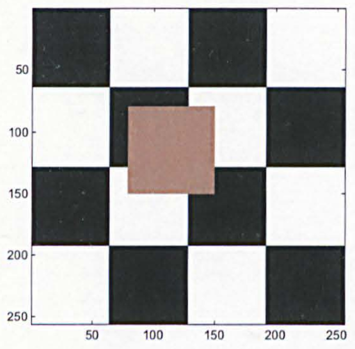
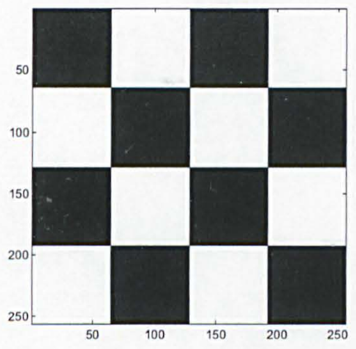


Figure 3.3: Here D is the inpainting domain or target region to be filled-in and $\Omega \setminus D$ is the source region. LEFT: A point $\boldsymbol{p}$ with the highest priority $P(\boldsymbol{p})$ is selected on the front, then a patch $\Psi(\boldsymbol{p})$ surrounding this point is created and filled-in using information from $\Omega \setminus D$. RIGHT: The unknown part of the patch $\Psi(\boldsymbol{p})$ has already been filled-in.

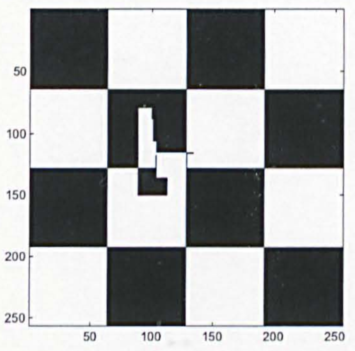
We now proceed to describe this technique: given an image $u(x, y)$ defined on $\Omega \subset \mathbb{R}^2$, the user selects the region D of the image to be filled-in and the size of the patch window Ψ. The algorithm automatically selects the fill front ∂D and, for the purpose of having a filling-in order the algorithm computes priorities $\mathcal{P}(\boldsymbol{p}) = \mathcal{C}(\boldsymbol{p})\mathcal{D}(\boldsymbol{p})$ for each patch $\Psi_{\boldsymbol{p}}$ centered at point $\boldsymbol{p}$ in the front. The priorities are computed using two terms: the *confidence* term $\mathcal{C}(\boldsymbol{p})$ and the *data* term $\mathcal{D}(\boldsymbol{p})$. During initialization, the function $\mathcal{C}(\boldsymbol{p})$ is set to $\mathcal{C}(\boldsymbol{p}) = 0 \ \forall \boldsymbol{p} \in D$, and $\mathcal{C}(\boldsymbol{p}) = 1 \ \forall \boldsymbol{p} \in \Omega \setminus D$. The idea of the confidence term is to have a measure of the reliability of the information surrounding the pixel $\boldsymbol{p}$. The data term $\mathcal{D}(\boldsymbol{p})$ on the other hand, is used to identify edges arriving at the front and to give preference to their propagation into D. They are computed



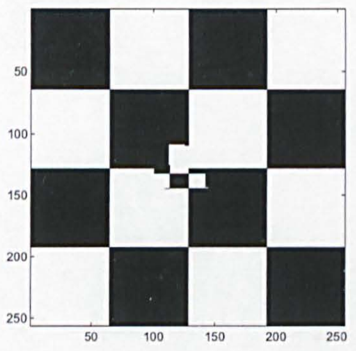
(a)



(b)



(c)



(d)

Figure 3.4: (a) Image to be inpainted, the inpainting domain in this case is the orange rectangle. (b) By selecting the patch-size equal to 24 pixels the inpainting result obtained is perfect. (c) With a wrong selection of the patch-size, 12 pixels this time, the result is totally wrong. (d) Again a bad selection of the patch-size, 6 pixels this time, yields a visually incorrect solution.

using the following formulas:

$$\mathcal{C}(p) = \frac{\sum_{q \in \Psi_p \cap \bar{D}} \mathcal{C}(q)}{|\Psi_p|}, \quad \mathcal{D}(p) = \frac{\nabla^{\perp} u_p \cdot \vec{n}_p}{\alpha}, \tag{3.7}$$

where $|\Psi_p|$ is the area of Ψ_p, α is a normalization factor and $\nabla^{\perp} u_p$ gives the isophote direction and intensity at point p, see Figure 3.3.

After $\mathcal{P}(p)$ has been computed for all p in the front, the patch with the highest priority $\Psi_{\hat{p}}$ is located. Then the most similar patch $\Psi_{\hat{q}}$ to $\Psi_{\hat{p}}$ is found in $\Omega \setminus D$ using the distance function $d(\Psi_{\hat{q}}, \Psi_{\hat{p}})$, this is

$$\Psi_{\hat{q}} = \min_{\Psi_q \in \Omega \setminus D} d(\Psi_{\hat{p}}, \Psi_q). \tag{3.8}$$

Usually the distance function is defined as the sum of the square differences (SSD) of the already filled-in pixels in the two patches. Finally, having found the source $\Psi_{\hat{q}}$, the value of each pixel $p' \in \Psi_{\hat{p}} \cap D$ to be filled is copied from its corresponding position inside $\Psi_{\hat{p}}$.

The whole process we just finished describing is expressed in algorithm form in Algorithm 7.

Algorithm 7 Patch-Based Image Completion

```
1: Manually select the initial front $\partial D^0$ or boundary of the missing section $D$.
2: while $D^k \neq 0$ do
3:     Identify the fill front $\partial D^k$.
4:     Compute priorities $\mathcal{P}(p)$  $\forall p \in \partial D^k$.
5:     Find the patch $\Psi_{\hat{p}}$ with the maximum priority, i.e., $\Psi_{\hat{p}}$ such that
           $\hat{p} = \min_{p \in \partial D^t} \mathcal{P}(p)$.
6:     Find the exemplar $\Psi_{\hat{p}} \in \Omega \setminus D$ that minimizes $d(\Psi_{\hat{p}}, \Psi_{\hat{q}})$.
7:     Copy image data from $\Psi_{\hat{q}}$ to $\Psi_{\hat{p}}$.
8:     Update $\mathcal{C}(p)$  $\forall p$ such that $p \in \Psi_{\hat{p}} \cap D$.
9: end while
```

3.3.2 Variational inpainting

In the variational approach for image inpainting, a damaged image u^0 defined in $\Omega \subset \mathbb{R}^2$ is considered as a collection of curves broken at some region D within the image and the goal is to find the best interpolant to recover the missing parts of those broken curves in D. The missing region D, which may have complicated topology, is known as the *inpainting domain* and $\Omega \setminus D$ may contain noise η as well, see Figure 3.5.

Typical interpolation techniques such as polynomial interpolation, splines, etc. fail here to deliver good results mainly for one simple reason: they are too smooth for image interpolation, hence jumps (edges) cannot be reconstructed.

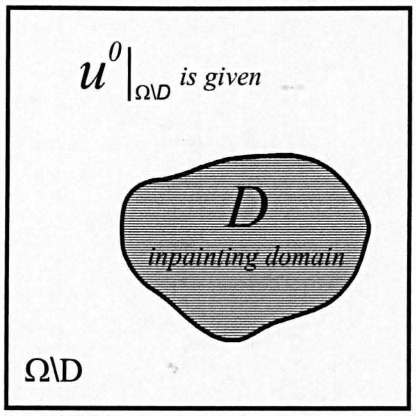


Figure 3.5: Illustration of a typical inpainting problem.

In the variational approach, a functional defined as the sum of an energy-image term $E(u)$ plus a fitting term is minimized

$$\min_{u \in BV} \left\{ \alpha E(u) + \int_{\Omega \backslash D} (u - u^0)^2 \, dxdy \right\}. \tag{3.9}$$

Here the first term $E(u)$ decides the class of the interpolant in D and the second fixes the ends of the image-curves of u in ∂D to be close to the values of u^0.

The mathematical interest on variational inpainting became increasingly active in the last decade since the very first works on image interpolation by Mumford *et al.* [100], Masnou and Morel [97], and Caselles *et al.* [30]. However, it was the pioneering work of Bertalmio *et al.* [10] who proposed an algorithm to imitate the work of inpainting artists who manually restore old damaged pictures which mainly motivated all the subsequent research in this field [121, 40].

Bertalmios's *et al.* [10] algorithm cleverly transports a smoothness image measure (namely the Laplacian of the image) along the level lines (contours of the same image intensity) directed into the inpainting domain; in their paper, they also showed that some intermediate steps of anisotropic diffusion are necessary to avoid blurring of edges. This algorithm was created mostly intuitively, but later on turned out to be closely related to the Navier-Stokes equation, as showed by Bertozzi *et al.* [11]. Since then, many other authors have proposed different models for digital inpainting.

Chan and Shen [121] introduced the TV model for local inpainting based on the celebrated total variation based image denoising model of Rudin, Osher and Fatemi [114]. Later on, the same authors modified this model to improve its performance for large scale inpainting, and created the so-called Curvature-Driven Diffusion (CDD) model [123]. Furthermore they, together with Kang, introduced a higher-order variational model [125] based on the Euler's elastica which connects the level lines by using Euler elastica curves [90] instead of using straight lines as the TV model does. Unfortunately, for the latter two models there appear to exist no fast methods to find the numerical solution. The aim of this thesis is to develop such a kind of fast algorithms.

A related inpainting model was proposed by Esedoglu and Shen [54] and is based on the very successful Mumford-Shah image segmentation model. This model is also good for local inpainting, but shares the same problem as the TV model in that it cannot reconnect separated parts of broken objects far apart. To fix this problem, the same authors of [54] proposed the Mumford-Shah-Euler inpainting model which, in the same fashion of the Euler's elastica model, uses the information encoded in the curvature to reconnect smoothly the level lines. More recently, in separate works, Bertozzi *et al.* [13] proposed a model to inpaint binary images based on the Cahn-Hilliard equation and Grossauer and Scherzer [68] proposed a model based on the complex Ginzburg-Landau equation. It remains to develop fast multigrid methods for these models.

Each one of the above models has its own particular features which may not suit all applications. However, as rightly remarked in [122], one of the most interesting open problems in digital inpainting (whatever the model) is the fast and efficient digital realization.

We remark that measuring the quality of restoration is non-trivial, as physical perceptions can be different and the 'true' solutions may not be unique [40].

3.3.3 Energy image models

The good performance of image restoration techniques based on variational methods strongly relies in the designing of a good energy image model. In order this energy model to be good, it must be able to capture in the best possible way the *hidden energy* of the image. In variational techniques this energy is part of the functional to be minimized and usually plays the role of regularizer for the inverse problem.

Using the level-set image representation, it is very easy to construct image models by direct functionalization [40]. To this end, at each γ_λ-level-set the energy is captured by selecting a curve model able to represent a relevant geometrical feature of the level curve. In the following we will review how to construct two of the most widely used image models : the *total variation* and the *Euler's elastica* models.

To construct a curve energy model first we need a good way to catch up the energy of the curve in the best possible way. The most simple way to do it is the first order *length* energy defined as

$$E[\gamma_\lambda] = length(\gamma_\lambda) = \int_{\gamma_\lambda} ds, \tag{3.10}$$

where ds stands for the arc-length element of the level set.

From here, what follows is to construct the energy image model $E[u]$ by adding up the energy of all the γ_λ-level-sets contained in Γ_u (the set containing all γ_λ-curves). That is,

$$E[u] = \int_{\Gamma_u} E[\gamma_\lambda]\, d\lambda = \int_{\Gamma_u} \int_{\gamma_\lambda} ds d\lambda. \tag{3.11}$$

By noticing that along any level-set of γ_λ,

$$d\lambda = |\nabla u| d\sigma$$

with $d\sigma$ standing for the projection of $d\lambda$ onto the γ_λ-level-set plane and therefore normal to ds, finally obtain that

$$E_{tv}[u] = \int_{\Gamma_u} \int_{\gamma_\lambda} |\nabla u| \; ds d\sigma = \int_\Omega |\nabla u| \; dx dy, \tag{3.12}$$

which is known as the *total variation energy* image model. In particular, $dsd\sigma = dxdy$ is the area element.

Similarly, we could have chosen

$$E[\gamma_\lambda] = \int_{\gamma_\lambda} (a + b\kappa^p) \; ds, \tag{3.13}$$

with $a, b, p > 0$ and $\kappa = \nabla \cdot \frac{\nabla u}{|\nabla u|}$ the curvature of γ_λ as our curve energy and obtain this time the *Euler's elastica energy* image model

$$E_{elastica}[u] = \int_{\Gamma_u} \int_{\gamma_\lambda} (a + b\kappa^p)|\nabla u| \; ds d\sigma = \int_\Omega (a + b\kappa^p)|\nabla u| \; dx dy. \tag{3.14}$$

The elastica functional $E_{elastica}[u]$ captures better the energy of the image, but leads to more difficult to solve partial differential equations.

3.3.4 The total variation model

In this section we review very briefly the TV inpainting model [121], which was constructed based on three principles that according to its creators a good inpainting model must satisfy. These are:

Locality. Meaning that to carry out the inpainting process only the information surrounding the inpainting domain must be used.

Restoration of narrow edges. Meaning that the model must be able to reconstruct the missing parts of the edges which give the most visual information in an image.

Robustness to noise. Meaning that it must get rid of any noise present and restore the missing part at the same time.

As usual assume that u and η are respectively the true image and the unknown additive Gaussian noise satisfying $u_0 = u + \eta$ in $\Omega \backslash D$. Following Rudin *et al.* [114], the TV inpainting model is as follows:

$$\min_{u \in BV} \left\{ \int_\Omega |\nabla u| \; dx dy \; + \; \frac{\lambda}{2} \int_{\Omega \backslash D} (u - u^0)^2 \; dx dy \right\}. \tag{3.15}$$

Although direct minimization ideas [32, 33] could be applied, so far the above minimization is mainly solved via its associated Euler-Lagrange equation:

$$\nabla \cdot \left(\frac{\nabla u}{|\nabla u|} \right) + \lambda_E (u^0 - u) = 0 \text{ in } \Omega, \quad \text{with } \frac{\partial u}{\partial \boldsymbol{\nu}} = 0 \text{ on } \partial\Omega, \tag{3.16}$$

where $\boldsymbol{\nu}$ is the unit outward normal on the boundary $\partial\Omega$ and λ_E is defined as

$$\lambda_E = \begin{cases} \lambda > 0 & (x,y), \in \Omega \backslash D \\ 0 & (x,y) \in D \,. \end{cases} \tag{3.17}$$

In D alone, where $\lambda_E = 0$, equation (3.16) reduces to an ill-posed boundary value problem with non-unique solution as it was shown in [30]. There the authors gave the following example: consider an inpainting domain with circular shape, this is D is a disk of radius equal to one and therefore its boundary ∂D is defined by $x^2 + y^2 = 1$. Assume that outside the disk, u is defined by a function $\varphi(x, y) = \lambda_1 x^2 + \lambda_2 y^2$ with $\lambda_1 > \lambda_2$. Therefore, the following equation is satisfied

$$\varphi(x,y) = \varphi(-x,y) = \varphi(x,-y)$$

and the functions

$$u_1(x,y) = \varphi(\sqrt{1-y^2}, y) \quad \text{and} \quad u_2(x,y) = \varphi(x, \sqrt{1-x^2})$$

are both solutions of equation (3.16).

TV model shortcomings

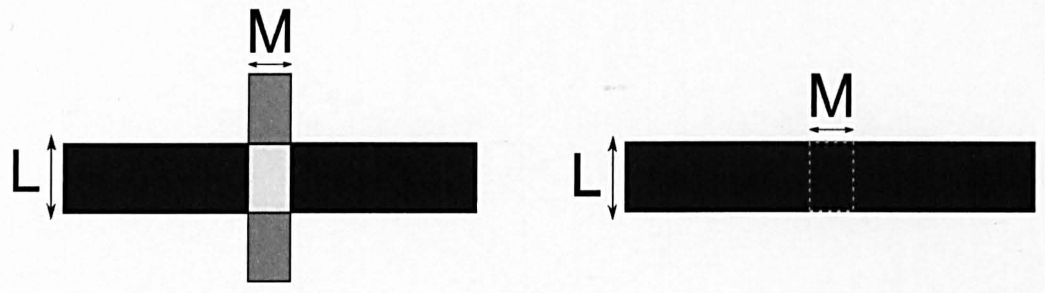
The main two drawbacks of the TV inpainting model are: (1) its inability to reconnect far apart separated parts of broken objects, and (2) its inability to smoothly reconstruct the curvature of the level sets.

For (1), the explanation can be found by looking at the energy values for the problem posed in Figure 3.6. This very illustrative example was presented in [121]. Assume that the intensity value of the horizontal black bar is u_b and the intensity value of the gray vertical bar is $u_w = u_b + \epsilon$ for some $\epsilon > 0$. It is not difficult to see that the solution u of (3.16) must be constant, say c. Thus, we only need to find the value of that constant c which minimizes (3.15). The functional gives us the correct clue; being all the intensity values constant, the total variation on the closure of D concentrates along ∂D. This is,

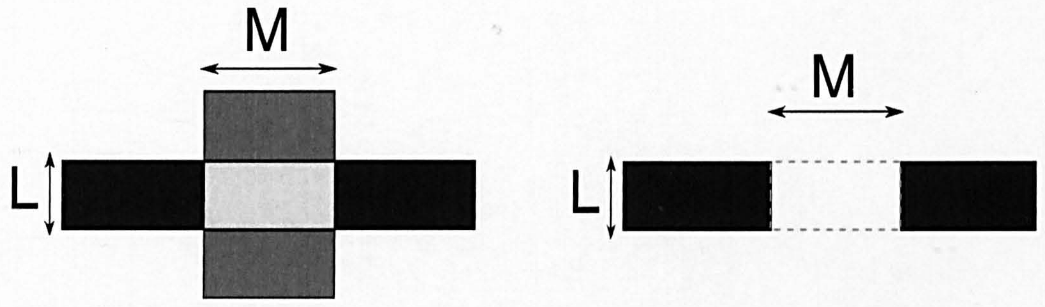
$$TV(u) = 2(|u_b - c| * L + |u_b + \epsilon - c| * M).$$

Now, the maximum principle [39] tell us to select c on the interval $u_b \leq c \leq u_b + \epsilon$. Thus, to minimize (3.15) we have two possibilities (other values increase the cost of the functional) : to select $c = u_b$ which yields

$$TV(u) = 2(|u_b - u_b| * L + |u_b + \epsilon - u_b| * M) = 2\epsilon M$$



(a)



(b)

Figure 3.6: (a) LEFT: Broken bar with a gap small in size compared with the characteristic feature of the image, this is $M < L$. RIGHT: The expected TV inpainting result, the gap is filled-in with the right intensity value achieving reconnection. (b) LEFT: Same broken bar as above but this time with the gap large in size compared with the characteristic feature of the image, this is $M > L$. RIGHT: This time the expected solution from the TV model is a broken bar since the cost of no-connection is less than the one of reconnection.

or to select $c = u_b + \epsilon$ which yields

$$TV(u) = 2(|u_b - u_b - \epsilon| * L + |u_b + \epsilon - u_b - \epsilon| * M) = 2\epsilon L.$$

Clearly if $M < L$, $c = u_b$ will be the correct solution achieving reconnection of the broken bar. However, with $M > L$, $c = u_b + \epsilon$ will be preferred keeping the gap open. Two real life examples of this drawback are illustrated in Figures 3.7 and 3.8.

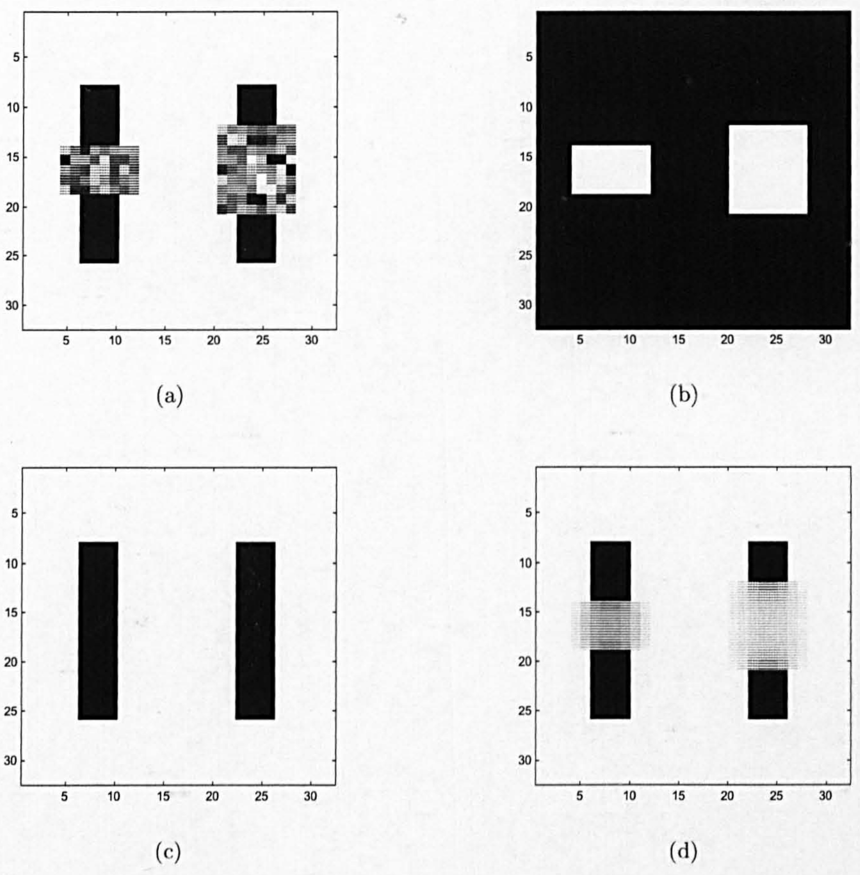


Figure 3.7: (a) An image with two broken bars. The missing region of both is large in size compared with the width of the bars. (b) The mask of the inpainting domain (c) The true solution. (d) The visually incorrect solution of the TV inpainting model. It cannot reconnect the bars as a human being would have expected.

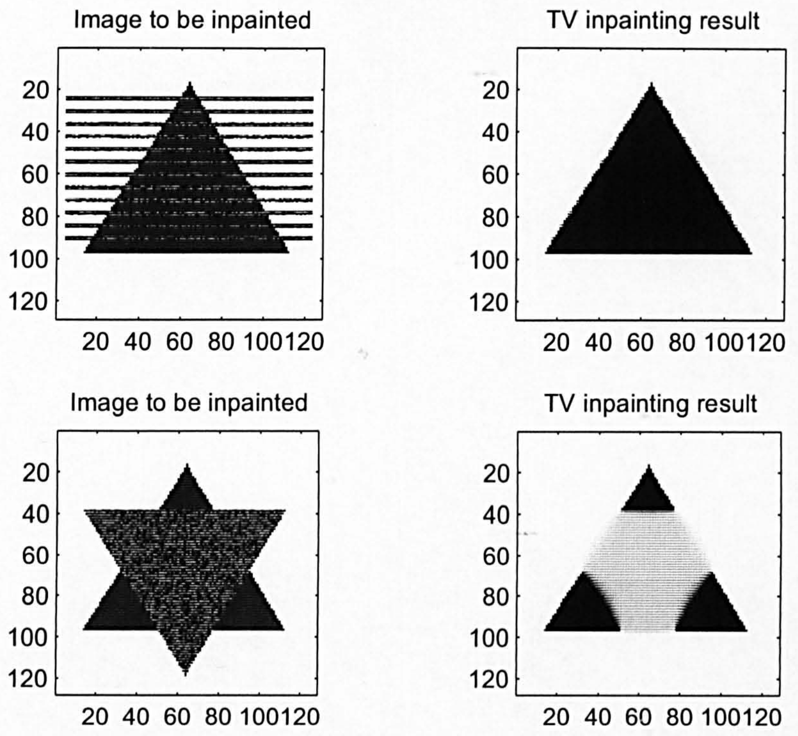


Figure 3.8: On one hand, at the top, we show a TV inpainting satisfying the connectivity principle [123, 121]. Here the inpainting domain D represented by the horizontal noisy bars is relatively small in size compared with the characteristic feature of the image, therefore the TV model performs very well and carries out a visually pleasant inpainting. On the other hand, at the bottom, D (represented by the noisy triangle) is relatively large and therefore, the TV model gives an unpleasant result.

For (2), there is a very simply explanation: in D, equation (3.16) reduces to $\kappa = 0$ (remember $\kappa = \nabla \cdot \frac{\nabla u}{|\nabla u|}$ is the curvature of the level sets). Therefore, curves used to inpaint are always straight lines creating this way very often artificial corners in the reconstructed image, see Figure 3.9.

3.3.5 The Euler's elastica inpainting model

As defined in [125], a curve Γ is said to be *Euler's elastica* if it is the equilibrium curve of the elasticity energy:

$$E_2[\gamma] = \int_\gamma (a + b\kappa^2)\, ds, \tag{3.18}$$

where ds denotes the arc length element, $\kappa(s)$ the scalar curvature, and a, b two positive constant weights. Euler obtained the above energy when studying the steady shape of a thin and torsion-free rod under external forces [90].

Even though this is a model for plane curves, it is possible to extend its use to an

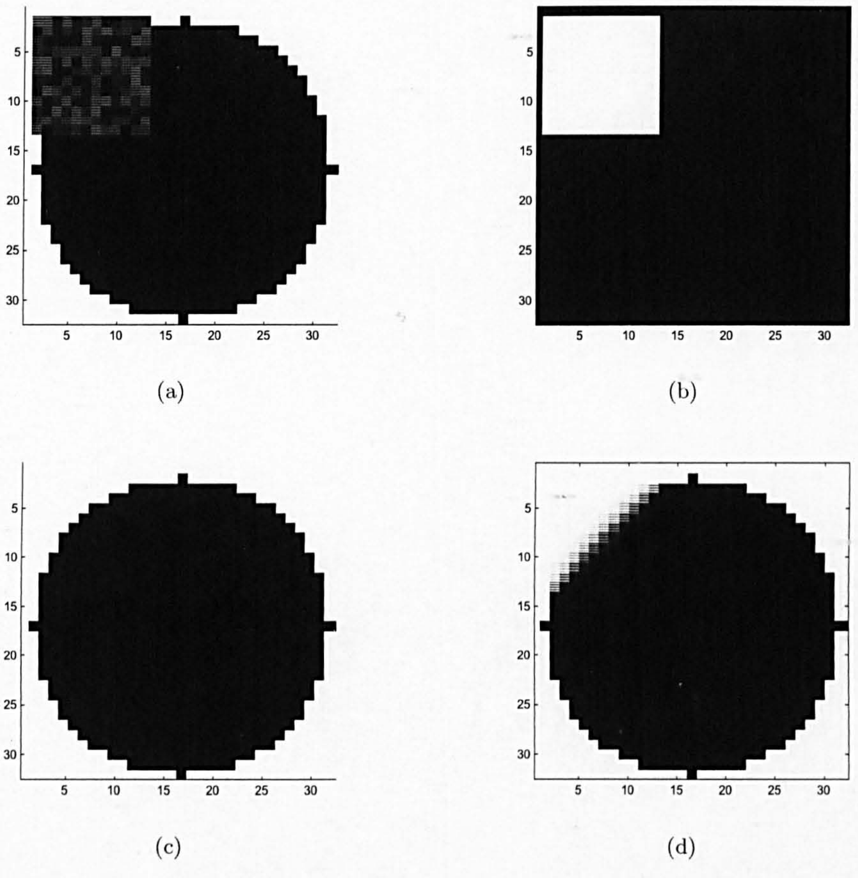


Figure 3.9: (a) A circle with a missing part. (b) The inpainting mask. (c) The true solution. (d) The TV inpainting result .

image model as we already have shown in Section 3.3.3.

As we know, variational models achieve their objective by minimizing some chosen energy functional. In particular, the Euler's elastica model [125] proposes the following minimization:

$$\min_{u \in BV} \left\{ E(u) = \int_\Omega (a + b\,|\kappa|^p)\,|\nabla u|\;dxdy + \frac{\lambda}{2}\int_{\Omega\setminus D} (u - u^0)^2\;dxdy \right\}, \tag{3.19}$$

where a and b are arbitrary positive constants, $\lambda > 0$ is a penalty parameter, $p = 2$ is usually chosen, u is the true image to be restored and $\kappa = \kappa(x, y) = \nabla \cdot \frac{\nabla u}{|\nabla u|}$ is the mean curvature. The above model allows u^0 to have additive Gaussian noise η present in $\Omega\setminus D$. The virtue of (3.19) is that it penalizes the integral of the square of the curvature along edges instead of only penalizing the length of edges as the TV model does [40, 24]. Consequently, the model can reconnect contours along large distances and recover the curvature of objects at the same time.

The derivation of the Euler-Lagrange equation

In [125] the Euler-Lagrange equation for the Euler's elastica model was derived in a geometrical form and for a general function $\phi = \phi(\kappa)$ of the curvature, i.e. the regularization term was written as $\int_\Omega \phi|\nabla u|\, dxdy$. Here we present a similar, but more detailed derivation of the Euler-Lagrange equation for the particular case $\phi = a + b\kappa^2$. Further, we introduce a different way to write this equation which in a future section, will let us implement a promising primal-dual method for this model.

Theorem 3.3.1 *Let $\phi = a + b\kappa^2 \in C^1(\Omega)$ and $E(u)$ defined as in (3.19). Then the Euler-Lagrange equation is given by*

$$\nabla \cdot \left[\left(a + b\kappa^2\right) \frac{\nabla u}{|\nabla u|} - \frac{2b}{|\nabla u|^3} \nabla^\perp u \nabla(\kappa|\nabla u|) \nabla^\perp u \right] + \lambda_E(u^0 - u) = 0 \tag{3.20}$$

with the boundary conditions along $\partial\Omega$

$$\frac{\partial u}{\partial \nu} = 0 \quad and \quad \frac{\partial((a + b\kappa^2)|\nabla u|)}{\partial \nu} = 0, \tag{3.21}$$

here as usual λ_E is defined as in (3.17), ∇u is a vector normal to the level sets and $\nabla^\perp u$ the corresponding tangential vector to them.

Proof. As it is well-known the Euler-Lagrange equation is found by computing the first variation, or optimality condition, of the functional $E(u)$ defined as in (3.19). That is,

$$\frac{d}{d\varepsilon} E(u + \varepsilon\varphi)\bigg|_{\varepsilon \to 0} = \frac{d}{d\varepsilon} \left\{ aE_1(u + \varepsilon\varphi) + bE_2(u + \varepsilon\varphi) + \frac{\lambda}{2} E_3(u + \varepsilon\varphi) \right\}\bigg|_{\varepsilon \to 0} = 0, \tag{3.22}$$

where to simplify we have split the energy in three parts $E_1(u), E_2(u)$, and $E_3(u)$. These parts are defined as follows:

$$E_1(u + \varepsilon\varphi) = \int_\Omega |\nabla(u + \varepsilon\varphi)|\, dxdy, \tag{3.23}$$

$$E_2(u + \varepsilon\varphi) = \int_\Omega \left(\nabla \cdot \frac{\nabla(u + \varepsilon\varphi)}{|\nabla(u + \varepsilon\varphi)|} \right)^2 |\nabla(u + \varepsilon\varphi)|\, dxdy, \tag{3.24}$$

$$E_3(u + \varepsilon\varphi) = \int_\Omega (u + \varepsilon\varphi - u^0)^2\, dxdy. \tag{3.25}$$

We need to compute the first variation for each one of these functionals, so we start with the first energy E_1:

$$\begin{aligned} \frac{d}{d\varepsilon} E_1(u + \varepsilon\varphi)\bigg|_{\varepsilon \to 0} &= \frac{d}{d\varepsilon} \int_\Omega |\nabla(u + \varepsilon\varphi)|\, dxdy \bigg|_{\varepsilon \to 0} \\ = \int_\Omega \frac{\nabla(u + \varepsilon\varphi)}{|\nabla(u + \varepsilon\varphi)|} \cdot \nabla\varphi\, dxdy \bigg|_{\varepsilon \to 0} &= \int_\Omega \frac{\nabla u}{|\nabla u|} \cdot \nabla\varphi\, dxdy \\ = -\int_\Omega \left(\nabla \cdot \frac{\nabla u}{|\nabla u|} \right) \varphi\, dxdy &+ \int_{\partial\Omega} \frac{\nabla u}{|\nabla u|} \cdot \nu\, ds. \end{aligned} \tag{3.26}$$

Here, to drop the boundary integral from integration by parts the following boundary condition needs to be satisfied

$$\nabla u \cdot \boldsymbol{\nu} = 0 \text{ along } \partial\Omega. \tag{3.27}$$

We now move to compute the first variation for the second energy E_2:

$$\begin{aligned}
\frac{d}{d\varepsilon}E_2(u+\varepsilon\varphi)\bigg|_{\varepsilon\to 0} &= \frac{d}{d\varepsilon}\int_\Omega \left(\nabla\cdot\frac{\nabla(u+\varepsilon\varphi)}{|\nabla(u+\varepsilon\varphi)|}\right)^2 |\nabla(u+\varepsilon\varphi)|\, dxdy\bigg|_{\varepsilon\to 0} \\
&= \int_\Omega \frac{d}{d\varepsilon}\left(\nabla\cdot\frac{\nabla(u+\varepsilon\varphi)}{|\nabla(u+\varepsilon\varphi)|}\right)^2\bigg|_{\varepsilon\to 0} |\nabla u|\, dxdy \\
&+ \int_\Omega \left(\nabla\cdot\frac{\nabla u}{|\nabla u|}\right)^2 \frac{d}{d\varepsilon}|\nabla(u+\varepsilon\varphi)|\bigg|_{\varepsilon\to 0} dxdy.
\end{aligned} \tag{3.28}$$

For the sake of clarity we proceed again by splitting this process in two parts. For the first integral, we have

$$\begin{aligned}
&\int_\Omega \frac{d}{d\varepsilon}\left(\nabla\cdot\frac{\nabla(u+\varepsilon\varphi)}{|\nabla(u+\varepsilon\varphi)|}\right)^2\bigg|_{\varepsilon\to 0} |\nabla u|\, dxdy \\
&= \int_\Omega 2\kappa|\nabla u|\frac{d}{d\varepsilon}\left(\nabla\cdot\frac{\nabla(u+\varepsilon\varphi)}{|\nabla(u+\varepsilon\varphi)|}\right)\bigg|_{\varepsilon\to 0} dxdy \\
&= \int_\Omega 2\kappa|\nabla u|\nabla\cdot\left(\frac{\nabla\varphi}{|\nabla u|} - \frac{\nabla u}{|\nabla u|^3}\nabla u\cdot\nabla\varphi\right) dxdy \\
&= \int_\Omega 2\kappa|\nabla u|\nabla\cdot\left(\frac{1}{|\nabla u|}\left(\nabla\varphi - \frac{\nabla\varphi\cdot\nabla u}{|\nabla u|^2}\nabla u\right)\right) dxdy.
\end{aligned} \tag{3.29}$$

In the integral in the last line, we observe that the term $\boldsymbol{n}\otimes\boldsymbol{n} = \frac{\nabla\varphi\cdot\nabla u}{|\nabla u|^2}\nabla u$ is the orthogonal projection onto the normal direction which satisfies $\boldsymbol{n}\otimes\boldsymbol{n} + \boldsymbol{t}\otimes\boldsymbol{t} = \boldsymbol{I}$ with $\boldsymbol{t}\otimes\boldsymbol{t}$ the tangential projection and $\boldsymbol{I}$ the identity projection. This means that we can rewrite the last integral as

$$\begin{aligned}
&\int_\Omega 2\kappa|\nabla u|\nabla\cdot\left(\frac{1}{|\nabla u|}\{\boldsymbol{t}\otimes\boldsymbol{t}\}\nabla\varphi\right) dxdy \\
&= -\int_\Omega \nabla(2\kappa|\nabla u|)\cdot\frac{1}{|\nabla u|}\{\boldsymbol{t}\otimes\boldsymbol{t}\}\nabla\varphi\, dxdy + \int_{\partial\Omega} 2\kappa\boldsymbol{\nu}\cdot\{\boldsymbol{t}\otimes\boldsymbol{t}\}\nabla\varphi\, ds.
\end{aligned} \tag{3.30}$$

Again we need to drop the boundary integral. To this end, we note that the first boundary condition (3.27) implies that $\boldsymbol{v} = \pm\boldsymbol{t}$ along $\partial\Omega$, so applying this into the second integral in (3.30) ,we get

$$\begin{aligned}
\int_{\partial\Omega} 2\kappa\boldsymbol{\nu}\cdot\{\boldsymbol{t}\otimes\boldsymbol{t}\}\nabla\varphi\, ds &= \int_{\partial\Omega} 2\kappa\boldsymbol{\nu}\cdot\{\boldsymbol{\nu}\otimes\boldsymbol{\nu}\}\nabla\varphi\, ds \\
&= \int_{\partial\Omega} 2\kappa\boldsymbol{\nu}\cdot\nabla\varphi\, ds = 0.
\end{aligned} \tag{3.31}$$

Now noticing that $\{\boldsymbol{t}\otimes\boldsymbol{t}\}$ is a symmetric operator we can rewrite the first integral in (3.30) as

$$\int_\Omega \nabla(2\kappa|\nabla u|)\cdot\frac{1}{|\nabla u|}\{t\otimes t\}\nabla\varphi\,dxdy=\int_\Omega\{t\otimes t\}\frac{1}{|\nabla u|}\nabla(2\kappa|\nabla u|)\cdot\nabla\varphi\,dxdy \tag{3.32}$$

and by integrating by parts

$$-\int_\Omega\{t\otimes t\}\frac{1}{|\nabla u|}\nabla(2\kappa|\nabla u|)\cdot\nabla\varphi\,dxdy=$$
$$\int_\Omega\left(\nabla\cdot\{t\otimes t\}\frac{1}{|\nabla u|}\nabla(2\kappa|\nabla u|)\right)\varphi\,dxdy-\int_{\partial\Omega}\{t\otimes t\}\frac{1}{|\nabla u|}\nabla(2\kappa|\nabla u|)\cdot\boldsymbol{\nu}\,ds. \tag{3.33}$$

The required boundary condition to drop the boundary integral in (3.33) is given by

$$\{t\otimes t\}\frac{1}{|\nabla u|}\nabla(2\kappa|\nabla u|)\cdot\boldsymbol{\nu}=0. \tag{3.34}$$

Here, as in (3.30), we can use the fact that $\boldsymbol{\nu}=\pm t$ along $\partial\Omega$, so (3.34) can be expressed simply as

$$\nabla(2\kappa|\nabla u|)\cdot\boldsymbol{\nu}=0. \tag{3.35}$$

Coming back to the second integral in (3.28), we proceed in a similar way. That is,

$$\begin{aligned}&\int_\Omega\left(\nabla\cdot\frac{\nabla u}{|\nabla u|}\right)^2\frac{d}{d\varepsilon}|\nabla(u+\varepsilon\varphi)|\bigg|_{\varepsilon\to0}dxdy\\ &=\int_\Omega\kappa^2\left(\frac{\nabla(u+\varepsilon\varphi)}{|\nabla(u+\varepsilon\varphi)|}\cdot\nabla\varphi\right)\bigg|_{\varepsilon\to0}dxdy=\int_\Omega\kappa^2\frac{\nabla u}{|\nabla u|}\cdot\nabla\varphi\,dxdy\\ &=-\int_\Omega\left(\nabla\cdot\left(\kappa^2\frac{\nabla u}{|\nabla u|}\right)\right)\varphi\,dxdy+\int_{\partial\Omega}\kappa^2\frac{\nabla u}{|\nabla u|}\cdot\boldsymbol{\nu}\,ds,\end{aligned} \tag{3.36}$$

where $\kappa^2\geq0$, so requiring again (3.27) is enough to drop the boundary integral.

Finally, we compute the first variation of the third energy E_3 as follows:

$$\frac{d}{d\varepsilon}E_3(u+\varepsilon\varphi)\bigg|_{\varepsilon\to0}=\frac{d}{d\varepsilon}\int_\Omega(u+\varepsilon\varphi-z)^2\,dxdy\bigg|_{\varepsilon\to0}=\int_\Omega2(u-z)\varphi\,dxdy. \tag{3.37}$$

As final step, we only need to put all the results together to make the Euler-Lagrange equation explicit. Thus, from (3.26), (3.33), (3.36) and (3.37) and using the boundary conditions (3.27) and (3.35), the Euler-Lagrange equation for the Euler's-elastica model is given by

$$-\nabla\cdot\left(\left(a+b\kappa^2\right)\frac{\nabla u}{|\nabla u|}-b\{t\otimes t\}\frac{1}{|\nabla u|}\nabla(2\kappa|\nabla u|)\right)+\lambda_E(u-u^0) \tag{3.38}$$
$$=-\nabla\cdot\left(\left(a+b\kappa^2\right)\frac{\nabla u}{|\nabla u|}-\frac{b}{|\nabla u|}\frac{\partial(\nabla(2\kappa|\nabla u|))}{\partial t}t\right)+\lambda_E(u-u^0)=0. \tag{3.39}$$

Since $\boldsymbol{n}=\nabla u=(u_x,u_y)$ and $t=\nabla^\perp u=(-u_y,u_x)$ it is then possible to rewrite

(3.39) as (3.20) □.

To write (3.20) in a more compact form, define a vector field $\boldsymbol{V} = \langle V^1, V^2 \rangle$ by $\boldsymbol{V} = (a + b\kappa^2)\frac{\nabla u}{|\nabla u|} - \frac{2b}{|\nabla u|^3}\nabla^{\perp}u\nabla(\kappa|\nabla u|)\nabla^{\perp}u$, that is,

$$V^1 = (a + b\kappa^2)\frac{\partial_x u}{|\nabla u|} + \frac{2b}{|\nabla u|^3}\left[-\partial_y u\ \partial_x(\kappa|\nabla u|) + \partial_x u\ \partial_y(\kappa|\nabla u|)\right]\ \partial_y u,$$
$$V^2 = (a + b\kappa^2)\frac{\partial_y u}{|\nabla u|} - \frac{2b}{|\nabla u|^3}\left[-\partial_y u\ \partial_x(\kappa|\nabla u|) + \partial_x u\ \partial_y(\kappa|\nabla u|)\right]\ \partial_x u. \qquad (3.40)$$

Here ∂_ω denotes the derivative with respect to any variable ω. Then with the above notation, equation (3.20) becomes $\nabla \cdot \boldsymbol{V} + \lambda_E(u^0 - u) = 0$ or

$$\alpha\nabla \cdot \boldsymbol{V} + \chi(u^0 - u) = 0, \qquad (3.41)$$

if we let

$$\alpha = \begin{cases} \frac{1}{\lambda} & \text{in } \Omega\backslash D \\ 1 & \text{in } D \end{cases} \quad \text{and } \chi = \begin{cases} 1 & \text{in } \Omega\backslash D \\ 0 & \text{in } D. \end{cases}$$

By splitting $\boldsymbol{V} = \boldsymbol{\mathcal{N}} + \boldsymbol{\mathcal{T}}$ further (along normal and tangent directions) with $\boldsymbol{\mathcal{N}} = (a + b\kappa^2)\frac{\nabla u}{|\nabla u|}$ and $\boldsymbol{\mathcal{T}} = -\frac{2b}{|\nabla u|^3}\nabla^{\perp}u\nabla(\kappa|\nabla u|)\nabla^{\perp}u$, the above PDE (3.41) can be written as

$$\alpha\,(\nabla \cdot \boldsymbol{\mathcal{N}} + \nabla \cdot \boldsymbol{\mathcal{T}}) + \chi(u^0 - u) = 0, \qquad (3.42)$$

which has a geometry interpretation on the transporting mechanisms of the level sets information [125]. In (3.42) two competitive terms transporting the information in different directions, $\nabla \cdot \boldsymbol{\mathcal{N}}$ across the level sets as in [122] and $\nabla \cdot \boldsymbol{\mathcal{T}}$ along the level sets as in [10].

Remark 3.3.2 *The elastica PDE can be written in a different form. This is done by starting from the general geometric form (3.38) and replacing the tangential projection term by $\boldsymbol{I} - \boldsymbol{n} \otimes \boldsymbol{n}$, where*

$$\boldsymbol{n} \otimes \boldsymbol{n} = \frac{\nabla\varphi \cdot \nabla u}{|\nabla u|^2}\nabla u. \qquad (3.43)$$

Hence we obtain

$$-\nabla \cdot \left(\frac{(a + b\kappa^2)\nabla u}{|\nabla u|} + \frac{2b\nabla u \cdot \nabla(\kappa|\nabla u|)}{|\nabla u|^3}\nabla u - \frac{2b\nabla(\kappa|\nabla u|)}{|\nabla u|}\right) + \lambda_E(u - u^0) = 0. \quad (3.44)$$

This equation shall be used in a future chapter to develop a promising primal-dual method for the Euler's elastica formulation

Virtues of the Euler's elastica inpainting model

As an illustration of the advantages of using the elastica model we show in the Figures 3.10 and 3.11 the results obtained from using the elastica model on the two problems we have already presented and where we know that the TV model fails.

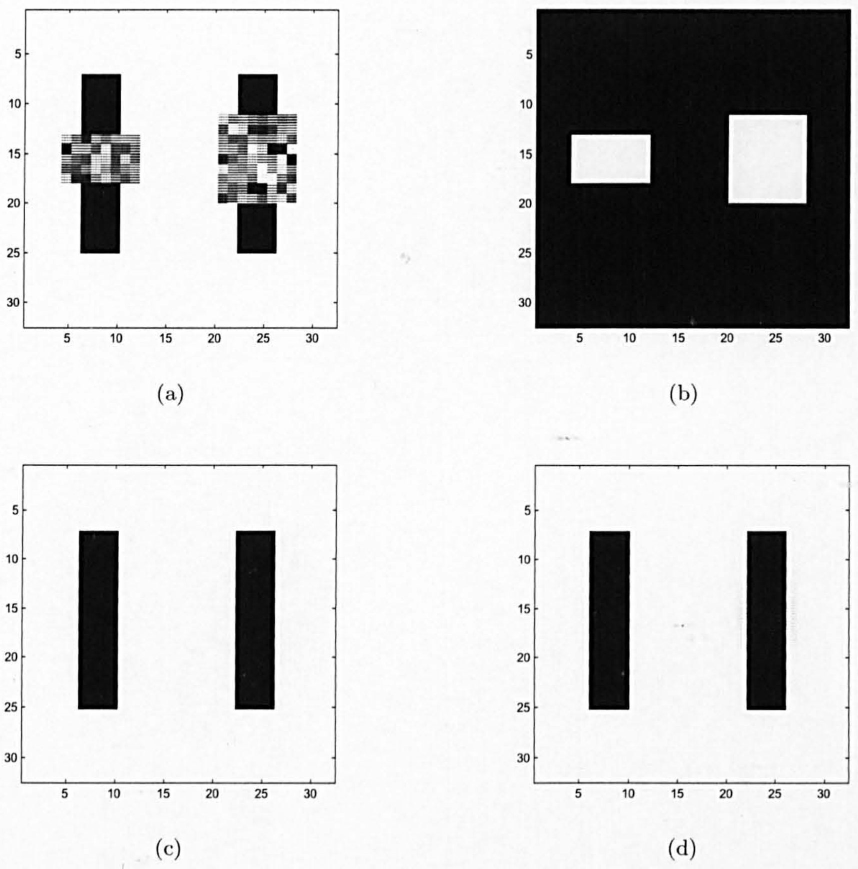


Figure 3.10: (a) An image with two broken bars. The missing region of both is large in size compared with the width of the bars. (b) The mask of the inpainting domain (c) The true solution (d) The visually correct solution from the Euler's elastica model. The broken bars are reconnected.

3.3.6 Level lines continuation

The Euler's elastica model of Chan *et al.* [125] was actually not the first to introduce the elastica energy in the variational image processing community. The elastica energy $\int_\Gamma (a + |\kappa|^p) d\mathcal{H}$ with Γ a curve, κ its curvature and $\mathcal{H}$ the one-dimensional Hausdorff measure is mainly known by the work of Mumford *et al.* [100] on disocclusion, although, as detailed in [30], it had been previously used by others in different contexts.

One of these works closely related to inpainting was presented by Masnou and Morel for the first time in [97] and later explained in more detail in [96]. There the authors follow the *amodal completion* technique studied by Kanizsa [79] consisting on the smooth continuation of object boundaries between T-junctions (points where the image edges form a "T") to reconstruct partially hidden objects.

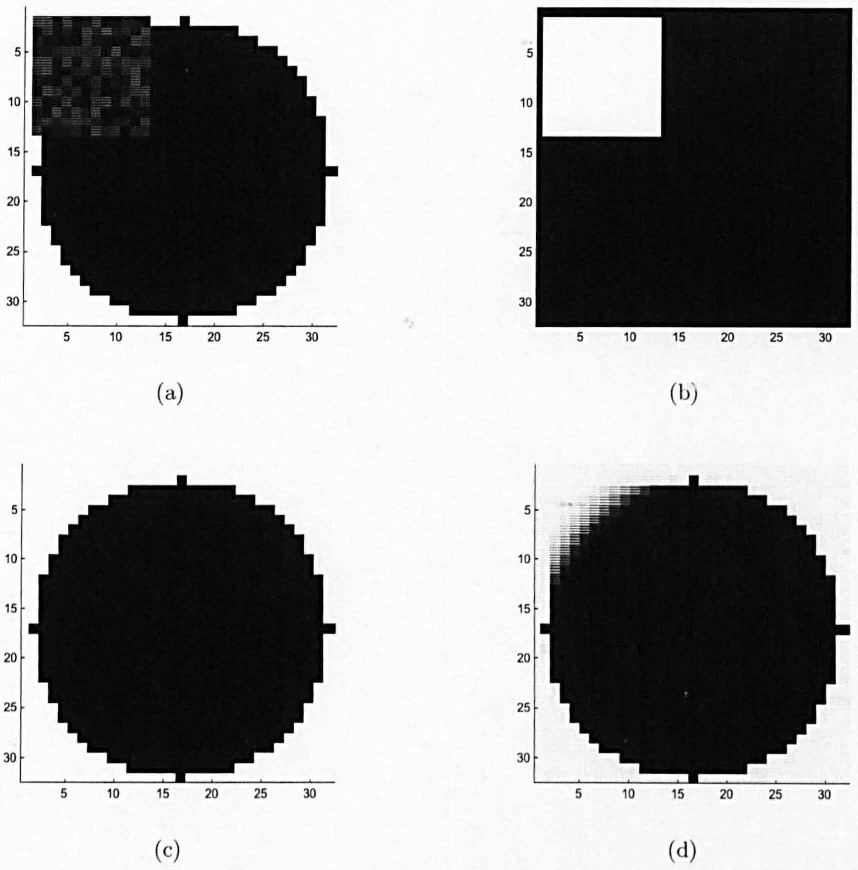


Figure 3.11: (a) A circle with a missing part. (b) The mask of the inpainting domain. (c) The true solution. (d) The Euler's elastica inpainting result. Notice this time the curvature of the circle is restored.

There are delicate differences between the approach of [96, 97] and the elastica model. For instance, Masnou-Morel's technique cannot recover curvy level lines as the elastica model (3.19) can do and it is also not robust when noise is added to the image. On the other hand, Masnou and Morel presented a practical algorithm which makes use of dynamic programming to carry out the disocclusion process. Advantages of this algorithm are: easily recovering of sharp edges and low order of complexity; see below. However, one strong limitation is that it is not topology free since it assumes the occlusion to be without hole. A very detailed explanation of this method can be found in [96] and here we just reproduce its basic steps in Algorithm 8.

The complexity of Masnou and Morel algorithm, as reported in [96], is

$$O(V^2 + VT^2 + T^3 + P) \text{ which can be reduced to } O(V^2 + VTlogT + T^3 + P),$$

where V is the number of vertices in the occlusion boundary, T is the number of T-junctions and P is the number of pixels within the occlusion.

Algorithm 8 Level lines disocclusion algorithm

```
1: Compute the polygonal line corresponding to the occlusion boundary.
2: Computation of the T-junctions on the occlusion boundary.
3: Triangulation of the occlusion.
4: Use of dynamic programming to compute the optimal set of level lines pairwise
   connecting T-junctions.
5: Drawing of the geodesic paths.
6: Use of the geodesic propagation to restore the image.
```

3.3.7 Filling-in through vector fields and gray levels

The second related work we review here is the filling-in model of Ballester *et al.* proposed in [9]. For convenience we shall refer to this model as the BBCSV model. The BBCSV model differs from the elastica model (3.19) in that it diffuses a vector field and gray levels at same time within the missing regions. The functional

$$\min_{u\in BV,\ \theta\in W^{1,p}} \left\{ \int_D (a + b|\nabla K * u|)|\nabla\cdot\boldsymbol{\theta}|^p \, dxdy + \int_D (|\nabla u| - \boldsymbol{\theta}\cdot\nabla u)\, dxdy \right\} \qquad (3.45)$$

is minimized by evolving in time a pair of PDE's, of second and third order respectively. Here a, b are positive parameters as before, $p > 1$, D the inpainting domain, K a convolution kernel and $\boldsymbol{\theta}$ a new variable representing the vector field in D.

The first term in (3.45) which can be rewritten as

$$\int_D (a + b|\nabla K * u|)|\nabla\cdot\frac{\nabla u}{|\nabla u|}|^p \, dxdy \qquad (3.46)$$

by using the change of variables $\boldsymbol{\theta} = \frac{\nabla u}{|\nabla u|}$, can be seen as a relaxation of the elastica energy $\int_D (a + b|\kappa|^p)|\nabla u|$ as stated in [9] and hence its relation with the elastica model (3.19). The major two differences between these two models are:

- The elastica model leads to morphologically invariant flows, [125].
- The numerical solution of the BBCSV model is more friendly since a system of three second order PDE's have to be solved. This compared with the elastica model where the solution of a fourth-order PDE (3.20) has to be found.

On the other hand, it is not clear from [9] if the BBCSV model or a modified version of it by adding a fitting term similar to (3.19) is robust when noise is added to the image.

3.3.8 The Cahn-Hilliard model

Due to the success of previous variational methods for image inpainting, in particular the Euler's elastica model, researchers started looking for possible different models with similar characteristics to the elastica model but with the additional aim of having a fast numerical solver. Under these requirements, a slightly modified Cahn-Hilliard equation was proposed and studied in [13, 12, 27]. Although the Cahn-Hilliard model satisfies these conditions for the case of binary images (two gray-level values only), its performance for general gray-level images is not as good as for the binary case.

This model proposes to minimize the following functional

$$\min_{u\in L^2}\left\{\int_\Omega \frac{\epsilon}{2}|\nabla u|^2+\frac{1}{\epsilon}W(u)\,dxdy+\lambda\int_{\Omega\setminus D}(u^0-u)^2\,dxdy\right\}, \tag{3.47}$$

where $W(u)$ is a nonlinear potential with wells or zeros corresponding to values of u that are taken on by most of the gray-scale values. An example is

$$W(u)=u^2(u-1)^2 \tag{3.48}$$

with minima at $u=0$ and $u=1$.

By using the interpolation inequality [12]

$$\int_\Omega |\nabla u|\,dxdy\le\delta\int_\Omega(\Delta u)^2\,dxdy+\frac{C}{\delta}\int_\Omega |u|^2\,dxdy, \tag{3.49}$$

where δ, C are positive constants. It is possible to show [12] that the corresponding Euler-Lagrange equation for the modified Cahn-Hilliard functional (3.47) is given by

$$-\Delta(\epsilon\Delta u-\frac{1}{\epsilon}W'(u))+\lambda(u^0-u)=0. \tag{3.50}$$

A convexity splitting semi-implicit method can be easily implemented together with the Fast-Fourier transform method. The Cahn-Hilliard model can reconnect separated parts of objects through long distances and is able to recover curvature as well.

3.3.9 The Mumford-Shah and Mumford-Shah-Euler models

These two models are based on the object-edge Mumford and Shah image model

$$E[u,\Gamma]=\frac{\gamma}{2}\int_{\Omega\setminus\Gamma}|\nabla u|^2\,dxdy+\alpha H^1(\Gamma), \tag{3.51}$$

where Γ denotes the collection of edges in the image and H^1 the one-dimensional Hausdorff measure. Computationally, $H^1(\Gamma)$ is substituted by length(Γ) since it is a generalization of the notion of length of regular curves.

Mumford-Shah inpainting model

In the Mumford-Shah model for image inpainting it is proposed to minimize with respect to u the following functional:

$$J_{ms}[u,\Gamma|u^0,D]=\frac{\lambda}{2}\int_\Omega(u-u^0)^2\,dxdy+E[u,\Gamma] \tag{3.52}$$

where

$$E[u,\Gamma] = \frac{\gamma}{2}\int_{\Omega\backslash\Gamma} |\nabla u|^2 \, dxdy + \alpha \text{length}(\Gamma). \tag{3.53}$$

The minimization however is not straight forward since the three terms in (3.52), via (3.53), are defined on different domains. The functional may be reformulated using the level-set representation [103], or approximated using gamma-convergence techniques. Here we present only the latter.

To this end, the Γ set is approximated by its signature function $z_\epsilon : \Omega \longrightarrow [0,1]$ which is nearly 1 almost everywhere within the image except on a tiny tubular neighborhood of Γ of radius ϵ where it is close to 0. Using the signature function z_ϵ let us represent all terms in $E[u,\Gamma]$ on the Ω domain. Further, by using the approximation of the length(Γ) function given in [3, 4], the new representation of $E[u,\Gamma]$ is as follows:

$$E_\epsilon[z|u] = \frac{\gamma}{2}\int_\Omega z^2|\nabla u|^2 \, dxdy + \alpha\int_\Omega \left(\epsilon|\nabla z|^2 + \frac{(1-z)^2}{4\epsilon}\right) dxdy. \tag{3.54}$$

The Mumford-Shah inpainting model is finally defined as the minimization of

$$J_{ms}[u,\Gamma|u^0,D] = \frac{\lambda}{2}\int_\Omega (u-u^0)^2 \, dxdy + E_\epsilon[z|u]. \tag{3.55}$$

Taking variations on u and z respectively yield two second order nonlinear partial differential equations for which semi-implicit methods can be implemented [54].

Unfortunately, this model shares the same drawbacks of the TV model: (1) failure of reconnection across long distances, and (2) emerging of artificial corners due to the lack of curvature information in the energy functional (3.55).

Mumford-Shah-Euler inpainting model

The Mumford-Shah-Euler model was created to fix the problems of the Mumford-Shah model. Similar to the Euler-elastica inpainting model [125], it proposes to replace the length function in (3.53) by the Euler's elastica curve model $e(\Gamma)$. This is,

$$E[u,\Gamma] = \frac{\gamma}{2}\int_{\Omega\backslash\Gamma} |\nabla u|^2 \, dxdy + e(\Gamma), \tag{3.56}$$

where

$$e(\Gamma) = \int_\Gamma (a + b\kappa^2) \, ds. \tag{3.57}$$

As before, we need all the integrals to be defined in the same domain Ω. In [54], the authors showed how to do this using the level-set method for a two-phase problem. However by doing so we are automatically limiting the inpainting model to binary or high contrast images, in similar situation to the Cahn-Hilliard model. Although multi-phase methods may be implemented they do not appear to be a viable option for general gray-scale images. The Γ-convergence approximation therefore looks again as the better approach.

To this end, considering a signature function z_ϵ as before and using the following De Giorgi's approximation [64, 93] to (3.57),

$$E_\epsilon[z] = a\int_\Omega \left(\epsilon|\nabla z|^2\, dxdy + \frac{W(z)}{4\epsilon}\right)\, dxdy + \frac{b}{\epsilon}\int_\Omega \left(2\epsilon\Delta z - \frac{W'(z)}{4\epsilon}\right)^2\, dxdy \quad (3.58)$$

with the potential function $W(z)$ defined this time as

$$W(z) = (1-z^2)^2. \quad (3.59)$$

The Γ-convergence approximation of the Mumford-Shah-Euler inpaiting model is expressed as the separately minimization of

$$J_{mse}[u,\Gamma|u^0,D] = \frac{\lambda}{2}\int_\Omega (u-u^0)^2\, dxdy + E_\epsilon[z|u] \quad (3.60)$$

with respect to u and z, yielding second and fourth-order nonlinear PDE respectively.

The Mumford-Shah-Euler model because of its own nature shares the nice properties of the Euler's elastica inpainting model. However, its minimization through Γ-convergence approximation introduces an extra second order PDE to solve. Recall from Section 3.3.5 that in the Euler's elastica model only one fourth-order equation had to be solved. Further, $J_{mse}[u,\Gamma|u^0,D]$ has many minima so numerical solvers can easily get stagnated at local minima.

Chapter 4

Multigrid Method for a Modified Curvature Driven Diffusion Model for Image Inpainting

As a brief note, the work presented in this chapter has already been published [24] in the *Journal of Computational Mathematics* under the title *Multigrid Method For a Modified Curvature Driven Diffusion Model for Image Inpainting* and it is coauthored together with my supervisor Professor Ke Chen.

4.1 Introduction

In this chapter we study the curvature driven diffusion (CDD) model [123] for image inpainting. The CDD model may be regarded as an improved version of the total variation (TV) [121] inpainting model which was designed for local inpaintings only.

For the TV and CDD models, their associated Euler-Lagrange equations are highly nonlinear partial differential equations of second and third order, respectively. For the TV model there exists a relatively fast and easy to implement fixed point method, so adapting the multigrid (MG) method of [119] to here is almost immediate. For the CDD model however, so far only the well-known, but usually very slow explicit time marching method has been reported. Here we explain why the implementation of a fixed point method for the CDD model is not straightforward and consequently, the multigrid method as in [119] will not work here. This fact represents a strong limitation to the range of applications of this model since usually fast solutions are expected.

The main contribution we present in this chapter is a modification of the CDD model designed to enable a fixed point method to work and to preserve the features of the original model. As a result, a fast and efficient multigrid method is developed. Numerical experiments are presented at the end to show the very good performance of the fast MG algorithm.

The rest of the chapter is organized as follows. In Section 4.2 we briefly review the image inpainting problem formulation, and two variational models, followed by the revision of a commonly-used numerical method in Section 4.3. Then in Section 4.4, we describe first the modified CDD model followed by the framework of a nonlinear multigrid method with emphasis on two smoothers: the global smoother and the local smoother. A local Fourier analysis is also shown to give an indication of the effectiveness of both smoothers. Next in Section 4.5 we present some testing results illustrating the virtues of the modification and the associated multigrid method. Finally in Section 4.6 we present our conclusions.

4.2 Problem formulation, the TV and CDD models

We use this Section, to carry out a quick revision of the inpainting's formulation as a variational problem, the TV and CDD models. The first two have been covered in Chapter 3 so we do not extend ourselves again with extensive explanations.

4.2.1 Inpainting formulation

Recall that we are given an image $u^0 = u^0(x, y)$ defined on a domain $\Omega \subset \mathbb{R}^2$ and one subset $D \subset \Omega$ where the pixel values of u^0 are missing or damaged due to some reason. The inpainting problem is to try to reconstruct the values of u^0 in D from the available information on $\Omega \backslash D$ which may contain noise. The subset D which is known as the inpainting domain may have complicated topology and be not necessarily connected.

4.2.2 The total variation model

Assume that $u = u(x, y)$ and $\eta = \eta(x, y)$ are, respectively, the true image and the unknown additive Gaussian noise satisfying $u^0 = u + \eta$ in $\Omega \backslash D$. Following Rudin *et al.* [114], the TV inpainting model is given by the minimization (3.15).

Although direct minimization ideas [32, 33] could be applied, so far the above minimization is mainly solved via its associated Euler-Lagrange equation (3.16).

Recall that limitations of the TV model are:

Short distances reconnection only

The TV model cannot reconnect separated parts of objects that are far apart because the cost of the functional (3.15) in doing so is higher than the cost of no reconnection.

No curvature recovery

The TV model cannot reconstruct the natural curvature of the missing parts. Recall that equation (3.16) reduces to $\kappa = 0$ in the inpainting domain D where $\kappa = \nabla \cdot \frac{\nabla u}{|\nabla u|}$ is the curvature of the level sets meaning that the inpainting process is carried out using straight lines only.

4.2.3 The curvature-driven diffusion model

The CDD model which we review in this section was designed to correct the inability of the TV model in reconnecting separated parts of broken objects far apart. In looking for a solution to this problem, Chan and Shen [123] realized that in the TV model the diffusion coefficient given by

$$\hat{D} = \frac{1}{|\nabla u|}, \tag{4.1}$$

only depends on the contrast or strength of the level lines and it does not depend on the geometric information of the level lines themselves. They found that the curvature could be used to modify the diffusion coefficient $\hat{D}$ by introducing a function $g = g(|\kappa|)$ within it. This way the geometric information encoded in κ is used to strengthen the diffusion coefficient where necessary. The new diffusion coefficient $\hat{D}$ is then given by

$$\hat{D} = \frac{g(|\kappa|)}{|\nabla u|}, \quad \text{with} \quad g(s) = \begin{cases} 0 & s = 0 \\ \infty & s = \infty \\ > 0, & 0 < s < \infty. \end{cases} \tag{4.2}$$

On one hand, the choice of $g(\infty) = \infty$ was selected to take advantage of those points with very high or infinity curvature and use them to encourage reconnection by increasing $\hat{D}$ as much as possible.

On the other hand, the choice of $g(0) = 0$ is to avoid the CDD model degenerating to the TV model. According to Chan and Shen the choice of $g(0) = a \neq 0$ could endanger the connectivity principle, see [123]. They suggested [123]

$$g(s) = s^p, \quad \text{with} \quad s > 0, \quad p \geq 1. \tag{4.3}$$

We shall find another way to satisfy the connectivity principle by allowing $g(0) \neq 0$ in Section 4.4.4.

To get rid of possible noise present in the initial image which could be propagated to the interior of the inpainting domain, a fidelity term is used as in the TV model. Thus, by defining the vector field $V = \langle V^1, V^2 \rangle$ by $V = G\frac{\nabla u}{|\nabla u|}$ with $G = G[(x,y), |\kappa|]$

$$G = \begin{cases} 1 & (x,y) \in \Omega \backslash D \\ |\kappa|^p & (x,y) \in D, \end{cases} \tag{4.4}$$

the CDD scheme is to solve the following third order nonlinear equation for u:

$$\nabla \cdot V + \lambda_E(u^0 - u) = 0 \quad \text{in} \quad \Omega, \tag{4.5}$$

with homogeneous Neumann boundary condition as before. Since V contains the term $|\nabla u|^{-1}$, to avoid the singularity at flat regions $|\nabla u|_\beta \overset{def}{=} \sqrt{|\nabla u|^2 + \beta}$ is used instead of

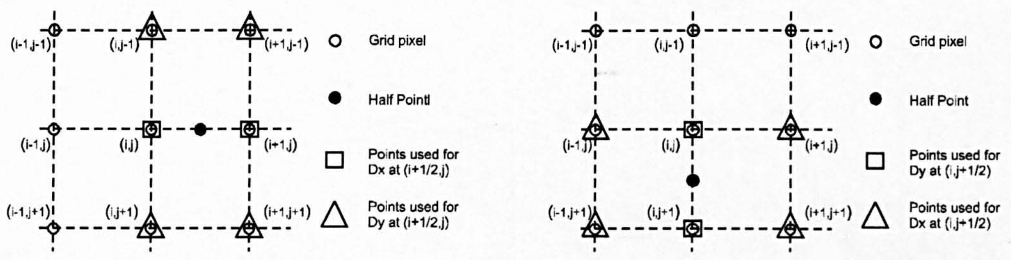

(i-1,j-1)
(i,j-1)
(i+1,j-1)
Grid pixel
Half Point
(i-1,j)
(i,j)
(i+1,j)
Points used for Dx at (i+1/2,j)
(i-1,j+1)
(i,j+1)
(i+1,j+1)
Points used for Dy at (i+1/2,j)
Points used for Dy at (i,j+1/2)
Points used for Dx at (i,j+1/2)

Figure 4.1: On the left side an x-half-point and on the right side a y-half-point.

$|\nabla u|$, where β is a small parameter. Equation (4.5) will become

$$\alpha \nabla \cdot \boldsymbol{V} + \chi(u^0 - u) = 0 \tag{4.6}$$

if we let

$$\alpha = \begin{cases} \frac{1}{\lambda} & \text{in } \Omega \backslash D \\ 1 & \text{in } D \end{cases} \quad \text{and} \quad \chi = \begin{cases} 1 & \text{in } \Omega \backslash D \\ 0 & \text{in } D. \end{cases}$$

4.3 Review of numerical methods

In this section, we intend to review the state-of-the-art methods for numerically solving the CDD model. Surprisingly, the list is very short and it only has been solved using an explicit time marching scheme.

4.3.1 Discretization

We start by discretizing the CDD differential equation (4.5) for a general coefficient G as follows:

$$\frac{V^1_{i+\frac{1}{2},j} - V^1_{i-\frac{1}{2},j}}{h_x} + \frac{V^2_{i,j+\frac{1}{2}} - V^2_{i,j-\frac{1}{2}}}{h_y} + \lambda_E(u^0_{i,j} - u_{i,j}) = 0, \tag{4.7}$$

where h_x and h_y are the grid spacing in the x and y-direction respectively and the discrete image $u^0 \in \mathbb{R}^{m \times n}$; we shall mainly consider the case of $n = m$ hence $h = h_x = h_y$.

We use staggered discretization so now we have to approximate V^1 and V^2 at the half-points (see Figure 4.1), for instance at $(i + \frac{1}{2}, j)$, u_x is approximated by central differences $(u_x)_{i+\frac{1}{2},j} = (u_{i+1,j} - u_{i,j})/h$, u_y by average approximation $(u_y)_{i+\frac{1}{2},j} = (u_{i+1,j+1} - u_{i+1,j-1} + u_{i,j+1} - u_{i,j-1})/4h$ and $|\nabla u|_\beta$ in the natural way :

$$\frac{1}{h}\sqrt{(u_{i+1,j} - u_{i,j})^2 + \left(\frac{1}{4}(u_{i+1,j+1} - u_{i+1,j-1} + u_{i,j+1} - u_{i,j-1})\right)^2 + h^2\beta}. \tag{4.8}$$

Hence equation (4.7) becomes

$$-G_{i+\frac{1}{2},j}\left(\frac{\alpha(u_x)_{i+\frac{1}{2},j}}{h|\nabla u|_{i+\frac{1}{2},j}}\right)+G_{i-\frac{1}{2},j}\left(\frac{\alpha(u_x)_{i-\frac{1}{2},j}}{h|\nabla u|_{i-\frac{1}{2},j}}\right)-G_{i,j+\frac{1}{2}}\left(\frac{\alpha(u_y)_{i,j+\frac{1}{2}}}{h|\nabla u|_{i,j+\frac{1}{2}}}\right)$$
$$+G_{i,j-\frac{1}{2}}\left(\frac{\alpha(u_y)_{i,j-\frac{1}{2}}}{h|\nabla u|_{i,j-\frac{1}{2}}}\right)+\chi u_{i,j}=\chi u^0_{i,j}. \tag{4.9}$$

To approximate the curvature term κ in G or in (4.2), we use the same idea of the ghost half points to approximate the divergence operator

$$\kappa=\nabla\cdot\frac{\nabla u}{|\nabla u|}=\frac{\partial}{\partial x}\left[\frac{u_x}{|\nabla u|}\right]+\frac{\partial}{\partial y}\left[\frac{u_y}{|\nabla u|}\right]. \tag{4.10}$$

By using again central differences and averages we have for example at $(i+\frac{1}{2},j)$ that

$$h\cdot\frac{\partial}{\partial x}\left[\frac{u_x}{|\nabla u|}\right]_{i+\frac{1}{2},j}=\left[\frac{u_x}{|\nabla u|}\right]_{i+1,j}+\left[\frac{u_x}{|\nabla u|}\right]_{i,j}. \tag{4.11}$$

$$4h\cdot\frac{\partial}{\partial y}\left[\frac{u_y}{|\nabla u|}\right]_{i+\frac{1}{2},j}=\left[\frac{u_y}{|\nabla u|}\right]_{i+1,j+1}-\left[\frac{u_y}{|\nabla u|}\right]_{i+1,j-1}$$
$$+\left[\frac{u_y}{|\nabla u|}\right]_{i,j+1}-\left[\frac{u_y}{|\nabla u|}\right]_{i,j-1}. \tag{4.12}$$

Finally (4.5) becomes a system of nonlinear algebraic equations denoted by

$$u_{i,j}\mathbb{S}_{i,j}-u_{i+1,j}\left(C_{i+\frac{1}{2},j}\right)-u_{i-1,j}\left(C_{i-\frac{1}{2},j}\right)-u_{i,j+1}\left(C_{i,j+\frac{1}{2}}\right)$$
$$-u_{i,j-1}\left(C_{i,j-\frac{1}{2}}\right)-\chi u^0_{i,j}=0 \tag{4.13}$$

where the new $C(\cdot,\cdot)$ notation represents the nonlinear terms. For instance,

$$C_{(i+\frac{1}{2},j)}=\frac{\alpha\, G_{(i+\frac{1}{2},j)}}{h|\nabla u|_{(i+\frac{1}{2},j)}} \tag{4.14}$$

and C at the other three half-points is computed in a similar way. Here $\mathbb{S}_{i,j}$ is defined as

$$\mathbb{S}_{i,j}=C_{i+\frac{1}{2},j}+C_{i-\frac{1}{2},j}+C_{i,j+\frac{1}{2}}+C_{i,j-\frac{1}{2}}+\chi. \tag{4.15}$$

The homogeneous Neumann boundary condition on $\partial\Omega$ is determined by the TV denoising model and therefore, is treated as

$$u_{i,0}=u_{i,1},u_{i,n+1}=u_{i,n},u_{0,j}=u_{1,j},u_{m+1,j}=u_{m,j}. \tag{4.16}$$

On the other hand, as in [123], the inpainting domain D is mathematically understood as an open set, i.e., not including its boundary and therefore away from $\partial\Omega$.

4.3.2 Explicit time marching method

In this method, solving (4.5) indirectly, one looks for the steady-state solution of a parabolic equation of the form:

$$\frac{\partial u}{\partial t} = r(u), \qquad \text{with } r(u) = \nabla \cdot V + \lambda_E(u^0 - u), \tag{4.17}$$

or, $r(u) = \alpha \nabla \cdot V + \chi(u^0 - u)$, with the initial condition $u(x, y, 0) = u^0(x, y)$ and appropriate boundary conditions and using an explicit Euler method for the left hand side, we get

$$u_{i,j}^{k+1} = u_{i,j}^k - \tau r(u_{i,j}^k), \;\; k = 0, 1, \ldots \tag{4.18}$$

Here a size restriction on the time step $\tau = \Delta t$ has to be imposed to guarantee the stability of the numerical solution. This is the main drawback of the time marching method, the problem being that due to its high nonlinearity, τ must be chosen very small which implies a large number of iterations to reach a meaningful solution. One option is to accelerate this method using the ideas developed in [95]. However, even in that case the CPU-time consumed by the resulting algorithm is still not appropriate for large images.

4.3.3 A possible fixed point method

For numerically solving the nonlinear algebraic equation (4.13) at each (i, j) point we fix the nonlinear terms C at some k-step and solve for the k+1 step as in [7, 144] for other problems. We then have that

$$\begin{aligned} u_{i,j}^{k+1}\mathbb{S}_{i,j} - u_{i+1,j}^{k+1}\left(C_{i+\frac{1}{2},j}^k\right) - u_{i-1,j}^{k+1}\left(C_{i-\frac{1}{2},j}^k\right) - u_{i,j+1}^{k+1}\left(C_{i,j+\frac{1}{2}}^k\right) \\ -u_{i,j-1}^{k+1}\left(C_{i,j-\frac{1}{2}}^k\right) = \chi u_{i,j}^0, \end{aligned} \tag{4.19}$$

and then such a fixed point method amounts to solving the linear system of equations (4.19)

$$A(\boldsymbol{u}^k)\boldsymbol{u}^{k+1} = \boldsymbol{u}^0, \tag{4.20}$$

where the vectors $\boldsymbol{u}^k$ and $\boldsymbol{u}^0$ are defined as $\boldsymbol{u}^k = [u_{1,1}^k, u_{2,1}^k \ldots, u_{n,1}^k, u_{1,2}^k, \ldots, u_{n,m}^k]$ and $\boldsymbol{u}^0 = [\chi u_{1,1}^0, \chi u_{2,1}^0 \ldots, \chi u_{n,1}^0, \chi u_{1,2}^0, \ldots, \chi u_{n,m}^0]$.

The selection of the diffusion coefficient G in D plays a crucial role on the feasibility of the implementation of the numerical scheme. According to Chan and Shen [123], on the inpainting domain, G must obey equations (4.3) and (4.4). Therefore, it must be

$$G = \begin{cases} 0 & \kappa = 0, \\ \infty & \kappa = \infty, \\ |\kappa|^p, & 0 < |\kappa| < \infty \text{ with } p \geq 1. \end{cases} \tag{4.21}$$

Since we allow G to be 0 when $\kappa = 0$, matrix $A(u^k)$ is singular i.e., whenever $\kappa = 0$ at one (i, j) pixel of the image then $A(u^k)$ losses one degree of its rank. Therefore, the fixed point (FP) method (4.20) does not work for the CDD model with (4.21).

One solution (motivated by numerical consideration) is to modify the above G to $\bar{G} = G + \epsilon$, where ϵ is a small and positive parameter. This idea will be tested shortly.

4.4 Nonlinear multigrid for a modified CDD model

Multigrid methods have proved to be very useful when solving many linear (and some nonlinear) partial differential equations (PDEs) such as those arising from image restoration problems and others, see [33, 34, 45, 119, 118, 139] for successful examples. Usually for a multigrid method to converge, a suitable smoother is the key and the task of finding one is nontrivial for a nonlinear problem. We now proceed to develop a multigrid algorithm for the CDD formulation (4.5):

$$\nabla \cdot \left(G_{i,j} \frac{(\nabla u)_{i,j}}{|\nabla u|_{i,j}} \right) + \lambda_E (u^0_{i,j} - u_{i,j}) = 0. \tag{4.22}$$

4.4.1 The generic multigrid components

First of all, we start by introducing new notation and rewriting the equation for the purpose of making it more tractable for computing implementation. Write (4.22) as

$$(Nu)_{i,j} = -\alpha \nabla \cdot \left(G_{i,j} \frac{(\nabla u)_{i,j}}{|\nabla u|_{i,j}} \right) + \chi u_{i,j} = \chi u^0_{i,j}, \tag{4.23}$$

after we have denoted by $Nu = \chi u^0$ the main nonlinear operator equation; see (4.6). Since we have to approximate this equation on grids of different sizes we will denote by $N_h u_h = \chi u^0_h$ the discrete equation (4.23) defined on the finest grid Ω_h of size h and similarly by $N_{2h} u_{2h} = \chi u^0_{2h}$ the same on the coarser grid Ω_{2h} which is obtained by standard coarsening, i.e., the nonlinear operator N_{2h} which results from defining equation (4.23) on the cell-centered grid Ω_{2h} with grid spacing $2h$. Likewise we can generate a sequence of L coarse levels $2h, 4h, 8h, \ldots, 2^L h$.

Next we briefly mention the standard intergrid transfer operators. Denote by R^{2h}_h (restriction) and I^h_{2h} (interpolation), respectively, two transfer operators between Ω_h and Ω_{2h} which on cell-centered grids are defined by the following equations [139]:

The *restriction operator* is defined by $R^{2h}_h u_h = u_{2h}$ as in (2.84) and the *interpolation operator* is defined by $I^h_{2h} u_{2h} = u_h$ as in (2.86).

4.4.2 Inpainting coarsening

Here we discuss how to coarsen the interfaces, unique to inpainting problems. It suffices to discuss two consecutive grids. First define as D_h the inpainting domain in the finest grid Ω_h and D_{2h} is the coarse grid counterpart to be constructed on Ω_{2h}. Basically, D_h

is identified using a binary mask matrix M_h composed of 1's for points in D_h and 0's for points in $\Omega_h \setminus D_h$. The question is then how to construct a similar M_{2h} on D_{2h}. A simple way is by restriction $M_{2h} = R_h^{2h} M_h$. The problem in doing so is that M_{2h} will have some orphan entries due to the action of the operator R_h^{2h} on M_h. Those entries precisely identify those interface points in Ω_{2h} which were not aligned to the coarse grid. In Figure 4.2 we give an example of such a case. There, the points a, b, c, d in Ω_h

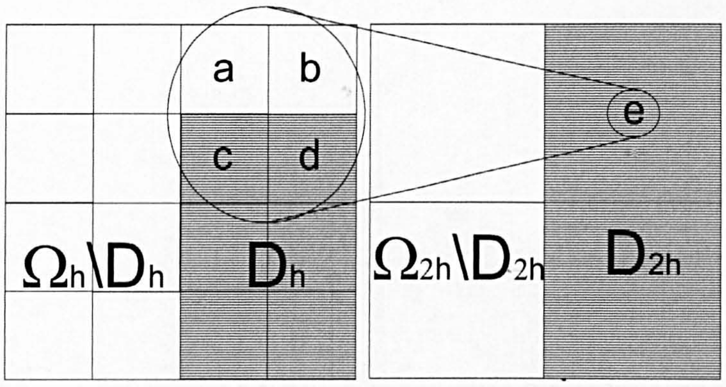


Figure 4.2: When applying standard coarsening to Ω_h, some points in Ω_{2h} are computed using partial information coming from D_h and thus we need to include those points in D_{2h}.

are used to compute the coarse point e in Ω_{2h}, however c and d belong to the inpainting domain D_h and therefore, e will be an orphan. For that reason we decided to include such type of points into the inpainting domain D_{2h} by setting their respective entries in M_{2h} to 1 (i.e. for M_{2h} to take all orphan points if any). The problem in applying this strategy is that the mask starts expanding when moving in the coarsest direction and eventually (depending on the size and topology of D_h) may reach the physical boundary $\partial\Omega$ at some coarse level; in other cases where D_h is disconnected some of its disconnected parts can merge themselves as well. Clearly for inpainting we have a very difficult interpolation problem and it is by no means an easy task to approximate the error at coarse levels. Furthermore, we observed that for some problems additional errors were incorporated by coarse levels. Hence we decided to stop the mask generation at the level for which the mask is just one pixel away from the boundary. This decision obviously prevents using those coarsest levels with only a few points; however this was the one that worked best.

We have also tried another strategy based on the idea of trying to reach the coarsest level using an adaptive mask generation. In this strategy the mask first starts growing till close to the boundary (like in our above idea), but then starts contracting or at least keeping one pixel away from the boundary. Unfortunately, this seemingly reasonable idea gave worst results than the above one (particularly for inpainting binary images).

4.4.3 The MG algorithm and the FAS scheme

To proceed, denote by *FPS* a general fixed-point type smoother; we shall define it shortly. Now use Algorithm 9 to state our V-cycling nonlinear MG, meaning that just one recursive call to the algorithm is made on each level to approximately solve a coarse grid problem.

Algorithm 9 Nonlinear Multigrid Method

```
Require: Select an initial guess $u_h$ on the finest grid $h$
1: $k \leftarrow 0$
2: $err \leftarrow tol + 1$
3: while $err < tol$ do
4:    $u_h^{k+1} \leftarrow FAS(u_h^k, N_h^k, u_h^0, M_h, \nu_0, \nu_1, \nu_2, gsiter, \alpha)$
5:    $err = \|u_h^k - u_h^{k-1}\|_2,$
6:    $k \leftarrow k + 1$
7: end while
```

Here the full approximation scheme (*FAS*) is defined [119, 139] recursively in Algorithm 10 with *gsiter* representing the number of inner Gauss-Seidel iterations at each pre or post-smoothing FPS step.

Algorithm 10 FAS Cycle $u_h \leftarrow FAS(u_h, N_h, u_h^0, M_h, \nu_0, \nu_1, \nu_2, gsiter, \alpha)$

```
1: if $\Omega_h$ = coarsest grid then
2:    solve $N_h u_h = \chi u_h^0$ accurately (i.e. $\nu_0$ iterations by FPS) and return.
3: else
4:    continue with step 6.
5: end if
6: Pre-smoothing: Do $\nu_1$ steps of, $u_h \leftarrow FPS(u_h, u_h^0, M_h, gsiter, \alpha, \nu_1)$
7: Restrict to the coarse grid, $u_{2h} \leftarrow R_h^{2h} u_h$
8: Set the initial solution for the next level, $\bar{u}_{2h} \leftarrow u_{2h}$
9: Compute the new right hand side $\chi u_{2h}^0 \leftarrow R_h^{2h}(\chi u_h^0 - N_h u_h) + N_{2h} u_{2h}$
10: Implement $u_{2h} \leftarrow FAS_{2h}(u_{2h}, N_{2h}, u_{2h}^0, \nu_0, \nu_1, \nu_2, gsiter, \alpha)$
11: Add the residual correction, $u_h \leftarrow u_h + I_{2h}^h(u_{2h} - \bar{u}_{2h})$
12: Post-smoothing: Do $\nu_2$ steps of $u_h \leftarrow FPS(u_h, u_h^0, M_h, gsiter, \alpha, \nu_2)$
```

4.4.4 The modified CDD model

Although the Algorithm 10 appears applicable to solving (4.23), it requires a suitable smoother FPS. To this end, we have tried various choices of FPS, but failed to find a working smoother; the fixed-point method from Section 4.3.3 cannot be used as an efficient smoother.

In trying to find a smoother that would work with Algorithm 10, our idea is to address the problem described in Section 4.3.3 by overcoming the singularity associated with the FP method. Motivated by the Euler's Elastica model [125], we decided to introduce two positive parameters a and b in the function g. Specifically our proposal for the diffusion coefficient G is

$$G = \begin{cases} a & \kappa = 0 \\ \infty & \kappa = \infty \\ a + b|\kappa|^p, & \text{with } p \geq 1 \text{ and } 0 < |\kappa| < \infty. \end{cases} \tag{4.24}$$

For a general p, our modified CDD takes the form

$$\nabla \cdot ((a + b|\kappa|^p)) \frac{\nabla u}{|\nabla u|} + \lambda_E(u^0 - u) = 0. \tag{4.25}$$

Since $\kappa \to 0$ in flatter regions, it is speculated in [123] that the connectivity principle is put at risk if the diffusion coefficient is not zero. We shall demonstrate via our numerical experiments that for the modified model (4.25), as long as a is sufficiently small, we can satisfy the connectivity principle, i.e., reconnect the contours across large distances. However, if we select a too small compared with b we introduce instability to the linear system (4.13). Experimentally we found that $20 < \frac{b}{a} < 250$ works well.

Note that, setting $b = 1$ and $a = 0$ reduces the modified model to the original CDD model [123]. On the other hand, setting $a = 1$ and $b = 0$ transforms the modified model to the TV model [121]. Therefore, our inpainting model (4.25) is half way between the CDD model and the TV model. Moreover, as we shall see, it inherits the virtue of the original CDD model in reconstructing large scale missing parts while sharing with the TV inpainting model the advantage of having a working fixed point (FP) method which in turn can provide a fast and efficient smoother for a multilevel method.

Remark 4.4.1 *Note that the CDD equation (4.5) has an interesting relation with the Euler's elastica model. Recall from Chapter 3 that the elastica model inpaints in two directions: the tangential and the normal to the level sets directions. By selecting $b = 1$ and $p = 2$ in the elastica model and by neglecting part of the normal direction setting $a = 0$ and also neglecting the tangential direction we obtain the CDD model of Chan and Shen. Actually, this omission of the tangential component is precisely the reason that prevents the CDD model to have the capability of denoising as well. In our modified CDD model we followed a similar reasoning, but keeping the full normal direction by selecting $a \neq 0$.*

4.4.5 A global smoother

Now we can similarly consider a fixed point method for numerically solving the discretized equations of (4.24-4.25) at each (i, j) point. To do this we fix the nonlinear terms G and $|\nabla u|$ at the current k-step and solve for the new $k+1$ step. Thus we again obtain that

$$\begin{aligned} u_{i,j}^{k+1}\mathbb{S}_{i,j} - u_{i+1,j}^{k+1}\left(C_{i+\frac{1}{2},j}^k\right) - u_{i-1,j}^{k+1}\left(C_{i-\frac{1}{2},j}^k\right) - u_{i,j+1}^{k+1}\left(C_{i,j+\frac{1}{2}}^k\right) \\ -u_{i,j-1}^{k+1}\left(C_{i,j-\frac{1}{2}}^k\right) = \chi z_{i,j} \end{aligned} \tag{4.26}$$

where

$$C^k_{(i+\frac{1}{2},j)} = \frac{\alpha\, G^k_{(i+\frac{1}{2},j)}}{h|\nabla u^k|_{(i+\frac{1}{2},j)}}, \tag{4.27}$$

and so on, and

$$\mathbb{S}_{i,j} = C^k_{i+\frac{1}{2},j} + C^k_{i-\frac{1}{2},j} + C^k_{i,j+\frac{1}{2}} + C^k_{i,j-\frac{1}{2}} + \chi, \tag{4.28}$$

whose corresponding linear system $A(u^k)u^{k+1} = \chi u^0$ is now never singular and hence solvable.

We can check that in this case $A(u^k)$ is a symmetric and sparse block-tridiagonal matrix with the important feature that it is weakly diagonally dominant. To show this, we can choose and arbitrary row of it and see that all the $a_{i,j}$ with $i \neq j$ entries but at most four are equal to zero. The nonzero entries in this row are given by $C^k_{i+\frac{1}{2},j}$, $C^k_{i-\frac{1}{2},j}$, $C^k_{i,j+\frac{1}{2}}$ and $C^k_{i,j-\frac{1}{2}}$, respectively, and all are positive. The diagonal entry $a_{i,i}$ on the other hand is computed using (4.28). Thus, we have that $a_{i,i} \geq \sum_{i\neq j}^{n} |a_{i,j}|$ in Ω with strict inequality in $\Omega\backslash D$ and therefore $A(u^k)$ is weakly diagonally dominant. Furthermore, because $G \equiv 1$ in $\Omega\backslash D$ and $G = a + b|\kappa|^p$ with $p \geq 1$ in D we can deduce by using the Gerschgorin theorem that $A(u^k)$ is positive semi-definite.

Therefore, with $A(u^k)$ having such a property, we can apply Gauss-Seidel (GS) iterations to solve the linear system (4.26): $A(u^k)u^{k+1} = \chi u^0$. As a smoother, we shall name this fixed-point method with GS iterations the FPGS smoother, which can be stated by as follows.

Algorithm 11 FPGS smoother $u \leftarrow FPGS(u_h, u_h^0, M_h, gsiter, h, \alpha, \nu)$

```
Require: On a grid with mesh size $h$, choose an initial guess $u_h$ for (4.26)
1: for $k = 1$ to $\nu$ do
2:     Apply $gsiter$ Gauss-Seidel iterations to the linear system $A_h(u_h^k)u_h^{k+1} = u_h^0$
3: end for
```

4.4.6 A local smoother

Following from [7, 45, 118] for solving other PDEs, we also considered a local-type fixed point smoother. The difference of this smoother from the global one is that we apply the relaxation steps *locally*. The scheme is the same as that of (4.26) with k representing the kth step. The idea is to update $u_{i,j}$ with the nonlinear terms C^k fixed at each k local step meaning that when we apply relaxation to the next (i,j) point, some C nonlinear terms have been updated. We call this smoother FPLS and is defined as follows:

Here $\bar{C}^k_{i,j}$ means C^k evaluated at $\bar{u}_{i,j}$ and the same applies to $\bar{\mathbb{S}}_{i,j}$, as in (4.28).

Algorithm 12 FPLS smoother $u \leftarrow FPLS(u, u^0, M_h, gsiter, h, \alpha, \nu)$

```
Require: On a grid with mesh size $h$, choose an initial guess
1: for $i = 1$ to $m$ do
2:   for $j = 1$ to $n$ do
3:     for $k = 1$ to $gsiter$ do
4:       $\bar{u}_h \leftarrow u_h$ and update $u_{i,j}$ by solving the linear equation
5:
```

$$u_{i,j}^{k+1}\bar{\mathbb{S}}_{i,j} - u_{i+1,j}^{k+1}\left(\bar{C}_{i+\frac{1}{2},j}^{k}\right) - u_{i-1,j}^{k+1}\left(\bar{C}_{i-\frac{1}{2},j}^{k}\right) - u_{i,j+1}^{k+1}\left(\bar{C}_{i,j+\frac{1}{2}}^{k}\right)$$
$$-u_{i,j-1}^{k+1}\left(\bar{C}_{i,j-\frac{1}{2}}^{k}\right) = \chi u_{i,j}^{0}. \tag{4.29}$$

```
6:     end for
7:   end for
8: end for
```

4.4.7 Use of the smoothers as independent methods

On their own, both smoothers are not always convergent; this situation is different from the image denoising case [144]. Consider the *test problem* illustrated at the top of Figure 4.3. In the middle and at the bottom of the same Figure we show the results obtained from using FPGS and FPLS as independent methods. The failure of both in trying to solve the problem is evident. The PSNR measure used is as defined in Section 4.5.

However, our main interest is not on the convergence of both smoothers, but in their smoothing capabilities. In this case as we shall show, FPGS has better smoothing rates than FPLS and therefore, it is our preferred smoother for a nonlinear MG algorithm.

4.4.8 Local Fourier analysis

We just have seen that both FPGS and FPLS are not always convergent. However, to be used in a multigrid algorithm we only need them to reduce (smooth) as much as possible the high frequency components of the error regardless the overall error itself.

When dealing with nonlinear problems the local Fourier analysis (LFA) can still be used as a tool to check if a given smoother is effective in reducing the high frequency components; see [7, 139, 34]. For simplicity, we consider the case of a square image of size $m \times m$. Let the local error functions $\xi_{i,j}^{k+1}$ and $\xi_{i,j}^{k}$ be defined as $\xi_{i,j}^{k+1} = u_{i,j} - u_{i,j}^{k+1}$ and $\xi_{i,j}^{k} = u_{i,j} - u_{i,j}^{k}$.

Then LFA involves expanding them in Fourier components as

$$\xi_{i,j}^{k+1} = \sum_{\phi_1,\phi_2=-m/2}^{m/2} \psi_{\phi_1,\phi_2}^{k+1} B_{\phi_1,\phi_2}(x,y) \quad \text{and} \quad \xi_{i,j}^{k} = \sum_{\phi_1,\phi_2=-m/2}^{m/2} \psi_{\phi_1,\phi_2}^{k} B_{\phi_1,\phi_2}(x,y) \tag{4.30}$$

and the Fourier component $B_{\phi_1,\phi_2}(x,y) = B_h(\boldsymbol{\theta},\cdot) = e^{i\theta_1 x/h}e^{i\theta_2 y/h}$ with $\boldsymbol{\theta} = (\theta_1,\theta_2) \in$

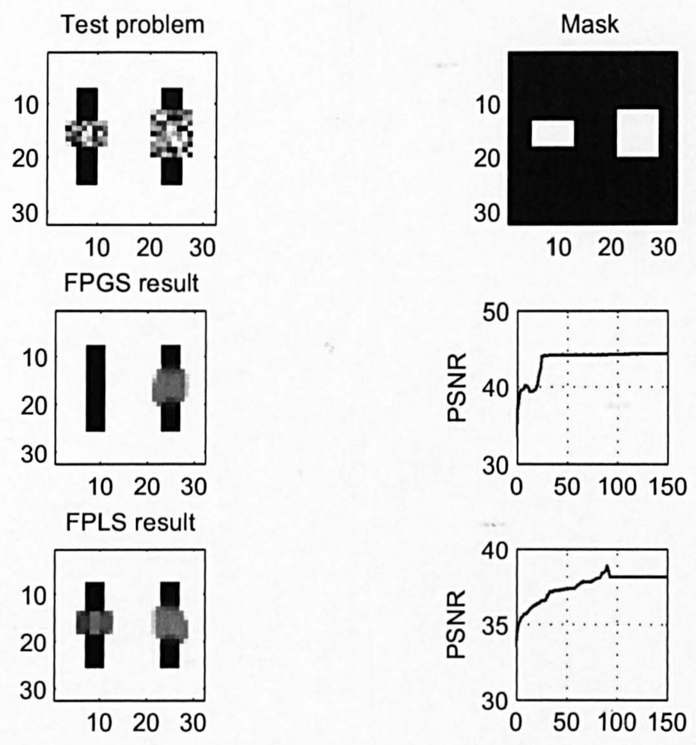


Figure 4.3: Test problem.

$\Theta = (-\pi, \pi]^2$, $\theta_1 = 2\pi\phi_1/m$, $\theta_2 = 2\pi\phi_2/m$ and $\boldsymbol{i} = \sqrt{-1}$. Now by substituting (4.30) into (4.26) we can obtain the error equation

$$-\mathbb{S}^k_{i,j}\xi^{k+1}_{i,j} + C^k_{i+\frac{1}{2},j}\ \xi^k_{i+1,j} + C^k_{i-\frac{1}{2},j}\ \xi^{k+1}_{i-1,j} + C^k_{i,j+\frac{1}{2}}\ \xi^k_{i,j+1} + C^k_{i,j-\frac{1}{2}}\ \xi^{k+1}_{i,j-1} = 0, \quad (4.31)$$

and for the linearized system (freezing all $C^k_{\cdot,\cdot}$ - coefficients), the local amplification factor is given by

$$\tilde{S}_h(\boldsymbol{\theta})_{i,j} = \frac{\left| C^k_{i+\frac{1}{2},j}\ e^{\boldsymbol{i}\theta_1} + C^k_{i,j+\frac{1}{2}}\ e^{\boldsymbol{i}\theta_2} \right|}{\left| \mathbb{S}^k_{i,j} - C^k_{i-\frac{1}{2},j}\ e^{-\boldsymbol{i}\theta_1} - C^k_{i,j-\frac{1}{2}}\ e^{-\boldsymbol{i}\theta_2} \right|}. \quad (4.32)$$

What follows is to compute the nonlinear coefficients $C_{(\cdot,\cdot)}$ at each (i, j) point to find the smoothing factor at the kth-step and at each (i, j)-location of the image as

$$\mu_{i,j} = \sup\{|\tilde{S}_h(\boldsymbol{\theta})_{i,j}| : \boldsymbol{\theta} \in \Theta^{high} = [-\pi, \pi)^2 \backslash [-\frac{\pi}{2}, \frac{\pi}{2})^2\}. \quad (4.33)$$

Recall that both (FPGS and FPLS) smoothers only differ in when we update the C coefficients.

Since our FPGS and FPLS smoothers are linearized at each step, their smoothing rates will change at every outer iteration. Then we have an $m \times m$ rate matrix $\bar{M}_k$, for the kth step, with entry $\mu_{i,j}$ representing the local smoothing rates at (i,j) point. Unfortunately, our initial tests showed that the maximum is close to 1 and yet the practical performance of the smoothers appears quite good. This prompted us to look for a better explanation.

In order to evaluate their effectiveness, it turned out that the (new) accumulated rate based on (old) consecutive smoothing rates is well below 1. That is to say, the above maxima are not achieved at the same location (otherwise LFA is not helpful)! For either smoother, suppose we have completed K (accumulated) inner relaxation steps. Let $\bar{M}_k$ denote the corresponding rate matrix (for $1 \le k \le K$); then define

$$\hat{\mu}_K = \max_{i,j}(\bar{M}_1)_{i,j}(\bar{M}_2)_{i,j}\cdots(\bar{M}_K)_{i,j} \tag{4.34}$$

as the accumulated smoothing rate of a relaxation step (over K iterations). Clearly this is a reasonable definition as it takes care of all iterations within a relaxation step; we expect $\hat{\mu}_K < 1$ and of course $\hat{\mu}_K \ll 1$ if the underlying smoother is good. For linear problems, we have a constant value $\bar{\mu}_{i,j} = \bar{\mu}$ so $\hat{\mu}_K = \bar{\mu}^K$.

Finally for completeness, we shall name the resulting global smoother based on modifying (4.21) by $\bar{G} = G + \epsilon$ as FPGS(ϵ). Likewise we shall denote the corresponding multigrid algorithm based on FPGS(ϵ) as MG(ϵ).

As an example, we present in Table 4.1 the accumulated local smoothing rates computed for the first five iterations for the test problem of Figure 4.3.

Up to outer	$\hat{\mu}_K$ Left bar region				$\hat{\mu}_K$ Right bar region			
iterations ν	FPGS	FPLS	F4	F8	FPGS	FPLS	F4	F8
1	0.5891	0.9620	0.7898	0.7911	0.8354	0.9690	0.8558	0.8568
2	0.5492	0.9448	0.7528	0.7538	0.2567	0.8986	0.7624	0.7651
3	0.5183	0.9333	0.7160	0.7201	0.0420	0.8861	0.7465	0.7506
4	0.4863	0.9327	0.6920	0.6958	0.0037	0.8804	0.6382	0.6337
5	0.3867	0.9320	0.6445	0.6483	0.0016	0.8633	0.5622	0.6216

Table 4.1: Illustration of accumulated smoothing rates for FPGS and FPGS(ϵ) smoothers with *gsiter* $= 10$ used for all tests. Here F4 means FPGS(10^{-4}) and F8 means FPGS(10^{-8}). Note $K =$ *gsiter* ν, where *gsiter* is the number of inner iterations.

From Table 4.1 we can argue that FPGS reduces the high frequency modes much faster than FPLS. Clearly the FPGS(ϵ) method is not as effective. We also tested on other inpaiting problems with various inpainting domains and noticed a similar behavior; therefore FPGS is our preferred smoother. Furthermore, Table 4.1 also suggests that 3 to 5 fixed point iterations are sufficient to get a smoothing rate comparable to GS for the Poisson equation [139].

Problem	Image Size	initial PSNR	final PSNR
Child, Figure 4.5	128 x 128	46	95
Bars, Figure 4.6	512 x 512	50	102
Ring, Figure 4.7	512 x 512	27	82
Lena, Figure 4.4	512 x 512	53	95

Table 4.2: Initial and final PSNR values after applying the modified CDD algorithm to the problems presented in this work.

4.5 Numerical results

In this section, we shall first give results of five different inpainting problems designed to test the performance of the multigrid algorithm, as illustrated in Figures 4.4-4.8. We use the problems in Figures 4.5 and 4.8 (tested in the original CDD paper [123]) to compare the performance of our MG algorithm with the method used in [123] and the MG(ϵ) method.

To measure the restoration quality, we found it useful to use the Peak-Signal-To-Noise-Ratio (PSNR) to check how similar two images u and u^0 of size $m \times n$ are each other. The PSNR is defined as

$$PSNR = 20\, log_{10}\left(\frac{255}{RMSE(u,u^0)}\right), \quad RMSE(u,u^0) = \sqrt{\frac{\sum_{i,j}(u_{i,j} - u^0_{i,j})^2}{mn}}. \qquad (4.35)$$

The larger a PSNR is, the better is the restored image. In real life situation, such a measure is not possible because u^0 is not known.

In Table 6.1 we show quantitative measure of the PSNR values for the damaged and restored images, and in Figures 4.5, 4.6, 4.7 and 4.4 the respective images. Here we only used a few MG cycles to obtain the results. Clearly the restored images are good.

4.5.1 Comparison with existent methods

We now address the efficiency gains. It turns out that our multigrid algorithm is many magnitudes faster, even for inpainting small images. For instance, for the inpainting problem in Figure 4.5 with only a 128×128 image, to reach the same accuracy, time marching requires 160,000 iterations and 7380 CPU seconds to converge, whilst our MG only requires 3 V-cycles and 9.6 CPU seconds. Therefore, it is not necessary to do extensive comparisons.

4.5.2 Full multigrid and MG(ϵ).

We observed that for very large scale inpainting domains such as those of the ring problem illustrated in Figure 4.7, the rate of convergence of our MG algorithm is slightly dependent on the initial guess, even though our MG always converges no matter what

	Image	MG			FMG			MG(10^{-4})			MG(10^{-8})		
Problem	Size	ν	#	CPU	ν	#	CPU	ν	#	CPU	ν	#	CPU
Lena	128	6	2	7	6	1	6	12	2	17	12	2	17
	256	8	3	47	8	1	22	15	3	87	15	4	116
	512	8	4	290	8	2	159	15	5	635	15	5	640
Bars	128	15	4	79	15	2	12	40	5	71	50	5	81
	256	20	6	350	20	3	193	50	7	667	50	8	760
	512	20	6	978	20	4	696	50	7	2871	50	8	3304
Ring	128	15	4	61	15	1	16	30	4	69	40	5	96
	256	15	5	192	15	1	46	50	6	606	50	8	813
	512	15	5	1165	15	1	241	50	7	4151	50	8	4732

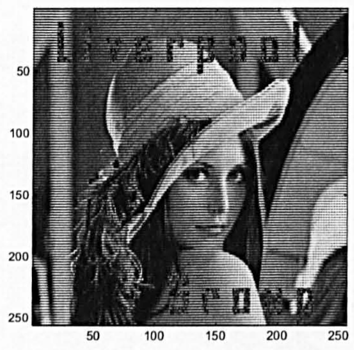
Table 4.3: Further improvements of MG results by the FMG and comparisons with MG(ϵ). Here '#' denotes the number of MG cycles and ν the number of smoothing steps per grid needed.

the initial guess is. In order to reduce this dependence and to improve even more the speed of convergence we adopted the Full Multigrid method (FMG), as described in [139]. Indeed, much better results are obtained and can be seen in Table 4.3, where we also give the results of MG(ϵ). Clearly, MG(ϵ) is less efficient than MG and FMG.

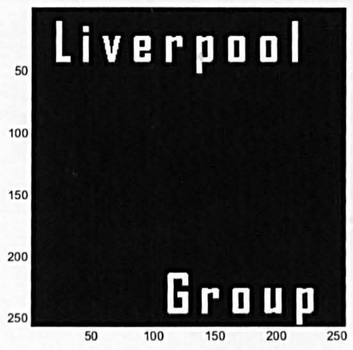
4.6 Conclusions

The original CDD model of [123] improves on the total-variation norm based inpainting model [121, 40]. The only reported time marching scheme is slow in convergence and therefore, it is only suitable to process small-sized images. Moreover, the nonlinear MG cannot be easily used to solve the model.

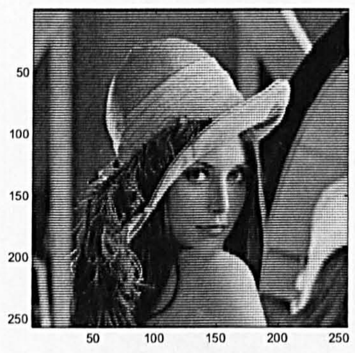
In this chapter we developed a fast and efficient nonlinear MG algorithm for solving a modified CDD model. By first finding out why a fixed-point method is not feasible for the original CDD model, we then proposed a modified CDD model for which a fixed-point method is feasible and developed a nonlinear MG for the modified model. A LFA shows that the global smoother is faster than the local smoother. Numerical results confirmed that the modified model retains the desirable property of the original model in reconnecting the level lines across large distances, and our multigrid method is very efficient.



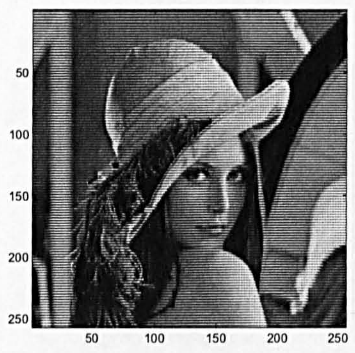
(a)



(b)



(c)



(d)

Figure 4.4: A practical text removal example. Notice the reconnection of the thin piece of hair (2 pixels-width) initially occluded by the thick letter C (4 pixels-width). The CPU-time used by our FMG algorithm was 22 seconds only.

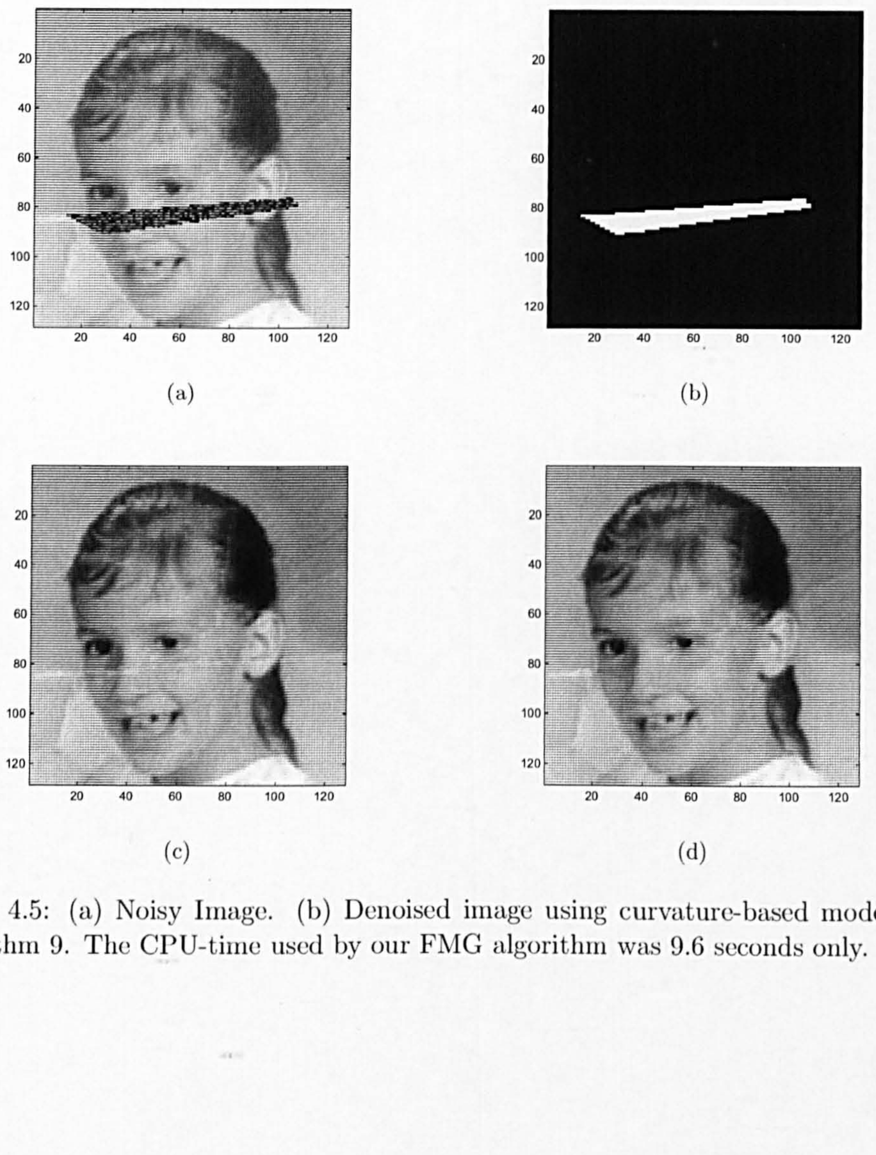
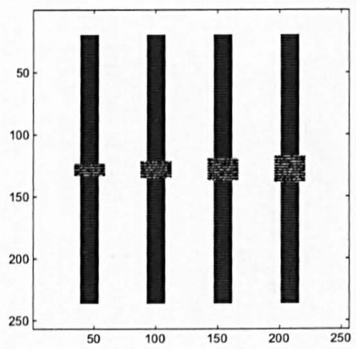
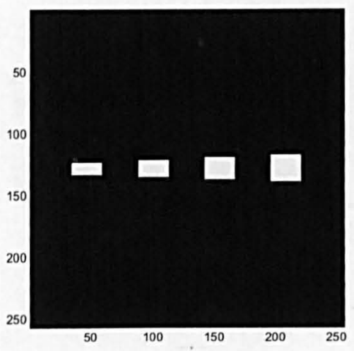


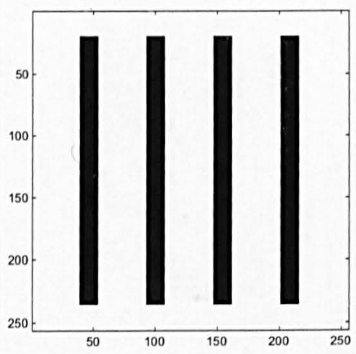
Figure 4.5: (a) Noisy Image. (b) Denoised image using curvature-based model and Algorithm 9. The CPU-time used by our FMG algorithm was 9.6 seconds only.



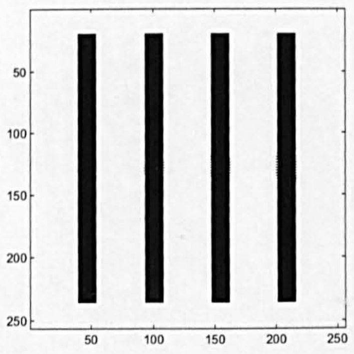
(a)



(b)

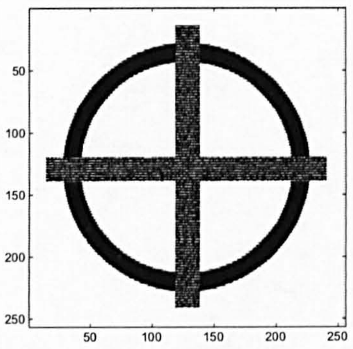


(c)

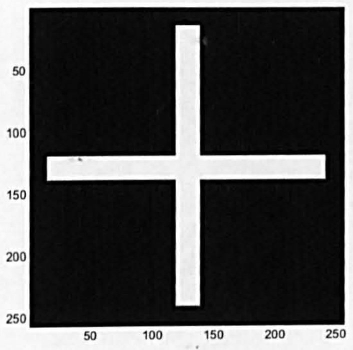


(d)

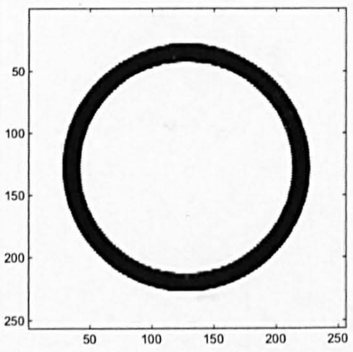
Figure 4.6: (a) Noisy Image. (b) Denoised image using curvature-based model and Algorithm 9. The CPU-time used by our FMG algorithm was 193 seconds.



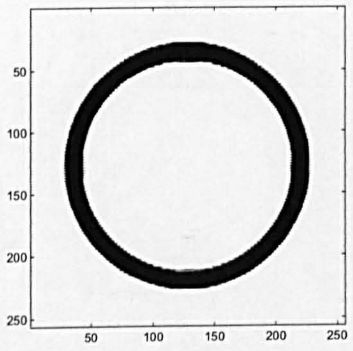
(a)



(b)



(c)



(d)

Figure 4.7: (a) Noisy Image. (b) Denoised image using curvature-based model and Algorithm 9. The CPU-time used by our FMG algorithm was 46 seconds.

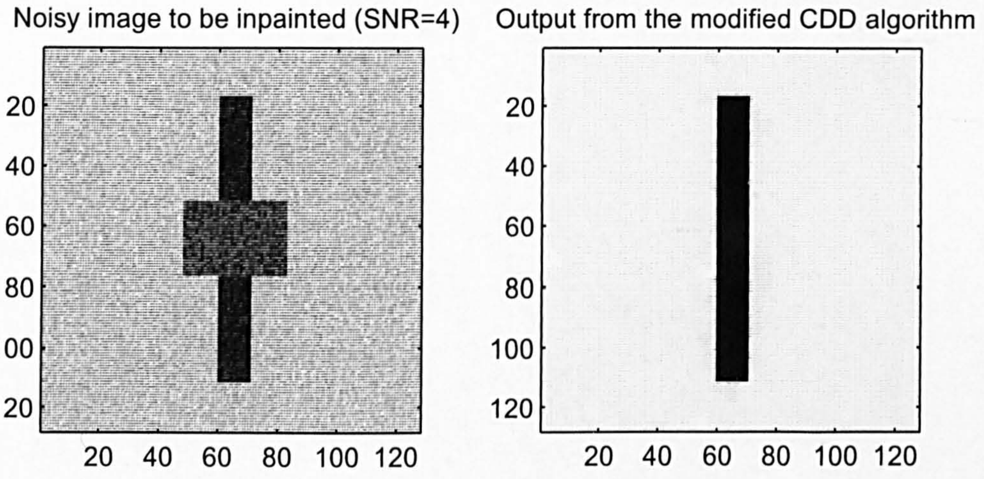


Figure 4.8: Our algorithm performs very well for noisy images. The CPU-time used by our FMG algorithm was 13 seconds only.

Chapter 5

Fast Numerical Algorithms for the Euler's Elastica Digital Inpainting Model

The Euler's elastica digital inpainting model very well-known for its attractive features of reconnecting contours along large distances, reconstructing the curvature of missing parts of objects and its ability to denoise outside the inpainting region if necessary. Since the underlying Euler-Lagrange partial differential equation (PDE) is of fourth-order and highly nonlinear, unfortunately, the usual numerical algorithm to find the solution is a very slow time marching method (due to stability restriction). In this chapter we address this fast solution issue by progressively proposing first, two new unconditionally stable time marching methods and then a novel fixed point method. The latter turns out to be two orders of magnitude faster than the explicit time marching method. Moreover, taking this new fixed point method as a smoother, we develop an even faster nonlinear multigrid method for optimal performance. Numerical results will be presented to illustrate the improved results obtained.

5.1 Introduction

Digital inpainting is a very usefull technique designed to restore the missing or damaged parts of digital images. The very first model of this technique was introduced by Bertalmio *et al.* [10] back in 2000. In the last few years, a number of PDE-based variational inpainting models (similar to [10]) have appeared. These include the TV model [121], the curvature-driven diffusion model [123], the Euler's elastica model [125], the Cahn-Hilliard equation based model [13], the complex Ginzburg-Landau equation based model [68], and the Mumford-Shah and Mumford-Shah-Euler models [54].

Of the above mentioned models, only the two by [125] and [54], respectively, can do the following at the same time: reconnection of level curves along large distances, curvature reconstruction and noise elimination. However, since the associated PDEs of both models are of fourth-order and highly non-linear, it is a non-trivial task to develop fast algorithms to solve them. In this chapter, we show how to develop fast numerical algorithms for the Euler's elastica model [125].

5.2 The Euler's elastica inpainting model

The Euler's elastica model recently introduced by Chan, Kang, and Shen [125] is reviewed here, as it offers us the capability of inpainting delicate fine features by respecting curvature.

For completeness we state here the inpainting formulation previously introduced in early chapters. Assume we are given a possible noisy image $u^0 = u^0(x, y)$ defined on a domain $\Omega \subseteq \mathbb{R}^2$ and there exists one subset $D \subset \Omega$ where the pixel values of u^0 are missing or damaged. The inpainting objective is to reconstruct the values of u^0 in D from the available information on $\Omega \backslash D$, see Fig. 3.5.

The elastica model

Variational models achieve their objective by minimizing some chosen energy functional. In particular, the Euler's elastica model [125] proposes to minimize (3.19). This model allows u^0 to have the Gaussian noise $\eta = \eta(x, y)$ present in $\Omega \backslash D$ such that $u^0 = u + \eta$. The virtue of (3.19) is that it penalizes the integral of the square of the curvature along edges instead of only penalizing the length of edges as the TV model does [40, 24]. Consequently, the model can reconnect contours along large distances and recover the curvature of objects at the same time. As we showed in Chapter 3, the Euler-Lagrange equation for the elastica model is given by the equations (3.20) and (3.21); see also the description at the end of Section (3.3.5) in equations (3.40)-(3.44).

5.3 Numerical solution of the elastica PDE

Whilst the quality of reconstruction of the elastica model is out of discussion, its efficient numerical realization is still an open challenge. In this section, firstly we introduce the numerical discretization of the model (3.20) and (3.21), secondly we review the time marching scheme proposed in [125] for its solution and finally, we move to reviewing two possible, but failed methods: a fixed point and a primal-dual method. It shall be in Section 5.4 where we introduce our new successful numerical algorithms for this model; both delivering faster solutions than the ones of the explicit Euler scheme reported in [125]. Latter in Section 5.6 we shall present an even faster nonlinear multigrid method.

5.3.1 Discretization

We start discussing the numerical approximation of the terms involved in (3.20) or (3.41), following [125]. In (3.41), to approximate $\nabla \cdot \boldsymbol{V} = \frac{\partial}{\partial x}V^1 + \frac{\partial}{\partial y}V^2$ at some pixel

(i,j) we use central differences between ghost half-points; see Fig. 4.1. That is

$$\nabla \cdot V_{i,j} = \frac{\left(V^1_{i+\frac{1}{2},j} - V^1_{i-\frac{1}{2},j}\right)}{h} + \frac{\left(V^2_{i,j+\frac{1}{2}} - V^2_{i,j-\frac{1}{2}}\right)}{h}, \tag{5.1}$$

where h is the spatial step-size that, for convenience, was chosen to be the same in both Cartesian directions. The next step is to approximate all the involved quantities at the half-points. First, to approximate $V^1_{i+\frac{1}{2},j}$ and $V^1_{i-\frac{1}{2},j}$, we do the computations as follows:

Curvature terms. These are approximated by the *min-mod* of two adjacent *whole* pixels.

$\kappa_{i+\frac{1}{2},j} = \textit{min-mod}(\kappa_{i+1,j}, \kappa_{i,j})$ and $\kappa_{i-\frac{1}{2},j} = \textit{min-mod}(\kappa_{i,j}, \kappa_{i-1,j})$,

where $\textit{min-mod}\ (\alpha,\beta) = \left(\frac{sgn\ \alpha + sgn\ \beta}{2}\right)\min(|\alpha|, |\beta|)$.

Partial Derivatives in x. By the central differencing of two adjacent *whole* pixels

$\partial_x(u_{i+\frac{1}{2},j}) = \frac{1}{h}(u_{i+1,j} - u_{i,j})$,

$\partial_x(u_{i-\frac{1}{2},j}) = \frac{1}{h}(u_{i,j} - u_{i-1,j})$,

$\partial_x(\kappa|\nabla u|)_{i+\frac{1}{2},j} = \frac{1}{h}\left(\kappa_{i+1,j}|\nabla u|_{i+1,j} - \kappa_{i,j}|\nabla u|_{i,j}\right)$ and

$\partial_x(\kappa|\nabla u|)_{i-\frac{1}{2},j} = \frac{1}{h}\left(\kappa_{i,j}|\nabla u|_{i,j} - \kappa_{i-1,j}|\nabla u|_{i-1,j}\right)$.

Here $|\nabla u|_{i,j} = \frac{1}{2h}\sqrt{(u_{i+1,j} - u_{i-1,j})^2 + (u_{i,j+1} - u_{i,j-1})^2 + 4h^2\varepsilon}$,

where a small parameter $\varepsilon > 0$ is used to avoid division by zero when $|\nabla u|_{i,j}$ is in the denominator.

Partial Derivatives in y. By the *min-mod* of ∂_y's at two adjacent *whole* points

$\partial_y(u_{i+\frac{1}{2},j}) = \textit{min-mod}\left(\frac{1}{2h}(u_{i+1,j+1} - u_{i+1,j-1}), \frac{1}{2h}(u_{i,j+1} - u_{i,j-1})\right)$,

$\partial_y(u_{i-\frac{1}{2},j}) = \textit{min-mod}\left(\frac{1}{2h}(u_{i,j+1} - u_{i,j-1}), \frac{1}{2h}(u_{i-1,j+1} - u_{i-1,j-1})\right)$,

$\partial_y(\kappa|\nabla u|)_{i+\frac{1}{2},j} = \textit{min-mod}\,(\alpha,\beta)$ with

$\alpha = \frac{1}{2h}\left(\kappa_{i+1,j+1}|\nabla u|_{i+1,j+1} - \kappa_{i+1,j-1}|\nabla u|_{i+1,j-1}\right)$ and

$\beta = \frac{1}{2h}\left(\kappa_{i,j+1}|\nabla u|_{i,j+1} - \kappa_{i,j-1}|\nabla u|_{i,j-1}\right)$.

$\partial_y(\kappa|\nabla u|)_{i-\frac{1}{2},j} = \textit{min-mod}\,(\alpha,\beta)$ with

$\alpha = \frac{1}{2}\left(\kappa_{i,j+1}|\nabla u|_{i,j+1} - \kappa_{i,j-1}|\nabla u|_{i,j-1}\right)$ and

$\beta = \frac{1}{2}\left(\kappa_{i-1,j+1}|\nabla u|_{i-1,j+1} - \kappa_{i-1,j-1}|\nabla u|_{i-1,j-1}\right)$.

Here $|\nabla u|$ is approximated using

$|\nabla u|_{i+\frac{1}{2},j} = \sqrt{(\partial_x(u_{i+\frac{1}{2},j}))^2 + (\partial_y(u_{i,j+\frac{1}{2}}))^2 + \varepsilon}$ and

$|\nabla u|_{i-\frac{1}{2},j} = \sqrt{(\partial_x(u_{i-\frac{1}{2},j}))^2 + (\partial_y(u_{i,j-\frac{1}{2}}))^2 + \varepsilon}$.

Then, by a similar procedure we can obtain the approximations for $V^2_{i,j+\frac{1}{2}}$ and $V^2_{i,j-\frac{1}{2}}$. Finally, the homogeneous Neumann boundary condition on $\partial\Omega$ is treated as in (4.16).

5.3.2 An accelerated time marching method

The first numerical method we study is the accelerated time marching (ATM) algorithm used in [125] to solve (3.20) and (3.21). The idea of an accelerating factor to speed up a time marching scheme was introduced in [95] for solving a nonlinear second order evolution equation.

Firstly equation (3.20) can be solved indirectly by looking for the steady-state solution of a parabolic equation of the form (4.17) and (4.18).

This numerical scheme which is accurate to $O(\Delta t)$ is known to have an stability restriction of $\Delta t \sim O(h^4)$ for fourth-order equations such as (3.20). In fact, for (3.20), due to its high nonlinearity, we have to use much smaller time-steps.

Secondly, the ATM idea, based on [95], is to multiply the right-hand side of (4.17) by $|\nabla u|$ with the purpose of reducing its stiffness [125]:

$$\frac{\partial u}{\partial t} = |\nabla u| r(u), \tag{5.2}$$

where as before an explicit Euler method is then used. Although effectively the dynamics is accelerated with this method, still a huge number of iterations are required to reach convergence. Hence, this method is not appropriate in terms of CPU-time for large images. As an example, for the *circle problem* of Fig. 5.1, the maximum stable time-step we were able to use was $\Delta t = 10^{-5}$ (using $\varepsilon = 10^{-2}$, $\lambda = 100$, $a = 1$, $b = 20$) for which convergence was obtained in 1.4×10^6 iterations and 20.7 hours of CPU-time for an image of size only 32×32 pixels. Here the inpainting domain D has size of 12×12 pixels and it was initialized with random noise.

5.3.3 Failure of traditional fixed point methods

Before introducing our new numerical algorithms, we shall explain roughly what the challenges one faces are when trying to solve numerically (3.20) or (3.41). We have seen already that a time marching method even with acceleration techniques is very slow. Although we have not implemented Newton-type methods, they are also expected to be not useful because of their reduced domain of convergence due to the terms $|\nabla u|$ and $|\nabla u|^3$ in the denominators of $\mathcal{N}$ and $\mathcal{T}$, respectively. This fact is very well explained and analyzed in [43] for a second-order equation of a similar type of singularity. Therefore, a natural idea to consider is a fixed-point-type method, as it was the case with second-order equations [144, 7, 118].

In looking for a fixed point method of the form $A(u)u = f(u)$ for a nonlinear PDE, the standard procedure is to linearize it at some k-step to construct a linear system of equations $A(u^k)u^{k+1} = f(u^k)$ and solve for u^{k+1}. Here $A(u^k)$ usually consist of all linear components and part of the linearized differential operator of such a PDE, while $f(u^k)$ gathers all other nonlinear components. Then, we repeat this procedure as many k-steps as necessary to reach convergence, this is to get $\|u^{k+1} - u^k\| < tol$, for given tolerance tol. If this fixed point procedure is convergent then for some $\bar{k}$ we take $u^{\bar{k}+1}$ as the solution. One necessary condition $A(u^k)$ must satisfy is to be invertible and because of this we always look for a diagonally dominant matrix $A(u^k)$ which guarantees a unique solution u^{k+1}.

For (3.42), we can follow this idea and make $A(u^k)$ weakly diagonally dominant if we collect all the contributions from $\mathcal{T}$ into $f(u^k)$. This suggests to solve

$$-\alpha\nabla\cdot\mathcal{N}^{k+1} + \chi u^{k+1} = \alpha\nabla\cdot\mathcal{T}^k + \chi u^0, \tag{5.3}$$

where $\nabla\cdot\mathcal{N}^{k+1}$ needs to be linearized and $f(u^k) \equiv \alpha\nabla\cdot\mathcal{T}^k + \chi u^0$. Unfortunately, this apparently feasible fixed point algorithm is neither convergent nor a good smoother for the high frequencies of the underlying error, as numerical tests showed.

There are two possible refinements on (5.3): one is to take some part from $\nabla\cdot\mathcal{N}$ into $f(u^k)$ and the other is to take some contribution from $\nabla\cdot\mathcal{T}$ into $A(u^k)$. Our experiments showed that the first option is not helpful, and the second is not much better because diagonal dominance cannot be guaranteed due to $\nabla(\kappa|\nabla u|)$ changing its sign and consequently $A(u^k)$ may not be even weakly diagonally dominant.

In Section 5.4.4 we will show by adding some stabilizing terms to both sides of (5.3) we can construct a fixed point method with weakly diagonally dominant matrix $A(u^k)$ that not only seems to be always convergent to the true solution of (3.20) and (3.21) but also a good smoother for a multigrid method.

5.3.4 A promising primal-dual method

Introducing a new variable into a high-order PDE such as (3.20) with views to reducing its order and solving the resulting lower order system of PDE's is a numerical technique which has been tested in similar to (3.20) anisotropic equations. For instance, the primal-dual Newton's method proposed in [35] for the TV denoising model [114] is maybe the most known successful example in the variational image processing community.

When designing primal-dual methods usually there is no unique way to introduce the new variable. Here we present a primal-dual method for the elastica PDE (3.20) which delivers very good results under suitable conditions, but fails for more general situations. To this end, we start by rewriting the elastica PDE (3.20) and (3.21) (see Chapter 3 for its derivation) in the form

$$-\nabla\cdot\left(\frac{(a+b\kappa^2)\nabla u}{|\nabla u|} + \frac{2b\nabla u\cdot\nabla(\kappa|\nabla u|)}{|\nabla u|^3}\nabla u - \frac{2b\nabla(\kappa|\nabla u|)}{|\nabla u|}\right) + \lambda_E(u-u^0) = 0, \tag{5.4}$$

and then introducing the change of variables $\theta = \kappa|\nabla u|$, which leads to solving the linear system

$$A(u,\lambda)f = b(u) \equiv \begin{pmatrix} \nabla \cdot \frac{\nabla}{|\nabla u|} & -1/|\nabla u| \\ \lambda_E & \nabla \cdot \frac{\nabla}{|\nabla u|} \end{pmatrix} \begin{pmatrix} u \\ \theta \end{pmatrix} = \begin{pmatrix} 0 \\ g(u) \end{pmatrix}, \qquad (5.5)$$

where $g(u) = \nabla \cdot \left(\frac{(a+b\kappa^2)\nabla u}{|\nabla u|} + \frac{2b\nabla u \cdot \nabla(\kappa|\nabla u|)}{|\nabla u|^3} \nabla u \right) + \lambda_E u^0$. To solve (5.5) we simple linearize it by lagging the nonlinear terms $|\nabla u|^{-1}$ in the left hand side of the equation and iterate.

A quick view to the system (5.5) reveals that the matrix $A(u,\lambda)$ may become ill-conditioned depending on the size of D and the value of λ_E; recall λ_E can take values from $[0,+\infty)$. For instance, in regions of $\Omega\backslash D$ with step gradients and $\lambda_E \gg 1$, all terms but λ_E in A will be very small; hence its condition number [73] will be very large.

On the other hand, even if λ_E is suitable selected for a particular problem there is still the need to carefully discretize the term $-1/|\nabla u|$ to guarantee diagonally dominance of $A(u,\lambda)$. Following the discretization process described before we carried out some tests using

$$|\nabla u| = \max\left\{ |\nabla u|_{i+\frac{1}{2},j}, |\nabla u|_{i-\frac{1}{2},j}, |\nabla u|_{i,j+\frac{1}{2}}, |\nabla u|_{i,j-\frac{1}{2}} \right\}.$$

This selection though it worked, it has a tendency to decimate the quality of the curvature of the restored missing regions, one of the main virtues of the elastica model.

Although the primal-dual algorithm just described still has some challenges to overcome, when properly tuned, it delivers a very fast speed of convergence - as an example it takes less than two hundred of iterations to solve the problem of Fig. 5.1 - and therefore it deserves a deeper investigation. For instance, Newton's method for this scheme still needs to be tested and may be reported in a future work.

Finally, we just quickly mention some other change of variables for (3.20) that we tested but did not work whatsoever: defining $\theta = V$ as in (3.40) yields an ill-conditioned system in D, or defining $\theta = \frac{(a+b\kappa^2)}{|\nabla u|}$ did not yield systems of PDE's of reduced order at all.

5.4 New methods for the elastica model

In this section, we shall proceed to present our new and faster numerical schemes for the solution of the elastica PDE (3.20) and (3.21). We start by presenting two unconditionally stable time marching (USTM) schemes. Here unconditionally stable means that there are no stability restrictions for the step-time Δt.

5.4.1 Unconditionally stable time marching methods - foundations

We base our USTM methods on a recently theorem by Eyre [55, 56] who proved the existence of unconditionally stable algorithms for gradient systems. Eyre showed that it is possible to construct such kind of algorithms through what is called a convexity splitting (CS) technique. This technique consists on dividing an energy functional $E(u)$ as the sum of two parts: one strictly convex $E_1(u)$ and one strictly concave $E_2(u)$. In doing so enables one to construct a gradient stable semi-implicit time marching method treating E_1 implicitly and E_2 explicitly. Of course, finding a suitable decomposition of the energy $E(u)$ is the essential part of the CS technique.

We now shall provide the basic theory for the construction of unconditionally stable schemes for gradient systems. To this end, assume a numerical method is needed for the initial value problem

$$\frac{\partial u}{\partial t} = -\nabla E(u), \ \ u(0) = u_0. \tag{5.6}$$

It is assumed [55] that $u : \mathbb{R}^+ \to \mathbb{R}^p$ is class C^1, $E(u) : \mathbb{R}^p \to \mathbb{R}$ is class C^2, $\nabla E(u)$ is the gradient of $E(u)$, and E satisfies the following three conditions:

$$\begin{cases} E(u) > 0 & \forall\, u \in \mathbb{R}^p \\ E(u) \to \infty & \text{as } \|u\| \to \infty \\ \langle H(u)u, u\rangle \geq \mu & \forall\, u \in \mathbb{R}^p \end{cases} . \tag{5.7}$$

Here $\|u\|^2 = \langle u, u\rangle$ is the norm on $\mathbb{R}^p$ defined by the inner product $\langle \cdot, \cdot \rangle$, $H(u)$ is the Hessian matrix of $E(u)$ and $\mu \in \mathbb{R}$.

Equations of the type of (5.6) are called gradient systems, their solutions gradient flows and they satisfy the property

$$\frac{\partial u}{\partial t} = \langle \nabla E(u), u_t \rangle = -\|\nabla E(u)\|^2, \tag{5.8}$$

meaning that $E(u(t)) \leq E(0)$ for all $t > 0$.

In the following, U^n will represent a numerical approximation of $u(n\Delta t)$, where Δt is the time-step. The following definition clarifies what it is understood as an unconditionally gradient stable scheme.

Definition 5.4.1 *A one-step numerical integration scheme (for (5.6) and (5.7)) is said to be unconditionally gradient stable if there exists a function $F(\cdot) : \mathbb{R}^p \to \mathbb{R}$, such that, for all $\Delta t > 0$ and all initial data:*

1. $F(U) \geq 0$ *for all* $U \in \mathbb{R}^p$.
2. $F(U) \to \infty$ *as* $\|U\| \to \infty$.
3. $F(U^{n+1}) \leq F(U^n)$ *for all* $U^n \in \mathbb{R}^p$.
4. *If* $F(U^n) = F(U_0)$ *for all* $n \geq 0$ *then* $U_0 \in \Lambda$, *the set of zeros of* ∇F.

The idea behind convexity splitting (CS), as we already have mentioned, is to split the energy E in two parts as $E = E_1 - E_2$ so the new scheme

$$\frac{\partial u}{\partial t} = -\nabla E_1(u) - \nabla E_2(u), \tag{5.9}$$

is implemented where the energies $E_1(u)$ and $E_2(u)$ are, respectively, strictly convex and strictly concave.

Further, in [55], it is proposed to construct a gradient stable time marching method treating ∇E_1 implicitly and ∇E_2 explicitly

$$U^{k+1} = U^k + \Delta t\,(-\nabla E_1(U^{k+1}) - \nabla E_2(U^k)). \tag{5.10}$$

The advantage of this CS method over the simple explicit Euler method is that Δt in (5.10) does not have to be small due to stability requirements, but its size is only up to accuracy requirements.

The detailed convergence proof of scheme (5.10) can be found in [56].

5.4.2 USTM1

We turn now to develop our first new USTM scheme for the elastica formulation (3.19). This method is not only newer but also faster than the accelerated time marching method [125] for the elastica PDE.

The elastica energy (3.19) can be naturally divided (depending upon the norm) in two parts as $J(u) = J_1(u) + J_2(u)$ with

$$J_1(u) = \int_\Omega (a + b\kappa^2)|\nabla u|\, dxdy \quad \text{and} \quad J_2(u) = \frac{\lambda}{2}\int_{\Omega\setminus D} (u - u^0)^2\, dxdy. \tag{5.11}$$

We note that $J_1(u)$ yields the gradient flow

$$\langle\frac{\partial u}{\partial t}, v\rangle_{TV} = -\langle\nabla J_1(u), v\rangle_{TV} \equiv -\frac{\partial}{\partial t}J_1(u+tv)\mid_{t=0} \quad \text{where}$$
$$-\frac{\partial}{\partial t}J_1(u+tv)\mid_{t=0} = -\nabla\cdot\left(\left(a + b\kappa^2\right)\frac{\nabla u}{|\nabla u|} - \frac{2b}{|\nabla u|^3}\nabla^\perp u \nabla(\kappa|\nabla u|)\nabla^\perp u\right) \tag{5.12}$$

and $\langle\cdot,\cdot\rangle$ denotes the inner product in the total variation space of functions.

Similarly using the L^2 norm in $J_2(u)$ yields the gradient flow

$$\langle\frac{\partial u}{\partial t}, v\rangle_{L^2} = -\langle\nabla J_2(u), v\rangle_{L^2} \equiv -\frac{\partial}{\partial t}J_2(u+tv)\mid_{t=0} \quad \text{where}$$
$$-\frac{\partial}{\partial t}J_2(u+tv)\mid_{t=0} = -\lambda(u - u^0). \tag{5.13}$$

However, $J(u)$ by itself is not strictly a gradient flow under any norm. Nevertheless, as we shall show, we still can apply the CS method successfully. As remarked, for the Cahn-Hilliard inpainting model which is also not strictly a gradient flow, it was found that the CS method was successful [13]. To apply the CS method to our case, we start by splitting

$$J_1(u) = J_{11}(u) - J_{12}(u) \tag{5.14}$$

$$\text{with} \quad J_{11}(u) = a\int_\Omega |\nabla u|\ dxdy + C_1\int_\Omega |\nabla u|\ dxdy \tag{5.15}$$

$$\text{and} \quad J_{12}(u) = -b\int_\Omega \kappa^2|\nabla u|\ dxdy + C_1\int_\Omega |\nabla u|\ dxdy, \tag{5.16}$$

and likewise we split $J_2(u) = J_{21}(u) - J_{22}(u)$ with

$$J_{21}(u) = \frac{C_2}{2}\int_{\Omega\setminus D} |u|^2\ dxdy \quad \text{and} \tag{5.17}$$

$$J_{22}(u) = -\frac{\lambda}{2}\int_{\Omega\setminus D} (u-u^0)^2\ dxdy + \frac{C_2}{2}\int_{\Omega\setminus D} |u|^2\ dxdy. \tag{5.18}$$

Actually for convexity splitting to work on the elastica energy (3.19), there might no need to carry out the last splitting since $J_2(u)$ is already a convex functional, however to construct our USTM1 algorithm, we follow the method described in [13], where the fitting functional was split as in (5.17) and (5.18) and test it for the elastica model. To this end, our USTM1 scheme is to solve

$$u^{k+1} = u^k + \Delta t\Big(-\nabla(J_{11}^{k+1} - J_{12}^k) - \nabla(J_{21}^{k+1} - J_{22}^k)\Big) \tag{5.19}$$

which leads to the numerical scheme

$$u^{k+1} + \Delta t\left(-\alpha\nabla\cdot\left((a+C_1)\frac{\nabla u^{k+1}}{|\nabla u^{k+1}|}\right) + C_2u^{k+1}\right) = u^k +$$
$$\Delta t\left(\alpha\nabla\cdot\left((b\kappa^2 - C_1)\frac{\nabla u^k}{|\nabla u^k|} - \frac{2b}{|\nabla u^k|^3}\nabla^\perp u\nabla(\kappa|\nabla u^k|)\nabla^\perp u^k\right) + \chi(u^0-u^k) + C_2u^k\right). \tag{5.20}$$

In order to ensure that the energies $J_{11}(u)$ and $J_{21}(u)$ are convex for the range of u we are interested in, we need to select appropriate positive constants C_1 and C_2. Experimentally we found that a good choice is $50 \le C_1 \le 200$ and $C_2 \sim \lambda$.

To address the problem of solving (5.20) at each time-step, we opted for constructing a fixed-point-type method by linearizing the term $|\nabla u|^{-1}$ on the left-hand side of (5.20). Therefore, after applying the discretization process described in Section 5.3.1 we get the following linear system of equations

$$\begin{aligned} u_{i,j}^{k+1}\mathbb{S}_{i,j} - u_{i+1,j}^{k+1}\left(C_{i+\frac{1}{2},j}^k\right) - u_{i-1,j}^{k+1}\left(C_{i-\frac{1}{2},j}^k\right) - u_{i,j+1}^{k+1}\left(C_{i,j+\frac{1}{2}}^k\right) \\ -u_{i,j-1}^{k+1}\left(C_{i,j-\frac{1}{2}}^k\right) = f_{i,j}(u^k,\lambda,\Delta t,C_1,C_2) \end{aligned} \tag{5.21}$$

where

$$C_{(i+\frac{1}{2},j)}^k = \frac{\alpha\Delta t\ (a+C_1)}{h^2|\nabla u^k|_{(i+\frac{1}{2},j)}} \tag{5.22}$$

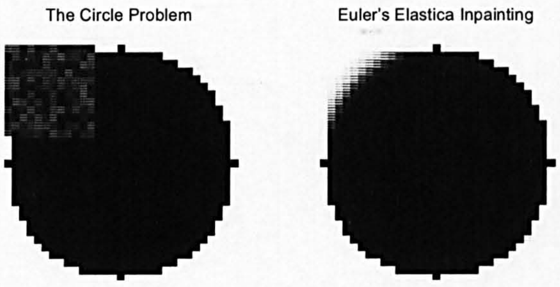


Figure 5.1: The circle problem. Reconstruction obtained by using the FPGS algorithm.

and so on, and

$$\mathbb{S}_{i,j} = (1 + \Delta t\, C_2) + C^k_{i+\frac{1}{2},j} + C^k_{i-\frac{1}{2},j} + C^k_{i,j+\frac{1}{2}} + C^k_{i,j-\frac{1}{2}}. \tag{5.23}$$

Then, such a fixed point method amounts to solving the linear system (5.21), i.e.

$$A(\mathbf{u}^k)\mathbf{u}^{k+1} = f(\mathbf{u}^k), \tag{5.24}$$

where $\mathbf{u}^k = [u^k_{1,1}, u^k_{2,1} \ldots, u^k_{n,1}, u^k_{1,2}, \ldots, u^k_{n,m}]$ and f is defined as the right-hand side of (5.20). In this case, $A(\mathbf{u}^k)$ is a sparse, symmetric and positive definite matrix.

The USTM scheme (5.20) is tested for a model problem, and comparative results with ATM are shown in Table 5.2; it is evident that the USTM scheme is many times faster than the ATM.

5.4.3 USTM2

As we have seen, convexity splitting guarantees unconditionally stability by treating the expanding term of the elastica energy, i.e. $-\nabla J_{12}(u)$ explicitly. Here we shall show that $J_{12}(u)$ can be further partitioned in such a way that one section of that energy can be implicitly treated without losing the unconditionally stable property of the scheme. We also follow the more natural approach of treating $J_2(u)$ implicitly without applying any artificial splitting as in (5.17)-(5.18). Numerical experiments show that the resulting new scheme which we name USTM2 is faster to converge than USTM1, but more importantly there is no need to select appropriate C_2.

Theorem 5.4.2 *If $J_{11}(u)$ and $J_{12}(u)$ are strictly convex functionals given by (5.14)-(5.16) and $J_{12}(u)$ satisfies (5.7) with $\langle H_{J_{12}}(u)u, u\rangle \geq -\mu$ when $\mu < 0$, then for any initial condition, and provided $C_1 > 0$ is sufficiently large, the numerical scheme*

$$u^{k+1} = u^k - \Delta t \left\{ \nabla J_{11}(u^{k+1}) - g_1(u^{k+1})\nabla g_2(u^{k+1}) - g_2(u^k)\nabla g_1(u^k) - \nabla J_{12b}(u^k) \right\}, \tag{5.25}$$

where

$$J_{12}(u) = J_{12_a}(u) + J_{12_b}(u) = \int_\Omega -b\kappa^2|\nabla u|\, dxdy + \int_\Omega C_1|\nabla u|\, dxdy \tag{5.26}$$

and $J_{12_a}(u)$ *further written as*

$$J_{12_a}(u) = \int_\Omega g_1(u)g_2(u)\, dxdy \;\; \text{with} \;\; g_1(u) = -b\kappa^2 \;\; \text{and} \;\; g_2(u) = C_1|\nabla u|, \tag{5.27}$$

is gradient stable for all $\Delta t > 0$.

Proof. Assuming $J_1(u)$ satisfies (5.7) and expanding it about u^{n+1} up to second order using Taylor's theorem, the following inequality holds

$$J_1(u^{k+1}) - J_1(u^k) \le \langle \nabla J_1(u^{k+1}), u^{k+1} - u^k\rangle + |\mu|\|u^{k+1} - u^k\|^2.$$

From here and using (5.14)-(5.16) and (5.25) we have that

$$\begin{aligned}
J_1(u^{k+1}) - J_1(u^k) &\le \langle \nabla J_{11}(u^{k+1}) - \nabla J_{12}(u^{k+1}), u^{k+1} - u^k\rangle + |\mu|\|u^{k+1} - u^k\|^2 \\
&- \langle \frac{1}{\Delta t}(u^{k+1} - u^k) + \nabla J_{11}(u^{k+1}) - g_1(u^{k+1})\nabla g_2(u^{k+1}) \\
&- g_2(u^k)\nabla g_1(u^k) - \nabla J_{12b}(u^k), u^{k+1} - u^k\rangle \\
&= -\langle g_1(u^{k+1})\nabla g_2(u^{k+1}) + g_2(u^{k+1})\nabla g_1(u^{k+1}) \\
&+ \nabla J_{12b}(u^{k+1}), u^{k+1} - u^k\rangle + \langle g_1(u^{k+1})\nabla g_2(u^{k+1}) + g_2(u^k)\nabla g_1(u^k) \\
&+ \nabla J_{12b}(u^k), u^{k+1} - u^k\rangle + \left(|\mu| - \frac{1}{\Delta t}\right)\|u^{k+1} - u^k\|^2 \\
&= -\langle g_2(u^{k+1})\nabla g_1(u^{k+1}) - g_2(u^k)\nabla g_1(u^k), u^{k+1} - u^k\rangle \\
&- \langle \nabla J_{12b}(u^{k+1}) - \nabla J_{12b}(u^k), u^{k+1} - u^k\rangle + \left(|\mu| - \frac{1}{\Delta t}\right)\|u^{k+1} - u^k\|^2.
\end{aligned}$$

Now from the convexity of $J_{12b}(u)$ there exists $\hat{\mu}$ such that

$$\langle \nabla J_{12b}(u^{k+1}) - \nabla J_{12b}(u^k), u^{k+1} - u^k\rangle \ge \hat{\mu}(C_1)\|u^{k+1} - u^k\|^2,$$

where $\hat{\mu}(C_1)$ indicates that $\hat{\mu}$ depends on C_1. Hence, we have that

$$\begin{aligned}
J_1(u^{k+1}) - J_1(u^k) &\le -\langle g_1(u^{k+1})\nabla g_2(u^{k+1}) - g_2(u^k)\nabla g_1(u^k), u^{k+1} - u^k\rangle \\
&+ \left(|\mu| - |\hat{\mu}(C_1)| - \frac{1}{\Delta t}\right)\|u^{k+1} - u^k\|^2 \le 0,
\end{aligned}$$

where the last inequality is satisfied provided C_1 is chosen large enough. □

Using Theorem 5.4.2 in the elastica energy (3.19) we get the following scheme which we name it as USTM2

$$u^{k+1} - \Delta t \left(\alpha \nabla \cdot \left((a + C_1 + b\kappa^2) \frac{\nabla u^{k+1}}{|\nabla u^{k+1}|} \right) + \chi u^{k+1} \right)$$
$$= u^k + \Delta t \left(\alpha \nabla \cdot \left(-C_1 \frac{\nabla u^k}{|\nabla u^k|} - \frac{2b}{|\nabla u^k|^3} \nabla^\perp u \nabla (\kappa |\nabla u^k|) \nabla^\perp u^k \right) + \chi u^0 \right). \qquad (5.28)$$

Although USTM2 is just slightly faster than USTM1, as can be seen from Table 5.2, in the USTM2 scheme there is no need to select C_2 which, if wrongly selected, may cause the USTM1 algorithm to diverge.

5.4.4 A working fixed point method

The main virtue of the USTM schemes is that the value of Δt is not an issue for stability matters. For similar, but different imaging problems see [146, 67, 129], however, Δt needs to be bounded to obtain a given accuracy at every iteration. For the elastica formulation (3.19) the aim is to find its minimum, therefore the value of Δt is less relevant for this purpose. This observation prompts us to construct a fixed point method from (5.28) by taking an infinity time-step Δt (i.e. steady state solution). This means to solve

$$-\alpha \nabla \cdot \left((a + C_1 + b\kappa^2) \frac{\nabla u^{k+1}}{|\nabla u^{k+1}|} \right) + \chi u^{k+1}$$
$$= \alpha \nabla \cdot \left(-C_1 \frac{\nabla u^k}{|\nabla u^k|} - \frac{2b}{|\nabla u^k|^3} \nabla^\perp u \nabla (\kappa |\nabla u^k|) \nabla^\perp u^k \right) + \chi u^0. \qquad (5.29)$$

At the present time a rigorous mathematical proof of the convergence of this fixed point scheme is not at hand although we expect this to be true since (5.29) is likely to inherit this property from the USTM2 scheme. Nonetheless, numerical experiments over a wide range of problems have always shown (5.29) to be a convergent and fast algorithm provided C_1 is rightly selected.

To obtain some numerical insight on the performance of this fixed point method, in Section 5.4.5 we shall follow the approach of using local Fourier analysis to study how fast does (5.29) eliminates the components of the error. In particular, we are mainly interested in the high-frequency interval $\Theta^{high} = [-\pi, \pi]^2 \setminus [-\pi/2, \pi/2]^2$ since our aim is to use this method as a smoother in a multigrid algorithm framework.

To solve (5.29), as with (5.21) for (5.20), we linearize the left-hand side and obtain the following linear system of equations

$$\begin{aligned} u_{i,j}^{k+1} \mathbb{S}_{i,j} - u_{i+1,j}^{k+1} \left(C_{i+\frac{1}{2},j}^k \right) - u_{i-1,j}^{k+1} \left(C_{i-\frac{1}{2},j}^k \right) - u_{i,j+1}^{k+1} \left(C_{i,j+\frac{1}{2}}^k \right) \\ - u_{i,j-1}^{k+1} \left(C_{i,j-\frac{1}{2}}^k \right) = f_{i,j}(u^k, \lambda, C_1) \end{aligned} \qquad (5.30)$$

where

$$C^k_{(i+\frac{1}{2},j)} = \frac{\alpha(a + C_1 + b\kappa^2)}{h^2|\nabla u^k|_{(i+\frac{1}{2},j)}} \tag{5.31}$$

and so on, and $\mathbb{S}_{i,j}$ given by (4.28).

Denote the resulting system of equations (5.30) by $A(u^k)u^{k+1} = f(u^k)$, with f defined as the right-hand side of (5.29), where the matrix A is semi-positive definite and weakly diagonally dominant. Further, we apply some Gauss-Seidel (GS) iterations and we call this fixed point method based on GS the FPGS algorithm.

A remark is due here. By re-writing (5.29), with $\mathcal{TV}(u) \equiv \nabla \cdot \frac{\nabla u}{|\nabla u|}$, as

$$-\alpha\left(\nabla \cdot \mathcal{N}^{k+1} + C_1\mathcal{TV}(u^{k+1})\right) + \chi u^{k+1} = \alpha\left(\nabla \cdot \mathcal{T}^k + C_1\mathcal{TV}(u^k)\right) + \chi u^0, \tag{5.32}$$

we can see that basically what we really did in (5.32) was to add stabilizing terms to both sides of our first failed fixed point method (5.3). One explanation may be that the domain of convergence of (5.3) is very small and therefore, it fails to converge for a bad initial guess. On the other hand, in (5.29) or (5.32), the $\mathcal{TV}$ term drives to convergence the algorithm when $\|u^{k+1} - u^k\|$ is large (at the beginning of iterations), however once $\|u^{k+1} - u^k\|$ starts decreasing then the $\mathcal{TV}$ terms on both sides of (5.32) tend to cancel each other and gradually (5.3) takes over.

5.4.5 Local Fourier analysis (LFA) for the fixed point method

For nonlinear problems such as the elastica PDE (3.20) and (3.21), evaluating how much a given algorithm reduces the high frequencies of the error is not an easy task. A very useful tool for this purpose is the LFA, see [139]. Here we use LFA to show that the linearized fixed point algorithm (5.30) certainly reduces the high frequency components with a good rate and therefore, is a suitable smoother for a MG method. Note that although (5.30) was numerically tested to be a good standalone method for (5.37), to this point we do not know its behavior in the high frequencies regime which only LFA can reveal.

For simplicity, we consider the case of a square image of size $m \times m$. Let the local error functions $e^{k+1}_{i,j}$ and $e^k_{i,j}$ be defined as $e^{k+1}_{i,j} = u_{i,j} - u^{k+1}_{i,j}$ and $e^k_{i,j} = u_{i,j} - u^k_{i,j}$. The LFA involves expanding

$$e^{(k+1)}_{i,j} = \sum_{\theta_1,\theta_2=-m/2}^{m/2} \psi^{k+1}_{\theta_1,\theta_2} B_{\theta_1,\theta_2}(x_i, y_j), \qquad e^{(k)}_{i,j} = \sum_{\theta_1,\theta_2=-m/2}^{m/2} \psi^{k}_{\theta_1,\theta_2} B_{\theta_1,\theta_2}(x_i, y_j) \tag{5.33}$$

in Fourier components $B_{\theta_1,\theta_2}(x_i, y_j)$ defined as

$$B_{\theta_1,\theta_2}(x_i, y_j) = \exp\left(\mathrm{i}\alpha_1\frac{x_i}{h} + \mathrm{i}\alpha_2\frac{y_j}{h}\right) = \exp\left(\frac{2\mathrm{i}\theta_1 x_i\pi}{m} + \frac{2\mathrm{i}\theta_2 x_j\pi}{m}\right) \tag{5.34}$$

with $\alpha_1 = 2\theta_1\pi/m$, $\alpha_2 = 2\theta_2\pi/m \in [-\pi, \pi]$. From (5.30)-(5.31) we obtain that

$$\begin{aligned}-(\chi + C_{i+\frac{1}{2},j} + C_{i-\frac{1}{2},j} + C_{i,j+\frac{1}{2}} + C_{i,j-\frac{1}{2}})e^{k+1}_{i,j} + C_{i+\frac{1}{2},j}\, e^{k}_{i+1,j} + C_{i-\frac{1}{2},j}\, e^{k+1}_{i-1,j} \\ + C_{i,j+\frac{1}{2}}\, e^{k}_{i,j+1} + C_{i,j-\frac{1}{2},j}\, e^{k+1}_{i,j-1} = 0.\end{aligned} \tag{5.35}$$

Therefore, the local amplification factor $\mu_{i,j} = \left|\psi^{k+1}_{\theta_1,\theta_2}/\psi^{k}_{\theta_1,\theta_2}\right| = \mu_{i,j}(\alpha_1, \alpha_2)$ is defined by

$$\mu_{i,j}(\alpha_1,\alpha_2) = \frac{\left|C_{i+\frac{1}{2},j}\, e^{\mathrm{i}\alpha_1} + C_{i,j+\frac{1}{2}}\, e^{\mathrm{i}\alpha_2}\right|}{\left|\chi + C_{i+\frac{1}{2},j} + C_{i-\frac{1}{2},j} + C_{i,j+\frac{1}{2}} + C_{i,j-\frac{1}{2}} - C_{i-\frac{1}{2},j}\, e^{-\mathrm{i}\alpha_1} - C_{i,j-\frac{1}{2}}\, e^{-\mathrm{i}\alpha_2}\right|}. \tag{5.36}$$

Then we compute the nonlinear coefficients $C_{(\cdot,\cdot)}$ at each (i, j) point and find the maximum for the kth step $\bar{\mu}_{i,j} = \max_{\alpha_1,\alpha_2} \mu_{i,j}(\alpha_1, \alpha_2)$ in the high frequency interval $(\alpha_1, \alpha_2) \in \Theta^{high}$.

Since our FPGS is linearized at each step, its smoothing rates will change at every outer iteration. Then we have an $m \times m$ rate matrix $\bar{M}_k$, for the kth step, with entry $\bar{\mu}_{i,j}$ representing the local smoothing rates at (i, j) point. As done in [24], in order to evaluate its effectiveness, we evaluate the accumulated rate based on consecutive smoothing rates $\bar{\mu}_{i,j}$. That is to say, suppose we have completed K (accumulated) inner relaxation steps. Let $\bar{M}_k$ denote the corresponding rate matrix (for $1 \le k \le K$); then define $\hat{\mu}_K$ given by (4.34), as the accumulated smoothing rate of a relaxation step (over K iterations). As an example, we present in Table 5.1 the smoothing rates $\bar{\mu}$ and accumulated smoothing rates $\hat{\mu}_K$ inside the inpainting domain D and outside it in $\Omega\backslash D$, computed for the first five iterations for the circle problem of Figure 5.1 with 20% of additive Gaussian noise.

	D		$\Omega\backslash D$	
Outer iterations ν	$\bar{\mu}$	$\hat{\mu}_K$	$\bar{\mu}$	$\hat{\mu}_K$
1	0.6267	0.6267	0.8850	0.8850
2	0.7049	0.4417	0.8882	0.7861
3	0.7559	0.3339	0.8680	0.6823
4	0.8220	0.2745	0.8582	0.5856
5	0.9115	0.2502	0.8445	0.4945

Table 5.1: Illustration of smoothing rates for the FPGS smoother. (Note $K = gsiter\ \nu$, where $gsiter$ is the number of inner iterations). Here $a = 1$, $b = 20$ and $gsiter = 10$ were used.

Clearly, the accumulated rates for FPGS are quite good for $K \le 5$. In practice for other problems, however, a bit more than 5 outer iterations may be needed to guarantee a fast and convergent MG algorithm.

Method	# iterations	CPU	PSNR
ATM	1.4×10^6	74,578	55
USTM1	6,500	120	60
USTM2	6,000	112	60
FPGS	3,500	73	60

Table 5.2: Performance of the two algorithms for the circle problem of Fig. 5.1 with $m = n = 32$. The following parameters were used for each one of them $a = 1$, $b = 20$, $\beta = 10^{-2}$, $\lambda = 100$. For ATM, $\Delta t = 10^{-5}$ and for USTM, $\Delta t = 1$. Finally, 10 GS steps were used in FPGS.

5.5 Comparison of unilevel methods

In Table 5.2 we present a performance summary of the three working algorithms we have presented so far. Here we use the Peak-Signal-to-Noise-Ratio (PSNR) measure as defined in (4.35) between images u and u^t of size $m \times n$ for a model problem where the true image u^t is known and u is the restored image. The larger a PSNR is, the better is the restored image. Clearly, even for this small image of 32×32, the FPGS algorithm is many times faster than ATM. However, it still can take time to process large images. As an optimal algorithm, the multigrid method has proved to be successful in solving a number of different image processing problems, see [118, 7, 24], therefore, we now proceed to develop such a multigrid method for the Euler's elastica PDE.

5.6 A nonlinear multigrid for the Euler's elastica model

When constructing a convergent MG method for a nonlinear problem the key task is to find a good smoother and this is by no means trivial. Fortunately, as we already have shown in Section 5.4.5 our fixed point FPGS is a good smoother. To proceed, we shall denote the elastica formulation (3.20) i.e.

$$\begin{aligned}(Nu)_{i,j} =& -\alpha\nabla \cdot \left(\left(a + b\kappa_{i,j}^2\right) \frac{\nabla u_{i,j}}{|\nabla u|_{i,j}} - \frac{2b}{|\nabla u|_{i,j}^3} \nabla^{\perp} u_{i,j} \nabla(\kappa_{i,j} |\nabla u|_{i,j}) \nabla^{\perp} u_{i,j} \right) \\ &+ \chi u_{i,j} = \chi u_{i,j}^0, \qquad (5.37)\end{aligned}$$

by $Nu = \chi u^0$ (as a nonlinear operator equation); see also (3.41).

5.6.1 The MG algorithm

To approximate equation (5.37) on grids of different sizes we will denote by $N_h u_h = \chi u_h^0$ the discrete equation defined on the finest grid Ω_h of size h and similarly by $N_{2h} u_{2h} = \chi u_{2h}^0$ the same on the coarser grid Ω_{2h} which is obtained by standard coarsening, i.e., the nonlinear operator N_{2h} which results from defining equation (5.37) on the cell-centered grid Ω_{2h} with grid spacing $2h$. Likewise we can generate a sequence of L

coarse levels $2h, 4h, 8h, \dots, 2^L h$.

Next we briefly mention the standard intergrid transfer operators. Denote by R_h^{2h} (restriction) and I_{2h}^h (interpolation), respectively, two transfer operators between Ω_h and Ω_{2h} which on cell-centered grids are defined by the following equations [139]: The Restriction operator is defined by $R_h^{2h} u_h = u_{2h}$ as in (2.84) and Interpolation operator $I_{2h}^h u_{2h} = u_h$ is defined as in (2.86).

To coarsen the interfaces, which is unique for inpainting problems, we use the same method as described in [24]. Briefly, we represent the inpainting domain by a binary mask M_h and coarsen this mask similarly to grid coarsening.

Multigrid schemes are designed to obtain fast solutions from numerical discretizations of PDE's similar to (3.20). In particular, when the PDE to solve is nonlinear as here, the full approximation scheme (FAS) [139] is highly efficient. This FAS method which we have adapted for the inpainting case and is described in Algorithm 14, performs by constructing a hierarchy of discretizations where at each level the error equation is partially solved or smoothed (step 2) and the new approximation transported to next coarser level (step 3). This process is recursively applied until reaching the coarsest level where an exact, but computationally cheap solution is obtained (step 1). Then the process move backwards on the hierarchical structure transporting the more accurate error (step 7) and updating the approximate solution at each level (step 8) then taking this new approximate solution as initial guess and smoothing again (step 9) repeating the process until reaching the finest level again. Usually standard coarsening is used to construct the hierarchical structure halving the number of variables on each dimension at each level.

Now we state our V-cycling nonlinear MG in Algorithm 13, meaning that just one recursive call to the algorithm is made on each level to approximately solve a coarse grid problem.

Algorithm 13 [Nonlinear Multigrid Method]

```
1: Select an initial guess $u_h$ on the finest grid $h$
2: Set $k = 0$ and $err = tol + 1$
3: while $err < tol$ do
4:     $u_h^{k+1} \leftarrow FAS(u_h^k, N_h^k, u_h^0, M_h, \nu_0, \nu_1, \nu_2, gsiter)$
5:     $err = \|u_h^{k+1} - u_h^k\|_2, \quad k = k + 1$
6: end while
```

Here *gsiter* represents the number of inner Gauss-Seidel iterations at each pre or post-smoothing FPGS step.

Finally, we remark that when implementing the MG method, the parameter λ and the balance at coarse levels between $\mathcal{N}$ and $\mathcal{T}$ defined by the parameters a and b in (5.37) need to be kept the same.

Algorithm 14 [FAS] $u_h \leftarrow FAS(h, u_h, N_h, u_h^0, M_h, \nu_0, \nu_1, \nu_2, gsiter)$

```
1: If $\Omega^h$ = coarsest grid, solve $N_h u_h = \chi u_h^0$ accurately (i.e. $\nu_0$ iterations by FPGS) and return    Else continue with step 2.
2: Pre-smoothing: Do $\nu_1$ steps of $u_h \leftarrow FPGS(u_h, u_h^0, gsiter, M_h)$
3: Restrict to the coarse grid, $M_{2h} \leftarrow R_h^{2h} M_h$ and $u_{2h} \leftarrow R_h^{2h} u_h$
4: Set the initial solution for the next level, $\bar{u}_{2h} \leftarrow u_{2h}$
5: Compute the new right hand side $\chi z_{2h} \leftarrow R_h^{2h}(\chi u_h^0 - N_h u_h) + N_{2h} u_{2h}$
6: Solve $N_{2h} u_{2h} = \chi u_{2h}^0$ by implementing
   $u_{2h} \leftarrow FAS_{2h}(2h, u_{2h}, N_{2h}, u_{2h}^0, M_{2h}, \nu_0, \nu_1, \nu_2, gsiter)$
7: Compute the error $e_{2h} = u_{2h} - \bar{u}_{2h}$ and move it back to the next grid by $e_h \leftarrow I_{2h}^h e_{2h}$
8: Add the residual correction, $u_h \leftarrow u_h + e_{2h}$
9: Post-smoothing: Do $\nu_2$ steps of $u_h \leftarrow FPGS(u_h, u_h^0, gsiter, M_h)$
```

5.6.2 The global smoother

As stated before, our fixed point smoother is simply the FPGS algorithm (Section 5.4.4), that is, Gauss-Seidel iterations to the linearised system (5.30). It is stated in Algorithm 15.

Algorithm 15 [FPGS Smoother] $u_h \leftarrow FPGS(u_h, u_h^0, gsiter, M_h)$

```
1: Choose an initial guess $u_h^0$ for (5.30)
2: for $k = 1$ to $gsiter$ do
3:     Apply $gsiter$ Gauss Seidel iterations to the linear system $A_h(u_h^k)u_h^{k+1} = u_h^0$
4: end for
```

5.7 Numerical results

In this section, we test the performance of our MG algorithm on four different problems. The problems and the MG restored results are shown in Figures 5.2, 5.3, 5.5 and 5.6 for $m = 256$.

5.7.1 Quality of reconstruction

Clearly, the restoration obtained with MG is visually pleasing in all of them. For instance, Figure 5.2 shows a very good reconstruction of the missing edges in the ear, nose and cheek of the girl. In Figure 5.3 we remark the reconnection of the thin piece of hair initially occluded by the letter "G" and in general the fair recovering of the geometrical structure of the missing regions. Figures 5.5 and 5.6 show the smooth reconstruction of curvy missing regions and, in particular, the latter illustrates the virtue of the Euler's elastica model in denoising and inpainting at the same the time; a feature only shared with the TV model.

For completeness, in Table 5.3 we present the PSNR values obtained from the restored images. In the first two problems, the PSNR values are high indicating a very good reconstruction. In the last two, they are not that high due to texture not recovered in the first case (grapes problem) and noise present in the second one (noisy circles problem).

5.7.2 Convergence comparison

We have already shown in Table 5.2 how slow the ATM can be (even for a small image), taking not thousands, but millions of iterations to converge. This kind of slow convergence is definitively not suitable for large images and barely enough for small ones. A quick review of Table 5.2 also reveals that our FPGS, as a standalone method, is two orders of magnitude faster than ATM for the circle problem and this relationship was confirmed through other experiments and different problems we tested. On the other hand, we present the results obtained with MG in Table 5.3, from which it can be seen that our MG can be used to obtain very fast inpaintings for large images.

It is difficult and maybe not fair to compare our MG algorithm for the elastica model against the numerical methods of the Masnou-Morel's and the BBCSV models. For instance, for the latter only a time evolution scheme for the system of PDE's has been reported. Also our MG method is designed to work over the whole Ω domain taking advantage of the denoising capabilities of the elastica model. Masnou-Morel's and BBCSV models on the other hand, only work over the inpainting domain D. In some situations like D being of small size and no noise present on the image maybe a better and faster solution is obtained by using our FPGS method instead of the MG scheme.

There is also a recently proposed method [131] to carry out a fast minimization of the elastica energy using using exemplar-based inpainting techniques with similar ideas proposed in [28]. These two methods however are not robust when the image is noisy.

5.7.3 Full multigrid

In order to provide an automatic and yet suitable initial guess, we adopted the Full Multigrid method (FMG) as described in [139]. FMG is based on the idea of nested iteration, this is, given the coarse grid Ω_h, one can apply a multigrid cycle (say a V-cycle) to obtain an approximate solution u_h at this h-level, then this u_h is interpolated to the next finer grid Ω_{h+1} to be used as initial guess for another multigrid cycle at the $h+1$-level. This process is carried out until reaching the finest level.

Notice that in FMG is the solution u_h and not the error e_h which is interpolated to the next finer level. Usually the operator used to interpolate the solution is denoted by Π_{h-1}^{h} and is of higher accuracy (cubic order in our Algorithm 16) than the interpolation operators used within the multigrid iteration.

Here u_h^{FMG} denotes the resulting FMG approximation on grid Ω_h. As expected, much better (faster) results are obtained as can be seen from the last two columns of Table 5.3 from testing all four inpainting problems.

Algorithm 16 Full Multigrid

```
For $h = 0$,
Solve $N_0 u_0 = u_0^0$, providing $u_0^{FMG} = u_0$.
for $h = 1$ to $\ell$ do
    $u_h^0 \leftarrow \Pi_{h-1}^h u_{h-1}^{FMG}$
    $u_h^{FMG} \leftarrow FAS(h, u_h, N_h, u_h^0, M_h, \nu_0, \nu_1, \nu_2, gsiter)$.
end for
```

		MG		FMG		
Problem	Image Size	MG cycles	CPU	MG cycles	CPU	PSNR
Child	128x128	10	171	2	27	91
	256x256	7	400	2	130	90
	512x512	6	1538	2	613	90
Lena	128x128	7	59	2	23	95
	256x256	6	268	2	103	94
	512x512	5	998	2	502	96
Grapes (*)	128x128	7	55	2	22	70
	256x256	5	219	2	101	70
	512x512	5	972	2	440	71
Noisy Circles	128x128	6	125	2	57	70
	256x256	4	356	2	193	71
	512x512	4	1514	2	851	70

Table 5.3: MG results and further improvements by the FMG. (*) results from inpainting the first (red) channel of the color image.

5.7.4 Color images

Our MG algorithm may be adapted to work with a new energy functional that involves all channels of a colour image. In [14], a new and generalized definition of the TV norm was introduced to the multidimensional case and applied successfully to denoising vector-valued images. This is, for any function $u(\mathbf{x}) = (u^1, \dots, u^m) : \mathbb{R}^n \to \mathbb{R}^m$ the multi-dimensional TV norm is given by

$$TV_{n,m}(u) = \sqrt{\sum_{i=1}^{m} [TV_{n,1}(u^i)]^2}, \tag{5.38}$$

which is invariant for functions with monotone components, and fixed end points [14] and reduces to the well-known one-dimensional TV norm

$$TV_{2,1}(u) = \int_\Omega |\nabla u| \, dxdy, \quad \Omega \subset \mathbb{R}^2. \tag{5.39}$$

The above $TV_{n,m}$ norm enabled the authors of [14] to couple the 3 channels of a colour image and obtain better results than the Channel-by-Channel (CbC) approach.

To extend this idea to the elastica inpainting, we may consider a coupling between the colour channels by minimizing the functional

$$\min_u \left\{ \sqrt{\sum_{i=1}^{m} \left(\int_\Omega (a + b|\kappa^i|^p) |\nabla u^i| \, dxdy \right)^2} + \frac{\lambda}{2} \sum_{i=1}^{m} \int_{\Omega \backslash D} (u^i - (u^0)^i)^2 \, dxdy \right\}. \quad (5.40)$$

We are planning to test this idea in the near future. Other possible approaches for colour images are described in [140, 150].

In Figure 5.5 we give an example of inpainting a color image by the Euler's elastica model, using the simple CbC approach. That is, each of the *RGB* channels was inpainted separately (i.e. solved by our MG algorithm) and then combined to form the inpainted colour image as displayed. Clearly, the result is encouraging.

5.8 Conclusions

The Euler's elastica inpainting model has two very desirable features: reconnecting far apart parts of broken objects and recovering the curvature of the missing parts of objects. So far, the lack of a fast numerical algorithm for this model represented a strong limitation for the range of its applications and wider use. In this chapter we first introduced two faster numerical algorithms (USTM and FPGS) than the existing accelerated time marching method (ATM). Then adopting our FPGS as a smoother, we were able to develop a fast and efficient nonlinear MG algorithm for solving this Euler's elastica model. A local Fourier analysis was done to demonstrate the effectiveness of the FPGS smoother. Numerical results confirmed that our multigrid method is very efficient.

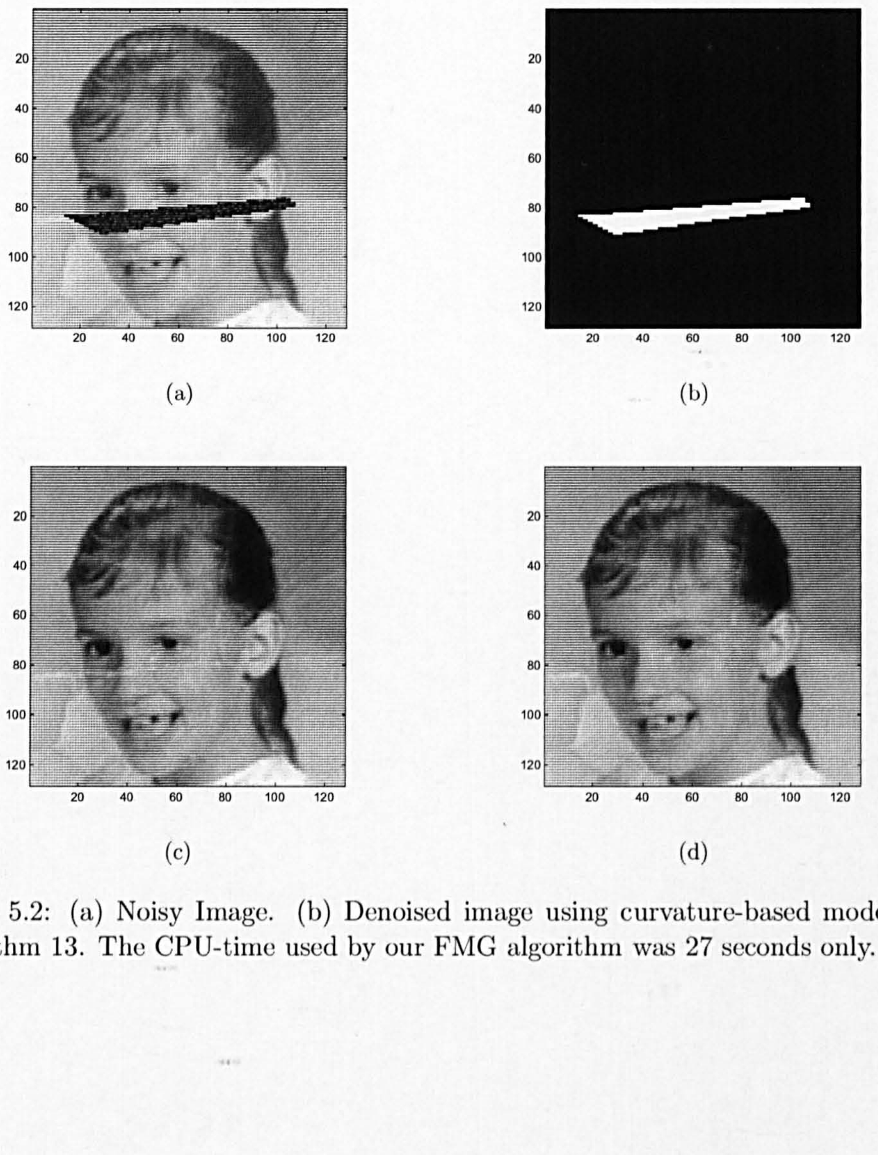
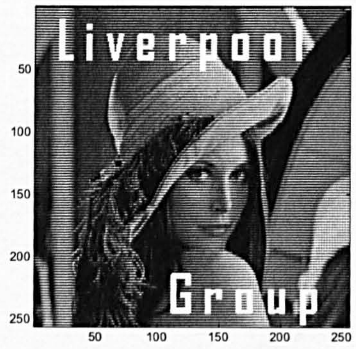
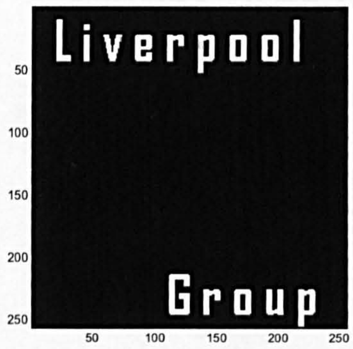


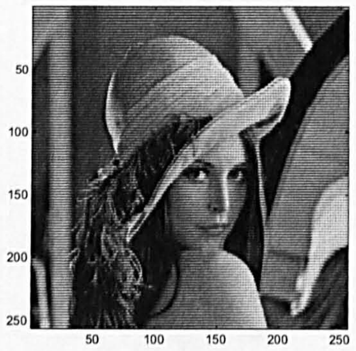
Figure 5.2: (a) Noisy Image. (b) Denoised image using curvature-based model and Algorithm 13. The CPU-time used by our FMG algorithm was 27 seconds only.



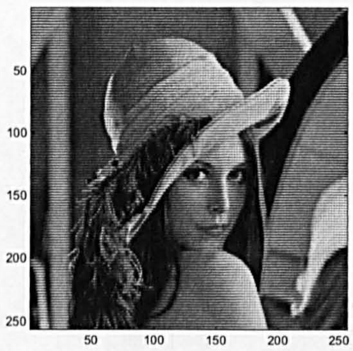
(a)



(b)

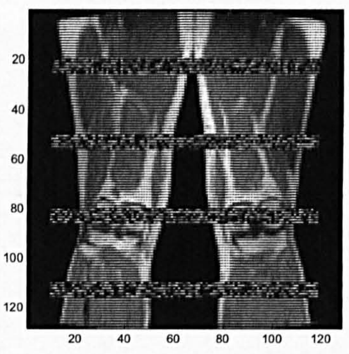


(c)

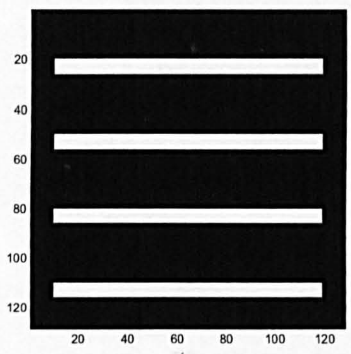


(d)

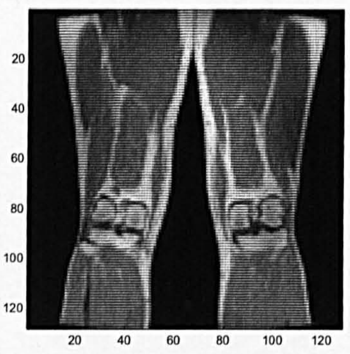
Figure 5.3: (a) Noisy Image. (b) Denoised image using curvature-based model and Algorithm 13. The CPU-time used by our FMG algorithm was 103 seconds only.



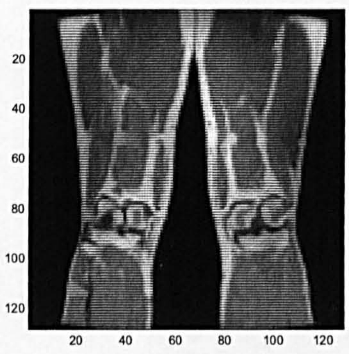
(a)



(b)



(c)



(d)

Figure 5.4: (a) Noisy Image. (b) Denoised image using curvature-based model and Algorithm 13. The CPU-time used by our FMG algorithm was 29 seconds only.

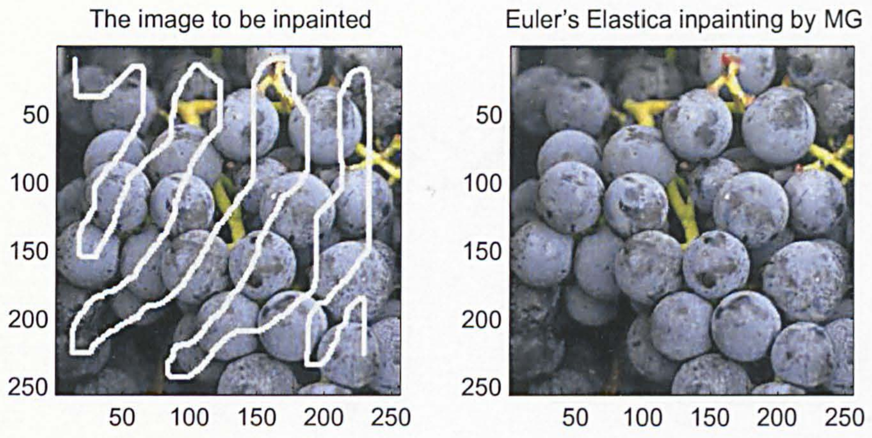


Figure 5.5: This example illustrates the recovering of the rounded parts of the objects (grapes). Also due to the fast processing obtained by using MG, a color image can be inpainted in suitable CPU-time. In this example, our FMG algorithm took 101 seconds for each channel to get the image inpainted.

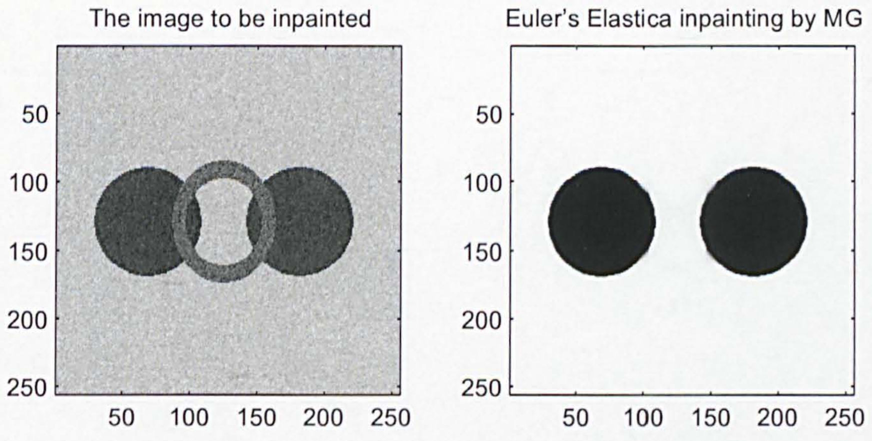


Figure 5.6: Example of the very good performance of MG processing noisy images. The CPU-time used by our FMG algorithm was 193 seconds.

Chapter 6

Multigrid Method for High-Order Denoising

Image denoising has been a research topic deeply investigated within the last two decades. Excellent results are obtained from using models like the Total Variation (TV) minimization by Rudin *et al.* [114] which involves solving a second order partial differential equation (PDE). In more recent years some effort has been made [157, 92, 91, 46] in improving these results by using higher order models and in particular to avoid the staircase effect inherent to the solution of the TV model. However, the construction of stable numerical schemes for the resulting PDE's arising from the minimization of such high order models has proved to be very difficult due to high nonlinearity and stiffness. In this chapter, we study a curvature-based energy minimizing model [159] for which one has to solve a fourth order PDE. The material presented here has been submitted for publication in [26].

For this curvature-based model we develop two new algorithms: a stabilized fixed point (SFP) method and, based upon this, an efficient nonlinear multigrid (MG) algorithm. We will show numerical experiments to demonstrate the very good performance of our MG algorithm.

6.1 Introduction

Image denoising is a basic, but very important image processing task that has been extensively investigated for many years. Although there exist different types of noise, here we study only algorithms to remove additive, zero-mean Gaussian noise. This can be modeled as

$$u^0(x,y) = u(x,y) + \eta(x,y), \tag{6.1}$$

where $u^0 = u^0(x,y)$ is the known noisy image, $u = u(x,y)$ is the unknown true image and $\eta = \eta(x,y)$ is the unknown additive noise all of which defined on a domain $\Omega \subseteq \mathbb{R}^2$. The task of removing noise can be accomplished by traditional ways such as linear filters which, though very simple to implement, may cause the restored image to be

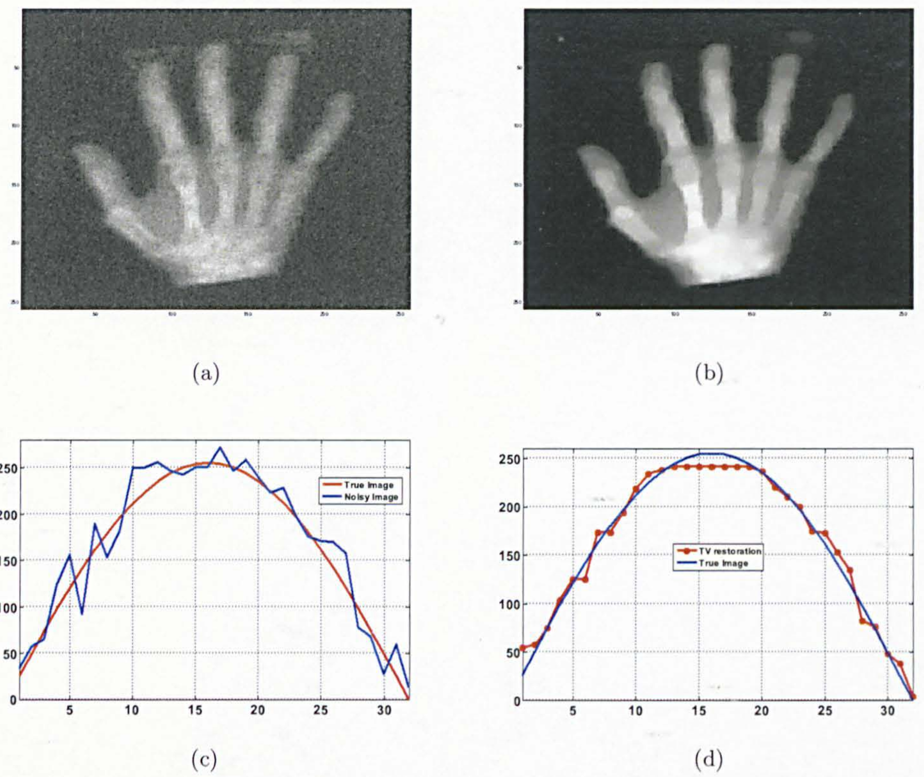


Figure 6.1: The staircase effect in the TV model transforms smooth functions into piecewise constant functions this causes restores to look blocky. (a) Noisy Image. (b) Restored image using TV model. (c) Smooth function and its noisy version. (d) After TV restoration the recovered function (in red) is now piecewise constant.

blurred at edges. A much better technique is to use nonlinear PDE's as anisotropic diffusion filters because they apply different strength of diffusivity to different locations in the image.

Usually anisotropic filters are implemented as second-order PDE's, see for instance [114]. The main drawback of these models is that they convert smooth functions into piecewise constant functions in a phenomenon known as *staircase effect* which causes images to look blocky; see Figure 6.1 for two examples.

Although some effort has been made (see for instance [95, 118, 46] and the references therein) to numerically reduce the staircase effect in second order models, some researchers have turned to higher-order models trying to avoid this problem. In this direction are for instance the works presented in [157, 92, 91, 159]. However, very few papers have touched on fast solvers for these high-order models and this work aims to partially fill in this gap.

The outline of this chapter is as follows. In Section 6.2 we introduce the model to solve. In Section 6.4 we review the existent numerical methods to solve this model. In Section 6.5 the numerical discretization of the resulting Euler-Lagrange (EL) equation is described. We use Section 6.6 to introduce our two new algorithms to solve this EL equation: a stabilized fixed point method and a nonlinear MG method. We introduce some early work on using MG algorithms for similar problems and explain the main difficulties to overcome. We also use this section to present a detailed LFA of the linearized problem to have an insight of the performance of the nonlinear MG algorithm. A complexity analysis is presented as well. In Section 6.7, we present numerical evidence to show the very good and fast performance of the MG algorithm and explain how it is affected by variations on the different parameters of the model and the numerical equation. Finally, in Section 6.8 we discuss how our algorithms can be adapted to solve similar high order problems and present our conclusions in Section 6.9.

6.2 High-order denoising

Additive noise in images is seen as random high frequency oscillations. Therefore, the key for its removal in energy based minimization techniques represented as

$$\min_{u \in BV(\Omega)} \left\{ \alpha R(u) + \int_\Omega |u - u^0|^2 \, dxdy \right\}, \tag{6.2}$$

is to select a regularization term $R(u)$ capable of efficiently measuring these oscillations. Different high order approaches for the regularization term $R(u)$ have been proposed. For example in [157, 92, 91, 36] all use second-order information (i.e. second-order derivatives) so it is expected them to be able to model noise better than those using only first-order information like, for instance, the TV model [114].

6.3 The curvature-based denoising model

In this paper, we study the model [159] resulting from selecting $R(u) = \int_\Omega \Phi(\kappa) \, dxdy$ with κ the curvature of the image and Φ defined either as $\Phi(\kappa) = |\kappa|$, $\Phi(\kappa) = \kappa^2$ or as in [159] as a combination of both.

To minimize (6.2) one could be tempted to use optimization algorithms such as Newton's method. However, there is a problem with this approach. After computing the first order condition, the resulting algebraic system of equations is highly nonlinear and has a reduced domain of convergence. This causes Newton's method to fail since it requires a very good initial guess to guarantee convergence. This is not surprising since a similar problem was reported in [145, 43, 37] when solving a similar formulation (TV denoising model). Multilevel optimization methods [33] on the other hand still need to be tested for high order problems.

Our approach instead, is to minimize (6.2) by solving its EL equation. This method has proved to deliver quality restoration results in a wide variety of image processing applications, see [40, 6, 144] for references. The EL equation we aim to solve is

$$\alpha\nabla\cdot\left(\frac{\nabla\Phi'(\kappa)}{|\nabla u|_\beta}-\frac{\nabla u\cdot\nabla\Phi'(\kappa)}{(|\nabla u|_\beta)^3}\nabla u\right)+u-u^0=0 \text{ in } \Omega \tag{6.3}$$

with homogeneous Neumann boundary condition $\nabla u\cdot\vec{\nu}=0$ on $\partial\Omega$. Note that we already have applied regularization to avoid division by zero by replacing $|\nabla u|$ with $|\nabla u|_\beta=\sqrt{|\nabla u|^2+\beta}$. For simplicity from now on we will write the derivative of $\Phi(\kappa)$ just as Φ' instead of $\Phi'(\kappa)$.

A remark is in order here, in [159] an image is understood as a surface represented by $(x,y,u(x,y))$ where initially $u(x,y)=u^0$. Then, the curvature term κ that appears in (6.3) is the curvature of the image surface given by

$$\kappa\equiv\kappa_S=\nabla\cdot\frac{\nabla u}{\sqrt{|\nabla u|^2+1}}. \tag{6.4}$$

One can adopt the more common understanding of an image as a collection of level sets and still obtain the same PDE, but this time with

$$\kappa\equiv\kappa_{LS}=\nabla\cdot\frac{\nabla u}{|\nabla u|} \tag{6.5}$$

standing for the curvature of the image at every level line or isophote, see [40, 6]. Notice that when κ_{LS} is regularized using a β-parameter as above, then $\kappa_{LS}=\nabla\cdot\frac{\nabla u}{|\nabla u|_\beta}$ equals κ_S for $\beta=1$.

The selection of β is actually of huge importance since for $\beta\ll 1$ the anisotropy of (6.3) is increased making this model less suitable for MG algorithms. Regardless of this fact, the MG algorithm we develop here is in this sense very general allowing to select a small value for β or $\beta=1$ and still obtaining a very good performance.

An interesting discussion about the *correct* value of β for a similar second order problem can be found in [5]. There, the authors showed that there is a range of values of β for which the model delivers a good reconstruction and these values are not necessarily extremely small.

6.4 Review of numerical methods

First, we remark that to solve (6.3) only an explicit scheme has been proposed [159]. Second, before introducing our numerical algorithm, we start by briefly explaining the main difficulty to solve (6.3) and exploring possible options based upon ideas developed in [95, 92, 143]. To this end, first re-write the EL equation as

$$\alpha\nabla\cdot\big(D_1(u)\nabla\Phi'-D_2(u)\nabla u\big)+u-u^0=0, \tag{6.6}$$

where

$$D_1(u) = \frac{1}{|\nabla u|_\beta} \quad \text{and} \quad D_2(u) = \frac{\nabla u \cdot \nabla \Phi'}{(|\nabla u|_\beta)^3} \tag{6.7}$$

are diffusion coefficients whose numerical values mainly depend upon the values of their respective denominators. Due to noise present in u and the edges of u itself, $D_1(u)$ and $D_2(u)$ are usually discontinuous coefficients on Ω causing (6.6) to be highly anisotropic.

We can find in the literature similar PDE's having only $D_1(u)$-type coefficients (TV denoising PDE for example [114]) and they have proved to be the main difficulty to implement fast and stable numerical algorithms, see [95, 145, 119] and references therein. For (6.3) things are even more difficult since it contains $D_2(u)$-type coefficients as well. For example, fixing $\beta = 10^{-2}$ in a plain region of the image yields $D_1 \sim O(10^2)$ compared to $D_2 \sim O(10^6)$. Hence depending upon the smoothness of the image and the level of noise this phenomenon can produce a very unbalanced discrete operator.

Solving (6.3) with an explicit method as in [159] has the drawback that the time-step needs to be selected very small for stability reasons. To implement this method, first (6.3) is transformed into the parabolic form

$$\frac{\partial u}{\partial t} = \alpha \nabla \cdot V(u) + u - u^0 \equiv r(u), \tag{6.8}$$

with $V = (D_1(u)\nabla\Phi' - D_2(u)\nabla u)$ and initial condition $u(x, y, 0) = u^0(x, y)$. Then this new PDE is evolved in time until reaching steady-state. An easy to implement, but very slow to converge, explicit Euler method is used yielding

$$u_{i,j}^{k+1} = u_{i,j}^k + \Delta t \, r(u)_{i,j}^k, \quad \text{with } k = 0, 1, \ldots \text{ and } \Delta t \text{ the time-step.} \tag{6.9}$$

One way to accelerate the convergence of the explicit method could be by multiplying $r(u)$ by $|\nabla u|$. This results in the scheme

$$\frac{\partial u}{\partial t} = |\nabla u|_{i,j}^k r(u)_{i,j}^k, \tag{6.10}$$

where again an explicit Euler scheme can be used for the time derivative. This idea was applied to similar PDE's in [95, 125] with some success. Applying this idea to solve (6.3) gave us however, very poor acceleration of the algorithm.

In summary, the above two explicit methods have the inconvenience of having to obey a very restrictive numerical condition on the time step. Usually $\Delta t \sim O((\Delta x)^4)$ for fourth order PDE's like (6.3) implying that both schemes are practically of no use for processing large images.

A third option could be to find a suitable change of variables obtaining an easier to solve system of second order equations. This was done in [92, 143] for harmonics maps letting the authors to solve the problem indirectly. Unfortunately, this does not seem to be straight forward here. A last option based upon convexity splitting ideas [55, 56] will be studied in Section 6.6.3.

6.5 Numerical implementation

We proceed to outline the discretization scheme we use. From now on, we assume[1] that the continuous domain $\Omega = [0,m] \times [0,n]$. By letting (h_x, h_y) to represent a vector of finite mesh sizes, we define the infinite grid G_h as

$$G_h = \{(x,y) : x = x_i = ih_x, \quad y = y_j = jh_y; \quad i,j \in \mathbb{Z}\}. \tag{6.11}$$

For simplicity, assume $m = n$ and $h = h_x = h_y$. We define the discrete grid as $\Omega_h = \Omega \cap G_h$ and $u_h = u_h(x,y) = u_h(x_i,y_j) = u_h(ih_x, jh_y)$ the discrete version of any function u defined on Ω_h. Here u and u^0 take on scaled values in the interval $[0,1]$. We also denote the derivative with respect to any variable ψ as $(\cdot)_\psi$.

For any $V = (V^1, V^2)$, to approximate $\nabla \cdot V = (V^1)_x + (V^2)_y$ at some pixel (i,j) we use central differences between ghost half-points. This is,

$$\nabla \cdot V_{i,j} = \frac{\left(V^1_{i+\frac{1}{2},j} - V^1_{i-\frac{1}{2},j}\right)}{h} + \frac{\left(V^2_{i,j+\frac{1}{2}} - V^2_{i,j-\frac{1}{2}}\right)}{h}, \tag{6.12}$$

where $h \times h$ is the size of one cell on the cell-centered grid we use. When appropriate we use *min-mod* derivatives since as noted in [114, 40] they help to recover sharp edges. The *min-mod* derivative is defined as

$$\textit{min-mod}\ (a,b) = \left(\frac{sgn\ a + sgn\ b}{2}\right) \min(|a|, |b|). \tag{6.13}$$

To approximate all the involved quantities at the half-points we proceed as follows. First consider $V^1_{i+\frac{1}{2},j}$ and $V^1_{i-\frac{1}{2},j}$:

Curvature by $\kappa_{i,j} = \frac{(u_x)_{i+\frac{1}{2},j}}{|\nabla u|_{i+\frac{1}{2},j}} - \frac{(u_x)_{i-\frac{1}{2},j}}{|\nabla u|_{i-\frac{1}{2},j}} + \frac{(u_y)_{i,j+\frac{1}{2}}}{|\nabla u|_{i,j+\frac{1}{2}}} - \frac{(u_y)_{i,j-\frac{1}{2}}}{|\nabla u|_{i,j-\frac{1}{2}}}$.

Partial Derivatives in x by the central differencing of two adjacent *whole* pixels

$$(u_x)_{i+\frac{1}{2},j} = (u_{i+1,j} - u_{i,j})/h, \quad (u_x)_{i-\frac{1}{2},j} = (u_{i,j} - u_{i-1,j})/h,$$
$$(\Phi'_x)_{i+\frac{1}{2},j} = \left(\Phi'_{i+1,j} - \Phi'_{i,j}\right)/h, \quad (\Phi'_x)_{i-\frac{1}{2},j} = \left(\Phi'_{i,j} - \Phi'_{i-1,j}\right)/h,$$
$$|\nabla u|_{i+\frac{1}{2},j} = \sqrt{((u_x)_{i+\frac{1}{2},j})^2 + ((u_y)_{i,j+\frac{1}{2}})^2 + \beta}.$$

Partial Derivatives in y by the *min-mod* of $(\cdot)_y$'s at two adjacent *whole* points

[1]The choice of $\Omega = [0,m] \times [0,n]$ ensures that $h = 1$ on the finest grid, without loss of generality.

$$(u_y)_{i+\frac{1}{2},j} = \textit{min-mod}\left(\tfrac{1}{2h}(u_{i+1,j+1} - u_{i+1,j-1}), \tfrac{1}{2h}(u_{i,j+1} - u_{i,j-1})\right),$$
$$(u_y)_{i-\frac{1}{2},j} = \textit{min-mod}\left(\tfrac{1}{2h}(u_{i,j+1} - u_{i,j-1}), \tfrac{1}{2h}(u_{i-1,j+1} - u_{i-1,j-1})\right),$$

$$(\Phi'_y)_{i+\frac{1}{2},j} = \textit{min-mod}\,(\zeta,\vartheta) \text{ with}$$
$$\zeta = \tfrac{1}{2h}\left(\Phi'_{i+1,j+1} - \Phi'_{i+1,j-1}\right) \text{ and } \vartheta = \tfrac{1}{2h}\left(\Phi'_{i,j+1} - \Phi'_{i,j-1}\right).$$

$$(\Phi'_y)_{i-\frac{1}{2},j} = \textit{min-mod}\,(\zeta,\vartheta) \text{ with}$$
$$\zeta = \tfrac{1}{2}\left(\Phi'_{i,j+1} - \Phi'_{i,j-1}\right) \text{ and } \vartheta = \tfrac{1}{2}\left(\Phi'_{i-1,j+1} - \Phi'_{i-1,j-1}\right).$$

$$|\nabla u|_{i-\frac{1}{2},j} = \sqrt{((u_x)_{i-\frac{1}{2},j})^2 + ((u_y)_{i,j-\frac{1}{2}})^2 + \beta}.$$

Then by a similar procedure we can obtain the approximations for $V^2_{i,j+\frac{1}{2}}$ and $V^2_{i,j-\frac{1}{2}}$. Finally, the homogeneous Neumann boundary condition on $\partial\Omega$ is treated as in (4.16).

6.6 A nonlinear MG for the fourth-order denoising model

Up to our knowledge no multigrid method has been reported. for the solution of a similar high-order denoising problem which presents at the same time the challenges to deal with nonlinearity, anisotropy and high-order derivatives. This situation is not surprising since the application of either standard linear or nonlinear MG with the known components do not converge. Below, in Section 6.6.3 we shall introduce a new non-standard smoother for a FAS multigrid algorithm.

6.6.1 Early works and numerical difficulties

First, we review some early works of MG algorithms mainly on the context of image processing techniques. This will help us to illustrate the main difficulties to implement optimal MG algorithms for these problems.

We start by reviewing works on *isotropic* nonlinear problems. Here, the functionals to minimize, mostly have been regularized using Tikhonov's idea with $\int_\Omega |\nabla u|^2$ and therefore, their correspondent EL equations include Laplacian-like differential operators with the nonlinearity usually coming from the fitting term. It is well-known that these strongly elliptic operators are suitable for MG algorithms and very good performance can be obtained without very much effort. The approach used in these cases is usually to linearize the problem and apply linear multigrid for the inner iterations. In this fashion are the works presented in [74, 133, 49, 85]. In [132], however, a nonlinear MG was reported.

In the context of *anisotropic* problems some work has been done as well. For example, the following developed MG algortihms for image processing problems: [75]

on image registration, [7, 104] on image segmentation, and [119, 118, 34, 59] on image denoising. All of the former, however, solve second-order PDE's.

In particular, previous image denoising works are of our interest since they give us a glimpse of the difficulties to develop optimal multigrids for this kind of problems. All, [119, 118, 34, 59], reported difficulties to obtain an optimal performance of geometric MG algorithms when the anisotropy of the problem associated with the TV regularizer $\int_\Omega |\nabla u|$ reached high levels. The anisotropy on denoising problems, is mainly due to the value of the regularization parameter β, the level of noise η and the smoothness of u itself, meaning the more strong edges present in u are the more anisotropic the PDE that need to be solved is.

In [59], to overcome this problem β was selected initially very large and a continuation method with $\beta \to 0$ was implemented. At every step of this continuation method a MG cycle was used to solve the problem. In [119, 118, 34], an small increment on the number of smoothing steps ν of the MG algorithm was applied and β no less than 10^{-2} was used.

For the denoising model we study here, a first issue is the level of discontinuity of the coefficients $D_1(u)$ and $D_2(u)$ in (6.3). This is, if they are moderately discontinuous then $N(u)$ can be fairly approximated on coarser grids using standard coarsening. However, for strongly discontinuous coefficients (say $\beta \ll 1$) the performance of the MG algorithm will be strongly affected. Notice that in image denoising problems, since every image is different and because of noise, we do not have *a priori* information about the location of discontinuities so we do not know what variables (if any) are strongly coupled and in what direction. Hence, standard methods as line relaxation and semi-coarsening are not useful here.

To overcome this challenge, some authors have proposed to use algebraic MG methods where the coarsening is adapted to the structure of the domain itself. Here, because the domain of every image is different, this type of coarsening has to be adaptive making it computationally very expensive, see [45] and references there in. In [153] a geometric MG algorithm with adaptive coarsening for the anisotropic Cahn-Hilliard equation was developed. In [153], the authors used the simple rule of coarsening only where discontinuities were not present. We believe this technique can be adapted to MG algorithms for image denoising by coarsening only at plain regions of the image. Although this idea looks promising it needs to be tested since it may not be efficient for images with many edges or high levels of noise. In this chapter, we adopted the geometrical MG scheme with standard coarsening due to its simplicity and focus on non-standard smoothers.

A second issue for this model is the transportation of the error from one grid to another. Geometrical MG algorithms assume that the error is smooth enough so it can be well approximated onto the next coarser grid by using a simple transportation operator. Designing good smoothers for anisotropic equations is however a very difficult task. We tackle this problem in Section 6.6.3 by introducing a new fixed point algorithm which performs very well in homogeneous regions of the image domain reducing the high frequency components of the error and can guarantee its smoothness (up to some degree) close to edges (inhomogeneous regions).

Solutions techniques for MG algorithms and interesting discussions related to linear anisotropic problems in a general context have been presented in [58, 42, 1, 112, 17].

Finally, MG's for high-order problems other than those for the bi-harmonic equation [139] are difficult to find, even more if the problem is anisotropic or nonlinear. High-order brings numerical difficulties one needs to be aware at. For example, in [141] an algebraic MG was proposed for anisotropic second and fourth-order equations. In [81] however, it is argued that discretizations of high-order problems might not satisfy the M-matrix condition for multigrid convergence [139]. Hence it is suggested to use instead geometric MG algorithms for these problems. In [81], the authors also reported to have observed that an inaccurate approximation of high-order derivatives at coarse levels can produce so poor representations of positive definite operators that they are no longer positive definite on coarser grids causing the MG algorithm to fail to converge.

As it can be seen from the above discussion, there are many difficulties when developing MG algorithms for nonlinear high-order anisotropic problems as it is the curvature-based denoising model given by equation (6.3). One of capital importance, as we mentioned above, is to guarantee the smoothness of the error. As we shall show standard smoothers do not work for (6.3), hence we focus our efforts in developing a good smoother for this problem and test this smoother into the standard framework of a nonlinear MG algorithm.

6.6.2 The MG algorithm

In this section, we introduce a nonlinear MG algorithm for the fast solution of the high order denoising formulation (6.3). To this end, we denote the nonlinear operator equation by

$$(Nu)_{i,j} \equiv \alpha\nabla \cdot \big(D_1(u)_{i,j}(\nabla\Phi')_{i,j} - D_2(u)_{i,j}(\nabla u)_{i,j}\big) + u_{i,j} = u^0_{i,j}, \qquad (6.14)$$

and construct a hierarchy of discretizations by approximating the operator $(Nu)_{i,j}$ at different grid levels. As it is common practice, we denote by $N_h u_h = u^0_h$ the discrete equation defined on the finest grid Ω_h of size h and similarly by $N_{2h}u_{2h} = u^0_{2h}$ the same on the coarser grid Ω_{2h} which is obtained by standard coarsening. We can carry on applying this process until generating a sequence of L coarse levels $2h, 4h, 8h, \dots, 2^L h$. We state in Algorithm 17 our MG method

Here FAS denotes the cycle of going through all fine grids (smoothing the iterates and passing on the residual information to next grid) to the coarsest grid, solving the equation on the coarsest grid accurately and coming through all coarse grids (interpolating to next grid and smoothing the iterates again) back to the finest grid, as shown below [20, 139, 44].

Algorithm 17 Nonlinear Multigrid Method

```
Require: Select an initial guess $u_h$ on the finest grid $h$
1: $k \leftarrow 0$
2: $err \leftarrow tol + 1$
3: while $err < tol$ do
4:     $u_h^{k+1} \leftarrow FAS(u_h^k, N_h^k, u_h^0, \nu_0, \nu_1, \nu_2, gsiter, \alpha, \gamma)$
5:     $err = \|E(u_h^k) - E(u_h^{k-1})\|_2$,
6:     $k \leftarrow k + 1$
7: end while
```

Algorithm 18 FAS Cycle $u_h \leftarrow FAS(u_h, N_h, u_h^0, \nu_0, \nu_1, \nu_2, gsiter, \alpha, \gamma)$

```
1: if $\Omega_h$ = coarsest grid then
2:     solve $N_h u_h = u_h^0$ accurately (i.e. $\nu_0$ iterations by SFPGS) and return.
3: else
4:     continue with step 6.
5: end if
6: Pre-smoothing: Do $\nu_1$ steps of,   $u_h \leftarrow SFPGS(u_h, u_h^0, gsiter, \alpha, \gamma, \nu_1)$
7: Restrict to the coarse grid, $u_{2h} \leftarrow R_h^{2h} u_h$
8: Set the initial solution for the next level, $\bar{u}_{2h} \leftarrow u_{2h}$
9: Compute the new right hand side $u_{2h}^0 \leftarrow R_h^{2h}(u_h^0 - N_h u_h) + N_{2h} u_{2h}$
10: Implement $u_{2h} \leftarrow FAS_{2h}(u_{2h}, N_{2h}, u_{2h}^0, \nu_0, \nu_1, \nu_2, gsiter, \alpha, \gamma)$
11: Add the residual correction, $u_h \leftarrow u_h + I_{2h}^h(u_{2h} - \bar{u}_{2h})$
12: Post-smoothing: Do $\nu_2$ steps of   $u_h \leftarrow SFPGS(u_h, u_h^0, gsiter, \alpha, \gamma, \nu_2)$
```

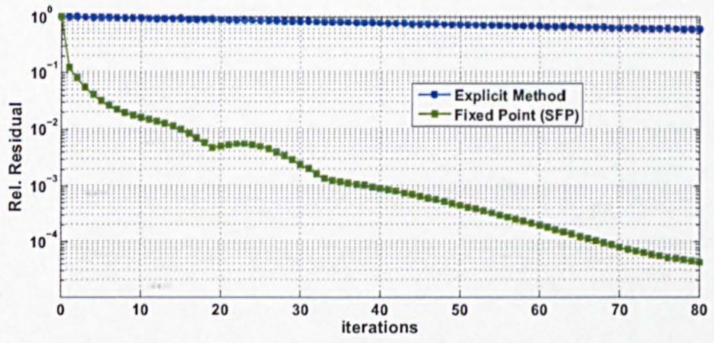


Figure 6.2: Comparison of speed of convergence between our stabilized fixed point (SFP) algorithm against the explicit time marching method. Clearly SFP is by far much faster.

6.6.3 The smoother - A new stabilized fixed point (SFP) method

Fixed point methods are usually fast algorithms, hence the wish to develop one for the curvature-based model. Unfortunately standard ways to implement such a kind of algorithm for (6.6) simply do not work. For instance, following [145, 144, 118] a possible scheme would be

$$-\alpha\nabla\cdot\left((D_2(u))_{i,j}^{k+1}(\nabla u)_{i,j}^{k+1}\right)+u_{i,j}^{k+1}=-\alpha\nabla\cdot\left((D_1(u))_{i,j}^{k}(\nabla\Phi')_{i,j}^{k}\right)+u_{i,j}^{0}. \qquad (6.15)$$

This scheme is however, neither stable nor convergent, the reason being that D_2 can easily change its sign so neither positive-definiteness nor diagonally dominance can be guaranteed for schemes of the form $A(u^k)u^{k+1}=f(u^k,u^0)$. In [23], the same sort of problem was observed in trying to develop a fixed point method for a fourth order PDE with similar structure to (6.6) giving already an indication that standard fixed point methods for this type of equations do not converge!

In this section, we discuss how to develop a working fixed point algorithm for (6.6) that is not only much faster than the explicit time-marching methods reviewed in Section 6.4, but it has also shown to be always convergent as a stand-alone algorithm in all the simulations we have carried out as well. Our aim however, goes further and it is oriented to develop and even faster multigrid algorithm; with this in mind we will not use our fixed point algorithm as standalone method, but as smoother for such a nonlinear multigrid algorithm.

To develop a working fixed point algorithm for (6.6), we will first analyze unconditionally stable time marching schemes based on convexity-splitting ideas developed in [55, 56]. The resulting semi-implicit scheme (6.16) improves the stability of time marching schemes since this is guaranteed for all possible time steps. In simple words, we solve

$$\frac{u_{i,j}^{k+1}-u_{i,j}^{k}}{\Delta t}=r(u)_{i,j}^{k}+\gamma\mathcal{N}_{i,j}^{k}-\gamma\mathcal{N}_{i,j}^{k+1}, \qquad (6.16)$$

where $r(u)$ is as in (6.8), $\gamma>0$ is an appropriate constant whose value depends on the selection of $\mathcal{N}$ and needs to be sufficiently large to bring the required stability to the new algorithm and $\mathcal{N}=\mathcal{N}(u)$ is the differential operator arising from the minimization of a convex functional such as $\int_\Omega|\nabla u|$ or $\int_\Omega|\nabla u|^2$. If $r(u)$ can be split in two parts (convex and non-convex) then the convex part is treated implicitly and the non-convex part explicitly. For more insight into convexity-splitting schemes and their implementation we refer the reader to [13, 146, 129, 23, 55, 56].

Based on the convexity-splitting scheme (6.16) we will refine the fixed point method (6.15) to make it stable. To start we borrow the idea of the stabilizing terms $\gamma\mathcal{N}^{k+1}$ and $\gamma\mathcal{N}^{k}$ and add them up to both sides of (6.15). Thus, our new proposed SFP algorithm for a general $\mathcal{N}$ takes the form

$$-\gamma\mathcal{N}_{i,j}^{k+1} - \alpha\nabla\cdot\left((D_2(u))_{i,j}^{k+1}(\nabla u)_{i,j}^{k+1}\right) + u_{i,j}^{k+1} = -\gamma\mathcal{N}_{i,j}^{k}$$
$$-\alpha\nabla\cdot\left((D_1(u))_{i,j}^{k}(\nabla\Phi')_{i,j}^{k}\right) + u_{i,j}^{0}. \tag{6.17}$$

Now we address the selection of $\mathcal{N}$ since it plays an important role on the performance of this new SFP scheme. We consider three possible options

$$\mathcal{N} = \begin{cases} u, \\ \Delta u, \\ \mathcal{TV}(u) = \nabla\cdot\frac{\nabla u}{|\nabla u|}. \end{cases} \tag{6.18}$$

In our experiments the first option $\mathcal{N} = u$ delivered very poor performance because very large γ needs to be selected resulting in very slow convergence. The second option $\mathcal{N} = \Delta u$ has been preferred by some authors for time-evolution schemes such as (6.16) in other contexts; see for instance [13, 129]. The third option $\mathcal{N} = \mathcal{TV}(u)$ is our recommended option, we illustrate the advantages of this selection using the two examples of Figure 6.3. First, Figure 6.3(a) from denoising a smoothly varying image (with no visible jumps in it) shows the performance for the first ten iterations of our SFP algorithm; for this problem option 2 is almost the same as option 3 in performance. However, Figure 6.3(b) from denoising an image with a lot of jumps (edges) shows that option 3 is clearly better. We also tried other refinements of this option 3 with $\mathcal{TV}_M(u) = \nabla\cdot\frac{\nabla u}{|\nabla u|^M}$ for $M = 2, 3$. Then we found that the resulting SFP method is more sensitive to the selection of γ.

Once the best $\mathcal{N}$ has been selected - in this case option 3 - it is useful to re-write (6.17) in the following way:

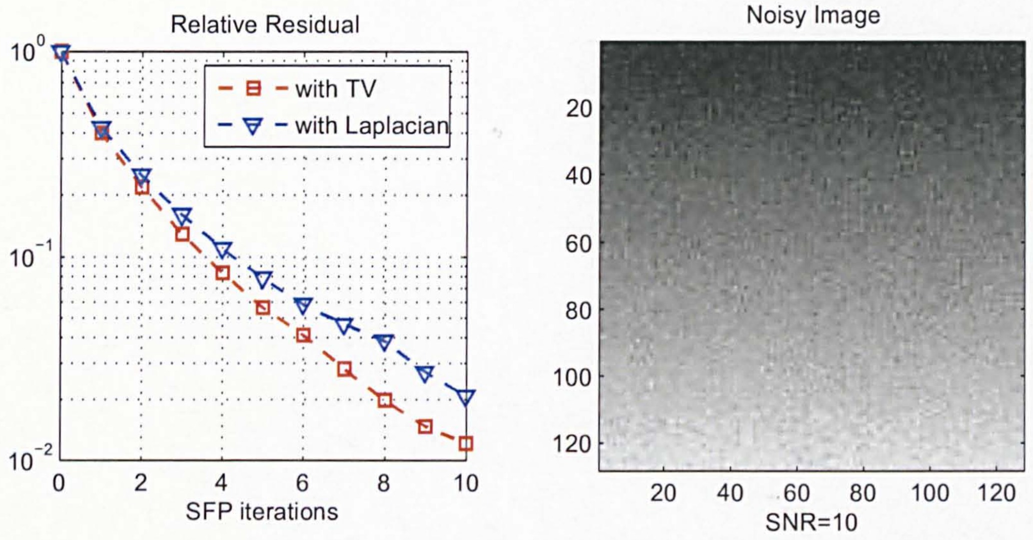
$$-\nabla\cdot\left(C_{i,j}^{k+1}(u)(\nabla u)_{i,j}^{k+1}\right) + g_{i,j}^{k}(u) + u_{i,j}^{k+1} = u_{i,j}^{0}, \tag{6.19}$$

where the diffusion coefficient is defined as $C_{i,j}^{k+1} \equiv \gamma(D_1(u))_{i,j}^{k+1} - \alpha(D_2(u))_{i,j}^{k+1}$ and the nonlinear term $g_{i,j}^{k}(u) \equiv \nabla\cdot\left(\gamma\frac{(\nabla u)_{i,j}^{k}}{|\nabla u|_{i,j}^{k}} + \alpha\frac{(\nabla\Phi')_{i,j}^{k}}{|\nabla u|_{i,j}^{k}}\right)$. To solve (6.19) we consider the following two different methods:

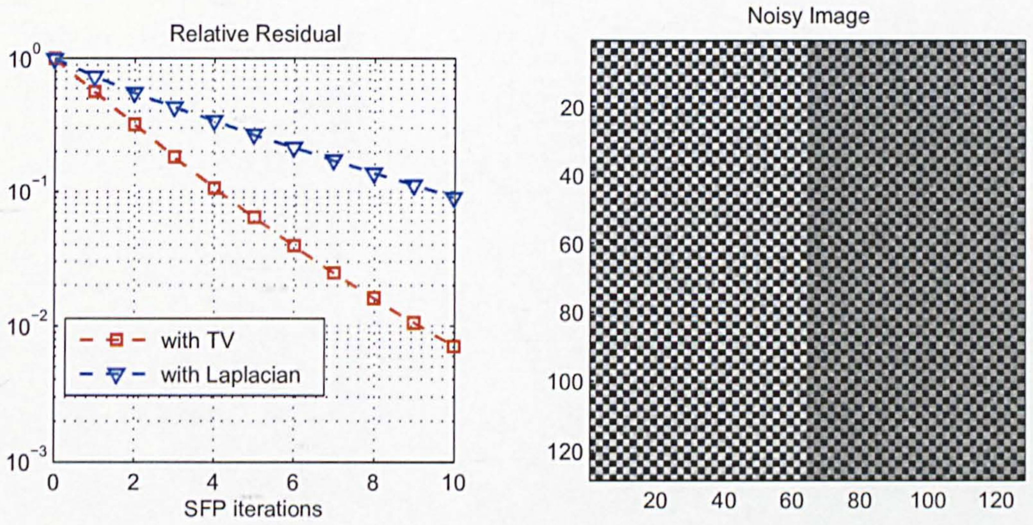
1. Gauss-Seidel-Picard method

The first is a nonlinear Gauss-Seidel-Picard (GSP) method. That is, we represent the system of algebraic equations (6.19) as $N_i(u_1, \ldots, u_r) = 0$ with $i = 1, \ldots, r$ and unknowns $u_1, \ldots, u_r$ and apply nonlinear iterations. For instance, a nonlinear Gauss-Seidel iteration to solve the i^{th} unknown from the i^{th} equation reads as

$$N_i(u_1^{p+1}, \ldots, u_{i-1}^{p+1}, u_i^{p+1}, u_{i+1}^{p}, \ldots, u_r^{p}) = 0, \quad i = 1, \ldots, r \tag{6.20}$$


Relative Residual
with TV
with Laplacian
10^0
10^{-1}
10^{-2}
0
2
4
6
8
10
SFP iterations
Noisy Image
20
40
60
80
100
120
SNR=10

(a)


Relative Residual
with TV
with Laplacian
10^0
10^{-1}
10^{-2}
10^{-3}
0
2
4
6
8
10
SFP iterations
Noisy Image
20
40
60
80
100
120

(b)

Figure 6.3

which together with Picard's method, i.e. freezing all $C_{\cdot,\cdot}$'s and $g_{i,j}(u)$ at the k^{th}-step to solve the single nonlinear equation for a unknown u_i^{p+1} yield the Gauss-Seidel-Picard iteration. We will name this SFP algorithm solved by GSP the SFP1 method.

2. Linearized fixed point method

The second is a linearization of the fixed point method (6.19) by also freezing both all $C_{\cdot,\cdot}$'s and $g_{i,j}(u)$ at the k^{th}-step to obtain

$$u_{i,j}^{k+1}\mathbb{S}_{i,j}^k - u_{i+1,j}^{k+1}C_{i+\frac{1}{2},j}^k - u_{i-1,j}^{k+1}C_{i-\frac{1}{2},j}^k - u_{i,j+1}^{k+1}C_{i,j+\frac{1}{2}}^k - u_{i,j-1}^{k+1}C_{i,j-\frac{1}{2}}^k = f_{i,j}^k \qquad (6.21)$$

where $C_{i+\frac{1}{2},j}^k = \gamma(D_1(u))_{i+\frac{1}{2},j}^k + \alpha(D_2(u))_{i+\frac{1}{2},j}^k$ and so on,

$$\mathbb{S}_{i,j}^k = 1 + C_{i+\frac{1}{2},j}^k + C_{i-\frac{1}{2},j}^k + C_{i,j+\frac{1}{2}}^k + C_{i,j-\frac{1}{2}}^k \qquad (6.22)$$

and $f_{i,j}^k = u_{i,j}^0 - g_{i,j}^k(u)$. Denote the resulting system by $A(u^k)u^{k+1} = f_{i,j}^k$ and notice that A is positive definite and diagonally dominant. If SFP is intended to be used as a stand-alone algorithm, then at each k^{th}-iteration, the system can be solved using for instance PCG or linear MG. However, in the context of a nonlinear MG with SFP used as smoother we found that partially solving the system with a few Gauss-Seidel or SOR iterations works better; we name this SFP algorithm solved up to some accuracy by any of the above linear solvers the SFP2 method.

Remark 6.6.1 *Notice in the SFP2 method with GS as inner solver all $C_{\cdot,\cdot}$'s and $g_{i,j}(u)$ remain frozen across a given number of GS iterations before being updated, whilst in the SFP1 method they are updated after every GS sweep. Therefore, SFP2 can be reduced to SFP1 by using only 1 GS sweep in this case. Since we will use LFA to study the smoothing properties of the SFP1 method this similarity suggests they may have close performance reducing the high frequency components of the error.*

6.6.4 Local Fourier analysis (LFA)

As stated in [139], LFA is a very useful tool for the quantitative analysis of MG methods. Theoretically, LFA is designed to study linear problems with constant coefficients on an infinite grid. Regardless of this strong limitation, LFA is still a recommended tool [20, 85, 1, 152] for the analysis of discrete nonlinear operators like (6.14). To this end, the first step is to neglect boundary conditions and to extend the discrete operator to an infinite grid, the second step assumes that any discrete nonlinear operator can be linearized locally (by freezing coefficients) and can be replaced locally by an operator with constant coefficients [139]. This method has been successfully applied to obtain a better understanding of MG algorithms applied to nonlinear problems for instance in [85, 1, 34, 24, 7] for problems in different contexts.

LFA for the SFP1 method

Although our intention is to analyze SFP2, its nonlinear terms are inconvenient to work with so we analyse instead a closely related variant of it - named as SFP1. The SFP1 method allows the following iteration

$$\frac{1}{h^2}\begin{bmatrix} 0 & 0 & 0 \\ -C^k_{i,j-\frac{1}{2}} & \mathbb{S}^k_{i,j} & 0 \\ 0 & -C^k_{i+\frac{1}{2},j} & 0 \end{bmatrix}(u^{k+1})^{(p+1)} = u^0_{i,j} - (g^{k+1}_{i,j})^{(p+1)}$$
$$-\frac{1}{h^2}\begin{bmatrix} 0 & -C^k_{i-\frac{1}{2},j} & 0 \\ 0 & 0 & -C^k_{i,j+\frac{1}{2}} \\ 0 & 0 & 0 \end{bmatrix}(u^{k+1})^{(p+1)}, \tag{6.23}$$

with $C_{\cdot,\cdot}$, $\mathbb{S}_{i,j}$ and $g_{i,j}$ defined as before and $p = 0, 1, \ldots, gsiter-1$ with $(u^{k+1}_{i,j})^{(0)} = u_{i,j}$. To apply LFA to (6.23) we start by defining the error functions $\xi^{p+1}_{i,j}$ and $\xi^p_{i,j}$ as it is common practice by $\xi^{p+1}_{i,j} = u^{k+1}_{i,j} - (u^{k+1})^{(p+1)}_{i,j}$ and $\xi^p_{i,j} = u^{k+1}_{i,j} - (u^k)^{(p)}_{i,j}$ and expanding them in Fourier components as

$$\xi^{p+1}_{i,j} = \sum_{\phi_1,\phi_2=-m/2}^{m/2} \psi^{p+1}_{\phi_1,\phi_2} e^{i\theta_1 x/h} e^{i\theta_2 y/h} \quad \text{and} \quad \xi^p_{i,j} = \sum_{\phi_1,\phi_2=-m/2}^{m/2} \psi^p_{\phi_1,\phi_2} e^{i\theta_1 x/h} e^{i\theta_2 y/h} \tag{6.24}$$

where $\boldsymbol{\theta} = (\theta_1, \theta_2) \in \Theta = (-\pi, \pi]^2$, $\theta_1 = 2\pi\phi_1/m$, $\theta_2 = 2\pi\phi_2/m$ and $\boldsymbol{i} = \sqrt{-1}$. We also need to linearize (6.23) by freezing the $C_{\cdot,\cdot}$ coefficients and using the Taylor expansion $g^{k+1}_{i,j} = g_{i,j}(u^k_{i,j}) + c_{i,j}((u^{k+1}_{i,j})^{(p+1)} - u^k_{i,j})$ with $c_{i,j} \equiv \frac{\partial g}{\partial u}(u^k_{i,j})$ which is reasonable when $(u^{k+1}_{i,j})^{(p+1)}$ is sufficiently close to $u^k_{i,j}$. After this has been done, we can substitute (6.24) into (6.23) to obtain the error equation

$$-\mathbb{S}^k_{i,j}\xi^{p+1}_{i,j} + C^k_{i+\frac{1}{2},j}\,\xi^p_{i+1,j} + C^k_{i-\frac{1}{2},j}\,\xi^{p+1}_{i-1,j} + C^k_{i,j+\frac{1}{2}}\,\xi^p_{i,j+1} + C^k_{i,j-\frac{1}{2}}\,\xi^{p+1}_{i,j-1} + c_{i,j}\xi^p_{i,j} = 0, \tag{6.25}$$

and, provided lexicographic Gauss-Seidel (GSLEX) is used, to solve (6.25). The local amplification factor from iteration p to $p+1$ is given by

$$\tilde{S}_h(\boldsymbol{\theta})_{i,j} = \left|\frac{\psi^{p+1}_{\phi_1,\phi_2}}{\psi^p_{\phi_1,\phi_2}}\right| = \frac{\left|C^k_{i+\frac{1}{2},j}\, e^{i\theta_1} + C^k_{i,j+\frac{1}{2}}\, e^{i\theta_2} + c_{i,j}h^2\right|}{\left|\mathbb{S}^k_{i,j} - C^k_{i-\frac{1}{2},j}\, e^{-i\theta_1} - C^k_{i,j-\frac{1}{2}}\, e^{-i\theta_2}\right|}, \tag{6.26}$$

and the smoothing factor at each (i,j)-location of the image is $\mu_{i,j} = \sup\{|\tilde{S}_h(\boldsymbol{\theta})_{i,j}|; \boldsymbol{\theta} \in \Theta^{high} = [-\pi,\pi)^2 \backslash [-\frac{\pi}{2}, \frac{\pi}{2})^2\}$.

A close examination of (6.26) reveals that $\mu_{i,j}$ will increase at in-homogenous regions of the image due to the values of the $C_{\cdot,\cdot}$'s there, but more importantly that the SFP1 scheme might no be a good smoother if $c_{i,j}h^2$ is large enough, a situation which is likely to happen in a MG algorithm at the coarse levels. Fortunately for us, a number of experiments showed that provided enough noise has been removed, the value of $g_{i,j}(u)$ barely changes across iterations indicating that $c_{i,j}$ is also very small and suggesting the need of a good initial guess for this method. This however does not represent much a problem since a simple convolution step or a full multigrid (FMG) are enough to provide the required very good initial guess.

Hence, we see that $c_{i,j}h^2$ has negligible effect on the smoothing properties of the SFP1 method if the above condition is satisfied. It is not the same for the $C_{\cdot,\cdot}$'s coefficients which strongly affect the performance as smoother of this method so we expect α, β, γ and the level of noise to have influence in this aspect.

Numerical simulations carried out over different images with fixed α, β and different noise levels yielded very much related to noise levels results. For example, the problem of Figure 6.5(a) with $SNR = 25$ yielded values of $\mu_{i,j} \approx 0.5$ in plain regions, but values of $\mu_{i,j} \approx 0.625$ close to edges which is not that bad. Other problems with much bigger levels of noise however, produced values as bad as $\mu_{i,j} \approx 0.90$ close to edges. In Table 6.1, we present the worst values across Ω_h i.e. $\bar{\mu} = \sup\{\mu_{i,j} : (i,j) \in \Omega_h\}$ for different noise levels at three different grids.

Noise Level	Iteration	Grid Size		
		$h=1$	$h=2$	$h=4$
$SNR = 15$	$k=1$	0.8331	0.8006	0.5974
	$k=2$	0.6653	0.6398	0.3617
	$k=3$	0.5204	0.5109	0.2206
$SNR = 10$	$k=1$	0.8325	0.8199	0.5765
	$k=2$	0.6733	0.6714	0.3374
	$k=3$	0.5515	0.5491	0.1992
$SNR = 3.5$	$k=1$	0.9014	0.8182	0.5606
	$k=2$	0.7800	0.6699	0.3177
	$k=3$	0.6591	0.5485	0.1811

Table 6.1: The values of $\bar{\mu}$ for 3 iterations of the SFP1 method with three different levels of noise are presented.

An interesting observation is that the bigger values of $\mu_{i,j}$ seem to always group around edges as illustrated in the Figure 6.4. This fact will prove to be important since we can use this a priory information to design an adaptive smoother which applies more relaxation at these difficult points. We will show how to do this later on.

Since $0.625^3 \approx 0.244$, this suggest using 3 iterations of the SFP1 method (in low noisy images) to obtain an smoothing factor comparable to 2 iterations of GS for the Poisson equation. Similar analysis can be done for different noise levels to obtain an optimal number of smoothing iterations. Although this analysis suggest a maximum of

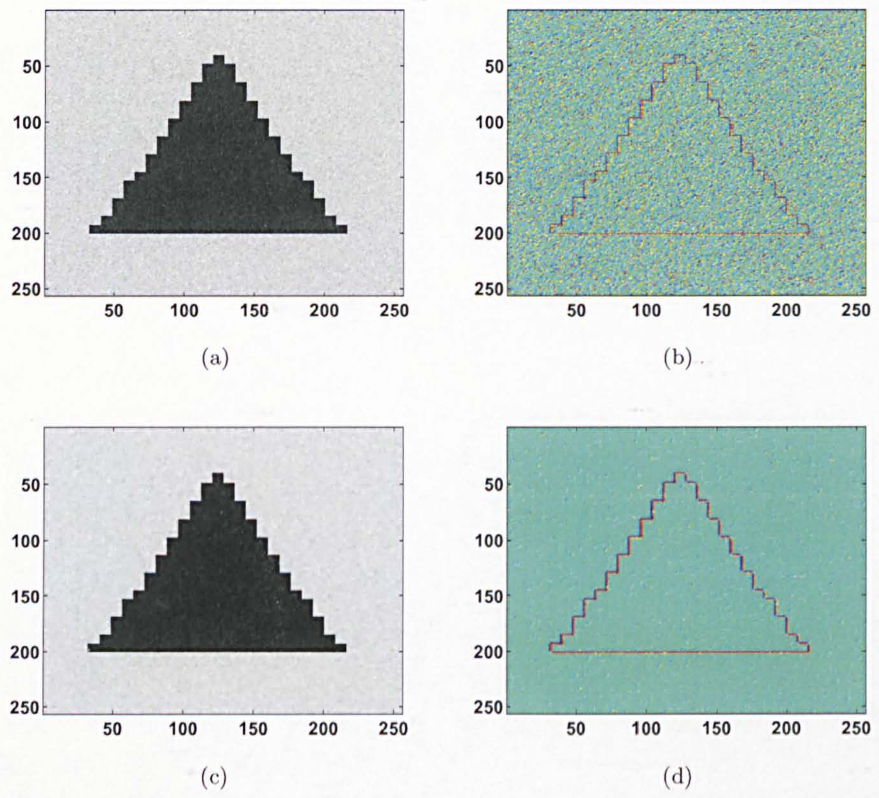


Figure 6.4: Example of the behavior of the smoothing amplification factor $\mu_{i,j}$. (a) Noisy image. (b) Smoothing rates computed for (a) presented as an image. Large values of $\mu_{i,j}$ (red points) are everywhere. (c) Partially denoised image after some iterations of the SFP1 method. (d) Computed values of $\mu_{i,j}$ for (c). Clearly the larger values of $\mu_{i,j}$, around 0.625 for this example, group along edges.

13 SFP1 smoothing steps (since $0.9^{13} \approx 0.25$), when these bad smoothing rates appear the MG algorithm also starts experimenting problems to approximate the nonlinear operator at coarse grids due to strongly discontinuous coefficients so extra local smoothing steps around the difficult points are recommended to partially overcome this problem.

LFA for the SFP2 method

Studying the smoothing properties of this method is much more complicated. We may try using LFA to obtain the smoothing factor of the linear problem for a fixed set of coefficients, but relating the linear with the nonlinear error is not an easy task. In particular partially solving (6.21) with GSLEX method provided good experimental results in the MG framework suggesting that the method may have good smoothing properties. We will provide more evidence through an example in the next section.

Criteria to select between the SPF1 and SFP2 methods

To address this we need to look at the computational costs of both methods. In each case, the action of computing/updating $C_{(\cdot,\cdot)}$ and $g(u)$ is the most costly process and for both this cost is equal to 195 flops per grid point against only 13 flops per grid point of the cost of updating the unknown using GSLEX method. Moreover, they seem to share similar smoothing properties hence in this fashion, the SFP2 method is more attractive since we do not have to update $C_{(\cdot,\cdot)}$ and $g(u)$, as often as in the SFP1 method.

6.6.5 Two-grid analysis

The two-grid analysis is a tool which helps to understand the convergence properties of a multigrid algorithm. Before proceeding to use such a tool in our Algorithm 18, first we will start by explaining its basic principles; a more detailed explanation can be found in [139, 152]. The notation we will use throughout this section is the following: L_h and L_{2h} will represent the linearized operator (6.21) on grids Ω_h and Ω_{2h} of size h and $2h$, respectively; an important assumption is that L_{2h}^{-1} exists. Similarly, we will represent by S_h the smoother operator, i.e. the SFP1 algorithm. In this way, the iteration operator for the $(h, 2h)$ two-grid cycle is given by

$$M_h^{2h} = S_h^{\nu_2} K_h^{2h} S_h^{\nu_1} \quad \text{with} \quad K_h^{2h} = I_h - I_{2h}^h L_{2h}^{-1} R_h^{2h} L_h. \tag{6.27}$$

It is important to note that M_h^{2h} above needs to be computed at each (i,j)-location, but in trying to keep notation simple we have not expressed this dependence explicitly. To calculate convergence factors for M_h^{2h} one needs to analyze how the operators I_{2h}^h, L_{2h}^{-1}, R_h^{2h} and L_h act on the Fourier components $B_h(\boldsymbol{\theta}^{(0,0)}, \cdot) = e^{i\theta_1 x/h} e^{i\theta_2 y/h}$ with $\boldsymbol{\theta}^{(0,0)} = (\theta_1, \theta_2)$; to this end we use the fact that quadruples of $B_h(\boldsymbol{\theta}^{(0,0)}, \cdot)$ coincide in Ω_{2h} with the respective grid function $B_{2h}(2\boldsymbol{\theta}^{(0,0)}, \cdot)$. Then, for any low frequency

$\boldsymbol{\theta} \in \Theta^{low} = [-\frac{\pi}{2}, \frac{\pi}{2})^2$ we consider the frequencies

$$\boldsymbol{\theta}^{(0,0)} = (\theta_1, \theta_2), \quad \boldsymbol{\theta}^{(1,1)} = (\bar{\theta}_1, \bar{\theta}_2), \quad \boldsymbol{\theta}^{(1,0)} = (\bar{\theta}_1, \theta_2), \quad \boldsymbol{\theta}^{(0,1)} = (\theta_1, \bar{\theta}_2), \tag{6.28}$$

where

$$\bar{\theta}_i = \begin{cases} \theta_i + \pi & \text{if } \theta_i < 0 \\ \theta_i - \pi & \text{if } \theta_i \geq 0 \end{cases}. \tag{6.29}$$

After defining $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)$, the corresponding four components $B(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \cdot)$ are called harmonics of each other and for $\boldsymbol{\theta} = \boldsymbol{\theta}^{(0,0)} \in \Theta^{low}$ they generate the four-dimensional space of harmonics $E_h^{\boldsymbol{\theta}} = \text{span}[B(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \cdot) : \boldsymbol{\alpha} \in \{(0,0), (1,1), (1,0), (0,1)\}]$. Hence assuming that I_{2h}^h, L_{2h}^{-1}, R_h^{2h} and L_h can be approximated on Ω_h and Ω_{2h} and $E_h^{\boldsymbol{\theta}}$ remains invariant under S_h, the two-grid operator M_h^{2h} can be represented on $E_h^{\boldsymbol{\theta}}$ for each $\boldsymbol{\theta} \in \Theta^{low}$ by the (4×4)-matrix

$$\hat{M}_h^{2h}(\boldsymbol{\theta}) = \hat{S}_h(\boldsymbol{\theta})^{\nu_2} \hat{K}_h^{2h}(\boldsymbol{\theta}) \hat{S}_h(\boldsymbol{\theta})^{\nu_1} \text{ with } \hat{K}_h^{2h}(\boldsymbol{\theta}) = \hat{I}_h - \hat{I}_{2h}^h(\boldsymbol{\theta}) \hat{L}_{2h}^{-1}(\boldsymbol{\theta}) \hat{R}_h^{2h}(\boldsymbol{\theta}) \hat{L}_h(\boldsymbol{\theta}) \tag{6.30}$$

and each matrix defined by

$$\begin{aligned} \hat{I}_h &= \text{diag}\{1,1,1,1\} \quad \in \mathbb{C}^{4\times4}, \quad \hat{L}_{2h}(\boldsymbol{\theta}) = \tilde{L}_{2h}(2\boldsymbol{\theta}^{(0,0)}) \quad \in \mathbb{C}^{1\times1}, \\ \hat{L}_h(\boldsymbol{\theta}) &= \text{diag}\{\tilde{L}_h(\boldsymbol{\theta}^{(0,0)}),\ \tilde{L}_h(\boldsymbol{\theta}^{(1,1)}),\ \tilde{L}_h(\boldsymbol{\theta}^{(1,0)}),\ \tilde{L}_h(\boldsymbol{\theta}^{(0,1)})\} \quad \in \mathbb{C}^{4\times4}, \\ \hat{R}_h^{2h}(\boldsymbol{\theta}) &= [\tilde{R}_h^{2h}(\boldsymbol{\theta}^{(0,0)})\ \tilde{R}_h^{2h}(\boldsymbol{\theta}^{(1,1)})\ \tilde{R}_h^{2h}(\boldsymbol{\theta}^{(1,0)})\ \tilde{R}_h^{2h}(\boldsymbol{\theta}^{(0,1)})] \quad \in \mathbb{C}^{1\times4}, \\ \hat{I}_{2h}^h(\boldsymbol{\theta}) &= \frac{1}{4}[\tilde{I}_{2h}^h(\boldsymbol{\theta}^{(0,0)})\ \tilde{I}_{2h}^h(\boldsymbol{\theta}^{(1,1)})\ \tilde{I}_{2h}^h(\boldsymbol{\theta}^{(1,0)})\ \tilde{I}_{2h}^h(\boldsymbol{\theta}^{(0,1)})]^T \quad \in \mathbb{C}^{4\times1}. \end{aligned} \tag{6.31}$$

Based on the above representation, we can calculate the asymptotic convergence factor of M_h^{2h} as follows:

$$\rho_{loc}(M_h^{2h})_{i,j} = \sup\{\rho_{loc}(\hat{M}_h^{2h}) : \boldsymbol{\theta} \in \Theta^{low}, \boldsymbol{\theta} \notin \Lambda\}, \tag{6.32}$$

where $\Lambda = \{\boldsymbol{\theta} \in \Theta^{low} : \tilde{L}_h(\boldsymbol{\theta}) = 0 \text{ or } \tilde{L}_{2h}(\boldsymbol{\theta}) = 0\}$. Again we will have different values for $\rho_{loc}(M_h^{2h})_{i,j}$ depending on the (i,j)-location so we define $\bar{\rho}_{loc}$ as the the maximum of $\rho_{loc}(M_h^{2h})$ over all (i,j) and take this value as the asymptotic convergence factor of M_h^{2h}.

In our Algorithm 18 the transfer operators we use are standard full weighting (FW) and bilinear interpolation and their *symbols* [139, 152] are defined by

$$\tilde{R}_h^{2h}(\boldsymbol{\theta}^{\boldsymbol{\alpha}}) = \frac{1}{4}(1 + \cos(\theta_1^{\alpha})(1 + \cos(\theta_2^{\alpha}), \tag{6.33}$$

$$\tilde{I}_{2h}^h(\boldsymbol{\theta}^{\boldsymbol{\alpha}}) = (1 + \cos(\theta_1^{\alpha})(1 + \cos(\theta_2^{\alpha}). \tag{6.34}$$

The symbols for the linearized operators are obtained on the other hand from (6.21) and they are defined by

$$\tilde{L}_h(\theta^\alpha) = -\mathbb{S}^k_{i,j} + C^k_{i+\frac{1}{2},j}\, e^{i\theta_1^\alpha} + C^k_{i-\frac{1}{2},j}\, e^{-i\theta_1^\alpha} + C^k_{i,j+\frac{1}{2}}\, e^{i\theta_2^\alpha} + C^k_{i,j-\frac{1}{2}}\, e^{-i\theta_2^\alpha}, \quad (6.35)$$

$$\tilde{L}_{2h}(\theta^\alpha) = -(\mathbb{S}^k_{i,j})_{2h} + (C^k_{i+\frac{1}{2},j})_{2h}\, e^{i2\theta_1^\alpha} + (C^k_{i-\frac{1}{2},j})_{2h}\, e^{-i2\theta_1^\alpha}$$
$$+(C^k_{i,j+\frac{1}{2}})_{2h}\, e^{i2\theta_2^\alpha} + (C^k_{i,j-\frac{1}{2}})_{2h}\, e^{-i2\theta_2^\alpha}, \quad (6.36)$$

where in the last equation $(\mathbb{S}^k_{i,j})_{2h}$ and $(C^k_{\cdot,\cdot})_{2h}$ are the frozen coefficients at coarse grid Ω_{2h}.

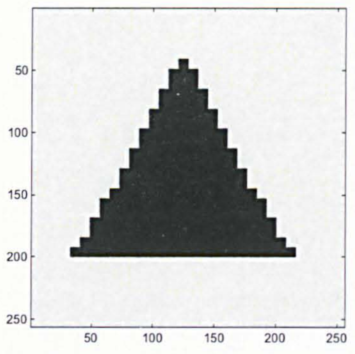
Due to the high-nonlinearity of (6.6) analyzing the behavior of $\rho_{loc}(\hat{M}^{2h}_h)i,j$ for this equation is pretty challenging. From (6.30), we see that the its final value will depend on the product of three different factors : the pre-smoothing steps $\hat{S}_h(\theta)^{\nu_1}$, the post-smoothing steps $\hat{S}_h(\theta)^{\nu_2}$ and the coarse grid operator $\hat{K}^{2h}_h(\theta)$. The latter will be affected by the level of noise, α, β, γ due to the influence of $\hat{L}_h(\theta)$ and $\hat{L}^{-1}_{2h}(\theta)$ on it and on how well these two operators approximate the problem at coarse levels. It is also assumed that the residual and errors are smooth enough so the restriction and interpolation operators can do their job properly. The pre- and post-smoothing steps are also influenced by the same factors and for them we already know that their value will increase if one of the following situations happens: noise, α or γ increase or β decreases.

We are now ready to apply two-grid analysis to a real problem and to this end we selected the problem of Figure 6.5(a) to carry out our tests. Numerical simulations using $\alpha = 1/250, \beta = 10^{-2}, \gamma = 150$, 15% of noise added to the image, $\nu_1 = \nu_2 = 3$ and SFP1 as smoother yielded the value $\bar{\rho}_{loc} = 0.1289$ for this problem. This result is perfectly in accordance with the performance of our MG algorithm to solve this problem since it predicts a relative residual of $0.1289^{10} = 1.26 \times 10^{-9}$ after 10 FAS-cycles which is congruent with the the residual of 5.7×10^{-9} that experimentally we obtained; see Figure 6.5(b).

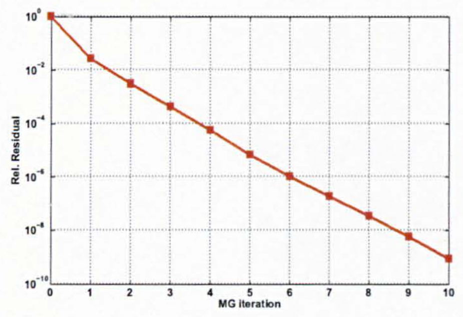
Notice that the problem of Figure 6.5 was designed (by keeping noise level low and few edges) to satisfy the requirements of LFA analysis: that is moderately discontinuous coefficients. Although, we do not expect this analysis to be accurate for large levels of noise, we believe to have gathered enough information from LFA to expect a good performance of our MG algorithm.

It is also important to remark that for the same problem with the same parameters as above but using this time SFP2 as smoother with $gsiter = \nu_1 = \nu_2 = 3$, experimentally we obtained a relative residual of 4.9×10^{-11} after the same number of FAS-cycles i.e. an extra reduction of two-orders of magnitude of the residual value!

This experiment suggests that SFP2 has similar or may be slightly better smoothing properties than SFP1. Recalling also that extra GS iterations are relatively very cheap (13 flops) this method appears to be more attractive for the FAS-cycle.



(a)



(b)

Figure 6.5: (a) Denoised image. (b) MG iteration history with $\nu_1 = \nu_2 = 3$ and the GSP method as smoother.

6.6.6 An adaptive SFP smoother

Initial tests on our multigrid algorithm with SFP2 as smoother showed very fast convergence on a wide range of problems when using $gsiter = \nu_1 = \nu_2 = 10$. For some very noisy problems however ν_1 and ν_2 had to be increased to keep the same fast convergence. Although with this selection our MG algorithm is by far much faster than the explicit method, it was a surprise to notice that our SFP2 with preconditioned conjugate gradient method (PCG), or GSLEX as inner solvers, or even SFP1 were nearly as efficient as the MG algorithm in many cases.

A careful analysis of our algorithm revealed that the inter-grid transfer operators were not working properly because very close to the edges of the image the residual and the error were not smooth enough even though in other parts they really were! Recalling that LFA already had warned us about this phenomenon for low noisy images we concluded that for medium and large noisy images the smoothing factors close to edges were much worse than initially we had thought. To solve this problem we considered two options: (1) one was to construct adaptive high-order inter-grid transfer operators as in [1], or (2) to apply extra smoothing steps locally around edges as in [19, 8]. We selected the second option, the reasons being that: on one hand our code needs only a little modification and the overall increment on the computational cost of the MG algorithm is very small; on the other hand constructing a successful adaptive interpolation operator is not an easy task and many different ways to do it need to be tested, see [1, 48] for comments on this respect; further, they are computationally costly and their storage requirements are large [48].

To select the regions where extra relaxation will be applied, a set of vectors whose entries indicate the difficult pixel points in each Ω_{h_ℓ} grid needs to be constructed. Algorithm 19 below shows how to construct such an indicator vector for one grid only; basically this algorithm is used in the first leg of the very first FAS-cycle (after step 3 in

Algorithm 18) and after that the vectors are not updated as redundant smoothing does no harm. The two scalar input entries to Algorithm 19, δ and T, define the percentage of the domain to be over-smoothed, i.e. edges and the initial threshold, respectively; we suggest to use $\delta < 0.2$.

Algorithm 19 $Q_{h_\ell} \leftarrow PV(u_{h_\ell}, \delta, T)$

```
1: Set $u^s = u_{h_\ell}$, $[m\ n] = size(u^s)$.
2: For $i = 1$ to m, For $j = 1$ to n,
   $|\nabla u^s|_{i,j} = \sqrt{(u^s_{i+1,j} - u^s_{i-1,j})^2 + (u^s_{i,j+1} - u^s_{i,j-1})^2}/2h$;
   End, End.
3: Initialize a vector $Q_{h_\ell}$
4: For $i = 1$ to m, For $j = 1$ to n,
     If $|\nabla u^s|_{i,j} > T$, add $(i,j)$ to $Q_{h_\ell}$; End.
   End, End.
5: If $size(T) > \delta * size(u^s)$ increase de value of $T$ and goto step 3
   Else, take $Q_{h_\ell}$ as the outcome; End.
```

Because the size of each Q_{h_ℓ} is very small, the extra storage added to the FAS algorithm is practically negligible. Once the indicator vector has been constructed, we can use it to implement our adaptive smoother and the way to do this is shown in Algorithm 20.

Algorithm 20 SFP $\mathbf{u} \leftarrow SFP(u, u^0, gsiter, h, \alpha, \gamma, \nu, Q, \omega)$

```
1: For $k = 1$ to $\nu$,
     Implement $A(u^k)u^{k+1} = f(u^k, u^0, \alpha, \gamma)$ using (6.21) and apply gsiter GSLEX iterations.
   End
   Name the outcome of the above for loop as $\bar{u}$.
2: If Q is not empty,
   using Q, extract from $\bar{u}$ and $u^0$ those pixels where Q points out to construct the vectors $u_Q$ and ${u^0}_Q$.
   For $k = 1$ to $\omega * \nu$,
     Implement $A(u^k_Q)u^{k+1}_Q = f(u^k_Q, u^0_Q, \alpha, \gamma)$ using (6.21) and apply gsiter GSLEX iterations.
   End
   Replace $u_Q$ into $\bar{u}$ properly and take $\bar{u}$ as the outcome of SFP.
3: If Q is empty,
   Take $\bar{u}$ as the outcome of SFP.
```

6.6.7 Complexity analysis

The main cost of our MG method is the cost of the smoothing steps. The cost of each outer iteration consists of the cost of each GSLEX step which is equal to 13 flops per grid point and the cost of the discretization which equals 195 flops per grid point. This makes the cost of every outer iteration equal to $(195+13gsiter)N$ where $N = nm$ is the total number of grid points. The other costs to be considered are: (*rhs*) which stands for the cost associated with the construction of the right-hand-side of the residual equation at the next coarser level (steps 3,4 and 5 of Algorithm 18) and (*irc*) representing the interpolation plus residual correction procedure (step 7 of Algorithm 18). In Ω_{2h} there are 1/4 the number of grid points that there are in Ω_h and in general if $p = 2^l$ there are $p^{-2} = (1/4)^l$ as many grid points on Ω_{ph} as there are in Ω_h . Hence an upper bound on the cost of one FAS-cycle is

$$\lim_{l\to\infty}\left(\underbrace{(1+\delta\omega)(\nu_1+\nu_2)(195+13gsiter)N}_{\text{pre+post-smoothing}}+\underbrace{306N}_{rhs}+\underbrace{4N}_{irc}\right)\sum_{n=0}^{l}(1/4)^n$$
$$= (1+\delta\omega)(\nu_1+\nu_2)(195+13gsiter)\frac{N}{0.75}+\frac{1240N}{3}. \tag{6.37}$$

Notice that this bound is strongly dominated by the cost of *smoothing* while the contribution from *rhs* and *irc* is relatively negligible. For instance, selecting $\delta = 0.2, \omega = 2, \nu_1 = \nu_2 = 10$ and $gsiter = 2$ we see that the cost of smoothing is approximately 26 times of that of the latter or put precisely the cost of $8250.6N\sum_{n=0}^{l}(1/4)^n$ versus $310N\sum_{n=0}^{l}(1/4)^n$.

6.7 Numerical results and experiments

In this section we present some results obtained with our MG algorithm to show that its convergence properties are good, the quality of restoration by the high order model is good as well and it is much faster than previous explicit methods. We also present some analysis to show how our MG algorithm is affected by changes on the values of parameters α, γ, h, β and the level of noise. All tests reported here will use $\Phi(\kappa) = \kappa^2$ in the regularization functional. For the selection of the initial guess u_h^0, there is no restriction whatsoever; we tested with $u_h^0 = u^0$, $u_h^0 = 0$ and $u_h^0 = G \star u$ with G a Gaussian convolution operator. For all these options, our MG algorithm converged, but convergence was slightly faster if the convolved initial guess was used and further the computational cost in doing this is very low.

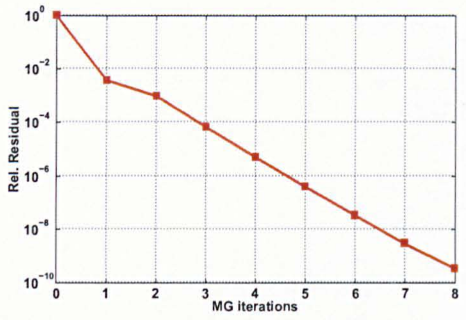
6.7.1 Convergence tests

To illustrate the convergence performance of our MG algorithm we present in Figure 6.6 the residual (R) and relative residual (RR) iteration results after 10 FAS-cycles when solving the three test problems of Figures 6.12, 6.13 and 6.14 with $SNR = 3.5$.

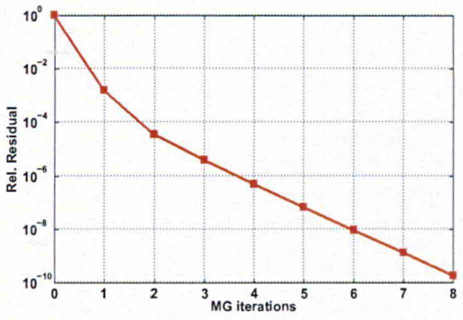
Here the residual and relative residual measures are defined as

$$R = \|Nu^k - u^0\|_2 \quad \text{and} \quad RR = \frac{\|Nu^k - u^0\|_2}{\|Nu^0 - u^0\|_2}.$$

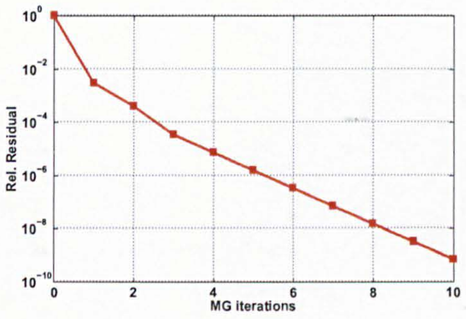
The MG algorithm was run with the following parameters: $\alpha = 1/200, \beta = 10^{-2}, \gamma =$



(a)



(b)



(c)

Figure 6.6: (a) Lena problem. (b) Peppers problem. (c) Brain problem.

$100, \nu_1 = \nu_2 = 10, gsiter = 2, \delta = 0.2, \omega = 2$. Clearly our MG algorithm shows very fast convergence in both cases. At the coarsest level we use $\nu_0 = 250$ or stop when the residual is less than 1×10^{-6} all tests in this section follow this approach.

6.7.2 Computational cost and speed comparison

The closer competitors to our MG algorithm happen to be our SFP1 and SFP2 smoothers used as standalone methods. Here we will analyze and compare the costs of both and MG when solving a real problem. The SFP2 will be solved up to an accuracy of 10^{-6} using GSLEX method, and 10^{-12} using PCG method. Another linear solver we considered was the SOR method; experiments showed that whilst SOR is quite good for lightly noisy images, it is divergent for the heavy noisy ones for any over-relaxation factor. PCG on the other hand makes SFP2 to get stuck at some point if each step is not solved very accurately and therefore, is very costly.

The cost of updating the unknown using GSLEX is equal to 13 flops per grid point, while the cost of using PCG is 15 flops per grid point. If we define a work unit (WU) as the cost of the GS updating, i.e. 1 WU = 13 flops then the cost of discretization which is the same for every method is $195/13 = 15$ WU.

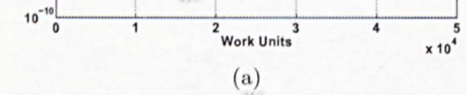
Our benchmark problem here consisted on denoising the problem of Figure 6.12(a) with $\alpha = 1/200, \beta = 10^{-2}, \gamma = 100, SNR = 3.5$ until reaching a relative residual below

1×10^{-9}. In Table 6.2 we present a summary of the results for different sizes of the image and for easier of visualization in Figure 6.7(a) we show the reduction in the relative residual per work unit for each method for the image size of 512^2. Clearly, our MG algorithm is by far the less costly.

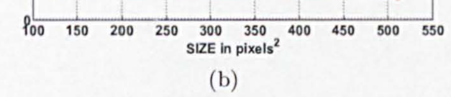
Figure 6.7(b) shows an extra advantage of our MG algorithm, while the cost per grid point grows in the SFP1 and SFP2 methods with the image size, the cost for the MG algorithm practically remains the same though it decreases indeed!

		SFP1	SFP2 GSLEX(10^{-6})	SFP2 PCG(10^{-12})	MG
128^2	inner iter.	1	21	36	10(2)
	outer iter.	982	716	517	8 FAS-cycles
	cost	15712 WU	25060 WU	29230 WU	5883 WU
256^2	inner iter.	1	20	36	10(2)
	outer iter.	1576	827	628	7 FAS-cycles
	cost	25216 WU	29772 WU	35506 WU	5229 WU
512^2	inner iter.	1	20	36	10(2)
	outer iter.	1747	974	859	6 FAS-cycles
	cost	27952 WU	34090 WU	48567 WU	4575 WU

Table 6.2: Here GSLEX(10^{-6}) means SFP2 method solved by GSLEX up to an accuracy of 10^{-6}, similarly PCG (10^{-12}) means SFP2 method solved by conjugate method with zero level incomplete Cholesky factorization up to an accuracy of 10^{-12}, finally 10(2) in the last column means that the MG algorithm uses $\nu_1 = \nu_2 = 10$ and $gsiter = 2$.



(a)



(b)

Figure 6.7: (a) Reduction in the relative residual per work unit for each method. (b) Increase of the cost per size of the image in pixel^2 for each method.

Lastly, we discuss the cost reduction with respect to the explicit method. Updating the unknown with it has a cost of only 5 flops however, at every step all derivatives, coefficients, etc. need to be updated increasing the total cost for each explicit step to

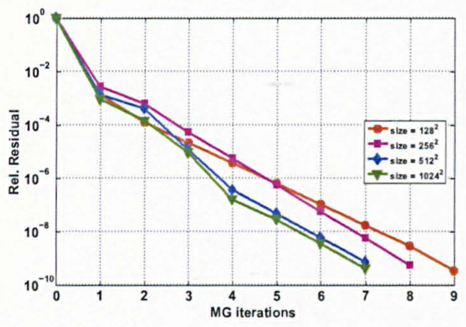


Figure 6.8: h dependence test.

$195 + 5$ flops $= 15.4$ WU. Remarking that the explicit method takes tens of thousands of iterations to converge, our MG algorithm is roughly two orders of magnitude less costly and faster than it.

6.7.3 Computational analysis

Here we analyze how the performance of our MG algorithm is affected when the value of any of the following: $\alpha, \beta\ \gamma, size, SNR$ is changed while values of the rest are kept fixed.

h-dependence test

In Figure 6.8 we illustrate the h-dependence of our MG algorithm, i.e. its performance with respect to different sizes of the image. The MG algorithm was run with the following parameters: $\alpha = 1/200, \beta = 10^{-2}, \gamma = 100, \nu_1 = \nu_2 = 10, gsiter = 2, SNR = 3.5$ for all tests. Clearly, the performance of our MG algorithm is not only not decimated, but gets better as the size of the image increases. One possible explanation to this unusual behavior is that SFP1 and SFP2 are h-dependent due to the $C_{\cdot,\cdot}$'s depending on h. This means that when h increases the value of the coefficients decrease making $\mathbb{S}_{i,j}$ the dominant term, hence obtaining better smoothing at coarser levels, see Table 6.1. In other words, the bigger the size of the image the more h grows at coarse levels, hence the better the efficiency is.

Noise level test

In Figure 6.9 the RR history against MG iterations (FAS-cycles) is presented for different noise levels. The MG algorithm was run with the following parameters: $\alpha = 1/200, \beta = 10^{-2}, \gamma = 100, \nu_1 = \nu_2 = 10, gsiter = 2$ and $size = 256^2$ for all tests. Although convergence is slower for noisier images, in general the number of FAS-cycles does not increase very much.

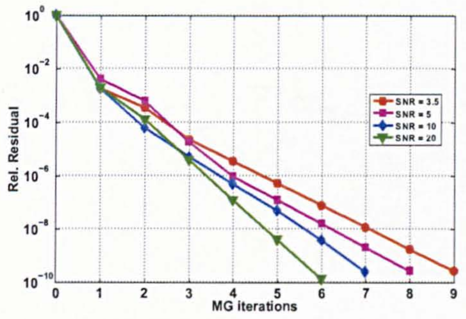


Figure 6.9: SNR dependence test.

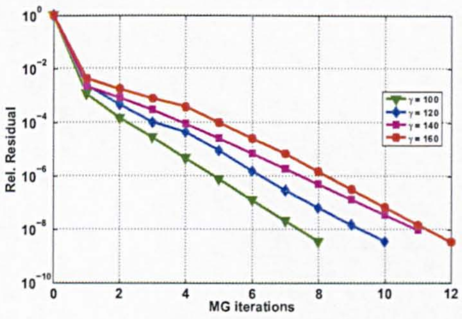


Figure 6.10: γ dependence test.

γ-dependence test

In Figure 6.10, the dependence with respect to the value of γ is presented. The MG algorithm was run with the following fixed parameters: $\alpha = 1/200, \beta = 10^{-2}, \nu_1 = \nu_2 = 10, gsiter = 2$, $SNR = 3.5$ and $size = 256^2$ for all tests, and γ was varied from 100 to 160. As can be seen, a bad selection of γ tends to reduce the performance of the MG algorithm.

α-dependence test

In Figure 6.11 we show how the MG algorithm is affected by selecting different values for the regularization parameter α. The MG algorithm was run with the following fixed parameters: $\beta = 10^{-2}, \gamma = 100, \nu_1 = \nu_2 = 10, gsiter = 2$, $SNR = 3.5$ and $size = 256^2$ for all tests, and α was varied from 0.001 to 0.1. Usually, the value of α needs to be increased as the level of noise gets higher. This has the effect to make the $D(u)$-coefficients more discontinuous affecting, as we already have mentioned, both the approximation of the nonlinear operator on coarse grids and the smoothing factor. Nonetheless, the performance of our MG algorithm is still quite good obtaining very low residuals with few MG cycles.

The results shown in Figure 6.11 need to be carefully interpreted. Although they show that the performance of the MG algorithm is worsened for large α it is also true that such large values are not used in practice. The purpose of α is to select the amount of noise to be removed so there is no point in choosing either a very large or very small α. In Table 6.3 we illustrate the effect different values of α have on the PSNR and the number of FAS-cycles used by our algorithm to solve the problem. The maximum PSNR was for small $\alpha = 0.005$ and only 8 FAS-cycles were needed.

β-dependence test

The most critical parameter affecting the performance of our MG algorithm is definitively β. On the benchmark problem we ran the MG algorithm with the following fixed parameters : $\alpha = 1/200, \gamma = 100, \nu_1 = \nu_2 = 10, gsiter = 2$, $SNR = 3.5$ and $size = 256^2$

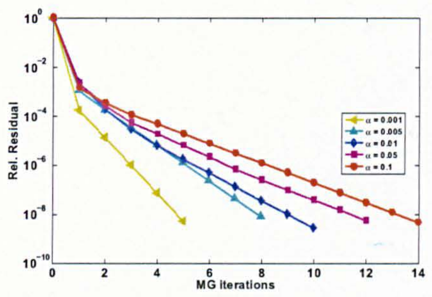


Figure 6.11: α dependence test.

α	FAS-cycles	PSNR
0.001	5	56.8
0.005	8	62.3
0.01	10	62.1
0.05	12	57
0.1	14	52

Table 6.3: Effect of α on the PSNR and number of FAS-cycles.

for all tests, while β was varied from 1 to 10^{-2}.

The results are presented in Table 6.4 and show a much better performance of our MG for large β which is not surprising since β strongly influences the values of the $D(u)$-coefficients. Theoretically, when κ is the curvature of level sets, β should be as small as possible, however in practice this is not only not necessary, but not recommendable from a practical point of view as well. This has been discussed for instance in [5], where it was shown that a smaller β not necessarily leads to a better reconstruction in the TV denoising model. For the curvature-based model something very similar happens and $\beta = 10^{-2}$ is a fair selection which provides a good quality of reconstruction and makes our MG algorithm to perform well.

Table 6.4 shows that decreasing β from 10^{-2} to 10^{-3} for fixed α does not improve the PSNR in a meaningful way, but there is a huge difference in the number of FAS-cycles used by our algorithm to solve both problems.

Table 6.5 presents the PSNR values obtained using different values of β - with α selected to deliver the best possible PSNR for that β value - and the number of FAS-cycles used by our MG algorithm to converge.

β	α	FAS-cycles	PSNR
1	1/200	3	48
10^{-1}	1/200	4	55.9
10^{-2}	1/200	10	60.5
10^{-3}	1/200	25	60.6

Table 6.4: Effect of β on the PSNR and number of FAS-cycles for fixed α.

β	α	FAS-cycles	PSNR
1	1	3	59.3
10^{-1}	1/40	5	59.9
10^{-2}	1/200	10	60.5
10^{-3}	1/200	25	60.6

Table 6.5: Effect of β on the PSNR and number of FAS-cycles for different α values.

Quality of restoration test

We show some qualitative results in Figures 6.12, 6.13 and 6.14. The added Gaussian noise is large so that all of images on the left column have $SNR = 3.5$. As can be appreciated the resulting denoised images do not show the undesirable stair case effect as expected and noise is fairly removed.

FAS-cycles	PSNR	Energy	Residual	Rel. Residual
0	49.38	223.4	20.655	1
1	62.257	190.92	0.02419	10^{-3}
2	62.259	190.91	0.00386	10^{-4}
3	62.259	190.91	0.00074	10^{-5}
4	62.259	190.90	0.00014	10^{-6}
5	62.259	190.90	0.00002	10^{-7}

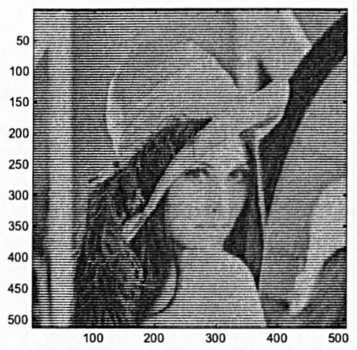
Table 6.6: Iteration history of the MG algorithm showing also the PSNR and residual values at each iteration.

Stopping criterion

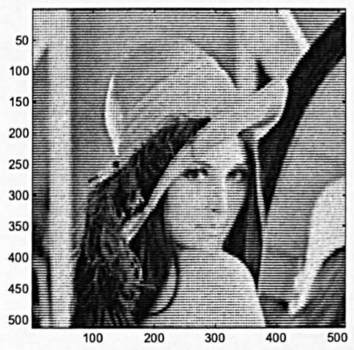
We already have shown the convergence properties of our MG algorithm and now we propose a practical stopping criterion. Although, R and RR defined in Section 6.7.1 can be used to stop the algorithm, we found more useful to use the energy values to execute this task. Our observations indicate that PSNR stop increasing when the change in the energy between two consecutive iterations is below 10^{-1}.. Therefore, in practice, we suggest to stop our MG algorithm when $\|E(u^k) - E(u^{k-1})\|_2 < tol$ with $tol = 10^{-1}$. This is roughly equivalent to stopping the algorithm when $RR < 10^{-4}$. To help the reader to understand our motives we show in Table 6.6 the data obtained from solving the benchmark problem with our MG algorithm using the following parameters: $\alpha = 1/200, \beta = 10^{-2}, \gamma = 100, \nu_1 = \nu_2 = 10, gsiter = 2$, $SNR = 3.5$ and $size = 256^2$.

6.8 Generalization

Fast algorithms for solving high-order PDE's are in high demand. In the image processing community some researchers have realized that using high-order models yield better results than second-order models. Two good examples are the curvature-based denoising model we studied here and the elastica inpainting model [125]. For the resulting PDE's from these high-order models only explicit or semi-implicit time marching methods have been reported in the literature. One of the advantages of our SFP1 and SFP2 algorithms is that they can be very easily generalized to solve other PDE's similar to (6.3). This is, we can implement fast FP algorithms for other models by splitting their differential operators and adding up suitable stabilizing terms. Then these FP algorithms can be used as smoothers in a MG context. For instance, we already have successfully applied this idea for solving the fourth-order PDE of the Euler's elastica digital inpainting model, see [23] for reported results. We also have encouraging results from using this method for solving a 3D denoising problem also known as surface fairing and studied by Elsey and Esedoglu [53]. For this we have implemented only the 2-dimensional case also called curve denoising.



(a)

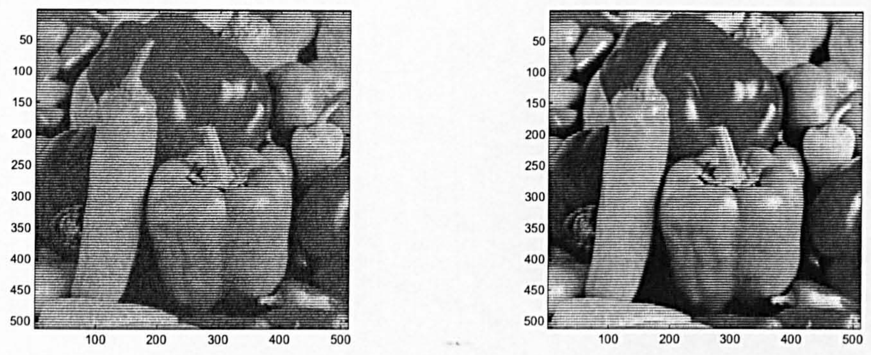


(b)

Figure 6.12: (a) Noisy Image. (b) Denoised image using curvature-based model and Algorithm 17.

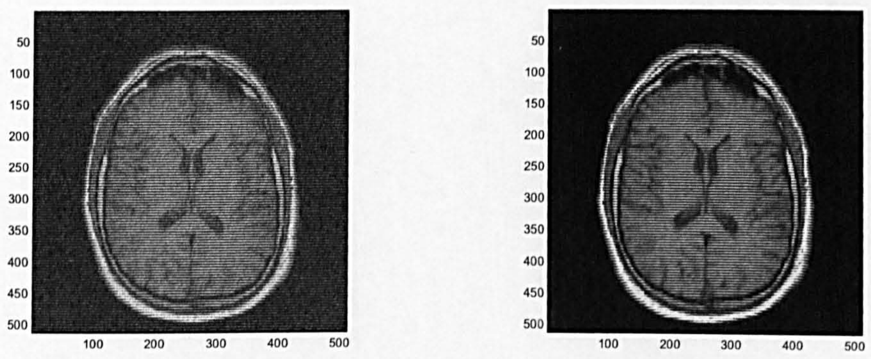
6.9 Conclusions

In this chapter, we introduced two algorithms (SFPGS and MG) for solving the curvature-based denoising model [159], which is high-order and capable of effective noise removal. The resulting Euler Lagrange is of fourth-order, anisotropic and highly nonlinear so conventional algorithms struggle to find the solution quickly and efficiently. In contrast, our MG algorithm is showed to be fast and robust to changes in noise level and parameters. We explained that a fixed point method using the Vogel and Oman [145] idea is unstable and simply does not work for this curvature-based formulation. We showed then how to stabilize this FP method and developed a stabilized fixed point (SFP) giving evidence through LFA of its smoothing properties. Based on this, we developed a fast nonlinear MG method. Finally, a generalization of our algorithms to similar problems was discussed.



(a) (b)

Figure 6.13: (a) Noisy Image. (b) Denoised image using curvature-based model and Algorithm 17.



(a) (b)

Figure 6.14: (a) Noisy Image. (b) Denoised image using curvature-based model and Algorithm 17.

Chapter 7

High-Order Color Image Restoration Models and Fast Algorithms

Denoising of gray-scale images has been extensively studied and investigated within the last decades. With the appearance of the total variation (TV) model of Rudin, Osher and Fatemi [114] back in 1992 became evident that variational approaches to the image denoising problem can yield excellent results. Hence, a lot of research to obtain improvements has been done around this model mainly looking for eliminating or minimizing the so called *staircase effect* inherent to it and which makes images to look blocky, see for instance [95, 118, 46] and references therein. Deep research has also been carried out to equip the TV model with fast and efficient numerical solvers.

The extension of the very successful TV model to color or vector-valued images has been unfortunately, less investigated although interesting works about this subject do exist [117, 14, 21, 130]. The obvious and natural method is the channel by channel (CbC) approach with implies using the TV model independently for each channel. There exists however the feeling among researchers that there must be some kind of coupling among channels although what the type of this coupling should be, it is not clear yet. Regardless of this uncertainty, coupled models have proved to deliver better results than the simple CbC approach in a number of situations.

Coming back to variational models for gray scale images, some of the proposals to avoid the staircase effect consist of increasing the order of the regularization term from first-order (based on first-order derivatives) as in the TV model to second-order. Examples of these methods can be found in [157, 36, 92, 91, 46, 159] and the references therein.

By combining the above two ideas: high-order regularization plus channel coupling; it would be nice to have a high-order color denoising model with coupling among channels. Surprisingly, and up to our knowledge, there is however no published work of such a kind of vector-valued high-order model. Of course, we expect such a model to deliver better results than its counterpart the CbC approach similar to what happens with the lower-order vector-valued TV model.

In this work, already accepted for publication in [25], we take the high-order and curvature-based gray-scale denoising model [159] as our starting point and introduce two different ways to generalize it to vector-valued images obtaining respectively what we call *global* and *local* coupling among the channels. We will analyze the properties of these two new models and will show some examples suggesting that our *global high-order model* is not only better than its correspondent CbC high-order competitor but also better than the TV based color models. Finally, we will show how to implement a fast multigrid algorithm for this model.

This work is organized as follows: first in Section 7.1 we state the variational denoising problem for vector-valued imags. Later in Section 7.2, Section 7.3 and Section 7.4 three total-variation-based models for vector-valued image denoising are reviewed. Then in Section 7.5 our two new high-order models (LM and GM) are introduced. In Section 7.6 the numerical implementation of these high-order models is presented and in Section 7.7 the proposed numerical algorithms with emphasis on a multigrid algorithm for the GM model are explained. Finally, numerical experiments Section 7.9 and conclusions (Section 7.10) will be presented.

7.1 Problem formulation

Define a vector-valued image as a function $\boldsymbol{u} = \Omega \subset \mathbb{R}^n \longrightarrow \mathbb{R}^m$ i.e., $\boldsymbol{u} = (u_1, \ldots, u_m)$ with $u_\ell = u_\ell(x_1, \ldots, x_n)$ $\forall\, \ell = 1, \ldots, m$. In the widely used RGB image color model there are three channels (red, green and blue) and therefore $n = 2$ and $m = 3$. A noisy image $\boldsymbol{u_0}$ is obtained by adding up Gaussian noise $\boldsymbol{\eta}$ to $\boldsymbol{u}$ i.e., $\boldsymbol{u_0} = \boldsymbol{u} + \boldsymbol{\eta}$. The variational approach to remove $\boldsymbol{\eta}$ from $\boldsymbol{u_0}$ is then

$$\min_{\boldsymbol{u}} \left\{ R(\boldsymbol{u}) + \frac{\lambda}{2} \int_\Omega |\boldsymbol{u} - \boldsymbol{u_0}|^2 \, \boldsymbol{dx} \right\} \tag{7.1}$$

where $\boldsymbol{dx} = (dx_1, \ldots, dx_n)$ and $R(\boldsymbol{u})$ is a regularization term selecting the space of functions were $\boldsymbol{u}$ will belong.

In the Figure 7.1 we show an example of a synthetic color image and the way each one of its channels looks in a one-dimensional space. We will use this image to test the models we review here. We now proceed to present in historical order three different ways to generalize the total variation model of Rudin *et al.* [114] from gray scale to vector-valued images. This will prove to be helpful at the moment of introducing our high order color models.

7.2 Vectorial model of Sapiro and Ringach - (SR)

This denoising model for RGB color images is as follows: Let $\boldsymbol{u}(x_1, x_2) : \mathbb{R}^2 \longrightarrow \mathbb{R}^3$ be a multivalued image with $u_\ell(x_1, x_2) : \mathbb{R}^2 \longrightarrow \mathbb{R}$, $\ell = 1, 2, 3.$. Let $P = (x_1^0, x_2^0)$ and

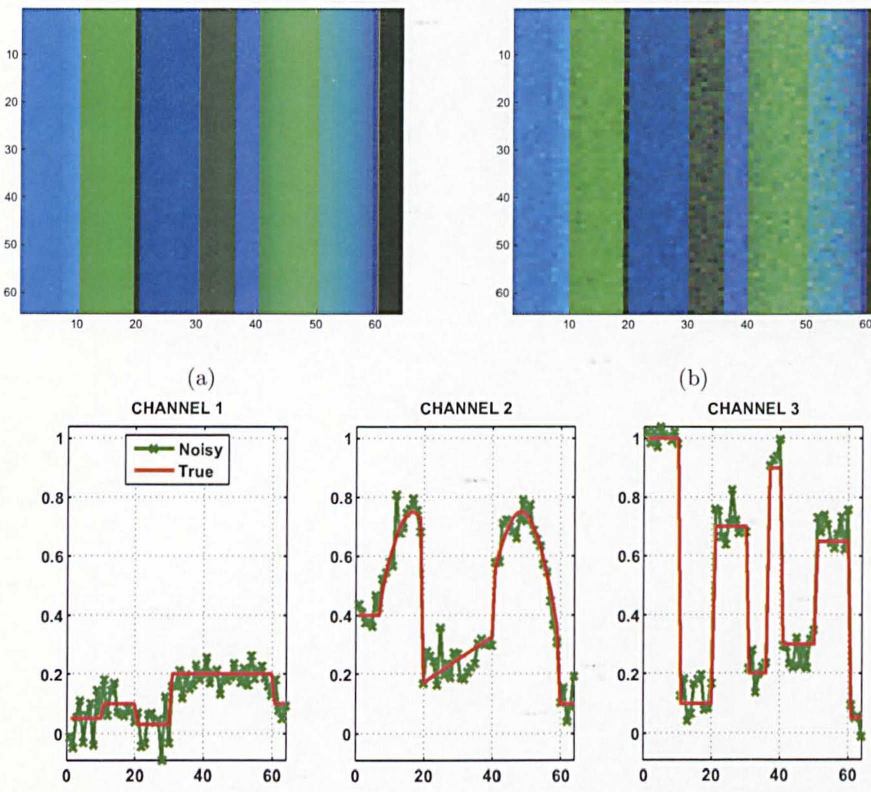


Figure 7.1: Model problem, an image with a weak piecewise constant channel on the left, a piecewise smooth channel in the middle and a strong piecewise constant channel on the right.

$Q = (x_1^1, x_2^1)$ be two points in the $3D$-dimensional image space, then the difference of image values at these two points is defined as $\triangle \boldsymbol{u} = \boldsymbol{u}(P) - \boldsymbol{u}(Q)$. When the Euclidean distance between two points tends to zero the difference becomes the arc element

$$d\boldsymbol{u} = \sum_{i=1}^{2} \frac{\partial \boldsymbol{u}}{\partial x_i} dx_i \tag{7.2}$$

and its squared norm also called the *first fundamental form* [29, 101] is defined by

$$d\boldsymbol{u}^2 = \sum_{i=1}^{2} \sum_{j=1}^{2} \frac{\partial \boldsymbol{u}}{\partial x_i} \frac{\partial \boldsymbol{u}}{\partial x_j} dx_i dx_j. \tag{7.3}$$

By also defining $g_{i,j} = \partial \boldsymbol{u}/\partial x_i \cdot \partial \boldsymbol{u}/\partial x_j$, equation (7.3) above can be rewritten as

$$d\boldsymbol{u}^2 = \sum_{i=1}^{2} \sum_{j=1}^{2} g_{i,j} dx_i dx_j = \begin{bmatrix} dx_i \\ dx_2 \end{bmatrix}^T \begin{bmatrix} g_{1,1} & g_{1,2} \\ g_{2,1} & g_{2,2} \end{bmatrix} \begin{bmatrix} dx_i \\ dx_2 \end{bmatrix}. \tag{7.4}$$

The first fundamental form allows the measurements of changes in the image. The extrema of (7.4) are obtained in the directions of the eigenvectors of the matrix $[g_{i,j}]$, and these are defined as $(\cos\theta_\pm, \sin\theta_\pm)$, where $\theta_\pm$ are given by

$$\theta_+ = \frac{1}{2}tan^{-1}\left(\frac{2g_{1,2}}{g_{1,1}-g_{2,2}}\right) \text{ and } \theta_- = \theta_+ + \frac{\pi}{2}.$$

The values attained at them are the corresponding eigenvalues defined by

$$\lambda_\pm = \frac{g_{1,1}+g_{2,2} \pm \sqrt{(g_{1,1}-g_{2,2})^2 + 4g_{1,2}^2}}{2}. \tag{7.5}$$

Here θ_+ is the direction of maximal change and λ_+ the maximal rate of change. As remarked in [117], by using this information it is possibie to detect image discontinuities by defining a function $f = f(\lambda_+, \lambda_-)$ that measures the dissimilarity between λ_+ and λ_-. In [117] it was proposed to use any decreasing function $g = g(\lambda_+ - \lambda_-)$ and the evolution equation

$$\frac{\partial u}{\partial t} = g(\lambda_+ - \lambda_-)\frac{\partial^2 u}{\partial \theta^2} \tag{7.6}$$

to denoise color images, but no specific g was given. By noticing that

$$\int_\Omega f(\lambda_+, \lambda_-)\, dx_1 dx_2 = \int_\Omega \sqrt{\lambda_+ - \lambda_-}\, dx_1 dx_2 = \int_\Omega |\nabla u|\, dx_1 dx_2 \tag{7.7}$$

for the gray-scale case, that is with $m = 1$, in [117] the authors proposed (7.7) as the possible analog total variation functional for vector-valued images. This proposition as later Blomgren and Chan suggested is not suitable from the variational point of view and $\int_\Omega \sqrt{\lambda_+ + \lambda_-}\, dx_1 dx_2$ was suggested instead.

7.3 Vectorial TV model of Blomgren and Chan - (BLC)

The second vectorial model we review is the Blomgren-Chan's model introduced in [14]. There, a generalization of the TV norm for gray-scale images was also proposed. Keeping similar notation to the one introduced in [14], we start by recalling that for a single channel or gray-scale image $u : \Omega \subset \mathbb{R}^n \to \mathbb{R}$, the total variation norm $\mathrm{TV}_{n,1}$ is defined by

$$\mathrm{TV}_{n,1}(u) = \int_\Omega |\nabla u|\, dx_1 dx_2. \tag{7.8}$$

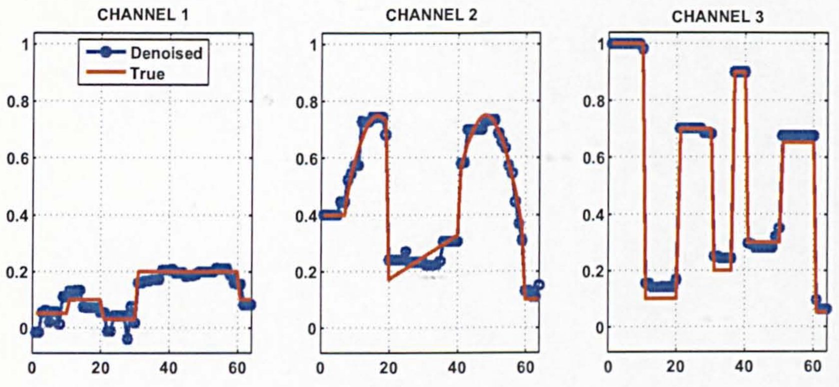
In [14], for any function $u : \mathbb{R}^2 \to \mathbb{R}^m$ the following multi-dimensional total variation norm was proposed:

$$\mathrm{TV}_{n,m}(u) = \sqrt{\sum_{\ell=1}^{m} [\mathrm{TV}_{n,1}(u_\ell)]^2} = \sqrt{\sum_{\ell=1}^{m} \left(\int_\Omega |\nabla u_\ell|\, dx_1 dx_2,\right)^2}. \tag{7.9}$$

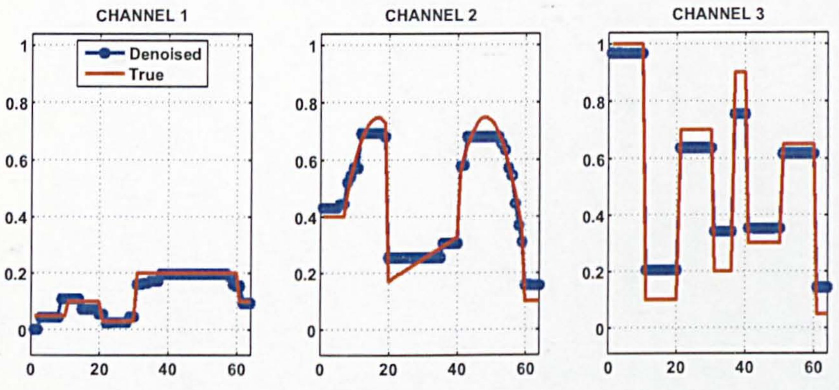
Using this norm in (7.1) as regularization term defines the BLC color denoising model with corresponding Euler-Lagrange equations

$$\frac{\mathrm{TV}_{n,1}(u_\ell)}{\mathrm{TV}_{n,m}(\boldsymbol{u})}\nabla\cdot\frac{\nabla u_\ell}{|\nabla u_\ell|}-\lambda(u_\ell-u_\ell^0)=0 \quad \text{in } \Omega, \quad \ell=1,\dots,m \tag{7.10}$$

and $\nabla u_\ell\cdot\vec{\nu}_\ell=0$, $\forall\ell=1,\dots,m$ on the boundary $\partial\Omega$, where $\vec{\nu}_\ell$ is the normal unit vector on the boundary of the ℓ^{th}-channel.



(a)



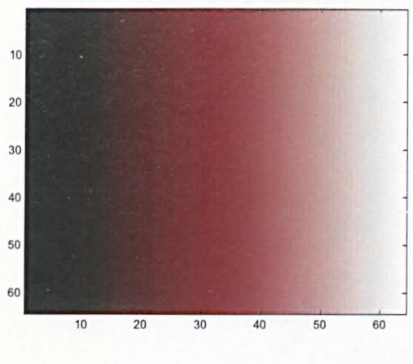
(b)

Figure 7.2: Results from the BLC model for two different values of α: (a) $\alpha=1/13$ and (b) $\alpha=1/5$.

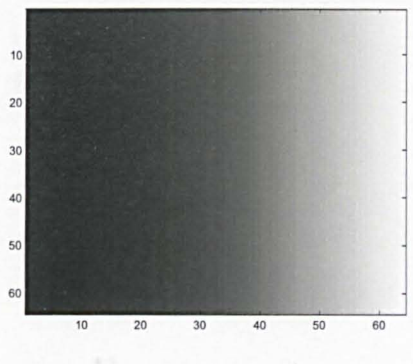
By looking at (7.10) it is easy to see how this model *globally* adjust through the factor $\mathcal{A}(u_\ell)=\frac{\mathrm{TV}_{n,1}(u_\ell)}{\mathrm{TV}_{n,m}(\boldsymbol{u})}$ the quantity of regularization applied to each channel depending on the total variation of the channel itself. This adjusting has the effect of preventing the wiping out of weak channels a problem that the total variation CbC approach does have. This phenomenon is illustrated with the help of the Figure 7.2.

7.4 Total Variation model of Bresson and Chan - (BRC)

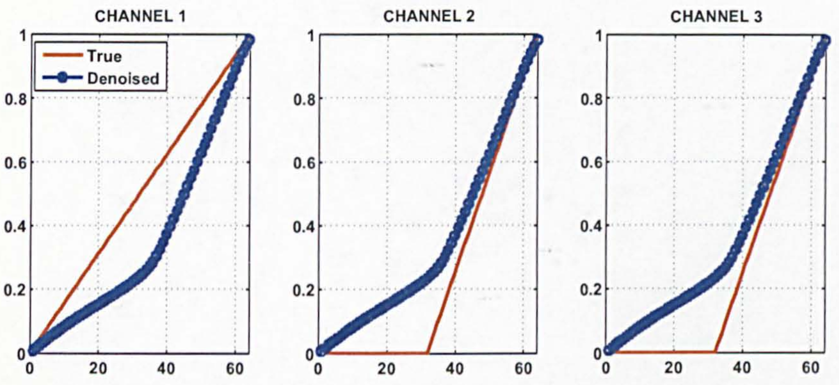
The last vectorial model we review is the Bresson-Chan's model [21]. Actually more than a new model it provides a rigorous mathematical way to construct a TV vector-



(a)



(b)



(c)

Figure 7.3: Results from the BRC model applied to a simple 1D color image with a kink in two of its channels. Dirichlet boundary conditions were used here. (a) Clean color image (b) BRC denoising result (c) 1D plot of each channel, in red is the true data and in blue the obtained result.

valued model that shows the theoretical link between the total variation CbC approach and the SR model with $f = \sqrt{\lambda_+ + \lambda_-}$. The BRC model proposes to construct the TV-norm for vector-valued functions as follows:

For a given vector valued function $\boldsymbol{u} : \Omega \subset \mathbb{R}^n \to \mathbb{R}^m$ the vectorial TV norm is denoted by the finite positive measure

$$\int_\Omega |D\boldsymbol{u}| := \sup_{\boldsymbol{p}\in P} \left\{ \int_\Omega < \boldsymbol{u}, \nabla \cdot \boldsymbol{p} > \, \boldsymbol{dx} \right\}, \tag{7.11}$$

where $\boldsymbol{p} := (\boldsymbol{p}_1, \dots, \boldsymbol{p}_m) : \Omega \to \mathbb{R}^{m\times n}$, $\boldsymbol{p}_\ell := (p_\ell^{x_1}, \dots, p_\ell^{x_n}) : \Omega \to \mathbb{R}^n$, $\forall \ell \in [1, m]$, $\nabla \cdot$ is the divergence operator such that $\nabla \cdot \boldsymbol{q} := (\nabla \cdot \boldsymbol{q}_1, \dots, \nabla \cdot \boldsymbol{q}_n) : \Omega \to \mathbb{R}^m$, $\forall \boldsymbol{q} : \Omega \to \mathbb{R}^{m\times n}$, $\nabla \cdot \boldsymbol{q}_\ell := \sum_{j=1}^n \partial_{x_j} q_\ell^{x_j} : \Omega \to \mathbb{R}$, $\forall \ell \in [1, m]$, the product $< .,. >$ is the Euclidean scalar product defined as $< \boldsymbol{v}, \boldsymbol{w} > := \sum_{\ell=1}^m < v_\ell, w_\ell >$, $\forall (\boldsymbol{v}, \boldsymbol{w}) \in (R^m)^2$, which implies that $< \boldsymbol{u}, \nabla \cdot \boldsymbol{p} > = \sum_{\ell=1}^m < u_\ell, \nabla \cdot \boldsymbol{p}_\ell >$ and the L^2 Euclidean norm $|\cdot|$ is naturally defined by $|\boldsymbol{v}| := \sqrt{< \boldsymbol{v}, \boldsymbol{v} >} = \sqrt{\sum_{\ell=1}^s v_\ell^2}$, $\forall \boldsymbol{v} \in \mathbb{R}^s$.

Depending on the set P of functions of the dual variable $\boldsymbol{p}$, the vectorial total variation norm (VTV) can be defined of different ways, in [21] Bresson and Chan considered the two sets[1]:

$$P_1 = \left\{p \in C_c^1(\Omega; \mathbb{R}^{m\times n}) : |p|_\infty \leq 1\right\} \text{ and } P_2 = \left\{p \in C_c^1(\Omega; \mathbb{R}^{m\times n}) : |p| \leq 1\right\},$$

where $|\cdot|_\infty$ in P_1 is the infinity norm such that $|p|_\infty = \max_{\ell=1,\dots,m} |p_\ell|$ and $|\cdot|$ in P_2 is the L^2 norm such that $|p| = \sqrt{\sum_{\ell=1}^m < p_\ell, p_\ell >} = \sqrt{\sum_{\ell=1}^m \sum_{j=1}^n (p_\ell^{x_j})^2}$.

By selecting the set P_1 it was proven in [21] that the vectorial TV norm can be expressed as

$$\int_\Omega |Du| = \sum_{\ell=1}^m \int_\Omega |\nabla u_\ell| \, dx_1 dx_2 = \sum_{\ell=1}^m TV(u_\ell) \tag{7.12}$$

i.e., the sum of the TV norms of each channel. This vectorial norm used as regularization term in (7.1) yields the total variation CbC color denoising approach.

On the other hand, by selecting P_2 as the set of functions of the dual variable $\boldsymbol{p}$, it can also be shown [21] that this time the vectorial TV norm reduces to

$$\int_\Omega |Du| = \int_\Omega \frac{\sum_{\ell=1}^m < \nabla u_\ell, \nabla u_\ell >}{|\nabla u|} \, dx_1 dx_2 = \int_\Omega \sqrt{\sum_{\ell=1}^m |\nabla u_\ell|^2 \, dx_1 dx_2} \tag{7.13}$$

which used in (7.1) yields the BRC color denoising model.

Thus, by using either (7.12) or (7.13) for regularization in (7.1), we can recover the total variation CbC or the BRC color denoising models, respectively. The latter with Euler-Lagrange equations is defined by

$$\nabla \cdot \frac{\nabla u_\ell}{\|\nabla u\|} - \lambda(u_\ell - u_\ell^0) = 0 \text{ in } \Omega, \; \ell = 1, \dots, m \tag{7.14}$$

where $\|\nabla u\| = \sqrt{|\nabla u_1|^2 + \ldots + |\nabla u_m|^2}$, and homogeneous Neumann boundary conditions for each channel.

From the above system of PDEs we can see that this time the coupling among the channels is obtained through the diffusion coefficient $D(u) = \|\nabla u\|^{-1}$. Further, since $D(u)$ takes different values across the image, the level of coupling varies from one region to another *locally* adjusting the level of regularization. The result of applying the BRC model to our test image can be seen in Figure 7.4. The quality of reconstruction is good but the staircase effect remains present.

Remark 7.4.1 *A connection between the BRC [14] and the SR [117] models can be obtained by selecting $f(\cdot) = \sqrt{\lambda_+ + \lambda_-}$ in (7.7). This specific function was actually not proposed in [117], but in [14], where the authors realized that norms of the form (7.7) with $f(\cdot) = \sqrt{\lambda_+ - \lambda_-}$ does not work well within the variational framework.*

[1]The notation $C_c(\Omega; \mathbb{R}^{m\times n})$ represents the space of scalar continuous functions with compact support in an open set $\Omega \subset \mathbb{R}^{m\times n}$.

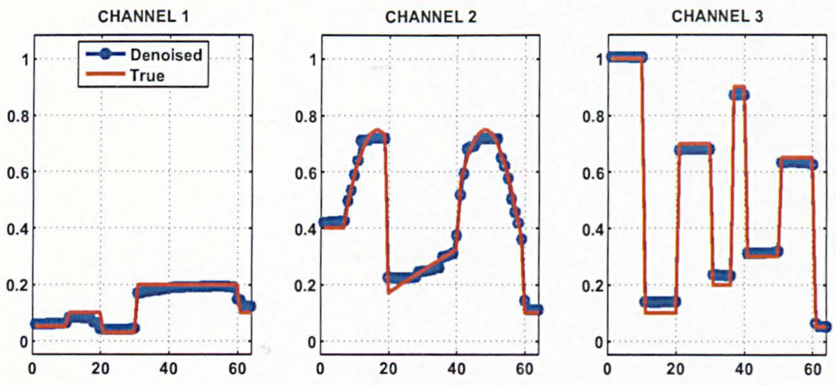


Figure 7.4: Result from the BRC model.

Happens to be that the coupling among channels in the BRC model is so strong that it tends to align all channels causing color smearing in the image. This is illustrated in Figure 7.3 where a very small λ in (7.14) was selected to make the phenomenon more evident.

In summary, and as remarked in [14, 21] the BLC model seems to be superior in quality of restoration to both the CbC approach and the BRC model although the latter is equipped with a faster numerical solver [21].

7.5 High-order vector-valued models

We are ready to state our proposed models. As in the vectorial TV case we need to have a starting point and from there start moving on. In other words, we have to select a working high-order model for gray-scale images and upgrade it to vector-valued images. The idea is also to have coupling among channels as in the vectorial TV models since this has proven to improve the quality of reconstruction.

To this end we decided to use the curvature-based model [159] as our starting point. Our motivations are three fold: (1) this model has nice properties like no staircase effect and contrast and corners preservation [159], (2) there is a fast multigrid solver already available for this model [26], and (3) curvature is an intrinsic geometric feature so generalization is easier.

First we review very briefly what it would be the CbC approach of this curvature-based model for color image denoising.

7.5.1 Channel by channel curvature-based model -(CbCM)

Denote as usual $\boldsymbol{u} = (u_1, \dots, u_m)$ the true vector-valued image, $\boldsymbol{u}^0 = (u_1^0, \dots, u_m^0)$ the noisy image and $\boldsymbol{\kappa} = (\kappa_1, \dots, \kappa_m)$ the curvature vector with $\kappa_\ell = \nabla \cdot \frac{\nabla u_\ell}{|\nabla u_\ell|}$ the curvature of the ℓ^{th}-channel of $\boldsymbol{u}$. Then minimize,

$$\min_{u_1 \in BV(\Omega),\dots,u_m \in BV(\Omega)} \left\{ \int_\Omega \sum_{\ell=1}^{m} \Phi_\ell(\kappa_\ell) \; dx_1 dx_2 + \frac{\lambda}{2} \int_\Omega |\boldsymbol{u} - \boldsymbol{u}^0|^2 \; dx_1 dx_2 \right\} \tag{7.15}$$

which may be done by solving the system of Euler-Lagrange equations

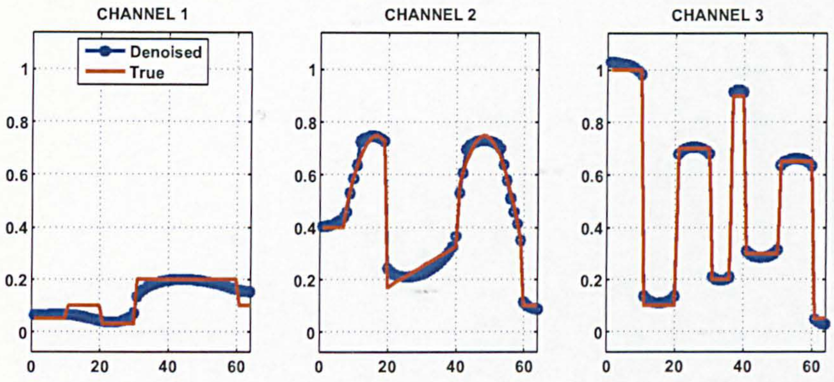


Figure 7.5: Result from the curvature-based CbC approach. The small step in the weak left channel has been almost wiped out.

$$\nabla \cdot \left(\frac{\nabla \Phi'_\ell}{|\nabla u_\ell|} - \frac{\nabla u_\ell \cdot \nabla \Phi'_\ell}{|\nabla u_\ell|^3} \nabla u_\ell \right) + \lambda(u_\ell - u_\ell^0) = 0 \text{ in } \Omega, \;\; \ell = 1, \dots, m \tag{7.16}$$

with boundary conditions $\frac{\nabla \Phi'_\ell}{|\nabla u_\ell|} \cdot \vec{\nu_\ell} = 0$ for each channel. In particular we use here and throughout the rest of this work $\Phi_\ell(k_\ell) = \kappa_\ell^2$ so $\Phi'_\ell = 2\kappa_\ell$.

Here as in the CbC approach for the vectorial TV model, λ is the same for all channels so we expect to have difficulties when denoising a color image having weak channels. This is, by selecting the best λ for one channel we may over-smooth the weak channel. We illustrate this effect in the Figure 7.5.

7.5.2 Local curvature-based color denoising model -(LM)

Here we introduce our first high order denoising model for vector-valued images. We construct this model based on ideas from the Bresson-Chan's model. This model is to minimize

$$\min_{u_1 \in BV(\Omega),\dots,u_m \in BV(\Omega)} \left\{ \int_\Omega \sqrt{\sum_{\ell=1}^{m} \Phi_\ell(\kappa_\ell)} \; dx_1 dx_2 + \frac{\lambda}{2} \int_\Omega |\boldsymbol{u} - \boldsymbol{u}^0|^2 \; dx_1 dx_2 \right\} \tag{7.17}$$

which leads to solve the Euler-Lagrange equations

$$\nabla \cdot \left(\frac{\nabla \Psi_\ell}{|\nabla u_\ell|} - \frac{\nabla u_\ell \cdot \nabla \Psi_\ell}{|\nabla u_\ell|^3} \nabla u_\ell \right) + \lambda(u_\ell - u_\ell^0) = 0 \text{ in } \Omega, \;\; \ell = 1, \dots, m \tag{7.18}$$

with boundary conditions $\frac{\nabla\Psi_\ell}{|\nabla u_\ell|}\cdot\vec{\nu_\ell}=0$ and Ψ_ℓ defined as

$$\Psi_\ell=\frac{\Phi'_\ell(\kappa_\ell)}{\sqrt{\sum_{\ell=1}^m\Phi_\ell(\kappa_\ell)}}. \tag{7.19}$$

As it can be observed the amount of diffusion in this model, is mainly affected by $\nabla\Psi_\ell$, a vector that *locally* varies across the image. Due to this, we name this model *the local curvature-based model* (LM).

By analyzing the above equation we observe that here we do not have the same problem as in the BRC model (its equivalent *local* TV model). In the BRC model the coefficient $\|\nabla\boldsymbol{u}\|^{-1}$ never stops diffusing across edges unless all are aligned. As a consequence color is smeared. In the LM model the two coefficients $|\nabla u_\ell|^{-1}$ and $|\nabla u_\ell|^{-3}$ only depend on u_ℓ so diffusion is properly stopped in every channel.

Unfortunately our experiments revealed that this model has a tendency to develop kinks (see Figure 7.6 channel 3) in the image not only decimating the quality of restoration but making more difficult to construct stable numerical solvers as well.

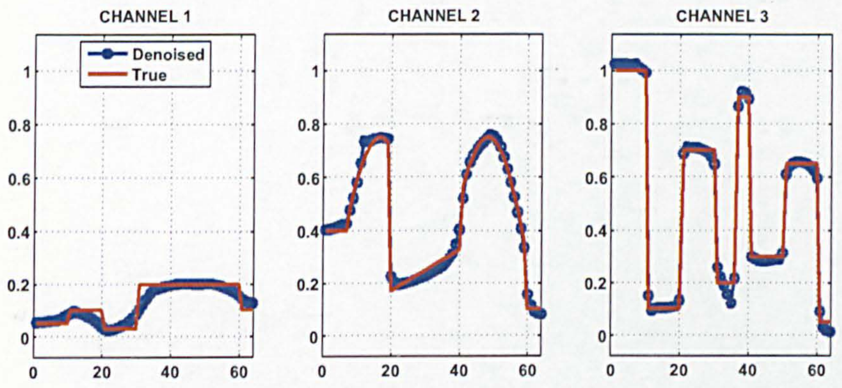


Figure 7.6: Result from the LM model. The small step in the weak left channel is better recovered but a kink appeared in the right channel.

7.5.3 Global curvature-based color denoising model -(GM)

Our second new high order model is inspired on the Blomgren-Chan's TV-norm. Here we propose to minimize.

$$\min_{u_1\in BV(\Omega),\dots,u_m\in BV(\Omega)}\left\{\sqrt{\int_\Omega\sum_{\ell=1}^m\Phi_\ell(\kappa_\ell)\;dx_1dx_2}+\frac{\lambda}{2}\int_\Omega|\boldsymbol{u}-\boldsymbol{u}^0|^2\;dx_1dx_2\right\} \tag{7.20}$$

which leads to solve this time the Euler-Lagrange equations

$$\frac{Q_\ell(u_\ell)}{Q(\boldsymbol{u})}\nabla\cdot\left(\frac{\nabla\Phi'_\ell}{|\nabla u_\ell|}-\frac{\nabla u_\ell\cdot\nabla\Phi'_\ell}{|\nabla u_\ell|^3}\nabla u_\ell\right)+\lambda(u_\ell-u^0_\ell)=0\;\text{ in }\Omega,\;\ell=1,\dots,m \tag{7.21}$$

with boundary conditions $\frac{\nabla \Phi'_\ell}{|\nabla u_\ell|} \cdot \vec{\nu_\ell} = 0$ and $Q(\boldsymbol{u})$ and $Q_\ell(u_\ell)$ defined by

$$Q(\boldsymbol{u}) = \sqrt{\int_\Omega \sum_{\ell=1}^{m} \Phi_\ell(\kappa_\ell) \; dx_1 dx_2} \quad \text{and} \quad Q_\ell(u_\ell) = \int_\Omega \Phi_\ell(\kappa_\ell) \; dx_1 dx_2. \tag{7.22}$$

Here as in the total variation BLC model, the amount of regularization or diffusion is defined by the relation $\mathcal{A}(u_\ell) = Q(u_\ell)/Q(\boldsymbol{u})$ which is the same for all pixels at each channel. For this reason we call this model *the global curvature-based model* (GM).

Thus, we expect this model to reduce regularization for weak channels avoiding smearing them. This is a very similar idea to the one used in the BLC model, however our experiments show that applied together with the curvature-based model, it delivers much better results. A clear example is given in Figure 7.7, where the reconstruction of the small step in the weak left channel is very good and the quality in the others channels is preserved.

An extra feature of this model is that a fast nonlinear multigrid algorithm can be constructed as we will show in Section 7.7.

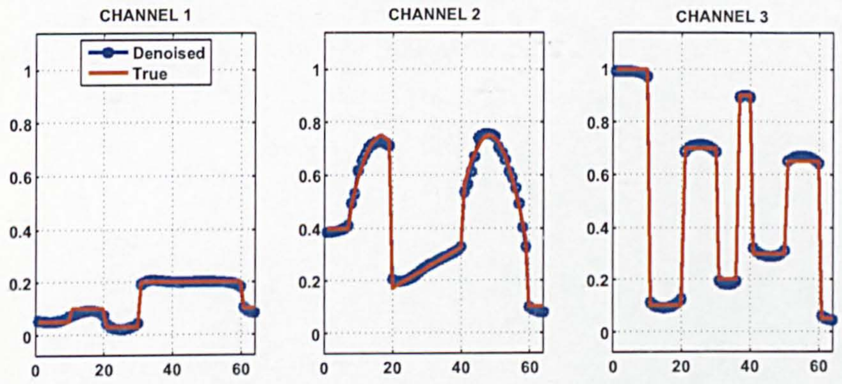


Figure 7.7: Result from the GM model. All channels, including the weak one, are very well reconstructed.

7.6 Numerical implementation

We now proceed to outline the discretization scheme we use for the three high order models i.e., CbCM, LM and GM. We assume the continuous domain $\Omega = [0,p] \times [0,q]$ and let (h_x, h_y) to represent a vector of finite mesh sizes. We also define the infinite grid G_h as $G_h = \{(x,y) : x = x_i = ih_x, \;\; y = y_j = jh_y; \;\; i,j \in \mathbb{Z}\}$ and for simplicity assume $p = q$ and $h = h_x = h_y$. Then the discrete grid is defined as $\Omega_h = \Omega \cap G_h$ and a discrete function on the grid Ω_h as $u_h = u_h(x,y) = u_h(x_i, y_j) = u_h(ih_x, jh_y)$. These discrete functions take on scaled values in the interval $[0,1]$. We also denote the derivative with respect to any variable ψ as $(\cdot)_\psi$.

Thus, for any vector $\boldsymbol{V} = (V^1, V^2)$ its divergence is approximated using central differences i.e., $\nabla \cdot V_{i,j} = (V^1_{i+\frac{1}{2},j} - V^1_{i-\frac{1}{2},j})/h + (V^2_{i,j+\frac{1}{2}} - V^2_{i,j-\frac{1}{2}})/h$, as described in Section 6.5.

For the GM model $Q_\ell(u_\ell)$ and $Q(u)$ are computed the following way:

$$Q_\ell(u_\ell) = \sum_{i=1}^{p}\sum_{i=1}^{q} \Phi_\ell(k_\ell)_{i,j}, \quad Q(u) = \sqrt{\sum_{i=1}^{m}\left(\sum_{i=1}^{p}\sum_{i=1}^{q} \Phi_\ell(k_\ell)_{i,j}\right)}. \tag{7.23}$$

For the LM model Ψ_ℓ is computed by

$$(\Psi_\ell)_{i+\frac{1}{2},j} = \frac{(\Phi_\ell)'_{i+\frac{1}{2},j}(\kappa_\ell)}{\sqrt{\sum_{\ell=1}^{m}(\Phi_\ell)_{i+\frac{1}{2},j}(\kappa_\ell)}}. \tag{7.24}$$

7.7 The numerical solution for the high order models

Here we present a general method to implement a fixed point algorithm for any of the high order models presented so far. Later in this section we will show how to modify slightly this method to obtain an optimal performance for the GM model. Then we will we introduce a nonlinear multigrid method for it.

The Euler-Lagrange equations of all of the high-order models can be written in the general form

$$\mathcal{A}(u_\ell)\nabla \cdot (\mathcal{B}(u_\ell) - \mathcal{C}(u_\ell)\nabla u_\ell) + \lambda(u_\ell - u_\ell^0) = 0, \quad \ell = 1,\dots,m, \tag{7.25}$$

where

$$\mathcal{A}(u_\ell) = 1, \mathcal{B}(u_\ell) = \frac{\nabla\Phi'_\ell(\kappa_\ell)}{|\nabla u_\ell|}, \mathcal{C}(u_\ell) = \frac{\nabla u_\ell \cdot \nabla\Phi'_\ell(\kappa_\ell)}{(|\nabla u_\ell|)^3}, \text{ for CU1-CbC,}$$
$$\mathcal{A}(u_\ell) = 1, \mathcal{B}(u_\ell) = \frac{\nabla\Psi_\ell}{|\nabla u_\ell|}, \mathcal{C}(u_\ell) = \frac{\nabla u_\ell \cdot \nabla\Psi_\ell}{(|\nabla u_\ell|)^3}; \text{ for CU2-LCM and}$$
$$\mathcal{A}(u_\ell) = Q(u_\ell)/Q(u), \mathcal{B}(u_\ell) = \frac{\nabla\Phi'_\ell(\kappa_\ell)}{|\nabla u_\ell|}, \mathcal{C}(u_\ell) = \frac{\nabla u_\ell \cdot \nabla\Phi'_\ell(\kappa_\ell)}{(|\nabla u_\ell|)^3}, \text{ for CU3-GCM.}$$

To obtain a fast solution of the above PDE a fixed point method as the ones described in [145, 144, 40, 119] for the TV model would be desirable. Straight implementation of such a FP method does not work for PDE's like (7.25), so we use the method described in [23, 26], where a stabilizing term $\mathcal{N}(u_\ell)$ is included and the following fast fixed point scheme is used:

$$\begin{aligned} &-\gamma\mathcal{N}(u_\ell^{k+1}) - \mathcal{A}(u_\ell^k)\nabla \cdot \left(\mathcal{C}(u_\ell^k)\nabla u_\ell^{k+1}\right) + \lambda u_\ell^{k+1} \\ &= -\gamma\mathcal{N}(u_\ell^k) - \mathcal{A}(u_\ell^k)\nabla \cdot \left(\mathcal{B}(u_\ell^k)\right) + \lambda u_\ell^0. \end{aligned} \tag{7.26}$$

For the CbC and GM models $\mathcal{N}(u_\ell) = \nabla \cdot \frac{\nabla u_\ell}{|\nabla u_\ell|}$ is selected as in [26]. For the LM model however, $\mathcal{N}(u_\ell) = \nabla \cdot \left(\frac{\nabla u_\ell}{|\nabla u_\ell|\sqrt{\sum_{\ell=1}^{m}(\Phi_\ell)_{i,j}(\kappa_\ell)}}\right)$ provides a better performance of the algorithm.

The fixed point scheme (7.26) leads to a linear system of equations of the form

$$\begin{aligned}(u_\ell)_{i,j}^{k+1}\mathbb{S}_{i,j}^k - (u_\ell)_{i+1,j}^{k+1}C_{i+\frac{1}{2},j}^k - (u_\ell)_{i-1,j}^{k+1}C_{i-\frac{1}{2},j}^k - (u_\ell)_{i,j+1}^{k+1}C_{i,j+\frac{1}{2}}^k \\ -(u_\ell)_{i,j-1}^{k+1}C_{i,j-\frac{1}{2}}^k = f(u_\ell^k, u_\ell^0, \alpha, \gamma)_{i,j}\end{aligned} \tag{7.27}$$

where for example for the CbC and GM models

$$C_{i+\frac{1}{2},j}^k = \frac{\gamma}{|\nabla u_\ell^k|_{i+\frac{1}{2},j}} + \alpha\left(\frac{\nabla u_\ell^k \cdot \nabla\Phi_\ell'(\kappa_\ell)}{(|\nabla u_\ell^k|)^3}\right)_{i+\frac{1}{2},j}$$

and so on, $\mathbb{S}_{i,j}^k$ is given by (6.22), $\alpha = 1/\lambda$ and f is defined as the right-hand side of (7.26). This linear system can be arranged in matrix form $A(u_\ell^k)u_\ell^{k+1} = f(u_\ell^k, u_\ell^0, \alpha, \gamma)$ with A a sparse, symmetric and positive definite matrix. To solve this system we use a simple lexicographic Gauss-Seidel method. The procedure is stated in Algorithm 21.

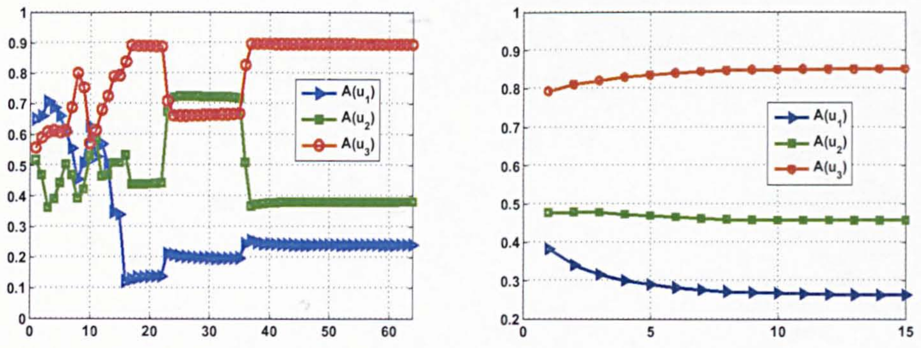
```
Algorithm 21 CFPGS   $u \leftarrow CFPGS(u, u^0, gsiter, h, \alpha, \gamma, \nu)$
Require: On a grid with mesh size $h$, choose an initial guess $u = (u_1, \ldots, u_m)$ for (7.26)
1: for $\ell = 1$ to $m$ do
2:    for $k = 1$ to $\nu$ do
3:       Apply gsiter Gauss-Seidel iterations to the linear system $A(u_\ell^k)u_\ell^{k+1} = f\ (u_\ell^k)$
4:    end for
5: end for
```

Algorithm optimization for the GM model

From the evidence presented before there is clear indication that the GM method is the best among the high order models. Due to this we now concentrate on optimizing the Algorithm 21 for this model. For the GM model $\mathcal{A}(u_\ell) = Q(u_\ell)/Q(u)$, $\mathcal{B}(u_\ell) = \frac{\nabla\Phi_\ell'(\kappa_\ell)}{|\nabla u_\ell|}$ and $\mathcal{C}(u_\ell) = \frac{\nabla u_\ell \cdot \nabla\Phi_\ell'(\kappa_\ell)}{(|\nabla u_\ell|)^3}$.

The correct selection of the value of the stabilizing constant γ is very important for the good performance of the numerical algorithm (7.26), see [26] for a discussion about this subject, and the value of γ among others strongly depends on the value of the regularization parameter α. In the GM model, the term $\mathcal{A}(u_\ell)$ is a constant that is varying as the iterative algorithm evolves and is directly affecting the value of $\bar{\alpha} = \alpha\mathcal{A}(u_\ell)$. This is we can see the GM model as the CbC model with varying regularization parameter $\bar{\alpha}$.

We show in Figure 7.8 an example of the evolution of $\mathcal{A}(u_\ell)$ for the GM and BLC models where for the latter variations are much more moderate. Clearly in our high order GM model $\mathcal{A}(u_\ell)$ will affect the performance of the algorithm if a fixed γ is selected. This was confirmed in our initial experiments. In view of this, we use $\gamma_\ell =$



(a) (b)

Figure 7.8: Evolution of the values of $\mathcal{A}(u_\ell)$ in the GM and BLC models when solving the problem of Figure 7.1. In the GM model and within the first 40 iterations the values continuously and strongly change until reaching a stable state.

$\gamma/\mathcal{A}(u_\ell)$ and solve instead the equation

$$\begin{aligned} -\gamma_\ell \mathcal{N}(u_\ell^{k+1}) - \alpha \mathcal{A}(u_\ell^k) \nabla \cdot \Big(\mathcal{C}(u_\ell^k) \nabla u_\ell^{k+1} \Big) + u_\ell^{k+1} = \\ -\gamma_\ell \mathcal{N}(u_\ell^k) - \alpha \mathcal{A}(u_\ell^k) \nabla \cdot \Big(\mathcal{B}(u_\ell^k) \Big) + u_\ell^0. \end{aligned} \tag{7.28}$$

This method automatically increases the value of γ_ℓ when $\bar{\alpha}$ is small and decreases it when $\bar{\alpha}$ is large. We now proceed to introduce a multigrid algorithm for the GM model.

7.8 Nonlinear multigrid algorithm

We can go one step further and use the fixed point method (7.28) as the foundation for a nonlinear MG algorithm [139]. This algorithm has been successfully tested in a number of imaging problems, for instance: [75] on image registration, [7, 104] on image segmentation, [119, 118, 34, 59, 49] on image denoising-deblurring, and [24, 23] on image inpainting.

Multigrid schemes considerably speed up numerical processes achieving fast results by constructing a hierarchy of discretizations where at each level the error equation is partially solved and the new approximation transported to next coarser level. This process is recursively applied until reaching the coarsest level where an exact, but computationally cheap solution is obtained. Then the process moves backwards on the hierarchical structure transporting the more accurate error and updating the approximate solution at each level until reaching the finest level again. Usually standard coarsening is used to construct the hierarchical structure halving the number of variables on each dimension at each level.

To apply this scheme to our problem we start by defining the nonlinear discrete equations $(N_\ell u_\ell)_{i,j} = (u_\ell^0)_{i,j}$ by

$$(N_\ell u_\ell)_{i,j} = \alpha \mathcal{A}_{i,j}(u_\ell)\nabla \cdot (\mathcal{B}_{i,j}(u_\ell) - \mathcal{C}_{i,j}(u_\ell)(\nabla u_\ell)_{i,j}) + (u_\ell)_{i,j} = (u_\ell^0)_{i,j} \tag{7.29}$$

for $\ell = 1,\ldots,m$ and define $\boldsymbol{N}_h = [N_1,\ldots,N_m]$ as the vector of nonlinear operators on the grid of mesh size h such that $\boldsymbol{N}_h \boldsymbol{u}_h = [N_1u_1,\ldots,N_mu_m]$. Define the residual equations as $r_\ell = N_\ell u_\ell - u_\ell^0$ and the correspondent vector $\boldsymbol{r} = [r_1,\ldots,r_m]$. Then the nonlinear MG scheme for the vector-valued problem is stated in Algorithms 22 and 23.

Algorithm 22 Nonlinear Multigrid Method

```
Require: Select an initial guess $\boldsymbol{u} = (u_1,\ldots,u_m)$ on the finest grid $h$
 1: $k \leftarrow 0$
 2: $err \leftarrow tol + 1$
 3: while $err < tol$ do
 4:     $\boldsymbol{u}_h^{k+1} \leftarrow FAS(\boldsymbol{u}_h^k, \boldsymbol{N}_h^k, \boldsymbol{u}_h^0, \nu_0, \nu_1, \nu_2, gsiter, \alpha, \gamma)$
 5:     $err = \max\left\{\|(r_1)_h^{k+1}\|_2,\ldots,\|(r_m)_h^{k+1}\|_2\right\}$
 6:     $k \leftarrow k+1$
 7: end while
```

Algorithm 23 FAS Cycle $\boldsymbol{u}_h \leftarrow FAS(\boldsymbol{u}_h, \boldsymbol{N}_h, \boldsymbol{u}_h^0, \nu_0, \nu_1, \nu_2, gsiter, \alpha, \gamma)$

```
 1: if $\Omega_h$ = coarsest grid then
 2:     solve $\boldsymbol{N}_h\boldsymbol{u}_h = \boldsymbol{u}_h^0$ accurately (i.e. $\nu_0$ iterations by CFPGS) and return.
 3: else
 4:     continue with step 6.
 5: end if
 6: Pre-smoothing: Do $\nu_1$ steps of,  $\boldsymbol{u}_h \leftarrow CFPGS(\boldsymbol{u}_h, \boldsymbol{u}_h^0, gsiter, \alpha, \gamma, \nu_1)$
 7: Restrict to the coarse grid, $\boldsymbol{u}_{2h} \leftarrow R_h^{2h}\boldsymbol{u}_h$
 8: Set the initial solution for the next level, $\bar{\boldsymbol{u}}_{2h} \leftarrow \boldsymbol{u}_{2h}$
 9: Compute the new right hand side $\boldsymbol{u}_{2h}^0 \leftarrow R_h^{2h}(\boldsymbol{u}_h^0 - \boldsymbol{N}_h\boldsymbol{u}_h) + \boldsymbol{N}_{2h}\boldsymbol{u}_{2h}$
10: Implement $\boldsymbol{u}_{2h} \leftarrow FAS_{2h}(\boldsymbol{u}_{2h}, \boldsymbol{N}_{2h}, \boldsymbol{u}_{2h}^0, \nu_0, \nu_1, \nu_2, gsiter, \alpha, \gamma)$
11: Add the residual correction, $\boldsymbol{u}_h \leftarrow \boldsymbol{u}_h + I_{2h}^h(\boldsymbol{u}_{2h} - \bar{\boldsymbol{u}}_{2h})$
12: Post-smoothing: Do $\nu_2$ steps of  $\boldsymbol{u}_h \leftarrow CFPGS(\boldsymbol{u}_h, \boldsymbol{u}_h^0, gsiter, \alpha, \gamma, \nu_2)$
```

7.9 Numerical results and experiments

We now proceed to show some results obtained using the GM model and the multigrid algorithm outlined above.

From the TV models, it seems to be a general agreement [14, 21] that the BLC model delivers the better results. Among the two high order models (LM and GM) we have introduced here and the CbCM approach, our experiments suggest that GM is better than the other two. Due to these facts, we start by comparing the quality of restoration yielded by the BLC and GM models.

Quality of restoration

The one dimensional plots shown in Figures 7.2 and 7.7 already suggest that our color global curvature-based model outperforms the BLC model. Maybe a more accurate comparison can be carried out by computing the PSNR values between the true image u^t and the denoised image u for each channel. The PSNR measure for one channel of size $p \times q$ is computed using the formula defined in (4.35). The larger the PSNR is, the better restoration of the image is obtained. In real life situation, such a measure is not possible because u^t is not known.

From the obtained PSNR values for the model problem of Figure 7.1 and presented in Table 7.1, it is even clearer that the GM model delivers a much better restoration than the BLC model.

	PSNR		
Model	Channel 1	Channel 2	Channel 3
BLC ($\alpha = 1/13$)	75.54	74.86	73.30
BLC ($\alpha = 1/5$)	79.89	74.52	63.36
GM	82.00	80.94	77.93

Table 7.1: PSNR values from the BLC and GM models.

Even more, carefully looking at Figure 7.9(b) we see that the denoised image coming from the BLC method with $\alpha = 1/13$ looks a bit dirty. This is a combined effect of staircase plus noise still present on the weak channel, this is backed up by the results shown in Figure 7.2(a). By increasing the regularization to $\alpha = 1/5$, now the denoised image in Figure 7.9(c) looks visually much better because noise has been removed from the weak channel which we can confirm in Figure 7.9(b). However by doing so, the third channel is over smoothed, see again Figure 7.2(b), and its correspondent PSNR value gets worsened as seen from Table 7.1. Also notice that the staircase effect still can be observed in Figure 7.9(c). After all, it seems to be that the BLC model cannot cope easily with unbalanced channels.

Finally, in the Figures 7.11, 7.12 and 7.13 we present some qualitative results obtained by using our GM model.

Multigrid performance

Now we proceed to illustrate the fast performance of the nonlinear multigrid algorithm for the GM model. In Figure 7.10, we present the history iteration for solving each one of the problems from Figures 7.11-7.13 all with $SNR \approx 20$. For these problems we used the following parameters: $\gamma = 40$, $\beta = 10^{-2}$, $\alpha = 1/350$, $\nu_1 = \nu_2 = 10$, $gsiter = 10$. Clearly, the MG iteration is very good reaching very quickly very small residuals. A good stopping criteria for the MG algorithm is to stop when the relative residual is less than 10^{-4}.

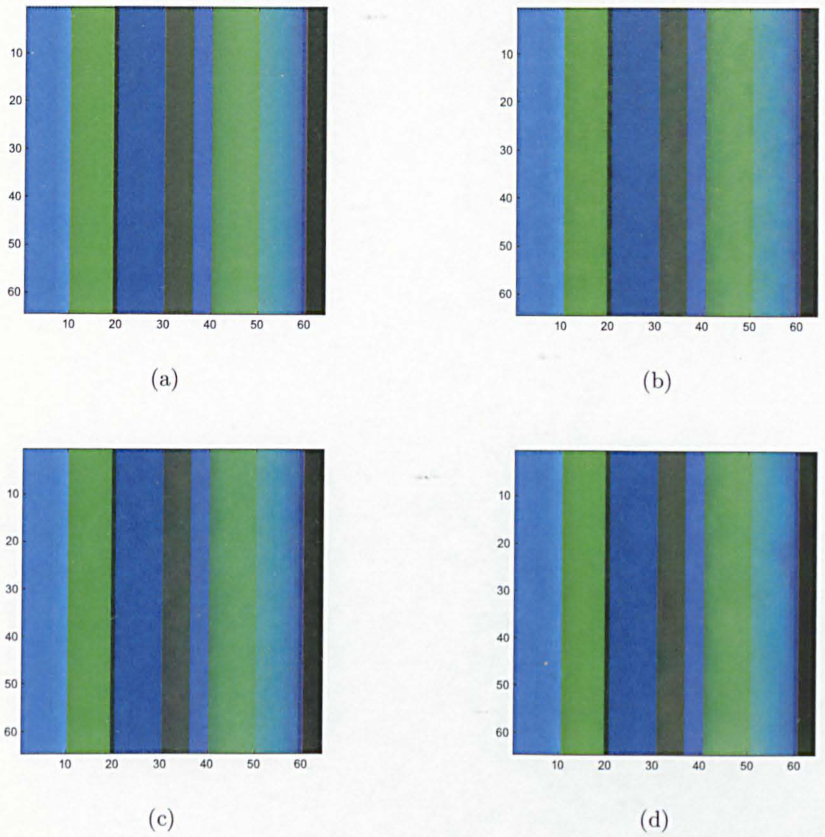


Figure 7.9: (a) Clean image. (b) Denoised image using the BLC model with $\alpha = 1/13$. (c) Denoised image using the BLC model with $\alpha = 1/5$. (d) Denoised image using the GM model.

In Table 7.2 we present the number of V-cycles and CPU-time consumed for the algorithm when solving the same problems with noisier images i.e, $SNR \approx 7$ and using the stopping criterion just described above. All simulations were carried out using Matlab® 2008a on a DELL Intel-Xeon-based computer.

7.10 Conclusions

In this chapter we have introduced two new high-order models for color image denoising. These two new models were designed from the curvature-based denoising model for gray-scale images originally published in [159].

Both models were designed to have coupling among the channels of the color image a characteristic that researchers have identified as highly desirable. From the two models the *global* version, i.e. the GM model showed to deliver the best results. Further, a fast numerical multigrid algorithm was constructed for this model.

A comparison between the new GM model and the BLC model was presented. The

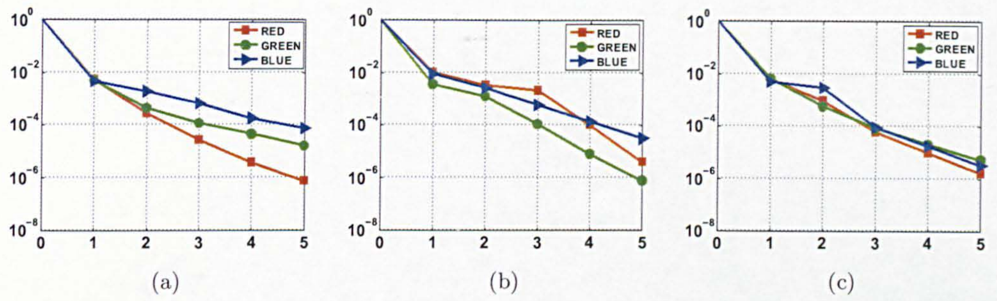
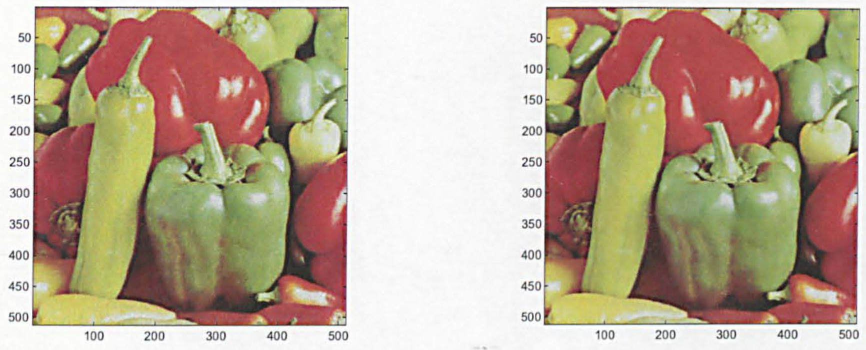


Figure 7.10: Performance of the nonlinear multigrid algorithm when solving the problems of (a) Figure 7.11, (b) Figure 7.12 and (c) Figure 7.13, all with size 256×256 and $SNR \approx 20$.

Image		Multigrid	
Problem	Size	# of V-cycles	CPU-time
Hats	128^2	7	79
	256^2	6	332
	512^2	6	1409
Flowers	128^2	7	78
	256^2	6	325
	512^2	6	1401
Peppers	128^2	8	91
	256^2	7	383
	512^2	6	1411

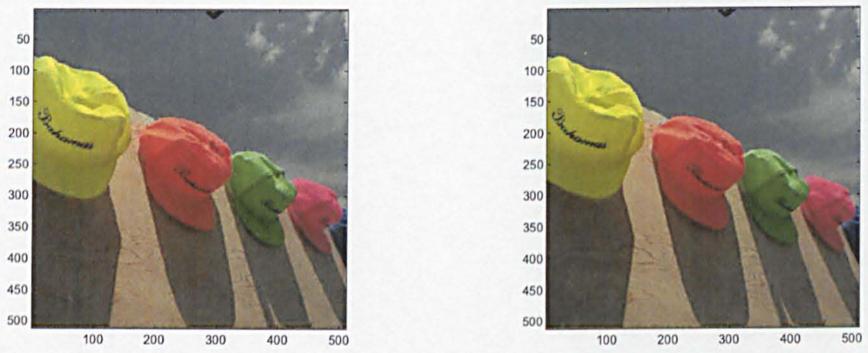
Table 7.2: Number of V-cycles and CPU-times from the MG algorithm when solving the problems of Figures 7.11 - 7.13

BLC model has been branded one of the best among the TV denoising models for color images. The GM model can cope better with unbalanced channels a situation that easily occur when different levels of illumination are present when a picture is taken.



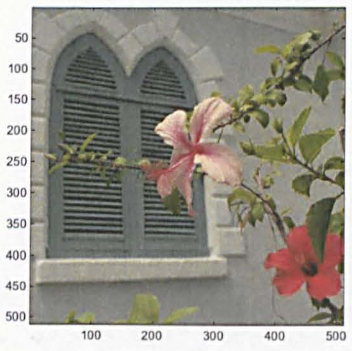
(a) (b)

Figure 7.11: Denoising example using the GM model. Our MG algorithm solved this problem in 1411 seconds.

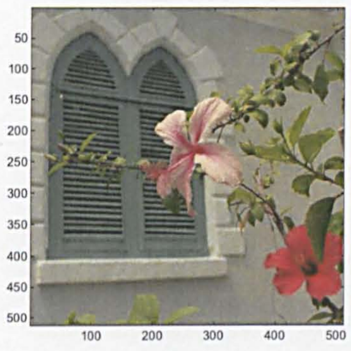


(a) (b)

Figure 7.12: Denoising example using the GM model. Our MG algorithm solved this problem in 1409 seconds.



(a)



(b)

Figure 7.13: Denoising example using the GM model. Our MG algorithm solved this problem in 1401 seconds.

Chapter 8

Future Work

There are different directions we can take from the work presented in this thesis. In the following we mention some of them:

1. Multi-resolution optimization methods would be interesting to test for the high order PDEs of Chapters 5, 6 and 7.

2. Extensions of our multigrid methods to three-dimensional problems look promising though not straightforward. In particular, some researchers have already started to study high-order models for surface denoising also known as fairing with very interesting results. No fast numerical methods for the solution of most of these models already exist.

3. The high-order vectorial models introduced in Chapter 7 for colour image denoising could be extended to colour image inpainting with possible improvements to current channel by channel inpainting techniques. This shall have to be tested.

4. The primal-dual method suggested in Chapter 5 for the Euler's elastica inpainting model deserves more investigation since our initial tests show very promising results, i.e very fast convergence. This technique could also be adapted to the image denoising problem of Chapters 6 and 7.

5. Due to the excellent results obtained from using models with high order regularization in image denoising and inpaiting problems, we already have started developing and testing such kind of techniques on image registration problems. This work is currently undergoing in collaboration with our colleague Noppadol Chumchob.

Bibliography

[1] R. E. Alcouffe, A. Brandt, J. E. Dendy, and J. W. Painter. The multigrid method for the diffusion equation with strongly discontinuous coefficients. *SIAM Journal on Scientific and Statistical Computing*, 2(4):430–454, 1981.

[2] L. Alvarez, F. Guichard, P.-L. Lions, and J.-M. Morel. Axioms and fundamental equations of image processing. *Archive for Rational Mechanics and Analysis*, 123(3):199–257, 1993.

[3] L. Ambrosio and V. M. Tortorelli. Approximation of functionals depending on jumps by elliptic functionals via γ-convergence. *Communications on Pure and Applied Mathematics*, 43:999–1036, 1990.

[4] L. Ambrosio and V. M. Tortorelli. On the approximation of free discontinuity problems. *Bolletin Un. Mat. Ital.*, 6:105–123, 1992.

[5] U. M. Asher, E. Haber, and H. Huang. On effective methods for implicit piecewise smooth surface recovery. *SIAM Journal on Scientific Computing*, 28(1):339–358, 2006.

[6] G. Aubert and P. Kornprobst. *Mathematical Problems in Image Processing - Partial Differential Equations and the Calculus of Variations.* Springer 1st edition; Applied Mathematical Sciences, New York, 2001.

[7] N. Badshah and K. Chen. Multigrid method for the Chan–Vese model in variational segmentation. *Communications in Computational Physics*, 4(2):294–316, 2008.

[8] D. Bai and A. Brandt. Local mesh refinement multilevel techniques. *SIAM Journal on Scientific and Statistical Computing*, 8(2):109–134, 1987.

[9] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera. Filling-in joint interpolation of vector fields and gray levels. *IEEE Transactions on Image Processing*, 10(8):1200–1211, 2001.

[10] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer*

Graphics and Interactive Techniques, pages 417–424, New York, 2000. ACM Press/Addison-Wesley Publishing Co.

[11] M. Bertalmo, A. L. Bertozzi, and G. Sapiro. Navier-Stokes, fluid dynamics, and image and video inpainting. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:355–362, 2001.

[12] A. L. Bertozzi, S. Esedoglu, and A. Gillette. Analysis of a two-scale Cahn-Hilliard model for binary image inpainting. *SIAM Journal on Multiscale Modeling and Simulation*, 6(3):913–936, 2007.

[13] A.L. Bertozzi, S. Esedoglu, and A. Gillette. Inpainting of binary images using the Cahn-Hilliard equation. *IEEE Transactions on Image Processing*, 16(1):285–291, 2007.

[14] P. Blomgren and T. F. Chan. Color TV: total variation methods for restoration of vector-valued images. *IEEE Transactions on Image Processing*, 7(3):304–309, 1998.

[15] T. Bonesky. Morozov's discrepancy principle and Tikhonov-type functionals. *Inverse Problems*, 25(1):015015 (11pp), 2009.

[16] R. S. Borden. *A Course in Advanced Calculus.* Elsevier Science Publishing, 52 Vanderbilt Av. New York, USA, 1983.

[17] J. H. Bramble and X. Zhang. Uniform convergence of the multigrid V-cycle for an anisotropic problem. *Mathematics of Computation*, 70(234):453–470, 2001.

[18] A. Brandt. Multi-level adaptive technique for fast numerical solution to boundary value problems. *Proceedings of the 3rd International Conference on Numerical Methods in Fluid Mechanics. Lecture Notes in Physics*, 18:82–89, 1973.

[19] A. Brandt. Multigrid techniques: 1984 guide with applications to fluid dynamics. *Gesellschaft für Mathematik und Datenverarbeitung*, (GMD-Studie Nr. 85), 1984.

[20] A. Brandt. Multi-level adaptive solutions to BVPs. *Mathematics of Computations*, 31(138):333–390, 1997.

[21] X. Bresson and T. F. Chan. Fast dual minimization of the vectorial total variation norm and applications to color image processing. *Inverse Problems and Imaging*, 2(4):455–484, 2008.

[22] W. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial.* SIAM, Philadelphia, 2000.

[23] C. Brito-Loeza and K. Chen. Fast numerical algorithms for the Euler's elastica digital inpainting model. *Journal of Mathematical Imaging and Vision*, Under review, 2008.

[24] C. Brito-Loeza and K. Chen. Multigrid method for a modified curvature driven diffusion model for image inpainting. *Journal of Computational Mathematics*, 26(6):856–875, 2008.

[25] C. Brito-Loeza and K. Chen. High-order denoising models for vector-valued images. *IEEE Transactions on Image Processing*, Provisionally accepted with mandatory minor revisions, 2009.

[26] C. Brito-Loeza and K. Chen. Multigrid algorithm for high-order denoising. *SIAM Journal on Imaging Sciences*, Under review, 2009.

[27] M. Burger, L. He, and C. Schönlieb. Cahn-Hilliard inpainting and a generalization for grayvalue images. *SIAM Journal on Imaging Sciences*, 2(4):1129–1167, 2009.

[28] F. Cao, Y. Gousseau, S. Masnou, and P. Pérez. Geometrically guided exemplar-based inpainting. *Submitted*, 2008.

[29] M. Do Carmo. *Differential Geometry of Curves and Surfaces.* Prentice Hall, New York, 1976.

[30] V. Caselles, J.-M. Morel, and C. Sbert. An axiomatic approach to image interpolation. *IEEE Transactions on Image Processing*, 7(3):376–386, 1998.

[31] A. Chambolle and P.-L. Lions. Image recovery via total variation minimization and related problems. *Numerische Mathematik*, 76:167–188, 1997.

[32] T. F. Chan and K. Chen. On a nonlinear multigrid algorithm with primal relaxation for the image total variation minimisation. *Numerical Algorithms*, 41:387–411, 2006.

[33] T. F. Chan and K. Chen. An optimization-based multilevel algorithm for total variation image denoising. *SIAM Journal on Multiscale Modeling and Simulation*, 5(2):615–645, 2006.

[34] T. F. Chan, K. Chen, and J. L. Carter. Iterative methods for solving the dual formulation arising from image restoration. *Electronic Transactions on Numerical Analysis*, 26:299–311, 2007.

[35] T. F. Chan, G. Golub, and P. Mulet. A nonlinear primal-dual method for total variation-based image restoration. *Lecture Notes in Control and Information Sciences*, 219:241–252, 1996.

[36] T. F. Chan, A. Marquina, and P. Mulet. High-order total variation-based image restoration. *SIAM Journal on Scientific Computing*, 22(2):503–516, 2000.

[37] T. F. Chan and P. Mulet. On the convergence of the lagged diffusivity fixed point method in total variation image restoration. *SIAM Journal on Numerical Analysis*, 36(2):354–367, 1999.

[38] T. F. Chan, M. K. Ng, A. C. Yau, and A. M. Yip. Superresolution image reconstruction using fast inpainting algorithms. *Applied and Computational Harmonic Analysis, Special Issue on Mathematical Imaging*, 23(1):3–24, 2007.

[39] T. F. Chan, S. Osher, and J. Shen. The digital TV filter and nonlinear denoising. *IEEE Transactions on Image Processing*, 10(2):231–241, 2001.

[40] T. F. Chan and J. Shen. *Image Processing and Analysis: Variational, PDE, Wavelet, and Stochastic Methods.* SIAM, Philadelphia, 2005.

[41] T. F. Chan, J. Shen, and H.-M. Zhou. Total variation wavelet inpainting. *Journal of Mathematical Imaging and Vision*, 25(1):107–125, 2006.

[42] T. F. Chan and W. L. Wan. Robust multigrid methods for nonsmooth coefficient elliptic linear systems. *Journal of Computational and Applied Mathematics*, 123:323–352, 2000.

[43] T. F. Chan, H. M. Zhou, and R. H. Chan. Continuation method for total variation denoising problems. *Advanced Signal Processing Algorithms*, 2563(1):314–325, 1995.

[44] K. Chen. *Matrix Preconditioning Techniques and Applications.* Cambridge Monographs on Applied and Computational Mathematics (No. 19), Cambridge University Press, UK, 2005.

[45] K. Chen and J. Savage. An accelerated algebraic multigrid algorithm for total-variation denoising. *BIT Numerical Mathematics*, 47(2):277–296, 2007.

[46] Y. Chen, S. Levine, and M. Rao. Variable exponent, linear growth functionals in image restoration. *SIAM Journal on Applied Mathematics*, 66(4):1383–1406, 2006.

[47] A. Criminisi, P. Perez, and K. Toyama. Object removal by exemplar-based inpainting. *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:721-728, 2003.

[48] Jr. J.E. Dendy. Black box multigrid. *Journal of Computational Physics*, 48:366–386, 1998.

[49] M. Donatelli. A multigrid for image deblurring with Tikhonov regularization. *Numerical Linear Algebra with Applications*, 12:715–729, 2005.

[50] I. Drori, D. Cohen-Or, and H. Yeshurun. Fragment-based image completion. *ACM Trans. Graph.*, 22(3):303–312, 2003.

[51] R. Duits, L. Florack, J. de Graaf, and Bart ter Haar Romeny. On the axioms of scale space theory. *Journal of Mathematical Imaging and Vision*, 20(3):267–298, 2004.

[52] A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. *ICCV'99: Proceedings of the International Conference on Computer Vision*, 2:1033–1038, 1999.

[53] M. Elsey and S. Esedoglu. Analogue of the total variation denoising model in the context of geometry processing. *SIAM Journal on Multiscale Modeling and Simulation*, 7(4):1549–1573, 2009.

[54] S. Esedoglu and J. Shen. Digital inpainting based on the Mumford-Shah-Euler image model. *European Journal of Applied Mathematics*, 13(4):353–370, 2002.

[55] D. J. Eyre. Unconditionally gradient stable time marching the Cahn-Hilliard equation. *J.W. Bullard, R. Kalia, M. Stoneham, L.Q. Chen (Eds.), Computational and Mathematical Models of Microstructural Evolution*, 53:1686–1712.

[56] D. J. Eyre. An unconditionally stable one-step scheme for gradient systems. *Unpublished article (http://www.math.utah.edu/ftp/u/ma/eyre/stable.ps.gz)*, 1998.

[57] A. Farcas, L. Elliott, D. B. Ingham, and D. Lesnic. An inverse dual reciprocity method for hydraulic conductivity identification in steady groundwater flow. *Advances in Water Resources*, 27(3):223–235, 2004.

[58] R. Fischer and T. Huckle. Multigrid solution techniques for anisotropic structured linear systems. *Applied Numerical Mathematics*, 58:407–421, 2008.

[59] C. Frohn-Schauf, S. Henn, and K. Witsch. Nonlinear multigrid methods for total variation image denoising. *Computing and Visualization in Science*, 7:199–206, 2004.

[60] I. M. Gelfand and S. V. Fomin. *Calculus of Variations.* Prentice-Hall, Inc., New Jersey, 1963.

[61] I. Giakoumis and I. Pitas. Digital restoration of paintings cracks. *Proceedings of the 1998 IEEE International Symposium on IEEE Transactions on Circuits and Systems*, 4:269–272, 1998.

[62] M. Giaquinta and S. Hildebrandt. *Calculus of Variations I, The Lagrangian Formalism.* Springer-Verlag, 1996.

[63] M. Giaquinta and S. Hildebrandt. *Calculus of Variations II, The Hamiltonian Formalism.* Springer-Verlag, 1996.

[64] E. De Giorgi. Frontiere orientate di misura minima. *Sem. Mat. Scuola Norm Sup. Pisa*, 1960-61.

[65] E. Giusti. *Minimal Surfaces and Functions of Bounded Variation. Monographs in Mathematics, Vol. 80.* Birkhauser, 1984.

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

[67] J. B. Greer, A. L. Bertozzi, and G. Sapiro. Fourth order partial differential equations on general geometries. *Journal of Computational Physics*, 216(1):216 – 246, 2006.

[68] H. Grossauer and O. Scherzer. Using the complex Ginzburg-Landau equation for digital inpainting in 2D and 3D. *Lecture Notes in Computer Sciences: Scale Space Methods in Computer Vision*, 2595/2003:1080, 2003.

[69] J. Gu, L. Zhang, G. Yu, Y. Xing, and Z. Chen. X-ray CT metal artifacts reduction through curvature based sinogram inpainting. *Journal of X-Ray Science and Technology*, 14(2):73–82, 2006.

[70] M. Hanke. Limitations of the L-curve method in ill-posed problems. *BIT*, 36(2):287–301, 1996.

[71] P.C. Hansen. Analysis of discrete ill-posed problems by means of the l-curve. *SIAM Review*, 34(4):561–580, 1992.

[72] P.C. Hansen and D.P. O'Leary. The use of the l-curve in the regularization of discrete ill-posed problems. *SIAM Journal on Scientific Computing*, 14(6):1487–1503, 1993.

[73] M. T. Heath. *Scientific Computing.* The McGraw-Hill Companies, Inc.; 2nd edition, New York, 2002.

[74] S. Henn and K. Witsch. A multigrid approach for minimizing a nonlinear functional for digital image matching. *Computing*, 64:339–348, 2000.

[75] L. Homke. A multigrid method for anisotropic PDE's in elastic image registration. *Numerical Linear Algebra with Applications*, 13:215–229, 2006.

[76] C.-H. Huaung and J.-L. Wu. Attacking visible watermarking schemes. *IEEE Transactions on Multimedia*, 6(1):16–30, 2004.

[77] Y.-T. Jia, S.-M. Hu, and R. R. Martin. Video completion using tracking and fragment merging. *The Visual Computer*, 21(8-10):601–610, 2005.

[78] W. Kahan. *Gauss-Seidel Methods of Solving Large Systems of Linear Equations.* PhD thesis, University of Toronto, Canada, 1958.

[79] G. Kanizsa. Organization in vision. Essays on Gestalt perception. 1979.

[80] B. Kawohl and N. Kutev. Maximum and comparison principles for one-dimensional anisotropic diffusion. *Mathematische Annalen*, 311(1):107–123, 1998.

[81] S. L. Keeling and G. Haase. Geometric multigrid for high-order regularizations of early vision problems. *Applied Mathematics and Computation*, 184:536–556, 2007.

[82] J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–370, 1984.

[83] J. Koenderink. Generic neighbourhood operators. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:597–605, 1992.

[84] N. Komodakis. Image completion using global optimization. *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:442–452, 2006.

[85] H. Kostler, K. Ruhnau, and R. Wienands. Multigrid solution of the optical flow system using a combined diffusion- and curvature-based regularizer. *Numerical Linear Algebra with Applications*, 15:201–218, 2008.

[86] V. Kwatra, A. Schodl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph.*, 22(3):277–286, 2003.

[87] D. Lesnic. Characterizations of the functions with bounded variations. *Acta Universitatis Apulensis*, 6:47–54, 2003.

[88] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum. Real-time texture synthesis by patch-based sampling. *ACM Trans. Graph.*, 20(3):127–150, 2001.

[89] T. Lindeberg. *Scale-Space Theory in Computer Vision.* Kluwer Academic Publishers, Norwell, MA, USA, 1994.

[90] A.E.H. Love. *A Treatise on the Mathematical Theory of Elasticity.* Dover Publicacions, Inc, New York, 1927.

[91] M. Lysaker, A. Lundervold, and X.-C. Tai. Noise removal using fourth-order partial differential equation with applications to medical magnetic resonance images in space and time. *IEEE Transactions on Image Processing*, 12(12):1579–1590, 2003.

[92] M. Lysaker, S. Osher, and X.-C. Tai. Noise removal using smoothed normals and surface fitting. *IEEE Transactions on Image Processing*, 13(10):1345–1357, 2004.

[93] R. March and M. Dozio. A variational method for the recovery of smooth boundaries. *Image and Vision Computing*, 15:705–712, 1997.

[94] L. Marin, L. Elliott, P. J. Heggs, D. B. Ingham, D. Lesnic, and X. Wen. Boundary element solution for the Cauchy problem associated to the Helmholtz equation by the Tikhonov regularisation method. *IV: International Symposium on Inverse Problems in Engineering Mechanics (ISIP2003), (ed. M.Tanaka), Elsevier, Amsterdam*, 6:485–494, 2003.

[95] A. Marquina and S. Osher. Explicit algorithms for a new time dependent model based on level set motion for nonlinear deblurring and noise removal. *SIAM Journal on Scientific Computing*, 22(2):387–405, 2000.

[96] S. Masnou. Disocclusion: a variational approach using level lines. *IEEE Transactions on Image Processing*, 11(2):68–76, 2002.

[97] S. Masnou and J.-M. Morel. Level lines based disocclusion. *Proceedings of 5th IEEE Intl Conf. on Image Processing*, 3:259–263, 1998.

[98] Y. Matsushita, E. Ofek, X. Tang, and H.-Y. Shum. Full-frame video stabilization. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005*, 1:50–57.

[99] W.-Q. Yan Mohan, W.-Q. Yan, and M. S. Kankanhalli. Erasing video logos based on image inpainting. *Proc. Int. Conf. on Multimedia and Expo ICME*, 2:521–524, 1998.

[100] M. Nitzberg, D. Mumford, and T. Shiota. *Filtering, Segmentation, and Depth.* Springer-Verlag, New York, 1993.

[101] B. O'Neil. *Elementary Differential Geometry.* Academic Press, San Diego, 1997.

[102] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, New York, 1970.

[103] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces.* Springer 1st edition October, New York, 2002.

[104] G. Papandreou and P. Maragos. Multigrid geometric active contour models. *IEEE Transactions on Image Processing*, 16:229–240, 2007.

[105] K.A. Patwardhan, G. Sapiro, and M. Bertalmio. Video inpainting of occluding and occluded objects. *IEEE International Conference on Image Processing*, 2:69–72, 2005.

[106] K.A. Patwardhan, G. Sapiro, and M. Bertalmio. Video inpainting under constrained camera motion. *IEEE Transactions on Image Processing*, 16(2):545–553, 2007.

[107] D. Pavić, V. Schönefeld, and L. Kobbelt. Interactive image completion with perspective correction. *The Visual Computer*, 22(9):671–681, 2006.

[108] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

[109] P. Perona, T. Shiota, and J. Malik. Anisotropic diffusion. *B.M. ter Haar Romeny (Ed.), Geometry Driven Diffusion in Computer Vision*, Chapter 2:72–92, 1994.

[110] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C.* Cambridge University Press, USA, 2002.

[111] A. Rap, L. Elliott, D. B. Ingham, D. Lesnic, and X. Wen. The inverse source problem for the variable coefficients convection-diffusion equation. *Inverse Problems in Science and Engineering*, 15(5):413–440, 2007.

[112] A. Reusken and M. Soemers. On the robustness of a multigrid method for anisotropic reaction-diffusion problems. *Computing*, 80:299–317, 2007.

[113] F. Riesz and B. SZ.-Nagy. *Functional Analysis.* Dover Publications Inc., Reprinted edition, 1991.

[114] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60(1-4):259–268, 1992.

[115] W. Rudin. *Principles of Mathematical Analysis.* McGraw-Hill Science/Engineering/Math; 3rd edition, USA, 1976.

[116] Y. Saad. *Iterative Methods for Sparse Linear Systems.* SIAM, 2nd edition, USA, 2003.

[117] G. Sapiro and D. L. Ringach. Anisotropic diffusion of multivalued images with applications to color filtering. *IEEE Transactions on Image Processing*, 5(11):1582–1586, 1996.

[118] J. Savage and K. Chen. On multigrids for solving a class of improved total variation based staircasing reduction models. *Image Processing Based On Partial Differential Equations, eds. X.-C. Tai, K.-A. Lie, T. F. Chan and S. Osher*, 82:69–94, 2006.

[119] J. Savage and K. Chen. An improved and accelerated non-linear multigrid method for total-variation denoising. *International Journal of Computer Mathematics*, 82(8):1001–1015(15), August 2005.

[120] O. Scherzer. The use of Morozov's discrepancy principle for Tikhonov regularization for solving nonlinear ill-posed problems. *Computing*, 51(1):45–60, 1993.

[121] J. Shen and T. F. Chan. Mathematical models for local nontexture inpaintings. *SIAM Journal on Applied Mathematics*, 62(3):1019–1043, 2002.

[122] J. Shen and T. F. Chan. Variational image inpainting. *Communications on Pure and Applied Mathematics*, 58(5):579–619, 2005.

[123] J. Shen and T. F. Chan. Nontexture inpainting by curvature-driven diffusions. *Journal of Visual Communication and Image Representation*, 12(4):436–449, December 2001.

[124] J. Shen, X. Jin, Z. Xiaogang, C. Zhou, and C. Wang. Gradient based image completion by solving the Poisson equation. *Computers and Graphics*, 31(1):119–126, 2007.

[125] J. Shen, S. H. Kang, and T. F. Chan. Euler's elastica and curvature-based inpainting. *SIAM Journal on Applied Mathematics*, 63(2):564–592, 2003.

[126] T. K. Shih, R.-C. Chang, and Y.-P. Chen. Motion picture inpainting on aged films. *MULTIMEDIA '05: Proceedings of the 13th Annual ACM International Conference on Multimedia*, pages 319–322, 2005.

[127] T. K. Shih, N. C. Tang, W.-S. Yeh, T.-J. Chen, and W. Lee. Video inpainting and implant via diversified temporal continuations. *MULTIMEDIA '06: Proceedings of the 14th Annual ACM International Conference on Multimedia*, pages 133–136, 2006.

[128] T. Shiratori, Y. Matsushita, S. B. Kang, and X. Tang. Video completion by motion field transfer. *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 411–418, 2006.

[129] P. Smereka. Semi-implicit level set methods for curvature and surface diffusion motion. *Journal of Scientific Computing*, 19(1-3):439–456, 2003.

[130] N. Sochen, R. Kimmel, and R. Malladi. A general framework for low level vision. *IEEE Transactions on Image Processing*, 7(3):310–318, 1998.

[131] B. Song. Topics in Variational PDE Image Segmentation, Inpainting and Denoising, Ph.D. Thesis. *CAM Report 03-27*, 2003.

[132] R. S. Spitareli, R. March, and D. Arena. A multigrid finite-difference method for the solution of Euler equations of the variational image segmentation. *Applied Numerical Mathematics*, 39:181–189, 2001.

[133] M. Sturmer, H. Kostler, and U. Rude. A fast full multigrid solver for applications in image processing. *Numerical Linear Algebra with Applications*, 15:187–200, 2008.

[134] J. Sun, L. Yuan, J. Jia, and H-Y. Shum. Image completion with structure propagation. *ACM Trans. Graph.*, 24(3):861–868, 2005.

[135] E. Tadmor. Separation and decomposition of scales in signals and images. *Plenary talk, FoCM, City University of Hong Kong. (preprint)*, 2008.

[136] Z. Tauber, Z.-N. Li, and M. S. Drew. Review and preview: disocclusion by inpainting for image-based rendering. *IEEE Transactions on Systems, Man. and Cybernetics*, 37(4):527–540, 2007.

[137] J. W. Thomas. *Numerical Partial Differential Equations - Finite Difference Methods.* Springer-Verlag, New York, 1995.

[138] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-posed Problems.* Wiston and Sons, Washington, D.C., 1977.

[139] U. Trottenberg, C. Oosterlee, and A. Schuller. *Multigrid.* Academic Press; 1st edition, Orlando, 2001.

[140] D. Tschumperle and R. Deriche. Vector-valued image regularization with PDEs: a common framework for different applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):506–517, 2005.

[141] P. Vanek, J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56:179–196, 1996.

[142] M. V. Venkatesh, S.-C. Cheung, and J. Zhao. Efficient object-based video inpainting. *Pattern Recognition Letters*, 30(2):168–179, 2009.

[143] L. A. Vese and S. Osher. Numerical methods for p-harmonic flows and applications to image processing. *SIAM Journal on Numerical Analysis*, 40(6):2085–2104, 2002.

[144] C. R. Vogel. *Computational Methods for Inverse Problems.* SIAM, 1st edition, Philadelphia, PA, USA, 2002.

[145] C. R. Vogel and M. E. Oman. Iterative methods for total variation denoising. *SIAM Journal on Scientific Computing*, 17(1):227–238, 1996.

[146] B. P. Vollmayr-Lee and A. D. Rutenberg. Fast and accurate coarsening simulation with an unconditionally stable time step. *Phys. Rev. E*, 68(6):066703, 2003.

[147] L.-Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 479–488, 2000.

[148] J. Weickert. *Anisotropic Diffusion in Image Processing.* Teuber Verlag, Stuttgart, 1998.

[149] J. Weickert and B. Benhamouda. Why the Perona–Malik filter works? *Technical Report DIKU-TR-97/22, Department of Computer Sciences, Copenhagen Denmark*, 1997.

[150] J. Weickert and T. Brox. Diffusion and regularization of vector and matrix-valued images. *Technical Report, Preprint 58, Universitat des Saarlandes*, 2002.

[151] Y. Wexler, E. Shechtman, and M. Irani. Space-time completion of video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):463–476, 2007.

[152] R. Wienands and W. Joppich. *Practical Fourier Analysis for Multigrid Methods.* Chapman and Hall/CRC, Florida, USA, 2005.

[153] S. Wise, J. Kim, and J. Lowengrub. Solving the regularized, strongly anisotropic Cahn-Hilliard equation by an adaptive nonlinear multigrid method. *Journal of Computational Physics*, 226:414–446, 2007.

[154] A. P. Witkin. Scale-space filtering. *Proc. 8th Int. Joint Conf. Art. Intell.*, pages 1019–1022, 1983.

[155] C. L. Wu, J. S. Deng, W. M. Zhu, and F. L. Chen. Inpainting images on implicit surfaces. *Pacific Graphics*, pages 142–144, 2005.

[156] H. Yamauchi, J. Haber, and H.-P. Seidel. Image restoration using multiresolution texture synthesis and image inpainting. *Computer Graphics International Conference*, pages 120–125, 2003.

[157] Y.-L. You and M. Kaveh. Fourth-order partial differential equations for noise removal. *IEEE Transactions on Image Processing*, 9(10):1723–1730, 2000.

[158] D. M. Young. *Iterative Solution of Large Linear Systems.* Academic Press, New York, 1971.

[159] W. Zhu and T. F. Chan. Image denoising using mean curvature. *Unpublished article (http://www.math.nyu.edu/~wzhu/)*, 2008.