A Fast Augmented Lagrangian Method for Euler's Elastica Model

Conference Paper *in* Numerical Mathematics Theory Methods and Applications · May 2011

DOI: 10.1007/978-3-642-24785-9_13 · Source: DBLP

4 authors:

Yuping Duan
Nanyang Technological University
28 PUBLICATIONS **183** CITATIONS
SEE PROFILE

Yu Wang
9 PUBLICATIONS **58** CITATIONS
SEE PROFILE

Xue-Cheng Tai
University of Bergen
243 PUBLICATIONS **6,509** CITATIONS
SEE PROFILE

Jooyoung Hahn
AVL LIST GMBH
25 PUBLICATIONS **272** CITATIONS
SEE PROFILE

A FAST ALGORITHM FOR EULER'S ELASTICA MODEL USING AUGMENTED LAGRANGIAN METHOD*

XUE-CHENG TAI†, JOOYOUNG HAHN‡, AND GINMO JASON CHUNG§

Abstract. Minimization of functionals related to Euler's elastica energy has a wide range of applications in computer vision and image processing. A high order nonlinear partial differential equation (PDE) needs to be solved and the gradient descent method usually takes high computational cost. In this paper, we propose a fast and efficient numerical algorithm to solve minimization problems related to the Euler's elastica energy and show applications to variational image denoising, image inpainting, and image zooming. We reformulate the minimization problem as a constrained minimization problem, followed by an operator splitting method and relaxation. The proposed constrained minimization problem is solved by using an augmented Lagrangian approach. Numerical tests on real and synthetic cases are supplied to demonstrate the efficiency of our method.

Key words. Euler's elastica, image inpainting, image denoising, image zooming, constrained minimization, augmented Lagrangian method

1. Introduction. Euler's elastica is defined as planar curves γ that minimize energy functionals of the form

$$E(\gamma) = \int_\gamma (a + b\kappa^2) ds, \tag{1.1}$$

where κ is the curvature of the curve, s is arc length, $a > 0$, and $b > 0$. Such curves have a number of interesting applications in elasticity, computer graphics and image processing. In modeling a thin beam, the desired shape of the beam is the curve minimizing its internal strain energy or the integral of curvature squared over arc length. In [32] the derivation of the curve of least energy passing through two points with prescribed orientation is given. Kallay [33,34] gives a simplified derivation of his result and solves the problem with prescribed length. In [9, 10] the authors propose to find piecewise linear curves with small segment that minimize an energy functional defined in terms of the segment length and vertex angles of the polygonal spline approximations.

One of the difficult problems arising in image processing is segmentation with depth: segmenting an image into objects which should be ordered according to their depth in the scene. To solve it, Mumford, Nitzberg, and Shiota [42] propose to look for a continuation curve Γ which minimizes (1.1); see also [24, 25].

Following the work [42] and observing the importance of level lines for image representation, the authors in [39] adapt their variational continuation framework [42] to the level line structures. They propose a variational formulation for the recovery of the missing parts of a grey level image. They refer to this interpolation process

*The research is supported by MOE (Ministry of Education) Tier II project T207N2202 and IDM project NRF2007IDMIDM002-010.

†Mathematics Institute, University of Bergen, Norway (xctai@ntu.edu.sg).

‡Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore. Jooyoung Hahn (joo.hahn@uni-graz.at), who is currently in Institute of Mathematics and Scientific Computing in University of Graz, Austria, has been supported by the Austrian Science Fund (FWF) under the START-Program Y305 "Interfaces and Free Boundaries" and the SFB "Mathematical Optimization and Its Applications in Biomedical Sciences" since November 2010.

§Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore (JCHUNGG@ntu.edu.sg).

as *disocclusion*. After omitting the angular terms and taking a domain $\widetilde{\mathcal{D}}$ slightly bigger than the inpainting domain $\mathcal{D}$ such that $\mathcal{D} \subset\subset \widetilde{\mathcal{D}}$, their minimization criterion becomes

$$E = \int_{-\infty}^{\infty} \sum_{\Gamma \in F_\lambda} \int_\Gamma (a + b|\kappa|^p)\, |d\mathcal{H}^1 d\lambda, \tag{1.2}$$

where $p \geq 1$, $\mathcal{H}^1$ denotes the one-dimensional Hausdorff measure, and the elements of F_λ are the union of completion curves and the restrictions to $\widetilde{\mathcal{D}} \setminus \mathcal{D}$ of the associated level lines.

In order to study the inpainting problem from the viewpoint of the calculus of variations, Ambrosio and Masnou [2] rewrite (1.2), assuming that the curves Γ are the level lines of a smooth function u, as

$$E(u) = \int_{-\infty}^{\infty} \left(\int_{\partial\{u \geq \lambda\} \cap \widetilde{\mathcal{D}}} (a + b|\kappa|^p)\, d\mathcal{H}^1 \right) d\lambda. \tag{1.3}$$

Using the change of variable and the coarea formula, the energy functional becomes:

$$E(u) = \int_{\widetilde{\mathcal{D}}} \left(a + b \left| \nabla \cdot \frac{\nabla u}{|\nabla u|} \right|^p \right) |\nabla u|. \tag{1.4}$$

As noted in [2], this criterion makes sense only for a certain class of smooth functions and needs to be relaxed in order to deal with more general functions. So they extend the energy functional (1.4) to the whole $L^1(\mathbb{R}^2)$:

$$E(u) = \begin{cases} \displaystyle\int_{\widetilde{\mathcal{D}}} \left(a + b \left| \nabla \cdot \frac{\nabla u}{|\nabla u|} \right|^p \right) |\nabla u| & \text{if } u \in C^2(\mathbb{R}^2), \\ \infty & \text{if } u \in L^1(\mathbb{R}^2) \setminus C^2(\mathbb{R}^2). \end{cases} \tag{1.5}$$

Then, they define the relaxed functional associated with (1.5) as:

$$\overline{E}(u) = \inf \left\{ \liminf_{h \to \infty} E(u_h) : \quad u_h \to u \in L^1 \right\}. \tag{1.6}$$

They show the existence of an optimal solution and the coincidence between E and the associated relaxed functional $\overline{E}$ on C^2 functions.

The authors in [17] derived the Euler-Lagrange equation associated with (1.4) in the case $N = 2$, $p > 1$, and proposed some explicit numerical schemes to solve the corresponding Euler Lagrangian equation. They also showed numerical results in image inpainting to explain the effect of parameters a and b in (1.4). Their resulting Euler-Lagrange equation is nonlinear fourth order, so the computational cost becomes an issue. Another approach by relaxation is reported in [5], where the authors proposed a relaxed formulation of (1.4) and the term $\frac{\nabla u}{|\nabla u|}$ is replaced with a vector field θ. Instead of solving the fourth order PDE in [17], the formulation yields a set of coupled second order PDEs.

In this paper, we propose a fast and efficient algorithm to solve the minimization problem related to Euler's Elastica energy. We shall show applications to variational image denoising, image inpainting, and image zooming with the Euler's elastica energy. First, we reformulate it as a constrained minimization problem by introducing a couple of new variables. In order to solve the constrained minimization problem more

efficiently, we shall employ an operator splitting idea combined with an augmented Lagrangian functional. Note that many fast algorithms based on dual approach to efficiently solve total variation (TV) minimization problem have been proposed since its introduction by Rudin, Osher and Fatemi [44] as a regularization criterion for image denoising. For interested readers, please refer to [14,15,27,28,46–48,51–53,55–57] and references therein. The approach here is motivated by the split-Bregman scheme of [30] and by the observations made in [48] between some dual methods and split-Bregman iterations. The purpose is to produce some fast algorithms for minimization problems related to curvatures for level curves. So far, there are not many available fast algorithms for Euler's elastica models. We are only aware of the graph cut approaches of [4, 23] and reformulation of the problem of region-segmentation with curvature regularity as an integer linear program (ILP) [45].

The organization of the rest of the paper is as follows. In Section 2, we describe our proposed algorithm. In Section 3, we explain numerical discretization of the subproblems associated with the augmented Lagrangian functional. In Section 4, we show some numerical experiments to illustrate the efficiency of our proposed algorithm.

2. Augmented Lagrangian method. In this section, we propose a fast and efficient algorithm for minimization problems related to Euler's elastica energy. We shall especially consider applications of this approach for variational image denoising, image inpainting, and image zooming problems. For clarity of presentation, we will only consider two-dimensional problems. However, it is straightforward to extend the proposed method into higher dimensional problems. For mathematical notations, we will use the standard definitions for Sobolev spaces. Following conventions, we will use boldface letters to denotes vector functions.

Given a domain Ω and a subset $\Gamma \subset \Omega \subset \mathbf{R}^2$, we assume that the following data is given:

$$u_0 : \Gamma \to [0, 1].$$

For applications to image processing, the domain Ω is normally taken as a rectangle or a grid for a rectangle domain. For different applications, the subset Γ and data u_0 have different meanings. For image denoising, u_0 is a noisy image and $\Gamma = \Omega$. For image inpainting, we have an inpainting domains $\mathcal{D} \subset \Omega$ where the image data is missing or degraded. The data u_0 is a given image in $\Gamma = \Omega \setminus \mathcal{D}$. For this application, the values of u_0 on boundaries of $\mathcal{D}$ need to be propagated into the inpainting domain via minimization of the Euler's elastica energy. For image zooming, we have a given image $\bar{u}_0$ whose size is $[1, M_1] \times [1, M_2]$. For a fixed ratio $r \in \mathbf{N}$, we take the image domain $\Omega = [1, r(M_1 - 1) + 1] \times [1, r(M_2 - 1) + 1]$ and define the set of points $\Gamma = \{(i, j) \in \Omega \mid i \equiv 1 \bmod r,\ j \equiv 1 \bmod r\}$. From the values of u_0 on Γ assigned by $\bar{u}_0$, we need to construct an enlarged images defined on Ω. The image values on $\Omega \setminus \Gamma$ are interpolated via Euler's elastica energy.

In order to use Euler's elastica energy minimization for the above mentioned applications, we proposed to minimize the following functional:

$$\int_\Omega \left(a + b \left(\nabla \cdot \frac{\nabla u}{|\nabla u|} \right)^2 \right) |\nabla u| + \frac{\eta}{s} \int_\Gamma |u - u_0|^s, \tag{2.1}$$

where u_0 is the given data, $\eta > 0$, and $s \geq 1$. In image denoising, the choice of s is determined by the type of noise found in u_0: e.g. $s = 2$ for Gaussian white noise and $s = 1$ for salt-and-pepper noise. For image inpainting and image zooming, we use

$s = 1$ to preserve the contrast of a given image. It is also possible to minimize the following energy functional for image inpainting:

$$\int_{\mathcal{D}} \left(a + b \left(\nabla \cdot \frac{\nabla u}{|\nabla u|} \right)^2 \right) |\nabla u| \quad \text{with} \quad u|_{\partial \mathcal{D}} = u_0, \tag{2.2}$$

where $\mathcal{D} \subsetneq \Omega$ is the inpainting domain and $\partial \mathcal{D}$ is the boundary of $\mathcal{D}$. If the size of $\mathcal{D}$ is much smaller than Ω, this approach may give faster numerical convergence than using (2.1) in image inpainting. In this paper, for the sake of simplicity, we will use the same model given in (2.1) for all applications.

Now, we propose to cast the functional (2.1) into a constrained minimization problem by introducing two variables $\mathbf{p}$ and $\mathbf{n}$:

$$\mathbf{p} = \nabla u \quad \text{and} \quad \mathbf{n} = \frac{\nabla u}{|\nabla u|}.$$

The last constraint above can be reformulated as $|\mathbf{p}|\mathbf{n} = \mathbf{p}$. We could use Lagrangian or augmented Lagrangian method to solve minimization problem (2.1) under these constraints. The corresponding augmented Lagrangian functional is:

$$\begin{aligned} \mathcal{L}\left(u, \mathbf{p}, \mathbf{n}; \lambda_1, \boldsymbol{\lambda}_2\right) = & \int_\Omega \left(a + b(\nabla \cdot \mathbf{n})^2\right) |\mathbf{p}| + \frac{\eta}{s} \int_\Gamma |u - u_0|^s \\ & + \frac{r_1}{2} \int_\Omega \left||\mathbf{p}|\mathbf{n} - \mathbf{p}\right|^2 + \int_\Omega \boldsymbol{\lambda}_1 \cdot (|\mathbf{p}|\mathbf{n} - \mathbf{p}). \\ & + \frac{r_2}{2} \int_\Omega |\mathbf{p} - \nabla u|^2 + \int_\Omega \boldsymbol{\lambda}_2 \cdot (\mathbf{p} - \nabla u). \end{aligned} \tag{2.3}$$

One could use some iterative algorithms to find a point which satisfies the first order condition. However, we shall introduce some operator splitting ideas and use some special penalization techniques to get an iterative algorithm that is practically simple to implement and very fast.

First, we note that the following result follows easily from the well-known Hölder inequality:

LEMMA 2.1. *Given two vectors* $\mathbf{n} \neq 0, \mathbf{p} \neq 0$. *If they satisfy*

$$|\mathbf{n}| \leq 1, \quad |\mathbf{p}| = \mathbf{n} \cdot \mathbf{p}, \tag{2.4}$$

then we have

$$\mathbf{n} = \frac{\mathbf{p}}{|\mathbf{p}|}. \tag{2.5}$$

In order to obtain an efficient algorithm, we shall use an operator splitting idea by introducing a new variable $\mathbf{m}$ with the constraint $\mathbf{m} = \mathbf{n}$. As a consequence of Lemma 2.1, it is easy to see that minimization of the functional in (2.1) is equivalent to the following minimization problem:

$$\begin{aligned} \min_{v,u,\mathbf{m},\mathbf{p},\mathbf{n}} & \int_\Omega \left(a + b(\nabla \cdot \mathbf{n})^2\right) |\mathbf{p}| + \frac{\eta}{s} \int_\Gamma |v - u_0|^s \\ & \text{with} \quad v = u, \quad \mathbf{p} = \nabla u, \quad \mathbf{n} = \mathbf{m}, \quad |\mathbf{p}| = \mathbf{m} \cdot \mathbf{p}, \quad |\mathbf{m}| \leq 1. \end{aligned} \tag{2.6}$$

When $s \neq 2$ or $\Gamma \subsetneq \Omega$, we also need to introduce another variable v to split the nonlinearity. In case that $s = 2$ and $\Gamma = \Omega$, this new variable v is not needed and the modifications for the resulting algorithm is easily obtained by slightly modifying the proposed algorithm. For the case $s = 2$ and $\Gamma = \Omega$, we refer to [48, 52, 54] for some more details about how to modify the resulting algorithm.

Before we show how to solve the proposed constrained minimization (2.6), let us explain about the variable $\mathbf{m}$. The use of $\mathbf{m}$ with $|\mathbf{m}| \leq 1$ can be viewed as relaxation, similar to the relaxation approach proposed in [5]. Moreover, the constraint $|\mathbf{m}| \leq 1$ is very crucial to prevent the unboundedness of $\mathbf{m}$ when $\mathbf{p} = \mathbf{0}$. In order to impose the constraint

$$|\mathbf{m}| \leq 1 \text{ a.e in } \Omega,$$

we define a set

$$\mathcal{R} = \{\mathbf{m} \in \mathbf{L}^2(\Omega) \mid |\mathbf{m}| \leq 1 \text{ a.e. in } \Omega\}$$

and a characteristic function $\delta_{\mathcal{R}}(\cdot)$ on $\mathcal{R}$

$$\delta_{\mathcal{R}}(\mathbf{m}) = \begin{cases} 0 & \mathbf{m} \in \mathcal{R}, \\ +\infty & \text{otherwise.} \end{cases}$$

As indicated in Lions-Mercier [35], the following minimization problem with a given $\mathbf{m}_0$

$$\min_{\mathbf{m}} \int_\Omega |\mathbf{m} - \mathbf{m}_0|^2 + \delta_{\mathcal{R}}(\mathbf{m})$$

has an explicit solution given by

$$\mathbf{m} = \text{proj}_{\mathcal{R}}(\mathbf{m}_0) = \begin{cases} \mathbf{m}_0 & |\mathbf{m}_0| \leq 1, \\ \mathbf{m}_0/|\mathbf{m}_0| & \text{otherwise.} \end{cases} \tag{2.7}$$

In order to efficiently solve the proposed constrained optimization problem (2.6), we define the following augmented Lagrangian functional:

$$\begin{aligned} \mathcal{L}(v, u, \mathbf{m}, \mathbf{p}, \mathbf{n}; \lambda_1, \boldsymbol{\lambda}_2, \lambda_3, \boldsymbol{\lambda}_4) = & \int_\Omega \left(a + b(\nabla \cdot \mathbf{n})^2\right) |\mathbf{p}| + \frac{\eta}{s} \int_\Gamma |v - u_0|^s \\ & + r_1 \int_\Omega (|\mathbf{p}| - \mathbf{m} \cdot \mathbf{p}) + \int_\Omega \lambda_1 (|\mathbf{p}| - \mathbf{m} \cdot \mathbf{p}) \\ & + \frac{r_2}{2} \int_\Omega |\mathbf{p} - \nabla u|^2 + \int_\Omega \boldsymbol{\lambda}_2 \cdot (\mathbf{p} - \nabla u) \\ & + \frac{r_3}{2} \int_\Omega (v - u)^2 + \int_\Omega \lambda_3 (v - u) \\ & + \frac{r_4}{2} \int_\Omega |\mathbf{n} - \mathbf{m}|^2 + \int_\Omega \boldsymbol{\lambda}_4 \cdot (\mathbf{n} - \mathbf{m}) + \delta_{\mathcal{R}}(\mathbf{m}), \end{aligned} \tag{2.8}$$

where λ_1, $\boldsymbol{\lambda}_2$, λ_3, and $\boldsymbol{\lambda}_4$ are Lagrange multipliers and r_1, r_2, r_3, and r_4 are positive penalty parameters. Due to the constraint $|\mathbf{m}| \leq 1$, we have that

$$|\mathbf{p}| - \mathbf{m} \cdot \mathbf{p} \geq 0, \text{ a.e. in } \Omega.$$

That is the reason why we do not use L^2 penalization for the term multiplied by r_1.

Table 2.1: Augmented Lagrangian method for the Euler's elastica model.

1. Initialization: v^0, u^0, $\mathbf{m}^0$, $\mathbf{p}^0$, $\mathbf{n}^0$, and λ_1^0, $\boldsymbol{\lambda}_2^0$, λ_3^0, $\boldsymbol{\lambda}_4^0$.
2. For $k \geq 1$, compute an approximate minimizer $\left(v^k, u^k, \mathbf{m}^k, \mathbf{p}^k, \mathbf{n}^k\right)$ of the augmented Lagrangian functional with the fixed Lagrange multipliers λ_1^{k-1}, $\boldsymbol{\lambda}_2^{k-1}$, λ_3^{k-1}, and $\boldsymbol{\lambda}_4^{k-1}$:
$$\left(v^k, u^k, \mathbf{m}^k, \mathbf{p}^k, \mathbf{n}^k\right) \approx \arg\min \mathcal{L}\left(v, u, \mathbf{m}, \mathbf{p}, \mathbf{n}; \lambda_1^{k-1}, \boldsymbol{\lambda}_2^{k-1}, \lambda_3^{k-1}, \boldsymbol{\lambda}_4^{k-1}\right). \tag{2.9}$$
3. Update Lagrange multipliers
$$\lambda_1^k = \lambda_1^{k-1} + r_1\left(|\mathbf{p}^k| - \mathbf{m}^k \cdot \mathbf{p}^k\right), \tag{2.10}$$
$$\boldsymbol{\lambda}_2^k = \boldsymbol{\lambda}_2^{k-1} + r_2\left(\mathbf{p}^k - \nabla u^k\right), \tag{2.11}$$
$$\lambda_3^k = \lambda_3^{k-1} + r_3\left(v^k - u^k\right), \tag{2.12}$$
$$\boldsymbol{\lambda}_4^k = \boldsymbol{\lambda}_4^{k-1} + r_4\left(\mathbf{n}^k - \mathbf{m}^k\right). \tag{2.13}$$
4. Measure the relative residuals and stop iteration of k if they are smaller than ϵ_r.

In convex optimization, it is known that one of saddle points for the augmented Lagrangian functional will give a minimizer for the constrained minimization problem. In practice, an iterative algorithm is used to find a point which satisfies the first order condition of (2.8). This iterative scheme is shown in Table 2.1. We initialize Lagrange multipliers λ_1^0, $\boldsymbol{\lambda}_2^0$, λ_3^0, and $\boldsymbol{\lambda}_4^0$ and the variables v^0, u^0, $\mathbf{m}^0$, $\mathbf{p}^0$, and $\mathbf{n}^0$ as zero. Note that it is possible to change the initialization to obtain faster numerical results in each application. For instance, the variables v^0, u^0, $\mathbf{m}^0$, and $\mathbf{p}^0$ can be initialized to a given noisy data u_0 in image denoising as follows:

$$u^0 = u_0, \quad \mathbf{p}^0 = \nabla u_0, \quad \text{and} \quad \mathbf{n}^0 = \mathbf{m}^0 = \frac{\nabla u_0}{|\nabla u_0|} \quad \text{on} \quad \Omega.$$

Even though more optimized initialization may give faster numerical results, we simply use zero to initialize all variables and Lagrange multipliers to make the comparisons with other methods simpler.

For $k \geq 1$, an alternating minimization method is used to approximately find a minimizer $\left(v^k, u^k, \mathbf{m}^k, \mathbf{p}^k, \mathbf{n}^k\right)$ of the functional $\mathcal{L}\left(v, u, \mathbf{m}, \mathbf{p}, \mathbf{n}; \lambda_1, \boldsymbol{\lambda}_2, \lambda_3, \boldsymbol{\lambda}_4\right)$ with the fixed Lagrange multipliers $\lambda_1 = \lambda_1^{k-1}$, $\boldsymbol{\lambda}_2 = \boldsymbol{\lambda}_2^{k-1}$, $\lambda_3 = \lambda_3^{k-1}$, and $\boldsymbol{\lambda}_4 = \boldsymbol{\lambda}_4^{k-1}$ and the previous variables u^{k-1}, v^{k-1}, $\mathbf{p}^{k-1}$, $\mathbf{n}^{k-1}$, and $\mathbf{m}^{k-1}$. The procedure is given in Table 2.2. First of all, we initialize the variables: $\tilde{v}^0 = v^{k-1}$, $\tilde{u}^0 = u^{k-1}$, $\tilde{\mathbf{m}}^0 = \mathbf{m}^{k-1}$, $\tilde{\mathbf{p}}^0 = \mathbf{p}^{k-1}$, and $\tilde{\mathbf{n}}^0 = \mathbf{n}^{k-1}$. For $l = 1, \cdots, L$, we find minimizers $\tilde{v}^l$, $\tilde{u}^l$, $\tilde{\mathbf{m}}^l$, $\tilde{\mathbf{p}}^l$, $\tilde{\mathbf{n}}^l$ in

Table 2.2: Alternating minimization method to solve the problem (2.9) in Table 2.1.

1. Initialization: $\tilde{v}^0 = v^{k-1}$, $\tilde{u}^0 = u^{k-1}$, $\tilde{\mathbf{m}}^0 = \mathbf{m}^{k-1}$, $\tilde{\mathbf{p}}^0 = \mathbf{p}^{k-1}$, and $\tilde{\mathbf{n}}^0 = \mathbf{n}^{k-1}$.
2. For $l = 1, \cdots, L$ and fixed Lagrange multipliers $\lambda_1 = \lambda_1^{k-1}$, $\boldsymbol{\lambda}_2 = \boldsymbol{\lambda}_2^{k-1}$, $\lambda_3 = \lambda_3^{k-1}$, and $\boldsymbol{\lambda}_4 = \boldsymbol{\lambda}_4^{k-1}$, solve the following problems alternatively:

$$\tilde{v}^l = \arg\min \mathcal{L}\left(v, \tilde{u}^{l-1}, \tilde{\mathbf{m}}^{l-1}, \tilde{\mathbf{p}}^{l-1}, \tilde{\mathbf{n}}^{l-1}; \lambda_1, \boldsymbol{\lambda}_2, \lambda_3, \boldsymbol{\lambda}_4\right), \tag{2.14}$$

$$\tilde{u}^l = \arg\min \mathcal{L}\left(\tilde{v}^l, u, \tilde{\mathbf{m}}^{l-1}, \tilde{\mathbf{p}}^{l-1}, \tilde{\mathbf{n}}^{l-1}; \lambda_1, \boldsymbol{\lambda}_2, \lambda_3, \boldsymbol{\lambda}_4\right), \tag{2.15}$$

$$\tilde{\mathbf{m}}^l = \arg\min \mathcal{L}\left(\tilde{v}^l, \tilde{u}^l, \mathbf{m}, \tilde{\mathbf{p}}^{l-1}, \tilde{\mathbf{n}}^{l-1}; \lambda_1, \boldsymbol{\lambda}_2, \lambda_3, \boldsymbol{\lambda}_4\right), \tag{2.16}$$

$$\tilde{\mathbf{p}}^l = \arg\min \mathcal{L}\left(\tilde{v}^l, \tilde{u}^l, \tilde{\mathbf{m}}^l, \mathbf{p}, \tilde{\mathbf{n}}^{l-1}; \lambda_1, \boldsymbol{\lambda}_2, \lambda_3, \boldsymbol{\lambda}_4\right), \tag{2.17}$$

$$\tilde{\mathbf{n}}^l = \arg\min \mathcal{L}\left(\tilde{v}^l, \tilde{u}^l, \tilde{\mathbf{m}}^l, \tilde{\mathbf{p}}^l, \mathbf{n}; \lambda_1, \boldsymbol{\lambda}_2, \lambda_3, \boldsymbol{\lambda}_4\right). \tag{2.18}$$

3. $\left(v^k, u^k, \mathbf{m}^k, \mathbf{p}^k, \mathbf{n}^k\right) = \left(\tilde{v}^L, \tilde{u}^L, \tilde{\mathbf{m}}^L, \tilde{\mathbf{p}}^L, \tilde{\mathbf{n}}^L\right)$.

the subproblems from (2.14) to (2.18) by minimizing the following energy functionals:

$$\mathcal{E}_1(v) = \frac{\eta}{s}\int_\Gamma |v - u_0|^s + \int_\Omega \frac{r_3}{2}\left(v - \tilde{u}^{l-1}\right)^2 + \lambda_3 v, \tag{2.19}$$

$$\mathcal{E}_2(u) = \int_\Omega \frac{r_2}{2}\left|\tilde{\mathbf{p}}^{l-1} - \nabla u\right|^2 - \boldsymbol{\lambda}_2 \cdot \nabla u + \frac{r_3}{2}\left(\tilde{v}^l - u\right)^2 + \lambda_3(-u), \tag{2.20}$$

$$\mathcal{E}_3(\mathbf{m}) = \delta_{\mathcal{R}}(\mathbf{m}) + \int_\Omega \frac{r_4}{2}\left|\tilde{\mathbf{n}}^{l-1} - \mathbf{m}\right|^2 - \boldsymbol{\lambda}_4 \cdot \mathbf{m} - (r_1 + \lambda_1)\mathbf{m} \cdot \tilde{\mathbf{p}}^{l-1}, \tag{2.21}$$

$$\mathcal{E}_4(\mathbf{p}) = \int_\Omega \left(a + b\left(\nabla \cdot \tilde{\mathbf{n}}^{l-1}\right)^2\right)|\mathbf{p}| + (r_1 + \lambda_1)\left(|\mathbf{p}| - \tilde{\mathbf{m}}^l \cdot \mathbf{p}\right) + \frac{r_2}{2}\left|\mathbf{p} - \nabla \tilde{u}^l\right|^2 + \boldsymbol{\lambda}_2 \cdot \mathbf{p}, \tag{2.22}$$

$$\mathcal{E}_5(\mathbf{n}) = \int_\Omega b(\nabla \cdot \mathbf{n})^2\left|\tilde{\mathbf{p}}^l\right| + \frac{r_4}{2}\left|\mathbf{n} - \tilde{\mathbf{m}}^l\right|^2 + \boldsymbol{\lambda}_4 \cdot \mathbf{n}. \tag{2.23}$$

After L^{th} iterations, we update $\left(v^k, u^k, \mathbf{m}^k, \mathbf{p}^k, \mathbf{n}^k\right) = \left(\tilde{v}^L, \tilde{u}^L, \tilde{\mathbf{m}}^L, \tilde{\mathbf{p}}^L, \tilde{\mathbf{n}}^L\right)$. Before we get into more details of this algorithm, we would like to make the following comments:

- For $s = 1$ or 2, the minimization problems for $\mathcal{E}_1, \mathcal{E}_3$, and $\mathcal{E}_4$, c.f. (2.14), (2.16),and (2.17) in Table 2.2, can be done by some simple arithmetic calculations and thresholding at each grid point. There is no need to solve any equations.
- The minimization problems for $\mathcal{E}_2$ and $\mathcal{E}_5$, c.f. (2.15) and (2.18), need to solve a linear equation over the whole domain Ω. For image processing, the mesh is uniform, FFT and AOS [36, 50] schemes can be used to solve these two subproblems with very low cost.

In Section 3.2, we show the details for the implementation for the algorithms given in (2.19) to (2.23). Especially, we shall present the details in a discrete setting. This will make it more clear for implementations. In this paper, we numerically observe that $L = 1$ is enough to obtain desirable results in image denoising, image inpainting, and image zooming.

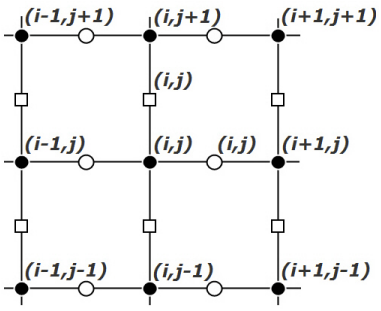


Fig. 3.1: The rule of indexing variables in the augmented Lagrangian functional (2.8): u, v, λ_1, and λ_3 are defined on $\bullet$-nodes. The first and second component of $\mathbf{p}$, $\mathbf{n}$, $\boldsymbol{\lambda}_2$, and $\boldsymbol{\lambda}_4$ are defined on $\circ$-nodes and $\square$-nodes, respectively.

3. Numerical discretization. We use a staggered grid system as in Figure 3.1 to solve the energy functional minimization from (2.19) to (2.23) and update the Lagrange multipliers from (2.10) to (2.13). Unknowns u, v, λ_1, and λ_3 are defined on $\bullet$-nodes. The first and second component of $\mathbf{p}$, $\mathbf{n}$, $\boldsymbol{\lambda}_2$, and $\boldsymbol{\lambda}_4$ are defined on $\circ$-nodes and $\square$-nodes, respectively. First of all, we introduce notations in Section 3.1. In Sections3.2 and 3.3, we shall present the details on how to solve the sub-minimization problems given in Table 2.2.

3.1. Notation. Let $\Omega = [1, N_1] \times [1, N_2]$ be a set of $N_1 N_2$ points in $\mathbf{R}^2$. For simplicity, we denote two inner product vector spaces:

$$X = \mathbf{R}^{N_1 N_2} \quad \text{and} \quad Y = X \times X.$$

For a given $(i, j) \in [1, N_1] \times [1, N_2]$, we see that

$$u \in X, \quad u(i, j) \in \mathbf{R} \quad \text{and} \quad p \in Y, \quad p(i, j) = (p_1(i, j), p_2(i, j)) \in \mathbf{R}^2,$$

we equip the standard Euclidean inner products as follows:

$$(u, v)_X \equiv \sum_{i,j} u(i, j)v(i, j) \quad \text{and} \quad (p, q)_Y \equiv (p_1, q_1)_X + (p_2, q_2)_X.$$

Note that the induced norms $|| \cdot ||_V$ are the ℓ_2-norm on vector spaces $V = X$ and Y.

The discrete backward and forward differential operators for $u \in X$ are defined

with the periodic condition:

$$\partial_1^- u(i,j) \equiv \begin{cases} u(i,j) - u(i-1,j), & 1 < i \le N_1, \\ u(1,j) - u(N_1,j), & i = 1, \end{cases}$$
$$\partial_2^- u(i,j) \equiv \begin{cases} u(i,j) - u(i,j-1), & 1 < j \le N_2, \\ u(i,1) - u(i,N_2), & j = 1, \end{cases}$$
$$\partial_1^+ u(i,j) \equiv \begin{cases} u(i+1,j) - u(i,j), & 1 \le i < N_1, \\ u(1,j) - u(N_1,j), & i = N_1, \end{cases}$$
$$\partial_2^+ u(i,j) \equiv \begin{cases} u(i,j+1) - u(i,j), & 1 \le j < N_2, \\ u(i,1) - u(i,N_2), & j = N_2, \end{cases}$$

We also define the discrete forward(+) and backward(−) gradient operator $\nabla^{\pm} : X \to Y$:

$$\nabla^{\pm} u(i,j) \equiv \left(\partial_1^{\pm} u(i,j), \partial_2^{\pm} u(i,j)\right).$$

Considering inner products on X and Y, the corresponding discrete backward(−) and forward(+) adjoint operator $\mathrm{div}^{\mp} : Y \to X$ of $-\nabla^{\pm}$ is obtained:

$$\mathrm{div}^{\mp} p(i,j) \equiv \partial_1^{\mp} p_1(i,j) + \partial_2^{\mp} p_2(i,j).$$

When a variable defined on $\circ$-nodes (or $\square$-nodes) needs to be evaluated at $(i,j) \in \square$-nodes (or $\circ$-nodes), we use the average operators:

$$\begin{aligned} \mathcal{A}_{i,j}^{\square}(n_1) &= \frac{n_1(i,j+1) + n_1(i-1,j+1) + n_1(i,j) + n_1(i-1,j)}{4}, \\ \mathcal{A}_{i,j}^{\circ}(n_2) &= \frac{n_2(i+1,j) + n_2(i,j) + n_2(i+1,j-1) + n_2(i,j-1)}{4}, \end{aligned} \tag{3.1}$$

where n_1 and n_2 are defined on $\circ$-nodes and $\square$-nodes, respectively. We also use a special operator to measure the magnitude of vector $\mathbf{p} = (p_1, p_2)$ at $(i,j) \in \bullet$-nodes, where p_1 and p_2 are defined on $\circ$-nodes and $\square$-nodes, respectively:

$$|\mathcal{A}|_{i,j}^{\bullet}(\mathbf{p}) = \left(\left(\frac{p_1(i,j) + p_1(i-1,j)}{2} \right)^2 + \left(\frac{p_2(i,j) + p_2(i,j-1)}{2} \right)^2 \right)^{\frac{1}{2}}, \tag{3.2}$$

When we compute the divergence of a vector $\mathbf{n} = (n_1, n_2)$ at $(i,j) \in \bullet$-nodes, where n_1 and n_2 are defined on $\circ$-nodes and $\square$-nodes, we use the following operator:

$$\mathrm{div}_{i,j}^{\bullet}(\mathbf{n}) = n_1(i,j) - n_1(i-1,j) + n_2(i,j) - n_2(i,j-1). \tag{3.3}$$

3.2. Subproblems. In this subsection, we explain how to find the minimizers of $\mathcal{E}_i$ $(1 \le i \le 5)$ shown from (2.19) to (2.23). For simplicity of notations, we denote the fixed Lagrange multipliers in the previous $(k-1)^{\text{th}}$ iteration as $\lambda_1 = \lambda_1^{k-1}$, $\boldsymbol{\lambda}_2 = \boldsymbol{\lambda}_2^{k-1}$, $\lambda_3 = \lambda_3^{k-1}$, and $\boldsymbol{\lambda}_4 = \boldsymbol{\lambda}_4^{k-1}$.

3.2.1. Minimization of $\mathcal{E}_1(v)$ in (2.19). We denote a fixed variable $\tilde{u}^{l-1}$ as u. We note that one can combine the last two terms in $\mathcal{E}_1(v)$ into one term. Let $w = u - \frac{\lambda_3}{r_3}$. Then we have

$$\mathcal{E}_1(v) = \frac{\eta}{s} \int_{\Gamma} |v - u_0|^s + \frac{r_3}{2} \int_{\Omega} (v - w)^2 + \bar{C}_1.$$

Since $\bar{C}_1$ does not depend on v, the minimization of $\mathcal{E}_1$ can be done by minimizing the following discrete energy functional for fixed u:

$$\tilde{\mathcal{E}}_1(v) = \tilde{\mathcal{E}}_{\Omega\setminus\Gamma}(v) + \tilde{\mathcal{E}}_{\Gamma}(v),$$

where

$$\tilde{\mathcal{E}}_{\Gamma}(v) = \sum_{(i,j)\in\Gamma} \left(\frac{\eta}{s}|v(i,j) - u_0(i,j)|^s + \frac{r_3}{2}|v(i,j) - w(i,j)|^2 \right),$$
$$\tilde{\mathcal{E}}_{\Omega\setminus\Gamma}(v) = \sum_{(i,j)\in\Omega\setminus\Gamma} \frac{r_3}{2}|v(i,j) - w(i,j)|^2, \quad (i,j) \in \bullet\text{-nodes}.$$

For a grid point $(i,j) \in \Omega \setminus \Gamma$, the minimizer $v(i,j)$ is

$$v(i,j) = w(i,j).$$

For a grid point $(i,j) \in \Gamma$, the minimizer $v(i,j)$ with $s = 2$ is

$$v(i,j) = \frac{\eta u_0(i,j) + r_3 w(i,j)}{\eta + r_3}.$$

When $s = 1$, we use the soft thresholding formula to find the closed-form formula for the minimizer $v(i,j)$ for a grid point $(i,j) \in \Gamma$:

$$v(i,j) = u_0(i,j) + M(i,j)(w(i,j) - u_0(i,j)),$$

where

$$M(i,j) = \max\left\{0, 1 - \frac{\eta}{r_3|w(i,j) - u_0(i,j)|}\right\}.$$

For $s \in (0,1)$, see more details in [53] on how to find a minimizer $v(i,j)$ for a grid point $(i,j) \in \Gamma$. To sum up, the updated $\tilde{v}^l$ is obtained by the following formulas:

$$\begin{aligned} &(i,j) \notin \Gamma \quad \Rightarrow \quad \tilde{v}^l(i,j) = w(i,j), \\ &(i,j) \in \Gamma \quad \Rightarrow \quad \begin{cases} \tilde{v}^l(i,j) = \dfrac{\eta u_0(i,j) + r_3 w(i,j)}{\eta + r_3}, & \text{if } s = 2, \\ \tilde{v}^l(i,j) = u_0(i,j) + M(i,j)(w(i,j) - u_0(i,j)), & \text{if } s = 1. \end{cases} \end{aligned} \tag{3.4}$$

3.2.2. Minimization of $\mathcal{E}_2(u)$ in (2.20). We denote fixed variables $\tilde{v}^l$ and $\tilde{\mathbf{p}}^{l-1}$ as v and $\mathbf{p}$, respectively. The Euler-Lagrange equation of (2.20) yields a linear equations:

$$-r_2 \operatorname{div}^- \nabla^+ u + r_3 u = -r_2 \operatorname{div}^- \mathbf{p} - \operatorname{div}^- \boldsymbol{\lambda}_2 + r_3 v + \lambda_3. \tag{3.5}$$

Since periodic boundary condition is imposed, it can be efficiently solved by the fast Fourier transform (FFT). Using the notation in Section 3.1, the above equation can be written as:

$$-r_2(\partial_1^-\partial_1^+ + \partial_2^-\partial_2^+)u + r_3 u = g,$$

where $g = -r_2(\partial_1^- p_1 + \partial_2^- p_2) - (\partial_1^-\lambda_{21} + \partial_2^-\lambda_{22}) + r_3 v + \lambda_3$. Introducing the identity operator $\mathcal{I}f(i,j) = f(i,j)$ and shifting operators,

$$\mathcal{S}_1^{\pm} f(i,j) = f(i \pm 1, j) \quad \text{and} \quad \mathcal{S}_2^{\pm} f(i,j) = f(i, j \pm 1), \tag{3.6}$$

the discretization of the equation (3.5) at •-node is as follows:

$$\left(-r_2\left(\mathcal{S}_1^- - 2\mathcal{I} + \mathcal{S}_1^+ + \mathcal{S}_2^- - 2\mathcal{I} + \mathcal{S}_2^+\right) + r_3\right) u(i,j) = g(i,j),$$

where

$$\begin{aligned} g(i,j) = &- r_2\left(\mathcal{I} - \mathcal{S}_1^-\right) p_1(i,j) - r_2\left(\mathcal{I} - \mathcal{S}_2^-\right) p_2(i,j) \\ &- \left(\mathcal{I} - \mathcal{S}_1^-\right) \lambda_{21}(i,j) - \left(\mathcal{I} - \mathcal{S}_2^-\right) \lambda_{22}(i,j) + r_3 v(i,j) + \lambda_3(i,j). \end{aligned}$$

Now, we apply the discrete Fourier transform $\mathcal{F}$. The shifting operator is a discrete convolution and its discrete Fourier transform is the componentwise multiplication in the frequency domain. For discrete frequencies, y_i and y_j, we have

$$\mathcal{F}\mathcal{S}_1^{\pm} f(y_i, y_j) = e^{\pm\sqrt{-1}z_i}\mathcal{F}f(y_i, y_j) \quad \text{and} \quad \mathcal{F}\mathcal{S}_2^{\pm} f(y_i, y_j) = e^{\pm\sqrt{-1}z_j}\mathcal{F}f(y_i, y_j), \tag{3.7}$$

where

$$z_i = \frac{2\pi}{N_1} y_i,\ y_i = 1, \cdots, N_1 \quad \text{and} \quad z_j = \frac{2\pi}{N_2} y_j,\ y_j = 1, \cdots, N_2. \tag{3.8}$$

It yields an algebraic equation:

$$\left(-2r_2\left(\cos z_i + \cos z_j - 2\right) + r_3\right) \mathcal{F}u(y_i, y_j) = \mathcal{F}g(y_i, y_j),$$

Note that $r_3 > 0$. The discrete inverse Fourier transform $\mathcal{F}$ gives the updated $\tilde{u}^l$.

3.2.3. Minimization of $\mathcal{E}_3(\mathbf{m})$ in (2.21). We denote a fixed variable $\tilde{\mathbf{n}}^{l-1}$ and $\tilde{\mathbf{p}}^{l-1}$ as $\mathbf{n}$ and $\mathbf{p}$, respectively. In a similar way as getting the closed-form solution of (2.19), we also obtain the close-form solution of (2.21) by re-writing the minimization functional in the form:

$$\mathcal{E}_3(\mathbf{m}) = \delta_{\mathcal{R}}(\mathbf{m}) + \frac{r_4}{2}\int_\Omega \left| \frac{(r_1 + \lambda_1)\mathbf{p} + \boldsymbol{\lambda}_4}{r_4} + \mathbf{n} - \mathbf{m} \right|^2 dx + \bar{C}_3, \tag{3.9}$$

where $\bar{C}_3$ does not depend on $\mathbf{m}$. For a fixed variable $\mathbf{n}$, we use (2.7) to obtain the closed-form solution of (3.9):

$$\tilde{\mathbf{m}}^l = \mathrm{proj}_{\mathcal{R}}\left(\mathbf{z}\right), \quad \text{where} \quad \mathbf{z} \equiv \frac{(r_1 + \lambda_1)\mathbf{p} + \boldsymbol{\lambda}_4}{r_4} + \mathbf{n}. \tag{3.10}$$

Since the first and second component of $\mathbf{p}$, $\mathbf{n}$, and $\boldsymbol{\lambda}_4$ are defined at ∘-nodes and □-nodes respectively, c.f. Figure 3.1, a discretization of $\mathbf{z}$ at $(i,j) \in$ ∘-node is

$$\begin{aligned} z_1^1(i,j) &= n_1(i,j) + \frac{1}{r_4}\left(\lambda_{41}(i,j) + \left(\frac{\lambda_1(i+1,j) + \lambda_1(i,j)}{2} + r_1\right) p_1(i,j)\right), \\ z_2^1(i,j) &= \mathcal{A}_{i,j}^{\circ}(n_2) + \frac{1}{r_4}\left(\mathcal{A}_{i,j}^{\circ}(\lambda_{42}) + \left(\frac{\lambda_1(i+1,j) + \lambda_1(i,j)}{2} + r_1\right) \mathcal{A}_{i,j}^{\circ}(p_2)\right). \end{aligned}$$

A discretization of $\mathbf{z}$ at $(i,j) \in$ □-node is similarly obtained with $\mathcal{A}^{\square}(\cdot)$:

$$\begin{aligned} z_1^2(i,j) &= \mathcal{A}_{i,j}^{\square}(n_1) + \frac{1}{r_4}\left(\mathcal{A}_{i,j}^{\square}(\lambda_{41}) + \left(\frac{\lambda_1(i,j+1) + \lambda_1(i,j)}{2} + r_1\right) \mathcal{A}_{i,j}^{\square}(p_1)\right), \\ z_2^2(i,j) &= n_2(i,j) + \frac{1}{r_4}\left(\lambda_{42}(i,j) + \left(\frac{\lambda_1(i,j+1) + \lambda_1(i,j)}{2} + r_1\right) p_2(i,j)\right). \end{aligned}$$

Therefore, we have the discretization of (3.10):

$$\tilde{m}_a^l(i,j) = \mathrm{proj}_{\mathcal{R}}\left(z_1^a(i,j), z_2^a(i,j)\right), \quad \text{where} \quad a \in \{1,2\}.$$

3.2.4. Minimization of $\mathcal{E}_4(\mathbf{p})$ in (2.22). We denote fixed variables $\tilde{u}^l$, $\tilde{\mathbf{m}}^l$, and $\tilde{\mathbf{n}}^{l-1}$ as u, $\mathbf{m}$, and $\mathbf{n}$, respectively. It is easy to see that $\mathcal{E}_4$ in (2.22) can be written as

$$\begin{aligned}\mathcal{E}_4(\mathbf{p}) = &\int_\Omega \left(a + b(\nabla\cdot\mathbf{n})^2 + r_1 + \lambda_1\right)|\mathbf{p}| \\ &+ \frac{r_2}{2}\int_\Omega \left|\mathbf{p} - \left(\nabla u + \left(\frac{r_1+\lambda_1}{r_2}\right)\mathbf{m} - \frac{\boldsymbol{\lambda}_2}{r_2}\right)\right|^2 + \bar{C}_4,\end{aligned} \tag{3.11}$$

where $\bar{C}_4$ does not depend on $\mathbf{p}$. Let

$$c = a + b(\nabla\cdot\mathbf{n})^2 + r_1 + \lambda_1 \quad \text{and} \quad \mathbf{q} = \nabla u + \left(\frac{r_1+\lambda_1}{r_2}\right)\mathbf{m} - \frac{\boldsymbol{\lambda}_2}{r_2}.$$

Then, for fixed u, $\mathbf{n}$ and $\mathbf{m}$, we apply the soft threshold formula to find the closed-form solution for the minimization of (3.11):

$$\tilde{\mathbf{p}}^l(i,j) = \max\left\{0, 1 - \frac{c}{r_2|\mathbf{q}(i,j)|}\right\}\mathbf{q}(i,j). \tag{3.12}$$

According to the rule of indexing variables in Figure 3.1, the first and second component of $\mathbf{p}$, $\mathbf{n}$, and $\boldsymbol{\lambda}_2$ are defined at $\circ$-nodes and $\square$-nodes, respectively. Now, a discretization of c and $\mathbf{q}$ at $(i,j) \in \circ$-node is obtained as follows:

$$\begin{aligned}c^1(i,j) &= a + r_1 + \frac{\lambda_1(i+1,j) + \lambda_1(i,j)}{2} \\ &\quad + b\left(\frac{n_1(i+1,j) - n_1(i-1,j)}{2} + \frac{n_2(i+1,j) + n_2(i,j) - n_2(i+1,j-1) - n_2(i,j-1)}{2}\right)^2, \\ q_1^1(i,j) &= u(i+1,j) - u(i,j) + \left(\frac{r_1}{r_2} + \frac{1}{r_2}\left(\frac{\lambda_1(i+1,j)+\lambda_1(i,j)}{2}\right)\right)m_1(i,j) - \frac{1}{r_2}\lambda_{21}(i,j), \\ q_2^1(i,j) &= \frac{1}{2}\left(\frac{u(i+1,j+1)+u(i,j+1)}{2} - \frac{u(i+1,j-1)+u(i,j-1)}{2}\right) \\ &\quad + \left(\frac{r_1}{r_2} + \frac{1}{r_2}\left(\frac{\lambda_1(i+1,j)+\lambda_1(i,j)}{2}\right)\right)\mathcal{A}^\circ_{i,j}(m_2) - \frac{1}{r_2}\mathcal{A}^\circ_{i,j}(\lambda_{22}).\end{aligned}$$

Similarly, a discretization of c and $\mathbf{q}$ at $(i,j) \in \square$-node is obtained as follows:

$$\begin{aligned}c^2(i,j) &= a + r_1 + \frac{\lambda_1(i,j+1) + \lambda_1(i,j)}{2} \\ &\quad + b\left(\frac{n_1(i,j+1) + n_1(i,j) - n_1(i-1,j+1) - n_1(i-1,j)}{2} + \frac{n_2(i,j+1) - n_2(i,j-1)}{2}\right)^2, \\ q_1^2(i,j) &= \frac{1}{2}\left(\frac{u(i+1,j+1)+u(i+1,j)}{2} - \frac{u(i-1,j+1)+u(i-1,j)}{2}\right) \\ &\quad + \left(\frac{r_1}{r_2} + \frac{1}{r_2}\left(\frac{\lambda_1(i,j+1)+\lambda_1(i,j)}{2}\right)\right)\mathcal{A}^\square_{i,j}(m_1) - \frac{1}{r_2}\mathcal{A}^\square_{i,j}(\lambda_{21}). \\ q_2^2(i,j) &= u(i,j+1) - u(i,j) + \left(\frac{r_1}{r_2} + \frac{1}{r_2}\left(\frac{\lambda_1(i,j+1)+\lambda_1(i,j)}{2}\right)\right)m_2(i,j) - \frac{1}{r_2}\lambda_{22}(i,j).\end{aligned}$$

Therefore, we have the discretization of (3.11):

$$\tilde{p}_a^l(i,j) = \max\left\{0, 1 - \frac{c^a(i,j)}{r_2\sqrt{(q_1^a(i,j))^2 + (q_2^a(i,j))^2}}\right\}q_a^a(i,j), \quad \text{where} \quad a \in \{1,2\}.$$

3.2.5. Minimization of $\mathcal{E}_5(\mathbf{n})$ in (2.23). We denote fixed variables $\tilde{\mathbf{p}}^l$ and $\tilde{\mathbf{m}}^l$ as $\mathbf{p}$ and $\mathbf{m}$, respectively. The Euler-Lagrange equation of (2.23) is given by

$$-2\nabla^+(b|\mathbf{p}|\,\mathrm{div}^-\,\mathbf{n}) + r_4(\mathbf{n}-\mathbf{m}) + \boldsymbol{\lambda}_4 = 0. \tag{3.13}$$

As a uniform grid is normally used for image processing, we shall employ a frozen coefficient method to solve the above linear equation for the purpose of easier implementations. For properly chosen c, we solve the following problem iteratively for $q = 1, 2, \cdots$:

$$-c\nabla^+\left(\mathrm{div}^-\,\mathbf{n}^q\right) + r_4\mathbf{n}^q = r_4\mathbf{m} - \boldsymbol{\lambda}_4 - \nabla^+\left((c-2b|\mathbf{p}|)\,\mathrm{div}^-\,\mathbf{n}^{q-1}\right), \tag{3.14}$$

where the initial condition $\mathbf{n}^{q=0} = \tilde{\mathbf{n}}^{l-1}$ is the solution at the previous iteration of the loop. In our simulations, we choose c as

$$c \equiv \max_{(i,j)\in\bullet\text{-nodes}} \left(2b|\mathcal{A}|^{\bullet}_{i,j}(\mathbf{p})\right).$$

The coupled linear equations (3.14) is efficiently solved by the FFT. In the frequency domain, we have N_1N_2 systems of two linear equations with two unknowns and the determinant of the coefficients matrix is not zero for all frequencies with a positive penalty constant r_4; see more details in [31]. If the sequence $\mathbf{n}^q$ satisfies with

$$\frac{||\mathbf{n}^q - \mathbf{n}^{q-1}||_{L^1}}{||\mathbf{n}^{q-1}||_{L^1}} < \epsilon_{\mathbf{n}}, \tag{3.15}$$

we stop the iteration and update $\tilde{\mathbf{n}}^l = \mathbf{n}^q$. In this paper, $\epsilon_{\mathbf{n}} = 10^{-3}$ is used for all numerical examples.

Now, we briefly give the details on how to solve the coupled linear equations (3.14). Using the notation in Section 3.1, the coupled equations (3.14) can be written as:

$$\begin{aligned} -c\left(\partial_1^+\partial_1^- n_1^q + \partial_1^+\partial_2^- n_2^q\right) + r_4 n_1^q &= r_4 m_1 - \lambda_{41} - \partial_1^+\left((c-2b|\mathbf{p}|)\,\mathrm{div}^-\,\mathbf{n}^{q-1}\right), \\ -c\left(\partial_2^+\partial_1^- n_1^q + \partial_2^+\partial_2^- n_2^q\right) + r_4 n_2^q &= r_4 m_2 - \lambda_{42} - \partial_2^+\left((c-2b|\mathbf{p}|)\,\mathrm{div}^-\,\mathbf{n}^{q-1}\right). \end{aligned} \tag{3.16}$$

Using the shifting operators (3.6), the discretization at $(i,j) \in \circ$-node of the first equation in (3.16) is:

$$-c\left(\left(\mathcal{S}_1^- - 2\mathcal{I} + \mathcal{S}_1^+\right) n_1^q(i,j) + \left(\mathcal{S}_1^+ - \mathcal{S}_1^+\mathcal{S}_2^- - \mathcal{I} + \mathcal{S}_2^-\right) n_2^q(i,j)\right) + r_4 n_1^q(i,j) = f_1(i,j), \tag{3.17}$$

where

$$f_1(i,j) = r_4 m_1(i,j) - \lambda_{41}(i,j) - (c-2b|\mathcal{A}|^{\bullet}_{i+1,j}(\mathbf{p}))\mathrm{div}^{\bullet}_{i+1,j}(\mathbf{n}) + (c-2b|\mathcal{A}|^{\bullet}_{i,j}(\mathbf{p}))\mathrm{div}^{\bullet}_{i,j}(\mathbf{n}).$$

Similarly, the second equation in (3.16) is discretized at $(i,j) \in \square$-node as:

$$-c\left(\left(\mathcal{S}_2^+ - \mathcal{I} - \mathcal{S}_1^-\mathcal{S}_2^+ + \mathcal{S}_1^-\right) n_1(i,j) + \left(\mathcal{S}_2^- - 2\mathcal{I} + \mathcal{S}_2^+\right) n_2(i,j)\right) + r_4 n_2^q(i,j) = f_2(i,j), \tag{3.18}$$

where

$$f_2(i,j) = r_4 m_2(i,j) - \lambda_{42}(i,j) - (c-2b|\mathcal{A}|^{\bullet}_{i,j+1}(\mathbf{p}))\mathrm{div}^{\bullet}_{i,j+1}(\mathbf{n}) + (c-2b|\mathcal{A}|^{\bullet}_{i,j}(\mathbf{p}))\mathrm{div}^{\bullet}_{i,j}(\mathbf{n}).$$

Table 3.1: Methods of solving subproblems in Table 2.2.

subproblems in Table 2.2	related energy functional	methods or solutions
(2.14)	$\mathcal{E}_1(v)$	(3.4): closed-form solution
(2.15)	$\mathcal{E}_2(u)$	(3.5): FFT of linear PDE
(2.16)	$\mathcal{E}_3(\mathbf{m})$	(3.10): closed-form solution
(2.17)	$\mathcal{E}_4(\mathbf{p})$	(3.12): closed-form solution
(2.18)	$\mathcal{E}_5(\mathbf{n})$	(3.14): FFT of coupled linear PDEs

We apply the discrete Fourier transform to solve (3.17) and (3.18). Using the notations of discrete frequencies in (3.8), we have a system of linear equations:

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} \mathcal{F}n_1^q(y_i, y_j) \\ \mathcal{F}n_2^q(y_i, y_j) \end{pmatrix} = \begin{pmatrix} \mathcal{F}f_1(y_i, y_j) \\ \mathcal{F}f_2(y_i, y_j) \end{pmatrix},$$

where the coefficients are

$$\begin{aligned} a_{11} &= r_4 - 2c\,(\cos z_i - 1)\,, \\ a_{12} &= -c\left(1 - \cos z_j + \sqrt{-1}\sin z_j\right)\left(-1 + \cos z_i + \sqrt{-1}\sin z_i\right), \\ a_{21} &= -c\left(1 - \cos z_i + \sqrt{-1}\sin z_i\right)\left(-1 + \cos z_j + \sqrt{-1}\sin z_j\right), \\ a_{22} &= r_4 - 2c\,(\cos z_j - 1)\,. \end{aligned}$$

We have $N_1 N_2$ numbers of 2×2 systems. The determinant of the coefficient matrix

$$D = r_4^2 - 2r_4 c(\cos z_i + \cos z_j - 2)$$

is always positive for all discrete frequencies if $r_4 > 0$. After the systems of linear equations are solved for each frequency, we use the discrete inverse Fourier transform to obtain $\mathbf{n}^q = (n_1^q, n_2^q)$:

$$n_1^q = \Re\left(\mathcal{F}^{-1}\left(\frac{a_{22}\mathcal{F}f_1 - a_{12}\mathcal{F}f_2}{D}\right)\right) \quad \text{and} \quad n_2^q = \Re\left(\mathcal{F}^{-1}\left(\frac{-a_{21}\mathcal{F}f_1 + a_{11}\mathcal{F}f_2}{D}\right)\right),$$

where $\Re(\cdot)$ is the real part of a complex number.

For clarity, we summarize the methods for the sub-minimization problems in Table 3.1. All subproblems can be efficiently solved with low computational costs.

3.3. Update Lagrange multipliers. After the variables v^k, u^k, $\mathbf{m}^k$, $\mathbf{p}^k$, and $\mathbf{n}^k$ in (2.9) are updated by the algorithm in Table 2.2, we update Lagrange multipliers λ_1^k, $\boldsymbol{\lambda}_2^k$, λ_3^k, and $\boldsymbol{\lambda}_4^k$ according the the algorithm given in Table 2.1. Using the staggered grid as shown in Figure 3.1, the discretized form for the equations from (2.10) to (2.13) is written as:

$$\begin{aligned} \lambda_1^k &= \lambda_1^{k-1} + r_1(|\mathcal{A}|_{i,j}^{\bullet}(\mathbf{p}^k) - \mathbf{m}^k \cdot \mathbf{p}^k) && \text{at } \bullet\text{-node}, \\ \lambda_{21}^k &= \lambda_{21}^{k-1} + r_2(p_1^k - \partial_1^+ u^k) && \text{at } \circ\text{-node}, \\ \lambda_{22}^k &= \lambda_{22}^{k-1} + r_2(p_2^k - \partial_2^+ u^k) && \text{at } \square\text{-node}, \\ \lambda_3^k &= \lambda_3^{k-1} + r_3(v^k - u^k) && \text{at } \bullet\text{-node}, \\ \lambda_{41}^k &= \lambda_{41}^{k-1} + r_4(n_1^k - m_1^k) && \text{at } \circ\text{-node}, \\ \lambda_{42}^k &= \lambda_{42}^{k-1} + r_4(n_2^k - m_2^k) && \text{at } \square\text{-node}. \end{aligned}$$

We use average of vectors to approximate $\mathbf{m}^k \cdot \mathbf{p}^k$ at $\bullet$-node:

$$\mathbf{p}^k(i,j) = \left(\frac{p_1^k(i,j) + p_1^k(i-1,j)}{2}, \frac{p_2^k(i,j) + p_2^k(i,j-1)}{2} \right),$$
$$\mathbf{m}^k(i,j) = \left(\frac{n_1^k(i,j) + n_1^k(i-1,j)}{2}, \frac{n_2^k(i,j) + n_2^k(i,j-1)}{2} \right).$$

When the residual $R_1^k = |\mathbf{p}^k| - \mathbf{m}^k \cdot \mathbf{p}^k$ is close to machine precision, it may cause numerical instabilities because we do not use L^2 penalization for the term multiplied by r_1 in the proposed augmented Lagrangian method (2.8). In order to reduce such instabilities, we update $\lambda_1^k(i,j) = \lambda_1^{k-1}(i,j)$ if $R_1^k(i,j) < 10^{-12}$

4. Numerical examples. In this section, we present numerical examples using the proposed algorithm in image inpainting, image denoising, and image zooming. The test system is a Intel(R) Core(TM) i7 CPU Q720 1.6GHz with 4GB RAM.

During the iterations, we always monitor the residuals defined by:

$$\left(\tilde{R}_1^k, \tilde{R}_2^k, \tilde{R}_3^k, \tilde{R}_4^k \right) = \left(|\mathbf{p}^k| - \mathbf{m}^k \cdot \mathbf{p}^k, \mathbf{p}^k - \nabla u^k, v^k - u^k, \mathbf{n}^k - \mathbf{m}^k \right).$$

Since the residuals do not explicitly depend on the choice of penalty parameters r_1, r_2, r_3, and r_4, it is reasonable to stop the iteration if the relative residuals are smaller than ϵ_r, that is,

$$R_i^k \equiv \frac{1}{|\Omega|} ||\tilde{R}_i^k||_{L^1} < \epsilon_r, \quad \forall i \in \{1,2,3,4\}, \tag{4.1}$$

where $||\cdot||_{L^1}$ is the L^1 norm on Ω and $|\Omega|$ is the area of domain. In all our numerical experiments except the computational time comparison test, we use (4.1) as our stopping criterion. We also monitor the relative errors of Lagrange multipliers:

$$\left(L_1^k, L_2^k, L_3^k, L_4^k \right) = \left(\frac{||\lambda_1^k - \lambda_1^{k-1}||_{L^1}}{||\lambda_1^{k-1}||_{L^1}}, \frac{||\boldsymbol{\lambda}_2^k - \boldsymbol{\lambda}_2^{k-1}||_{L^1}}{||\boldsymbol{\lambda}_2^{k-1}||_{L^1}}, \frac{||\lambda_3^k - \lambda_3^{k-1}||_{L^1}}{||\lambda_3^{k-1}||_{L^1}}, \frac{||\boldsymbol{\lambda}_4^k - \boldsymbol{\lambda}_4^{k-1}||_{L^1}}{||\boldsymbol{\lambda}_4^{k-1}||_{L^1}} \right), \tag{4.2}$$

the relative error of the solution $\{u^k \mid k = 1, 2, ...\}$

$$\frac{||u^k - u^{k-1}||_{L^1}}{||u^{k-1}||_{L^1}}, \tag{4.3}$$

and the numerical energy

$$E^k = \int_\Omega \left(a + b(\nabla \cdot \mathbf{n}^k)^2 \right) |\mathbf{p}^k| + \frac{\eta}{s} \int_\Gamma |v^k - u_0|^s. \tag{4.4}$$

All quantities are plotted in log scale except the x-axis for the numerical energy and SNR (4.6). Note that (4.2) can be quite small if the penalty parameters are large. It is caused by the explicit dependency on the penalty parameters.

The variation of the residuals as well as the relative errors and energy will give us important information about the convergence of the iterations. While we show numerical results obtained at the iteration when the stopping criterion is met, all graphs related to the proposed algorithm are drawn up to 10^3 iterations. Before we show our numerical results, we give some remarks on choosing tuning parameters.

There are three parameters coming from the elastica energy and data fitting term: a, b, and η. The ratio between a and b has to do with the connectivity and smoothness of the level lines: larger b encourages the connection of disconnected level lines and smoother level lines; see also [17]. The parameter η depends on how close we want u to be u_0. In image inpainting, we need u to be as close to u_0 on $\Omega \setminus \Gamma$ as possible. So we choose large value for η. In image denoising, we tune η according to the amount of noise found in image: the noisier u_0 is, the smaller value we choose for η. We also have the parameters associated with lagrange multipliers: r_1, r_2, r_3, and r_4. In order to reduce the number of parameters to tune, we set $r_1 = 1$ in our numerical experiments. To understand the role of the parameters r_2 and r_3, we go to the equation (3.5). The parameter r_2 controls the amount of diffusion of u: the larger r_2 is, the more diffusion u has. The parameters r_3 and r_4 control the closeness between v and u and $\mathbf{n}$ and $\mathbf{m}$, respectively. While using these as a guideline, we tried to keep the same parameters whenever possible. In particular, our results for the images corrupted by gaussian noise were obtained using the same parameters. Therefore, we can say that the number of parameters associated with our algorithm is not too much of concern.

4.1. Image inpainting. In this subsection, we illustrate the efficiency of the proposed algorithm via many examples in image inpainting. The process of restoring missing or damaged areas in digital images is known as image inpainting or image interpolation. Interesting applications of digital inpainting are explored in [7], where the term *image inpainting* was introduced. Numerous approaches to image inpainting have been proposed in literature: an axiomatic approach to image interpolation [13], variational geometric approach [1, 2, 17–20, 26, 39–41], inpainting by vector fields and gray levels [5, 6], simultaneous structure and texture image inpainting [8], wavelet and framelet based image inpainting [11, 16], examplar-based inpainting methods [21, 22, 29, 49], exemplar-based inpainting method with a geometric guide [12], analytical modeling of exemplar-based inpainting method [3], reformulation of the problem of region-segmentation with curvature regularity as an integer linear program (ILP) [45], and graph cut formulation of the elastica model [4, 23].

The inpainting domain $\mathcal{D}$ is always shown in red color. The integration domain Γ in the energy functional (2.1) is $\Omega \setminus \mathcal{D}$. The proposed (ALM) algorithm in Table 2.1 can be applied for $s = 1$ or 2 in (2.1). We use $s = 1$ for all examples in image inpainting. Note that the main idea of proposed algorithm can be used to solve image inpainting model (2.2). Recently, some fast algorithms to minimize curvature are proposed in image segmentation [23] and image inpainting [4, 45].

First of all, the computational time (sec) of the proposed algorithm (ALM) is compared with the numerical algorithm (CKS) in [17] for image inpainting. Whereas the algorithm (CKS) involves time step, the strict stability constraint on the time step for gradient descent method does not apply for the parameters of our algorithm. For a fair comparison, we stop the iteration if the relative difference error of the solution $\{u^k \mid k = 1, 2, ...\}$ satisfies

$$\frac{1}{|\Omega|}||u^k - u^{k-1}||_{L^1} < \epsilon_d. \tag{4.5}$$

In Table 4.1, the red regions on the left images in (a) and (b) are inpainting domains and the right images in (a) and (b) are the results of our proposed algorithm in Table 2.1. We use $\epsilon_d = 10^{-5}$ for the example (a) and $\epsilon_d = 2 \cdot 10^{-6}$ for the example (b). Since we have two projection steps and two linear PDEs in subproblems, the

Table 4.1: The computational time (sec) of our proposed (ALM) algorithm in Table 2.1 is compared with the method (CKS) in [17]. The red regions on the left images in (a) and (b) are inpainting domains. The right images in (a) and (b) are the results from the proposed algorithm.

	size	unknowns	ALM		CKS	
			# of iteration	time (sec)	# of iteration	time (sec)
(a)	28×28	50	55	0.177	427000	412.043
(b)	28×28	256	432	0.807	1301000	1397.371

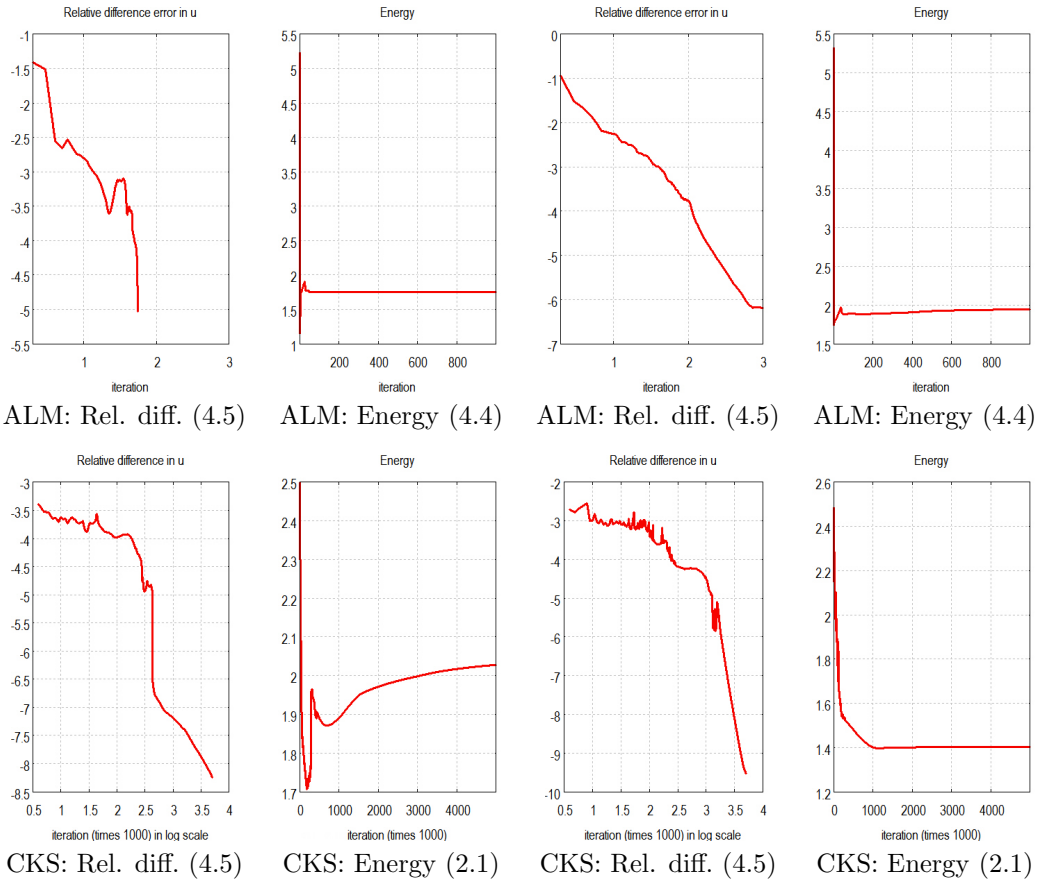
(a) (b)

proposed algorithm takes very low computational cost. Comparing with CKS, the proposed algorithm is approximately thousands times faster in computational time. The constants $a = 1$, $b = 20$, and $\eta = 5 \cdot 10^2$ are used for (a) and (b) in ALM and CKS. In (a), the penalty parameters in our algorithm are $r_1 = 1$, $r_2 = r_3 = 10$, and $r_4 = 50$. For the algorithm in CKS, the time step 10^{-5} is used. In (b), the penalty parameters in our algorithm are $r_1 = 1$, $r_2 = 4 \cdot 10^2$, and $r_3 = r_4 = 10^2$. For the algorithm in CKS, the time step 10^{-4} is used. In Figure 4.1, plots of energy and relative errors versus iteration numbers are shown for examples (a) and (b) in Table 4.1. As we mentioned at the beginning of Section 4, all graphs related to our algorithm are drawn up to 10^3 iterations. By doing so, we can easily observe the convergence of our algorithm. In the case of the algorithm in CKS, it involves time step. So it needs more iteration numbers to see its convergence. From this test, we can see that our algorithm takes very few iterations to converge compared with the CKS method. The two algorithms produce similar inpainting results.

In Figure 4.2, we show the inpainting results for some synthetic images. The red regions in the first and third rows are the inpainting domains. The second and fourth rows are inpainting results. As we expect, the Euler's elastica energy shows a property of long connectivity in (a2). If the total variation is used, that is, $b = 0$ in (2.1), the inpainting region in (a2) will be filled as black color because the width of red region is thicker than the height of horizontal white bar. From other synthetic examples, we observe that the curvature term makes smooth connection along the level curves of images on inpainting domains. Even though there are no theoretical convergence guarantees in this paper, these examples show that our proposed method is indeed working because visually correct results are obtained. In (e1) and (f1), we intentionally use complicated the inpainting domains. Even though the portion of unknown pixels are 69.96% in (e1) and 84.96% in (f1), the Euler's elastica energy recovers the main shapes in (e2) and (f2) very well. We use $r_1 = 1$ and $\eta = 10^3$ for (a2)-(f2) and $a = 1$ and $b = 20$ for (b2)-(f2). The rest of the tuning parameters are shown in Figure 4.2.

In Figure 4.3, we show the results of real image inpainting. (a2) is a restoration

Example in Table 4.1-(a) Example in Table 4.1-(b)



ALM: Rel. diff. (4.5) ALM: Energy (4.4) ALM: Rel. diff. (4.5) ALM: Energy (4.4)

CKS: Rel. diff. (4.5) CKS: Energy (2.1) CKS: Rel. diff. (4.5) CKS: Energy (2.1)

Fig. 4.1: Plots of energy and relative errors of u^k for examples (a) and (b) shown in Table 4.1. ALM is our proposed algorithm in Table 2.1 and CKS is the method in [17]. Since the relative difference in Table 4.1-(a) is zero after 55 iterations, we do not plot the relative difference afterwards.

of degraded image and (b2) is a recovery of corrupted data. For (a2) and (b2), we use the same tuning parameters. They are shown in Figure 4.3. In Figure 4.4, we illustrate relative residuals (4.1), relative errors of Lagrange multipliers (4.2), relative error of u^k (4.3), and numerical energy (4.4) along the outer iteration k. The graphs on the first and second rows are related to Figures 4.2-(c2) and 4.3-(a2), respectively. The graphs of the other examples in Figures 4.2 and 4.3 also have a similar profiles in Figure 4.4. Even though values of graphs are not monotonically decreasing, the values are stable and steady after a few iterations. In early stages of the outer iteration, the increasing profile in the numerical energy is reasonable because all variables v^k, u^k, $\mathbf{m}^k$, $\mathbf{p}^k$, and $\mathbf{n}^k$ are initially zero. Since η is large in image inpainting, the fidelity term in E^k (4.4) dramatically tends to be zero in a few iterations. Starting with zero values of variables, the regularity term in E^k is increasing and then it is eventually steady after some iterations.

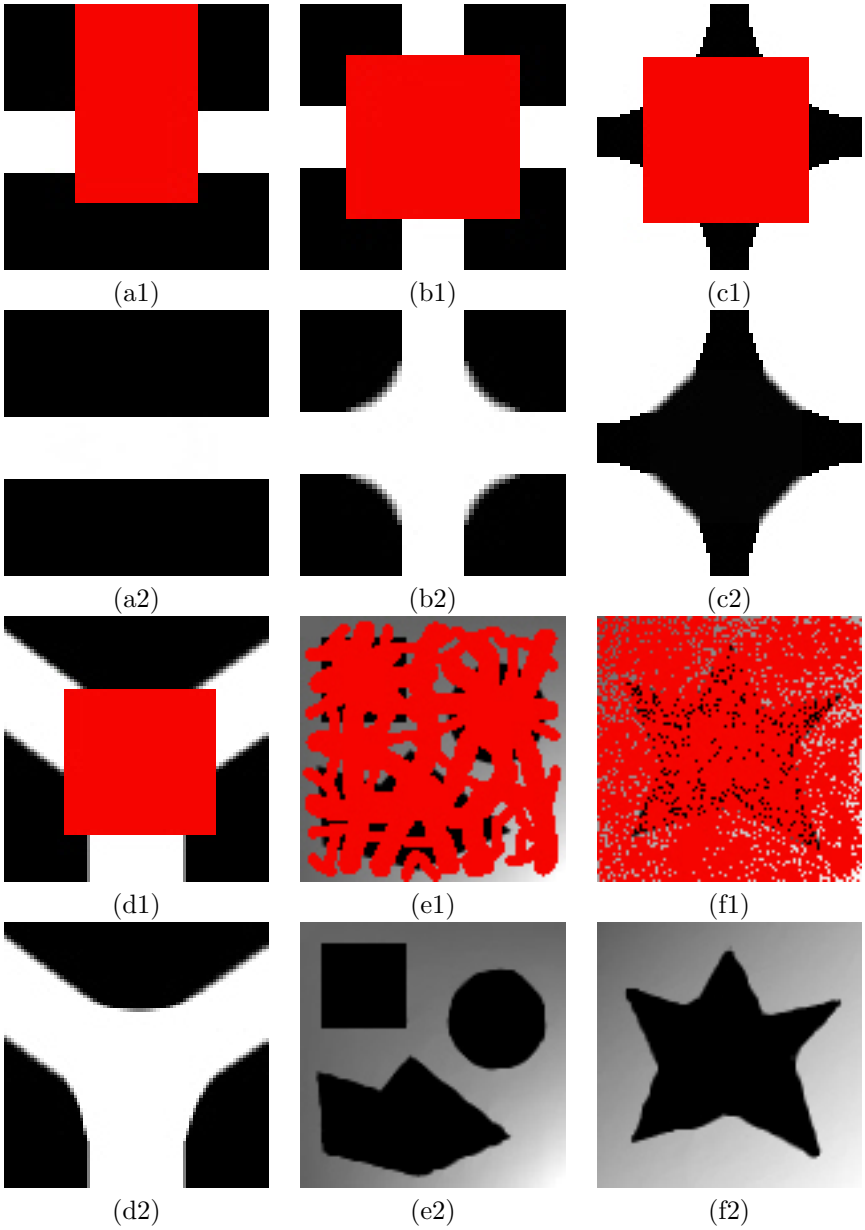
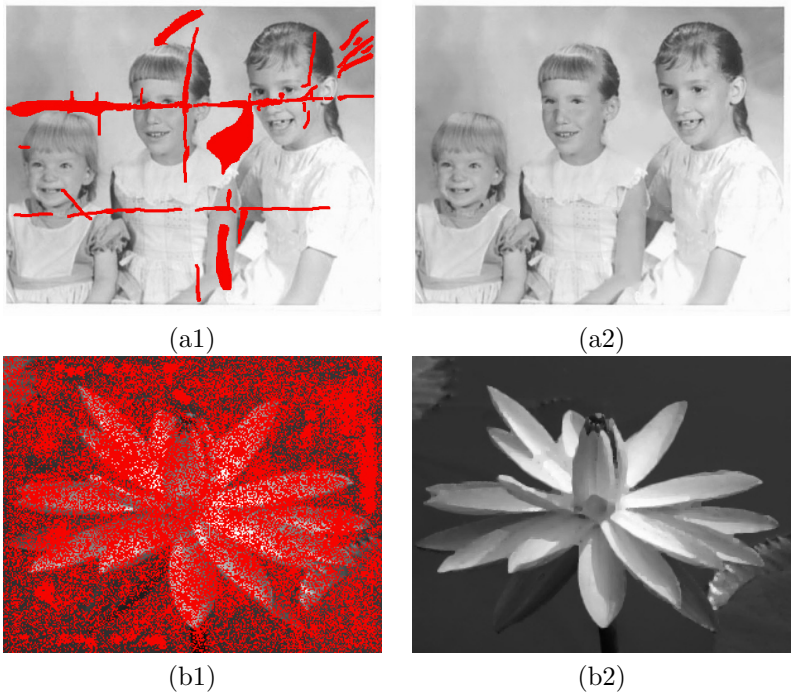


Fig. 4.2: The red regions in the first and third rows are inpainting domains. The second and fourth rows are inpainting results based on Euler's elastica functional minimization. Parameters: (a2): $r_2 = r_3 = 1$, $r_4 = 6 \cdot 10^2$, $\epsilon_r = 1.2 \cdot 10^{-2}$, (b2): $r_2 = 3 \cdot 10^2$, $r_3 = 10^2$, $r_4 = 6 \cdot 10^2$, $\epsilon_r = 10^{-3}$, (c2): $r_2 = r_3 = 2 \cdot 10^2$, $r_4 = 6 \cdot 10^2$, $\epsilon_r = 2 \cdot 10^{-3}$, (d2): $r_2 = 2 \cdot 10^2$, $r_3 = 10$, $r_4 = 10^2$, $\epsilon_r = 7 \cdot 10^{-4}$, (e2): $r_2 = r_3 = 50$, $r_4 = 5 \cdot 10^2$, $\epsilon_r = 5 \cdot 10^{-3}$, (f2): $r_2 = 50$, $r_3 = 10^2$, $r_4 = 5 \cdot 10^2$, $\epsilon_r = 5 \cdot 10^{-3}$.



(a1) (a2)

(b1) (b2)

Fig. 4.3: The red regions in the left column are inpainting domains. The right column is the corresponding inpainting result based on Euler's elastica functional minimization. Parameters: (a2) and (b2): $a = 1$, $b = 20$, $\eta = 10^3$, $r_1 = 1$, $r_2 = 2 \cdot 10^2$, $r_3 = 10^2$, and $r_4 = 3 \cdot 10^2$, and $\epsilon_r = 8 \cdot 10^{-3}$.

In Table 4.2, we show the number of unknown pixels in inpainting domains, the total number of outer iteration, and computational time (sec) for the results in Figures 4.2 and 4.3. If the size of inpainting domain is relatively smaller than the size of image, one may obtain better computational efficiency using the energy functional (2.2). However, there are some advantages to use the functional (2.1) in image inpainting. It is very flexible to choose inpainting domains. Since we have to use the boundary conditions for the image gradient in the model (2.2), it cannot be applied to the inpainting domain whose boundary encloses two or three isolated pixels such as red regions in Figures 4.2-(f1) and 4.3-(b1). Comparing the computational time in Figures 4.2-(e2) and 4.2-(f2), we have a better performance in Figure 4.2-(f1) even though the missing parts in Figure 4.2-(e1) is much smaller. It usually takes more time to obtain an inpainted result from the model (2.2) if the inpainting domain is larger and larger.

4.2. Image denoising. We illustrate the results of image denoising with Euler's elastica energy functional. Recently, the graph cuts algorithm to minimize Euler's elastica energy has been proposed for image denoising in [4]. In this subsection, we use $s = 1$ and 2 in (2.1) for salt-and-pepper noise and Gaussian white noise, respectively. In image denoising, Γ in (2.1) is the same as the domain of image Ω. The proposed algorithm can be also extended to Poisson type noise following [53].

In Figure 4.5, we show some results for some synthetic images based on our

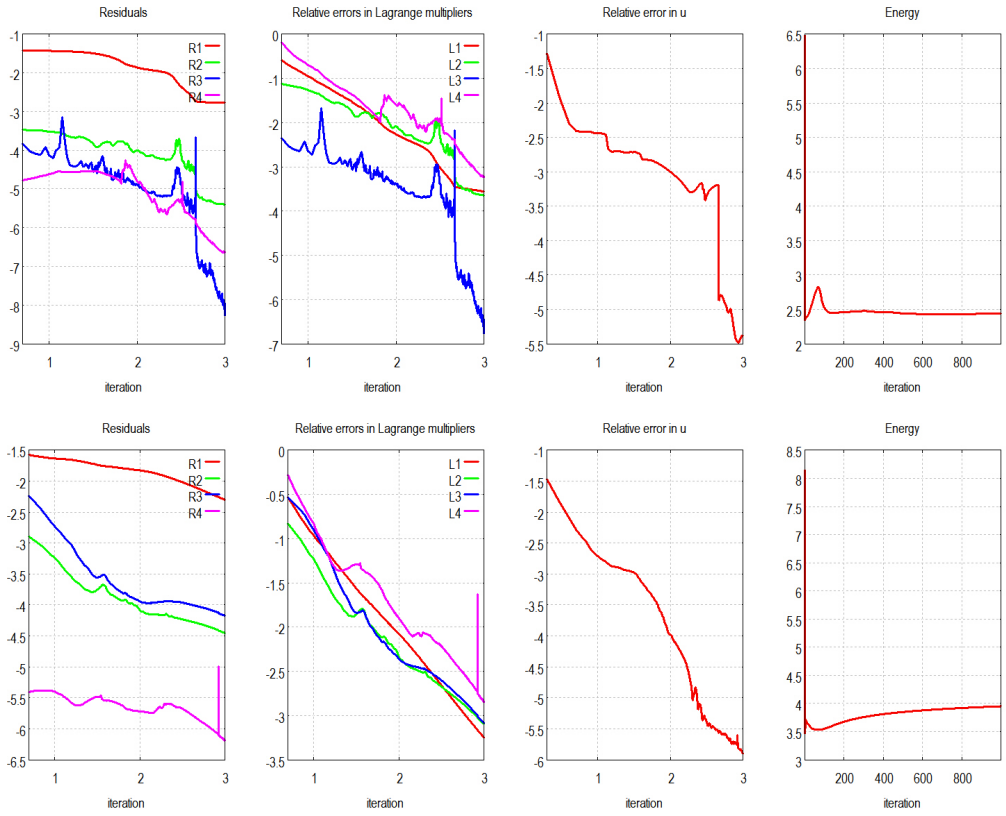


Fig. 4.4: From the left, the plots of residuals (4.1), relative errors in Lagrange multipliers (4.2), relative error in u (4.3), and energy (4.4) for examples shown in Figures 4.2-(c2) (top) and Figure 4.3-(a2) (down). From these plots, it is easy to see that the algorithm has converged before 10^3 iterations. The plots of the residuals R_i^k also gives important information about the choosing and tuning of the parameters r_i. With a bigger r_i, the residual R_i^k will converge to zero faster. It is better to choose the parameters r_i to make the residuals R_i^k converging to zero in a similar rate.

proposed algorithm in Table 2.1. The noisy images are shown in the left column and the restored images are shown in the right column. Gaussian white noise with zero mean and the standard deviation 10 are used for all test images. We acknowledge that the image in Figure 4.5-(d1) is taken from [43]. The test image in Figure 4.5-(c1) seems to be first used in [37]. According to the Euler's elastica energy, denoised images have smooth connections in the level curves of images. Moreover, the total variation preserves jump discontinuities in images. Some very nice analysis for these properties are given in [58]. Figure 4.6 shows the results of some real images. Note that we do not need to use the extra variable v and the Lagrange multiplier λ_3 in the functional (2.8) for this case where $s = 2$.

In Table 4.3, we show the size of images, SNR, number of outer iteration k, and computational time for the test images in Figures 4.5 and 4.6. The signal-to-noise

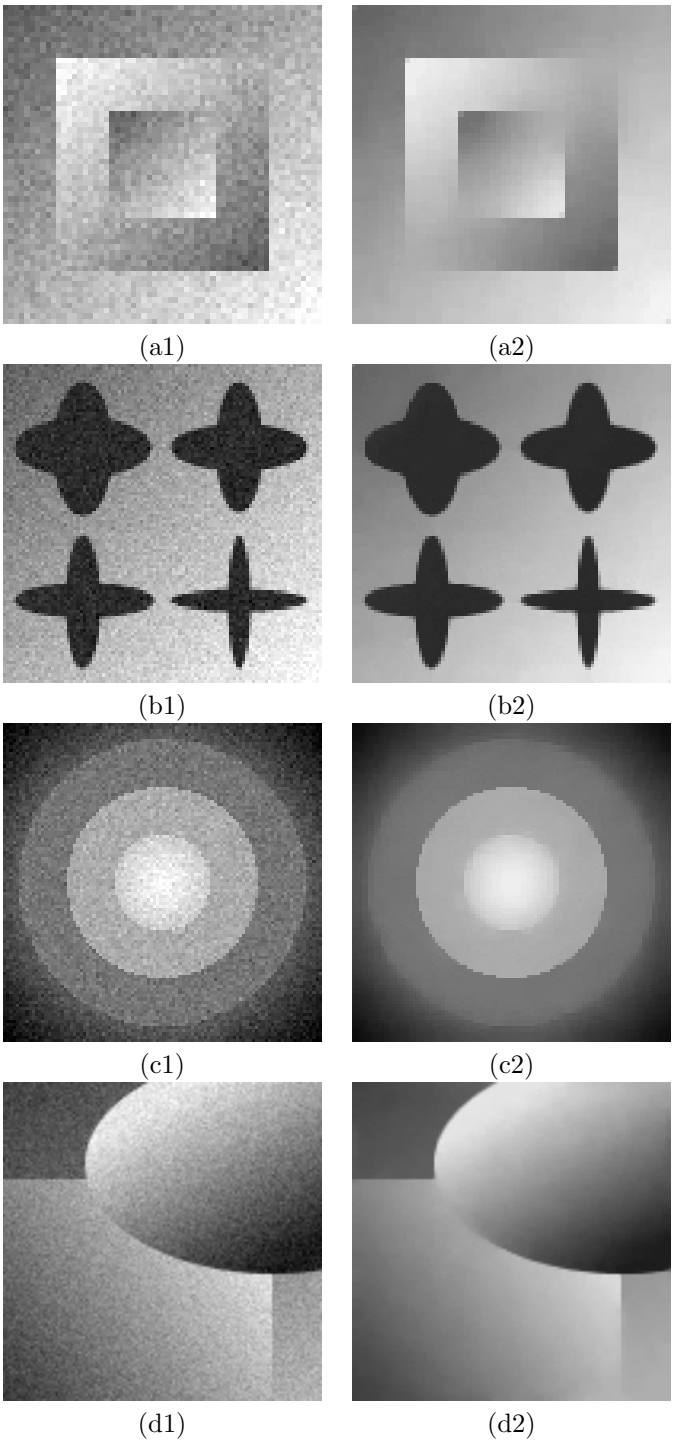


Fig. 4.5: Euler's elastica based image denoising: Left column: noisy images. Right column: denoised images. Gaussian white noise with zero mean and the standard deviation 10 is used for all images. For all examples, we use $a = 1$, $b = 10$, $\eta = 10^2$, $r_1 = 1$, $r_2 = 2 \cdot 10^2$, and $r_4 = 5 \cdot 10^2$. The remaining parameter ϵ_r is $1.3 \cdot 10^{-3}$ in (a2), $7 \cdot 10^{-3}$ in (b2), 10^{-2} in (c2), and $8 \cdot 10^{-3}$ in (d2).

Table 4.2: The number of unknowns in the inpainting domains in Figures 4.2 and 4.3 are shown. The computational time is measured in seconds. The number of iteration is the number of total outer iteration in Table 2.1.

images	size	unknowns	# of iteration	time (sec)	percentage of unknown pixels
Fig. 4.2-(a2)	52×52	936	403	2.527	34.62%
Fig. 4.2-(b2)	52×52	1088	333	2.200	40.24%
Fig. 4.2-(c2)	80×80	2500	449	6.568	39.06%
Fig. 4.2-(d2)	80×80	2024	352	5.319	31.63%
Fig. 4.2-(e2)	100×100	6996	307	7.753	69.96%
Fig. 4.2-(f2)	100×100	8496	278	6.927	84.96%
Fig. 4.3-(a2)	484×404	14258	443	298.611	7.29%
Fig. 4.3-(b2)	300×235	42114	329	80.641	59.74%

Table 4.3: The size of images and the SNR for images in Figures 4.5 and 4.6 are shown. The computational time is measured in seconds. The number of iteration is the number of total outer iteration in Table 2.1.

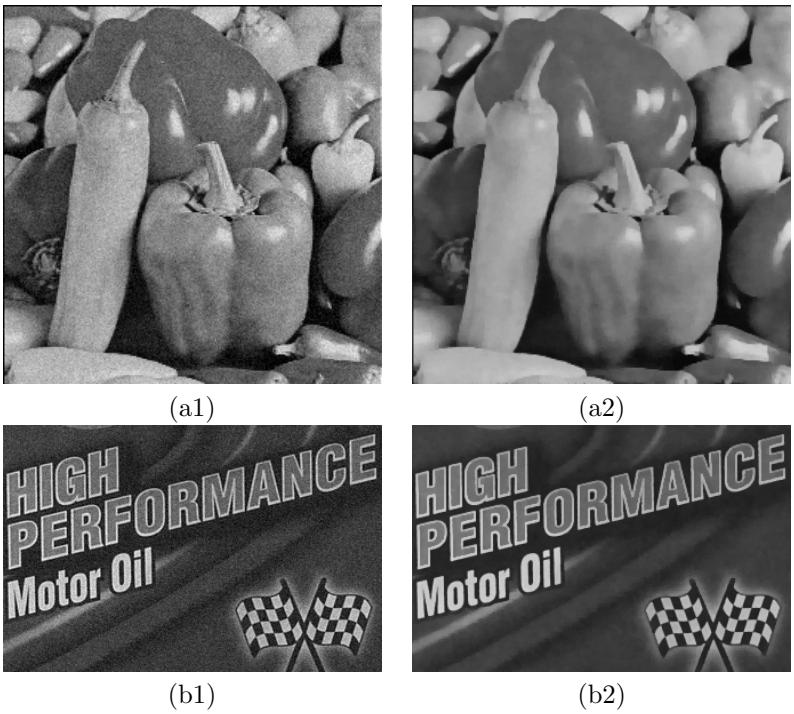
images	size	SNR	# of iteration	time (sec)
Fig. 4.5-(a2)	60×60	21.96	314	3.366
Fig. 4.5-(b2)	100×100	27.74	283	7.056
Fig. 4.5-(c2)	100×100	24.28	402	11.598
Fig. 4.5-(d2)	128×128	16.15	290	13.434
Fig. 4.6-(a2)	256×256	18.62	274	55.369
Fig. 4.6-(b2)	332×216	17.24	272	77.410

ratio (SNR) is defined as

$$10 \log_{10} \left(\frac{\sum_{i,j} (u^k(i,j) - a_1)^2}{\sum_{i,j} (|u^k(i,j) - u_c(i,j)| - a_2)^2} \right), \tag{4.6}$$

where u_c is an original image and a_1 and a_2 are average of u^k and $u^k - u_c$, respectively. The number of iteration is the total number of outer iteration in Table 2.1. The computational time is measured in seconds. Since the proposed algorithm consists of simple loops and discrete Fourier transforms, the speed of computation can be accelerated by parallel programming. Specially, we address that the computational time is easily reduced several times via CUDA based programming. In our numerical experiment with GeForce 9600M GT, we approximately obtain four times faster computation.

In Figure 4.7, we show the relative residuals (4.1), relative errors of Lagrange multipliers (4.2), relative error in u^k (4.3), numerical energy (4.4), and SNR (4.6) versus the outer iteration k. The graphs on the first and second rows are related to the examples in Figures 4.5-(b2) and 4.6-(a2), respectively. The plots of the other examples in Figures 4.5 and 4.6 also have a similar profiles in Figure 4.7. Since the Euler's



(a1) (a2)

(b1) (b2)

Fig. 4.6: Euler's elastica based image denoising: Left column: noisy images. Right column: denoised images. Gaussian white noise with zero mean and the standard deviation 10 is used for all images. For all examples, we use $a = 1$, $b = 10$, $\eta = 2 \cdot 10^2$, $r_1 = 1$, $r_2 = 10^2$, $r_4 = 5 \cdot 10^2$, and $\epsilon_r = 1.5 \cdot 10^{-2}$.

elastica energy functional is not convex, we cannot expect monotonic decreasing in the values of the energy, relative residuals, and relative errors of Lagrange multipliers. From these plots, we can see that the algorithm is stable and has converged. It is important that the residuals R_i^k converge to zero with similar rate for different i. It is also clear that the energy and SNR have converged to a steady state.

In Figures 4.8, we show image denoising results and tuning parameters for some real images with salt-and-pepper noise. The noisy images are shown in the left column and the restored images are shown in the right column. Salt-and-pepper noise with a noise density 0.4 are used for all test images in Figures 4.8. As we can observe from the results in Figures 4.8, our algorithm for the model (2.1) recovers jump discontinuities and smooth regions better compared to total variation based image restoration which may have suffered from stair-case effect.

In Table 4.4, we show the size of images, SNR, number of iterations and computational time for the test images in Figures 4.8. The number of iterations is the total number of outer iteration in Table 2.1. The computational time is measured in seconds. From Tables 4.3 and 4.4, we can observe that the computational cost depends on the size of image and the number of outer iteration. In our numerical experiments, the condition (3.15) with $\epsilon_{\mathbf{n}} = 10^{-3}$ is satisfied mostly less than 5 iterations in an early stage of outer iteration k. As long as the outer iteration increases, one or two iterations are enough to solve the equation (3.14). It explains that total

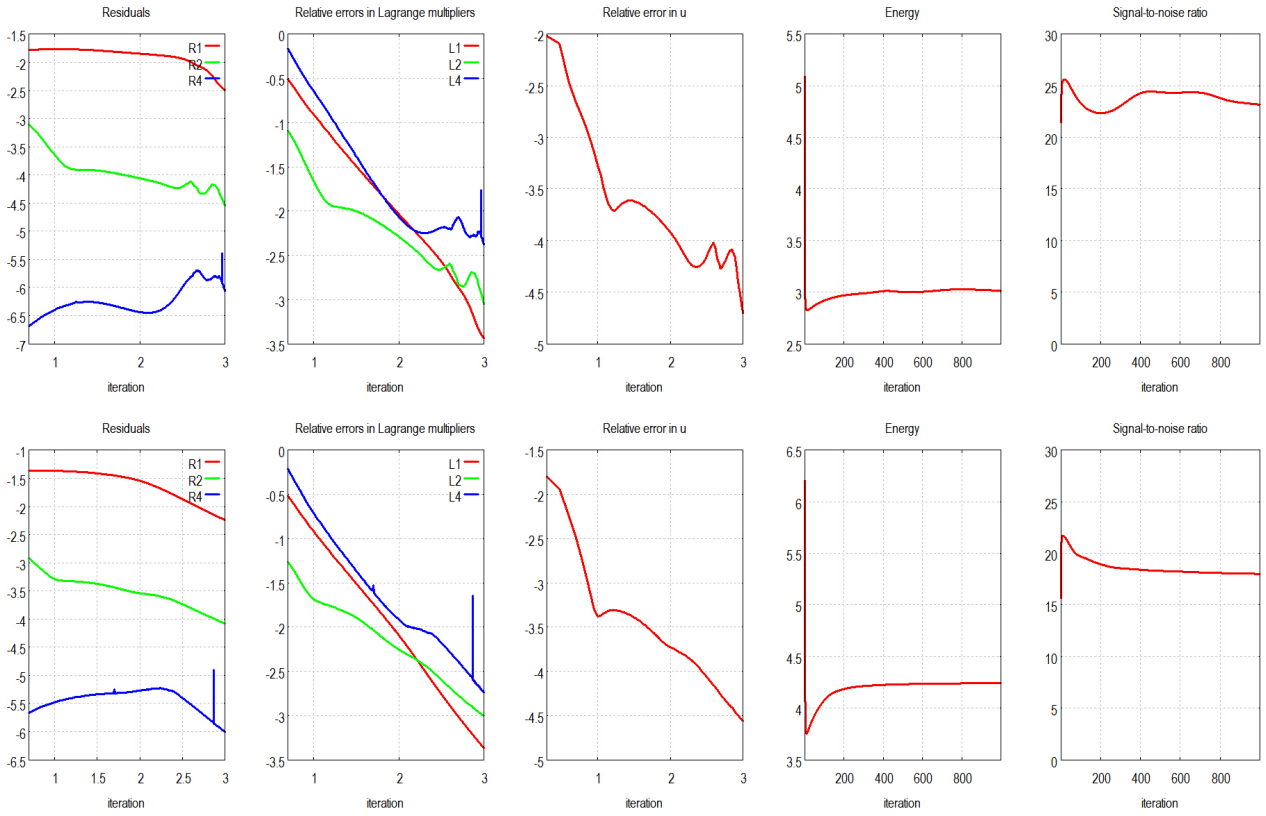
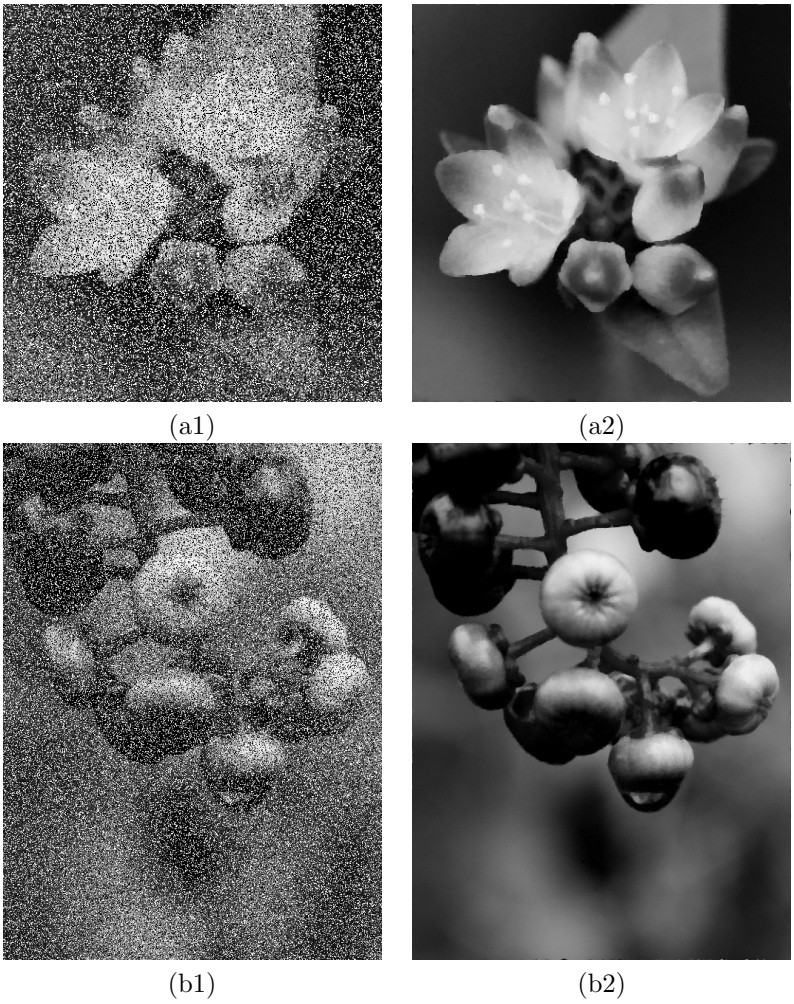


Fig. 4.7: From the left, the plots of residuals (4.1), relative errors in Lagrange multipliers (4.2), relative error in u (4.3), energy (4.4), and SNR (4.6) for examples shown in Figures 4.5-(c2) (top) and Figure 4.6-(a2) (down). From these plots, we can see that the algorithm is stable and has converged. It is important that the residuals R_i^k converge to zero with similar rate for different i. It is also clear that the energy and SNR have converged to a steady state.

computational cost depends on how many outer iterations are needed.

In Figure 4.10, we show the evolution of variables $\mathbf{n}^k$, $\mathbf{p}^k$, and v^k in order to have a better understanding the change of the numerical energy (4.4) for the example in Figure 4.9. We use the standard hue-saturation-value (HSV) color map to represent a vector field. Since every vector field in this paper is defined on two dimensional space, the value in the HSV color map is fixed to 1. For example, the white color in the HSV color map presents the zero vector. The first row in Figure 4.10 is the plot of the values for the fidelity and regularity terms; see (4.4). We also show the computed $\mathbf{n}^k$, $\mathbf{p}^k$, and v^k at $k = 10$, 50, 110, and 350. From the plots, we see that v first goes to a noisy image from $v^0 = 0$ and then become smoother as the iteration number increases. This explains why the blue curve in Figure 4.10 first goes down and then goes up again to a steady state. Since all variables are initialized to zero, the algorithm is "wandering" in the beginning and then stabilized after 150 iterations. Along the outer iterations, we observe that the variable $\mathbf{p}^k$ recovers image gradient very well.



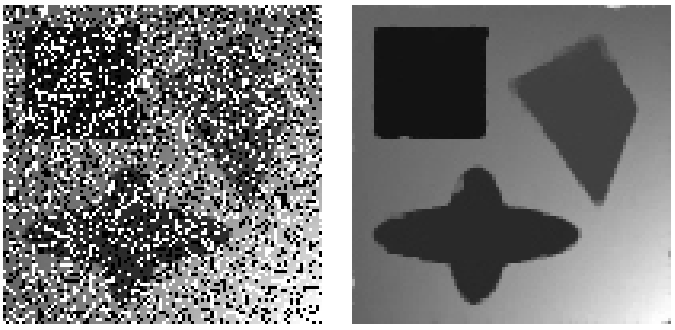
(a1) (a2)

(b1) (b2)

Fig. 4.8: Euler's elastica based image denoising: Left column: noisy images. Right column: denoised images. Salt-and-pepper noise with a noise density 0.4 is used for all images. Parameters: (a2): $a = 1$, $b = 20$, $\eta = 10$, $r_2 = 7 \cdot 10^2$, $r_3 = 10^2$, $r_4 = 5 \cdot 10^2$, and $\epsilon_r = 10^{-2}$.

Table 4.4: The size of images and the SNR for the images in Figures 4.8 are shown. The computational time is measured in seconds. The number of iteration is the number of total outer iteration in Table 2.1.

images	size	SNR	# of iteration	time (sec)
Fig. 4.8-(a2)	400×420	20.87	239	113.147
Fig. 4.8-(b2)	512×700	20.32	174	182.359



(a1) (a2)

Fig. 4.9: Euler's elastica based image denoising: Left column: noisy images. Right column: denoised images. Salt-and-pepper noise with a noise density 0.4 is used. Same parameters in Figure 4.8 are used except $\epsilon_r = 6 \cdot 10^{-3}$.

These plots show the quick convergence and the cost efficiency of our algorithm. It is amazing to see that such a complicated energy can be minimized by such a simple iterative procedure. Note that $\mathbf{p}$ takes (nearly) zero values on homogeneous regions and this may cause problems for some other approaches. However, it is of no problem for the iterative scheme given here.

4.3. Image zooming. Image zooming is one of basic operations in image processing, which changes a given image into a higher or lower resolution image. In this subsection, we only deal with image resolution enhancement by the factor $r \in \mathbf{N}$. More precisely, a given image $\bar{u}_0$ whose size is $[1, M_1] \times [1, M_2]$ is interpolated onto the domain $\Omega = [1, r(M_1 - 1) + 1] \times [1, r(M_2 - 1) + 1]$. For this purpose, we define the set of points $\Gamma = \{(i, j) \in \Omega \mid i \equiv 1 \bmod r,\ j \equiv 1 \bmod r\}$ and use the image u_0, whose value is same as $\bar{u}_0$ on Γ, for the energy functional given in (2.1):

$$u_0(i,j) = \begin{cases} \bar{u}_0\left(\dfrac{i-1}{r} + 1, \dfrac{j-1}{r} + 1\right) & \text{if} \quad (i,j) \in \Gamma, \\ 0 \quad \text{if} & (i,j) \in \Omega \setminus \Gamma. \end{cases}$$

The image values on $\Omega \setminus \Gamma$ are interpolated via Euler's elastica energy. More suitable fidelity based on a down-sampling operator in [38] may give better quality. In the energy functional (2.1), $s = 1$ is used for image zooming.

In Figure 4.11, we show the results by magnifying a synthetic image by a factor 8. The size of input image in (a) is 64×64 and the enlarged image is 505×505. In (b) and (c), box and bicubic filters are used respectively. (d) shows the result by minimizing the Euler's elastica energy using our proposed algorithm. The computational time is 461.964 seconds and the total number of outer iteration is 515.

From this test, the advantage of the proposed algorithm is clear. It is very fast. The discontinuity in the recovered image by Euler's elastica energy is very sharp. Corners and shapes are preserved. There is no blurring. The zig-zag boundary from the box filter and the blurring from the cubic filter are avoided. More tests of the proposed method for image zooming will be reported elsewhere.

5. Conclusion. We presented a fast and efficient method for the minimization of energy functional related to Euler's elastica. It is very difficult to minimize the energy

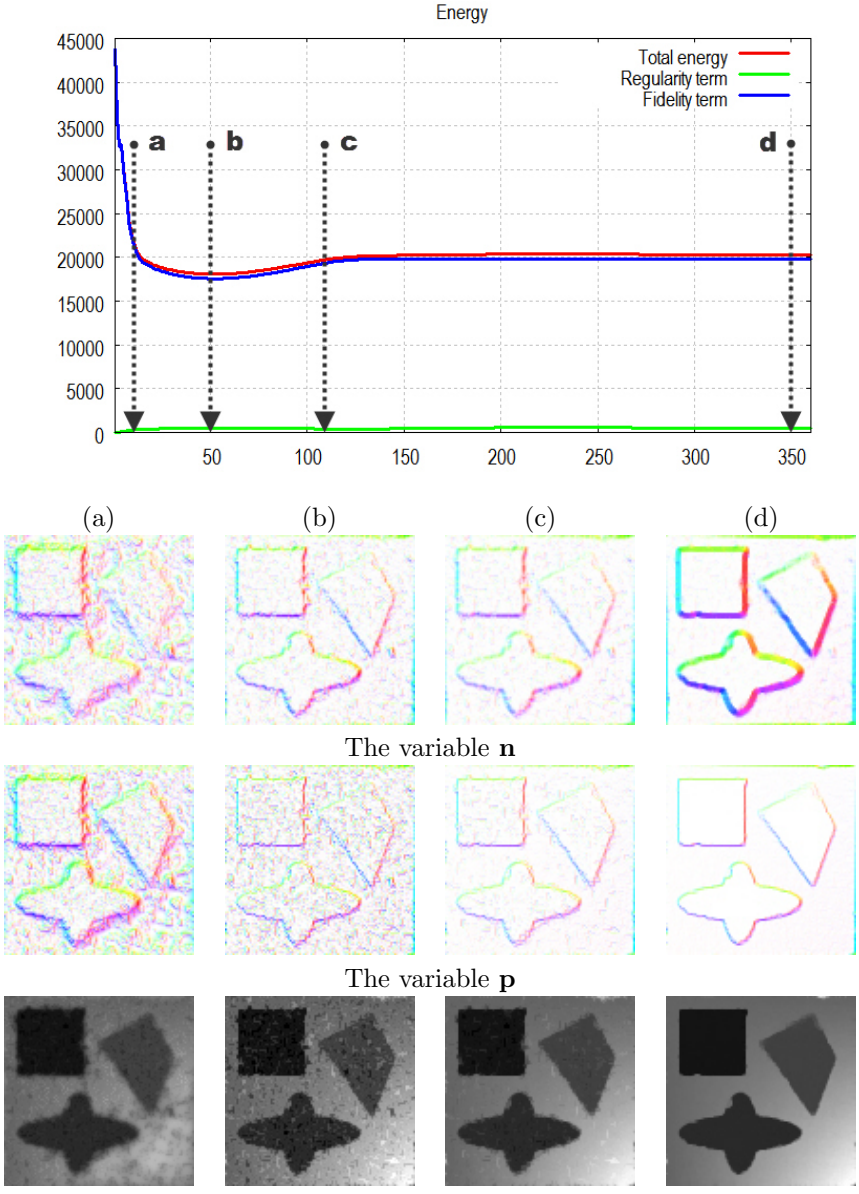
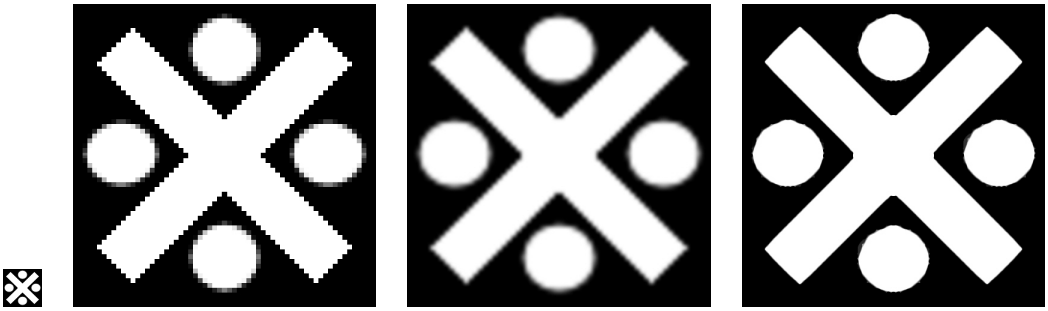


Fig. 4.10: Evolution of the energy value and the computed $\mathbf{n}, \mathbf{p}, v$ versus the iteration number for Figure 4.9-(a2). The first row is the plot of the numerical energy value in decimal scale. From (a) to (d), the number iterations are 10, 50, 110, and 350, respectively. The vector field $\mathbf{n}$ and $\mathbf{p}$ are rendered by the standard HSV color map with the unit value. These plots show the quick convergence and the cost efficiency of our algorithm.



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

Fig. 4.11: (b), (c), and (d) are resized by the factor 8 from (a). The size of image (a) is 64×64 and the others are 505×505. (b) and (c) are obtained by box and bicubic filters, respectively. (d) is the result of Euler's elastica energy. Parameters: $a = 1$, $b = 10$, $\eta = 10^2$, $r_1 = 1$, $r_2 = 5 \cdot 10^2$, $r_3 = 10^2$, $r_4 = 5 \cdot 10^2$, and $\epsilon_r = 3 \cdot 10^{-4}$.

due to its non-convexity, nonlinearity (highly nonlinear), and high order derivatives. The proposed algorithm is based on augmented Lagrangian method with specially designed constraints and penalization techniques. Due to these special features, the computational cost is very low per iteration. The method is fast to converge and simple to implement. As our numerical results indicate, our method yields good results in terms of computational time. It is believable that this kind of algorithm will have good application in a number of real industrial problem related to image processing and computer vision.

Acknowledgements. We would like to thank S.H. Kang (Georgia Tech) for sharing her C++ code in order to measure the computational time in Table 4.1. We also thank Ron Kimmel (Technion) for introducing this challenging subject to us. The copyright of original images in Figures 4.3-(b1), 4.8-(a1), and 4.8-(b1) is owned by Mr. Byung Keun Hahn. The authors thank technical staffs in Media Nature for providing the image in Figure 4.6-(b1).

REFERENCES

[1] L. Ambrosio and S. Masnou, *A direct variational approach to a problem arising in image reconstruction*, Interfaces Free Bound., 5 (2003), pp. 63–81.

[2] L. Ambrosio and S. Masnou, *On a variational problem arising in image reconstruction*, vol. 147, Birkhauser, Basel, 2004.

[3] J.F. Aujol, S. Ladjal, and S. Masnou, *Exemplar-based inpainting from a variational point of view*, SIAM J. Math. Anal., (To appear).

[4] E. Bae, J. Shi, and X.-C. Tai, *Graph cuts for curvature based image*, IEEE Trans. Image Process., (To appear).

[5] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera, *Filling-in by joint interpolation of vector fields and gray levels*, IEEE Trans. Image Processing, 10 (2001), pp. 1200–1211.

[6] C. Ballester, V. Caselles, and J. Verdera, *Disocclusion by joint interpolation of vector fields and gray levels*, Multiscale Model. Simul., 2 (2003), pp. 80–123.

[7] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, *Image inpainting*, in Computer Graphics, SIGGRAPH 2000, 2000, pp. 417–424.

[8] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher, *Simultaneous structure and texture image inpainting*, IEEE TIP, 12 (2003), pp. 882–889.

[9] A. Bruckstein, R. Holt, and A. Netravali, *Discrete elastica*, Appl. Anal., 78 (2001), pp. 453–485.
[10] A. Bruckstein, A. Netravali, and T. Richardson, *Epi-convergence of discrete elastica*, Appl. Anal., 79 (2001), pp. 137–171.
[11] J. Cai, R. Chan, and Z. Shen, *A framelet-based image inpainting algorithm*, Appl. Comput. Harmon. Anal., 24 (2008), pp. 131–149.
[12] F. Cao, Y. Gousseau, S. Masnou, and P. Pérez, *Geometrically guided exemplar-based inpainting*, Submitted.
[13] V. Caselles, J.M. Morel, and S. Catalina, *Axiomatic approach to image interpolation*, IEEE Trans. Image Process., 7 (1998), pp. 376–386.
[14] A. Chambolle, *An algorithm for total variational minimization and applications*, J. Math. Imaging Vis., 20 (2004), pp. 89–97.
[15] T.F. Chan, G.H. Golub, and P. Mulet, *A nonlinear primal-dual method for total variation-based image restoration*, SIAM J. Sci. Comput., 20 (1999), pp. 1964–1977.
[16] T. Chan, J. Shen, and H. M. Zhou, *Total variation wavelet inpainting*, J. Math. Imaging Vision, 25 (2006), pp. 107–125.
[17] T. F. Chan, S.-H. Kang, and J. Shen, *Euler's elastica and curvature based inpaintings*, SIAM J. Appl. Math., 63 (2002), pp. 564–594.
[18] T. F. Chan and J. Shen, *Nontexture inpainting by curvature driven diffusion (CDD)*, J. Visul Comm. Image Rep., 12 (2001), pp. 436–449.
[19] ———, *Mathematical models for local nontexture inpaintings*, SIAM J. Appl. Math, 62 (2002), pp. 1019–1043.
[20] T. F. Chan and J. Shen, *Variational image inpainting*, Comm. Pure Applied Math., 58 (2005), pp. 579–619.
[21] A. Criminisi, P. Pérez, and K. Toyama, *Object removal by exemplar-based inpainting*, in IEEE Int. Conf. Comp. Vision and Pattern Recog., 2003, pp. 721–728.
[22] A. Efros and T. Leung, *Texture synthesis by non-parametric sampling*, in IEEE International Conference on Computer Vision, Corfu, Greece, 1999, pp. 1033–1038.
[23] N. El-Zehiry and L. Grady, *Fast global optimization of curvature*, in IEEE Conference on CVPR, 2010, pp. 3257 – 3264.
[24] S. Esedoḡlu and R. March, *Segmentation with depth but without detecting junctions*, J. Math. Imag. Vis., 18 (2003), pp. 7–15.
[25] S. Esedoḡlu, S. Ruuth, and R. Tsai, *Threshold dynamics for shape reconstruction and disocclusion*, in In Proc. Int. Conf. Image Processing, 2005, pp. 502–505.
[26] S. Esedoḡlu and J. Shen, *Digital inpainting based on the Mumford-Shah-Euler image model*, European J. Appl. Math., 13 (2002), pp. 353–370.
[27] E. Esser, *Applications of Lagrangian-based alternating direction methods and connections to split Bregman*, tech. report, UCLA CAM Report 09-31, 2009.
[28] E. Esser, X. Zhang, and T. Chan, *A general framework for a class of first order primal-dual algorithms for TV minimization*, tech. report, UCLA CAM Report 09-67, 2009.
[29] W. Freeman, E. Pasztor, and O. Carmichael, *Learning low-level vision*, Int. J. Comput. Vis, 40 (2000), pp. 25–47.
[30] T. Goldstein and S. Osher, *The split Bregman method for L1-regularized problems*, SIAM J. Img. Sci., 2 (2009), pp. 323–343.
[31] J. Hahn, C. Wu, and X.-C. Tai, *Augmented Lagrangian method for generalized TV-Stokes model*, tech. report, UCLA CAM Report, 2010.
[32] B. Horn, *The curve of least energy*, ACM Transactions on Math. Software, 9 (1983), pp. 441–460.
[33] M. Kallay, *Plane curves of minimal energy*, ACM Transactions on Math. Software, 12 (1986), pp. 219–222.
[34] ———, *Method to approximate the space curve of least energy and prescribed length*, Computer-Aided Design, 19 (1987), pp. 73–76.
[35] P. L. Lions and B. Mercier, *Splitting algorithms for the sum of two nonlinear opertors*, SIAM J. Num. Anal., 16 (1979), pp. 964–979.
[36] T. Lu, P. Neittaanmaki, and X.-C. Tai, *A parallel splitting up method for partial differential equations and its application to Navier-Stokes equations*, RAIRO Math. Model. and Numer. Anal., 26 (1992), pp. 673–708.
[37] M. Lysaker and X.-C. Tai, *Iterative image restoration combining total variation minimization and a second-order functional*, Int. J. Comput. Vis., 66 (2006), pp. 5–18.
[38] A. Marquina and S. Osher, *Image super-resolution by TV-regularization and Bregman iteration*, J. Sci. Comput., 37 (2008), pp. 367–382.
[39] S. Masnou and J.-M. Morel, *Level lines based disocclusion*, in Proc. IEEE Int. Conf. on

Image Processing, Chicago, IL, 1998, pp. 259–263.
[40] S. Masnou and J. M. Morel, *Disocclusion: A variational approach using level lines*, IEEE Trans. Image Process., 11 (2002), pp. 68–76.
[41] S. Masnou and J.-M. Morel, *On a variational theory of image amodal completion*, Rend. Sem. Mat. Univ. Padova, 116 (2006), pp. 211–252.
[42] M. Nitzberg, D. Mumford, and T. Shiota, *Filtering, Segmentation and Depth*, vol. 662, Springer-Verlag, Berlin, 1993.
[43] T. Pock, D. Cremers, H. Bischof, and A. Chambolle, *An algorithm for minimizing the Mumford-Shah functional*, in IEEE International Conference on Computer Vision (ICCV) Kyoto, Japan, 2009.
[44] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Physica D, 60 (1992), pp. 259–268.
[45] T. Schoenemann, F. Kahl, and D. Cremers, *Curvature regularity for region-based image segmentation and inpainting: A linear programming relaxation*, in IEEE Int. Conf. Comp. Vision, Kyoto, Japan, 2009.
[46] S. Setzer, *Splitting Bregman algorithm, Douglas-Rachford splitting and frame shrinkage*, in SSVM '09: Proceedings of the Second International Conference on Scale Space and Variational Methods in Computer Vision, Berlin, Heidelberg, 2009, Springer-Verlag, pp. 464–476.
[47] G. Steidl and T. Teuber, *Removing multiplicative noise by Douglas-Rachford splitting methods*, J. Math. Imaging Vision, 36 (2010), pp. 168–184.
[48] X.-C. Tai and C. Wu, *Augmented Lagrangian method, dual methods and split Bregman iteration for ROF model*, in SSVM '09: Proceedings of the Second International Conference on Scale Space and Variational Methods in Computer Vision, Berlin, Heidelberg, 2009, Springer-Verlag, pp. 502–513.
[49] L.Y. Wei, M. Levoy, H. Bischof, and A. Chambolle, *Fast texture synthesis using tree-structured vector quantization*, in ACM Trans. On Graphics (SIGGRAPH00), 2000, pp. 479–488.
[50] J. Weickert, B. M. ter Harr Romeny, and M. A. Viergever, *Efficient and reliable schemes for nonlinear diffusion filtering*, IEEE Trans. Image Process., 7 (2001), pp. 398–410.
[51] P. Weiss, L. Blanc-Féraud, and G. Aubert, *Efficient schemes for total variation minimization under constraints in image processing*, SIAM J. Sci. Comp., 31 (2009), pp. 2047–2080.
[52] C. Wu and X.-C. Tai, *Augmented Lagrangian method, dual methods, and split Bregman iteration for ROF, vectorial TV, and high order models*, tech. report, UCLA CAM Report 09-76, 2009.
[53] C. Wu, J. Zhang, and X.-C. Tai, *Augmented Lagrangian method for total variation restoration with non-quadratic fidelity*, tech. report, UCLA CAM Report 09-82, 2009.
[54] Y. Yang, Y. Zhang, and W. Yin, *An efficient TVL1 algorithm for deblurring multichannel images corrupted by impulsive noise*, SIAM J. Sci. Comput., 31 (2009), pp. 2842–2865.
[55] W.T. Yin, S. Osher, D. Goldfarb, and F. Darbon, *Bregman iterative algorithms for compressed sensing and related problems*, SIAM J. Img. Sci., 1 (2008), pp. 143–168.
[56] X. Zhang, M. Burger, and S. Osher, *A unified primal-dual algorithm framework based on Bregman iteration*, tech. report, UCLA CAM Report 09-99, 2009.
[57] M. Zhu and T. Chan, *An efficient primal-dual hybrid gradient algorithm for total variation image restoration*, tech. report, UCLA CAM Report 08-34, 2008.
[58] W. Zhu and T. Chan, *Image denoising using mean curvature Available in: http://www.cims.nyu.edu/ wzhu/meancurvature06Nov30.pdf*, (2009).