Andrés Bruhn
Thomas Pock
Xue-Cheng Tai (Eds.)

LNCS 8293

Efficient Algorithms for Global Optimization Methods in Computer Vision

International Dagstuhl Seminar
Dagstuhl Castle, Germany, November 20–25, 2011
Revised Selected Papers

Lecture Notes in Computer Science **8293**

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

For further volumes:
http://www.springer.com/series/7412

Andrés Bruhn · Thomas Pock
Xue-Cheng Tai (Eds.)

Efficient Algorithms for Global Optimization Methods in Computer Vision

International Dagstuhl Seminar
Dagstuhl Castle, Germany, November 20–25, 2011
Revised Selected Papers

Springer

Editors
Andrés Bruhn
University of Stuttgart
Stuttgart
Germany

Thomas Pock
Graz University of Technology
Graz
Austria

Xue-Cheng Tai
University of Bergen
Bergen
Norway

ISSN 0302-9743 ISSN 1611-3349 (electronic)
ISBN 978-3-642-54773-7 ISBN 978-3-642-54774-4 (eBook)
DOI 10.1007/978-3-642-54774-4
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014934686

LNCS Sublibrary: SL6 – Image Processing, Computer Vision, Pattern Recognition, and Graphics

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Most of the leading algorithms in computer vision are based on global optimization methods. Such methods compute the solution of a given problem as minimizer of a suitable cost functional that penalizes deviations from previously made assumptions and integrates them in a global manner, i.e., over the entire image domain. Since this way of modelling is very transparent and intuitive, it is not surprising that such methods have become popular and successful tools to tackle many fundamental problems in computer vision such as, e.g., motion estimation, stereo reconstruction, image restoration, and object segmentation. However, there is also a price to pay when employing global optimization methods. The corresponding cost functionals often lead to optimization problems that are both mathematically challenging and computationally expensive.

In order to discuss recent advances and challenges in the design and the solution of global optimization methods, Dagstuhl Seminar 11471 on *Efficient Algorithms for Global Optimisation Methods in Computer Vision* was held during November 20–25, 2011, at the International Conference and Research Center (IBFI), Schloss Dagstuhl, near Wadern in Germany. The seminar focused on the *entire* algorithmic development pipeline for global optimization problems in computer vision: *modelling, mathematical analysis, numerical solvers,* and *parallelization*. In particular, the goal of the seminar was to bring together researchers from all four fields to analyze and discuss the connections between the different stages of the algorithmic design pipeline. The seminar included researchers from the fields of *computer science* and *mathematics alike*.

From all submissions, eight high-quality full articles were finally accepted after a strict reviewing process. Each article was reviewed by at least two international experts in the field and only articles with exclusively positive reviews were accepted for publication. The accepted articles reflect the state of the art in the field and focus on recent developments in efficient approaches for continuous optimization and related parallelization aspects on high-end cluster systems.

We would like to thank the team at castle Dagstuhl for the professional support and the perfect atmosphere during the seminar. Furthermore, we would like to thank the participants of the Dagstuhl seminar for their active discussions, their dedication during the seminar as well as for the quality of their timely reviews. Apart from all authors, we would also like to thank Ke Chen and Martin Welk for providing additional reviews. The organization of this event would not have been possible without the effort and the enthusiasm of many people. We would like to thank all who contributed to the success of this seminar.

November 2013

Andrés Bruhn
Thomas Pock
Xue-Cheng Tai

Contents

Dense Elastic 3D Shape Matching

Frank R. Schmidt[1(✉)], Thomas Windheuser[2],
Ulrich Schlickewei[2], and Daniel Cremers[2]

[1] BIOSS Center of Biological Signalling Studies,
University Freiburg, Freiburg, Germany
schmidt@cs.uni-freiburg.de
[2] Technische Universität München, München, Germany

Abstract. We propose a novel method for computing a geometrically consistent and spatially dense matching between two 3D shapes X and Y by means of a convex relaxation. Rather than mapping points to points we match infinitesimal surface patches while preserving the geometric structures. In this spirit, we consider matchings between objects' surfaces as diffeomorphisms which are by definition geometrically consistent. Since such diffeomorphisms can be represented as closed surfaces in the product space $X \times Y$, we are led to a minimal surface problem in a four-dimensional space. The proposed discrete formulation describes the search space with linear constraints which leads to a binary linear program. We propose an approximation approach to this potentially NP-hard problem. To overcome memory limitations, we also propose a multi-scale approach that refines a coarse matching until it reaches the finest level. As cost function for matching, we consider a thin shell energy, measuring the physical energy necessary to deform one shape into the other. Experimental results demonstrate that the proposed LP relaxation allows to compute high-quality matchings which reliably put into correspondence articulated 3D shapes. To our knowledge, this is the first solution to dense elastic surface matching which does not require an initialization and provides solutions of bounded optimality.

1 Introduction

1.1 Shape Similarity and Elastic Matching

Computing the similarity of 3D objects is among the central challenges both for humans and computers in order to structure the world around them. How can one determine that two hands are similar, that two faces are similar? A closer analysis of this problem reveals that the estimation of shape similarity is tightly coupled to the estimation of correspondence: Two hands in different articulation, for example, turn out to be similar because respective fingers all match a corresponding finger. The similarity of two given shapes can therefore be determined by finding the minimal elastic deformation which matches one shape into the other. While there exist algorithms for dense elastic surface matching, these algorithms typically only determine a locally optimal solution. They

A. Bruhn et al. (Eds.): Global Optimization Methods, LNCS 8293, pp. 1–18, 2014.
DOI: 10.1007/978-3-642-54774-4_1,

Fig. 1. Geometrically Consistent Elastic Matching. We propose to cast the dense elastic matching of surfaces in $3D$ as a codimension-two minimal surface problem which aims at minimizing the distortion when transforming one shape into the other. We show that a consistent discretization of this minimal surface problem gives rise to an integer linear program. By means of LP relaxation we can compute near-optimal matchings such as the one shown above. These matchings are dense triangle-wise matchings. (For visualization we combined triangles to patches and colored them consistently with their corresponding patch.)

require appropriate initialization and can therefore not be employed in a fully unsupervised manner. In particular, the accurate and unsupervised comparison of more sophisticated shapes remains an important challenge.

In this work, we propose a novel framework for finding an optimal geometrically consistent matching between two surfaces. We formulate shape matching as a minimal surface problem which allows for a linear programming discretization. This model comes with a sound physical interpretation and allows to compute high-quality matching without need for initialization. In parts, this work has been presented at vision conferences [1,2]. The goal of this paper is to combine both, the linear programming approach and the related multi-scaling approach into one comprehensive paper.

1.2 Related Work

While the matching of two different 3D shapes constitutes a very difficult problem, the problem becomes much easier in one less dimension. It is interesting to note that the matching of mere *planar* shapes can be solved by means of dynamic programming in runtimes which are subcubic in the number of points on each shape [3]. This is because the matching of planar shapes can be cast as a problem of finding a shortest closed curve of certain homotopy in a planar graph.

Unfortunately, the concepts of dynamic programming and variants of Dijkstra's algorithm do not extend to the third dimension where the solution is no

longer a shortest path but a minimal closed surface in a higher-dimensional space. Therefore, existing approaches for three-dimensional shape matching typically rely on *local* optimization techniques. Our approach tries to overcome this limitation by describing these closed surfaces with linear constraints involving the boundary operator. Inspired by Sullivan [4], the boundary operator was previously introduced in the context of image segmentation by Grady [5] and Schoenemann et al. [6].

The paradigm of the Gromov–Hausdorff framework, proposed by Mémoli and Sapiro in [7], is to find the correspondence which minimizes the geodesic distortion. Bronstein et al. [8] proposed an efficient method for computing such correspondences in a coarse-to-fine strategy much akin to optical flow algorithms. In [9] the same Gromov–Hausdorff framework was merged with the idea of diffusion distances. Other approaches to shape matching employ techniques from conformal geometry [10,11] or Riemannian geometry [12]. The physically motivated energy model we use in this work is related to the works of Litke et al. [13] and of Rumpf and Wirth [14].

All the above-mentioned methods have in common that they use a local optimization technique to minimize a non-convex energy. As a consequence, the quality of solutions depends heavily on a good initialization and an appropriately designed coarse-to-fine strategy. In addition, solutions do not come with any optimality guarantees, which implies that in principle they can be arbitrarily bad. To overcome these problems, methods with a more *global* flavor have been recently proposed.

On the one hand, Zeng and coworkers [15] formulate shape matching as a graph matching problem of third order and apply the QPBO algorithm [16]. Although the overall approach does not guarantee globally optimal solutions, it is able to detect when a proposed matching pair is globally optimal. Two major drawbacks of this approach are that firstly it suffers from a very high computational complexity, considering all triples of possible matchings. In practice it allows only the matching of a few feature points which is then post-processed with a local method. Secondly, this approach lacks a continuous counterpart, as it merely matches discrete points rather than surface elements.

On the other hand, Lipman and Daubechies [17] recently proposed to compare surfaces of genus zero and open surfaces using optimal mass transport and conformal geometry. Computationally, this amounts to solving a linear program in n^2 variables where n is the number of vertices used in the discretization of the surfaces. The problem with this approach is that no spatial regularity is imposed on the matchings.

1.3 Contribution

We propose a novel formulation for the shape matching problem based on finding an optimal surface of codimension 2 in the product of the two shape surfaces. This surface minimizes the physical deformation energy needed for deforming one shape into the other. We derive a consistent discretization of the continuous framework and show that the discrete minimal surface problem amounts to a

linear program. Compared to existing approaches the proposed framework has the following advantages:

- The LP formulation is a global approach allowing to compute matchings which are independent of initialization with no post-processing.
- The proposed method guarantees a geometrically consistent matching in the sense that the surfaces are mapped into one another in a continuous and orientation preserving manner.
- We provide a discretization of the set of diffeomorphisms by means of linear constraints. This is remarkable because in previous formulations the diffeomorphism constraint is non-linear and computationally very difficult [18].
- The algorithmic formulation is independent of the particular choice of deformation energy and can be applied universally. As an example, we show that one can also incorporate local feature similarity in order to improve performance.
- In order to be independent of potential memory limitations, we propose a multiscale-approach that starts with a coarse matching that is then refined in every iteration.
- Experiments demonstrate that reliable and dense matchings are obtained even for larger problem instances with no need for post-processing.

The paper is organized as follows. In Sect. 2 we present the relationship between 3D shape matching and the computation of a minimal surface in a 4D manifold. In Sect. 3 we present the discretization that we use in order to model arbitrary surfaces in the induced 4D space and in Sect. 4 we address the problem of finding an approximation of the involved integer linear program. After introducing in Sect. 5 a multi-resolution approach in order to also compute dense shape matching, we provide shape matching results in Sect. 6. Section 7 concludes this work.

2 From Continuous Shape Matching to Minimal Surfaces

One of our goal is to cast the shape matching problem as the computation of a minimal surface in a four-dimensional space. In Sect. 2.1 we formulate the overall energy that we want to minimize. It combines a physically motivated membrane energy with a bending energy. Subsequently, we show in Sect. 2.2 how this problem can be translated into an equivalent problem of finding a minimal codimension-two surface in the product space of the two involved shapes.

2.1 Shape Matching Based on Minimizing Deformation Energies

In the following, we assume that the two shapes $X, Y \subset \mathbb{R}^3$ are differentiable, oriented, closed surfaces. While most 3D shape matching approaches like to interpret a matching just as a bijective mapping between the surface points of these shapes, we pursue a fundamentally different approach. The main reason is

that in general, bijections do not respect the underlying two-dimensional structure of surfaces. In fact, there are even continuous bijections between a line and a two-dimensional patch like the continuous *space-filling curve* as shown by Hilbert [19]. Therefore, we propose to search for diffeomorphisms instead of bijections.

Diffeomorphisms $f : X \to Y$ are bijections for which both, f and f^{-1} are differentiable. This does not only cope with the dimensionality problem presented above, but it also helps us to propose an energy function that is symmetric in f and f^{-1}. Since both f and its inverse exist, the optimal matching f between X and Y also gives rise to the optimal matching between Y and X, namely $f^{-1} : Y \to X$. The set of diffeomorphisms $\mathrm{Diff}(X, Y)$ can be separated in two different classes, into the class of orientation preserving diffeomorphisms $\mathrm{Diff}^+(X, Y)$ and the class of orientation reversing diffeomorphisms $\mathrm{Diff}^-(X, Y)$.

In the following, we formulate the shape matching problem as an optimization problem over $\mathrm{Diff}^+(X, Y)$:

$$\min_{f \in \mathrm{Diff}^+(X,Y)} E(f) + E(f^{-1}) \tag{1}$$

where E is a suitable energy on the class of all diffeomorphisms between surfaces. Note that we choose a symmetric problem formulation, penalizing at the same time deformation energy of X into Y and of Y into X. This is necessary because usually E takes different values on f and on f^{-1}.

The energy functional we use is borrowed from elasticity theory in physics [20]. Here, we interpret the shapes X and Y as surfaces or "thin shells". If we now try to find the deformation of X into Y, it requires a certain amount of stretching and bending. This results in an energy that usually combines a membrane energy E_{mem} and a bending energy E_{bend} penalizing deformations in the first and in the second fundamental forms of the surfaces. In this work we use the following formulation:

$$E(f) = \underbrace{\int_X (\mathrm{tr}_{g_X} \mathbf{E}) + \mu \mathrm{tr}_{g_X}(\mathbf{E}^2)}_{E_{\mathrm{mem}}} + \lambda \underbrace{\int_X (H_X(x) - H_Y(f(x))^2}_{E_{\mathrm{bend}}} \tag{2}$$

where $\mathbf{E} = f^* g_Y - g_X$ is the difference between the metric tensors of X and Y, typically called the *Lagrange strain tensor*, $\mathrm{tr}_{g_X}(\mathbf{E})$ is the norm of this tensor (see [21]), H_X and H_Y denote the mean curvatures and μ and λ are parameters which determine the elasticity and the bending property of the material. This energy is a slightly simplified version of Koiter's thin shell energy [22].

After presenting the overall energy $E(f) + E(f^{-1})$ that we want to minimize, we will reformulate this problem in the next section into a problem of finding a minimal surface in a four-dimensional space.

2.2 Shape Matchings and Their Graph Surfaces

As we mentioned in Sect. 2.1, every matching function $f : X \to Y$ between two shapes X and Y is an orientation preserving diffeomorphism. Given such a

matching, its graph

$$\Gamma = \{\, (x, f(x)) \,|\, x \in X \} \tag{3}$$

becomes a surface in the four-dimensional[1] product space $X \times Y$. This surface Γ comes with the two natural projections:

$$\begin{aligned} \pi_X : \Gamma &\to X & \qquad \pi_Y : \Gamma &\to Y \\ (x, y) &\mapsto x & \qquad (x, y) &\mapsto y \end{aligned}$$

that will help us to characterize a diffeomorphism completely by its graph:

Proposition 1 (Graph Surfaces). *Let Γ be the graph of a diffeomorphism $f : X \to Y$. Then*

1. *Γ is a differentiable, connected, closed surface in the product space $X \times Y$.*
2. *The projections π_X and π_Y are both diffeomorphisms.*
3. *The two orientations which Γ naturally inherits from X and Y coincide.*

Vice versa, any surface $\Gamma \subset X \times Y$ which satisfies Conditions 1–3 is the graph of an orientation-preserving diffeomorphism between X and Y. We call such surfaces **graph surfaces**.

The energy $E(f) + E(f^{-1})$ can be expressed as $\widetilde{E}(\Gamma)$ via

$$E(f) + E(f^{-1}) = E(\pi_Y \circ (\pi_X)^{-1}) + E(\pi_X \circ (\pi_Y)^{-1}) =: \widetilde{E}(\Gamma). \tag{4}$$

Concluding the above discussion, we have transformed the optimization problem (1) into an optimization problem over the set of all *graph surfaces* in $X \times Y$, namely

$$\begin{aligned} &\min && \widetilde{E}(\Gamma) \\ &\text{subject to} && \Gamma \subset X \times Y \text{ is a graph surface} \end{aligned} \tag{5}$$

We remark that the idea of casting optimal diffeomorphism problems as minimal surface problems has been applied previously in the theory of nonlinear elasticity [23]. In the setup of shape matching, it is related to the approach that Tagare [24] proposed for the matching of 2D shapes. Its connection to orientation preserving diffeomorphisms was made in [3].

3 The Discrete Setting

In this section we develop a discrete representation of graph surfaces introduced in Sect. 2.2. We start in Sect. 3.1 with the definition of discrete surface patches in $X \times Y$. These patches are derived from a given discrete triangulation of X and Y itself. The surface patches in the product space $X \times Y$ are the building blocks for discrete graph surfaces that we introduce in Sect. 3.2. Finally in Sect. 3.3 we give a discrete version of the energy minimization problem (5).

[1] $X \times Y$ is a 4-manifold embedded in $\mathbb{R}^6$.

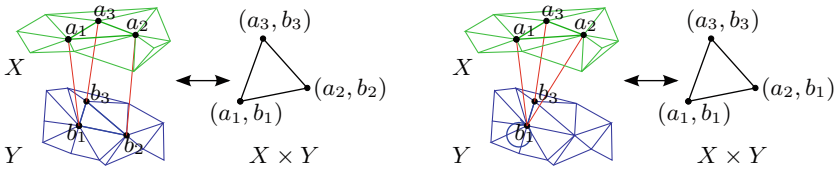


Fig. 2. Product Triangles. To assure a geometrically consistent, elastic matching from mesh X to mesh Y, we define a space of feasible solutions which is spanned by a set of 45 basic matchings among triangles, edges and vertices on either mesh. Two representative matchings and their corresponding representation in the product space $X \times Y$ are shown. ***Left image:*** The triangle $(a_1, a_2, a_3)^T$ on surface X is matched to triangle $(b_1, b_2, b_3)^T$ on Y by assigning vertex a_i to vertex b_i. This directly corresponds to the triangle with vertices (a_i, b_i) in the product graph. ***Right image:*** The triangle $(a_1, a_2, a_3)^T$ is matched to the edge $(b_1, b_3)^T$, represented here as degenerate triangle $(b_1, b_1, b_3)^T$.

3.1 Discrete Surface Patches

In the following, we assume that a shape X is given as a triangulated oriented surface mesh $G_X = (V_X, E_X, F_X)$, consisting of a set of vertices V_X, a set of directed edges E_X and a set of oriented triangles F_X.

While the orientation of X defines a natural orientation of the faces in F_X, such a natural orientation does not exist for the edge in E_X. Moreover, two faces $f_1, f_2 \in F_X$ that touch each other along an edge $e \in E_X$ induce opposite orientations onto this edge e. Since edges on X do not have a preferable orientation, we fix an orientation for each edge on X. Thus, whenever two vertices a_1 and a_2 of X are connected by an edge, either

$$\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \in E_X \qquad \text{or} \qquad \begin{pmatrix} a_2 \\ a_1 \end{pmatrix} = -\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \in E_X.$$

For simplicity, we extend the set of edges by *degenerate edges*

$$\overline{E}_X = E_X \cup \left\{ \begin{pmatrix} a \\ a \end{pmatrix} \;\middle|\; a \in V_X \right\}. \tag{6}$$

By assumption, the triangular faces of X are oriented. If the vertices a_1, a_2, a_3 build an oriented triangle on X, then

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} a_2 \\ a_3 \\ a_1 \end{pmatrix} = \begin{pmatrix} a_3 \\ a_1 \\ a_2 \end{pmatrix} \in F_X$$

and analogously to the edges, we extend the set of triangles by *degenerate triangles*

$$\overline{F}_X = F_X \cup \left\{ \begin{pmatrix} a_1 \\ a_2 \\ a_2 \end{pmatrix} \;\middle|\; a_1, a_2 \in V_X,\ \pm \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \in \overline{E}_X \right\}. \tag{7}$$

Due to the definition of $\overline{E}_X$, degenerate triangles can consist of two vertices or even of only one vertex. The existence of these degenerate triangles will allow stretching or compression of parts of the shapes as we will see below (cf. right image of Figs. 2 and 6).

Next, we define triangles F, edges E and vertices V that operate as building blocks for the graph surfaces introduced in Sect. 2.2. To this end, let us assume that two shapes X and Y are given as triangulated oriented surface meshes $G_X = (V_X, E_X, F_X)$ resp. $G_Y = (V_Y, E_Y, F_Y)$ and that $\overline{E}_X, \overline{F}_X, \overline{E}_Y, \overline{F}_Y$ are defined as above. Then define the graph $G = (V, E, F)$ of the product space $X \times Y$ by

$$\begin{aligned} V :=& V_X \times V_Y \\ E :=& \overline{E}_X \times \overline{E}_Y \\ F :=& (\overline{F}_X \times F_Y) \cup (F_X \times \overline{F}_Y) \end{aligned}$$

The product triangles in F are the pieces which are later glued together in order to obtain discrete graph surfaces. For shape matching, a product triangle

$$\begin{pmatrix} (a_1, b_1) \\ (a_2, b_2) \\ (a_3, b_3) \end{pmatrix} \in F \tag{8}$$

is interpreted as setting vertex $a_i \in V_X$ in correspondence with vertex $b_i \in V_Y$. While a triangle provides us with such a point-to-point matching, it also takes care of the geometric structure within the two shapes X and Y. In that sense it is more powerful than a mere point matching.

Given two non-degenerate triangles $a \in F_X$ and $b \in F_Y$, we allow for 45 different matchings between them:

- 3 orientation-preserving bijective matchings,
- 36 triangle-to-edge matchings and
- 6 triangle-to-vertex matchings.

The degenerate triangle-to-edge and triangle-to-vertex matchings allow us to handle infinitesimal stretching and compression in the proposed framework. Visualizations for two of the 45 possibilities is given in Fig. 2.

3.2 Discrete Surfaces

Following Sect. 2, a diffeomorphism can be represented as a surface $\Gamma \subset X \times Y$ satisfying conditions 1–3 of Proposition 1. In this section we derive discrete versions of these three properties. First, we define a surface in the product space:

Definition 1. *A* **discrete surface** *in* $G = (V, E, F)$ *is a subset* $\Gamma \subset F$*. The set of all discrete surfaces is denoted by* $\mathrm{surf}(G)$.

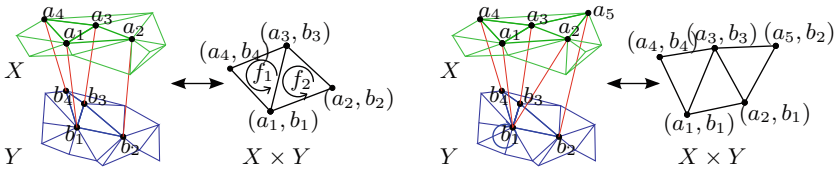


Fig. 3. Geometric Consistency. To ensure that neighboring triangles on X are matched with neighboring triangles on Y, we impose the closedness condition. **Left image (general case):** The triangles $(a_1, a_2, a_3)^T$ and $(b_1, b_2, b_3)^T$ are matched, thereby activating f_2. The boundary condition $\partial\Gamma = 0$ ensures that the matching continues with a correspondence whose triangles in X and Y are positively incident to $(a_1, a_3)^T$ and $(b_1, b_3)^T$ respectively. This constraint is satisfied for example by triangle f_1 which is visualized here. **Right image (stretching):** The stretching is achieved by matching triangle $(a_1, a_2, a_3)^T$ to edge $(b_3, b_1)^T$. Again, the geometric consistency is granted by the boundary condition evaluated on the product edges $((a_2, b_1), (a_3, b_3))^T$ and $((a_3, b_3), (a_1, b_1))^T$.

As we have seen in Sect. 3.1, any triangle in F can be interpreted as matching a (possibly degenerated) triangle of $\overline{F}_X$ to a (possibly degenerated) triangle of $\overline{F}_Y$. Thus, the intuitive meaning of a discrete surface $\Gamma \subset F$ is a set of point correspondences between the shapes X and Y. Imposing the discrete counterparts of 1–3 on such a discrete surface will result in a geometrically consistent matching that approximates a diffeomorphism between the continuous counter-parts of X and Y.

Discrete Version of Condition 1. In the following we will find a condition which guarantees the continuity of our matching. Recall that the boundary operator for triangle meshes [25] maps triangles to their oriented boundary. We extend this definition to the product graph G.

As for the sets E_X and E_Y we choose arbitrary orientations for each product edge $e \in E$. By means of these orientations we define for any edge $(v_1, v_2)^\top \in E$ connecting two vertices $v_1, v_2 \in V$ a vector $O\left((v_1, v_2)^\top\right) \in \mathbb{Z}^{|E|}$ whose e-th entry is given by

$$O\left((v_1, v_2)^\top\right)_e = \begin{cases} 1 & \text{if } e = \binom{v_1}{v_2} \\ -1 & \text{if } e = \binom{v_1}{v_2} \\ 0 & \text{else.} \end{cases} \tag{9}$$

The triangles in F naturally inherit orientations from the triangles in F_X and F_Y. This allows us to define the boundary operator as follows.

Definition 2. *The* boundary operator $\partial : F \to \mathbb{Z}^{|E|}$ *is defined by*

$$\partial \begin{pmatrix} (a_1, b_1) \\ (a_2, b_2) \\ (a_3, b_3) \end{pmatrix} := O \begin{pmatrix} (a_1, b_1) \\ (a_2, b_2) \end{pmatrix} + O \begin{pmatrix} (a_2, b_2) \\ (a_3, b_3) \end{pmatrix} + O \begin{pmatrix} (a_3, b_3) \\ (a_1, b_1) \end{pmatrix}, \tag{10}$$

where the $a_i \in V_X$ *and* $b_i \in V_Y$ *form triangles on* X *resp. on* Y *and* $\binom{(a_i,b_i)}{(a_j,b_j)} \in E$ *connects the vertices* $(a_i, b_i) \in V$ *with* $(a_j, b_j) \in V$*. The boundary operator is linearly extended to a map*

$$\partial : \mathrm{surf}(G) \to \mathbb{Z}^{|E|}. \tag{11}$$

and a discrete surface Γ *in* G *is* **closed** *if* $\partial\Gamma = 0$*.*

The closedness condition ensures that adjacent triangles on X are in correspondence with adjacent triangles on Y and therefore guarantees the geometric consistency (see Fig. 3). The natural discrete version of Condition 1 is a closed, connected discrete surface in G.

Discrete Version of Condition 2. Analogously to the continuous case, we can project product triangles of F to triangles of the surfaces X and Y.

Definition 3. *The projection* $\pi_X : F \to \mathbb{Z}^{|F_X|}$ *is defined by*

$$\pi_X \begin{pmatrix} (a_1,b_1) \\ (a_2,b_2) \\ (a_3,b_3) \end{pmatrix} := \begin{cases} e_a & \text{if } a = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \in F_X \\ (0,\ldots,0) & \text{else} \end{cases}. \tag{12}$$

Here, e_a *is the base vector with 1 in the* a*-entry and 0 in all other entries. We extend the projection* π_X *linearly to a map* $\pi_X : \mathrm{surf}(G) \to \mathbb{Z}^{|F_X|}$*. The projection* $\pi_Y : F \to \mathbb{Z}^{|F_Y|}$ *and its linear extension* $\pi_Y : \mathrm{surf}(G) \to \mathbb{Z}^{|F_Y|}$ *are defined analogously.*

Let now Γ be a discrete surface in G. Then we say that the projections of Γ to X and Y are discrete diffeomorphisms if and only if

$$\pi_X(\Gamma) = (1,\ldots,1) \in \mathbb{Z}^{|F_X|} \qquad \text{and} \qquad \pi_Y(\Gamma) = (1,\ldots,1) \in \mathbb{Z}^{|F_Y|}. \tag{13}$$

This gives a discrete version of Condition 2.

Note that in this definition we do not ask for injectivity on the vertices set. This is necessary for modelling discretely strong compressions. However, conditions (13) ensure a global bijectivity property which is sufficient in our context.

Discrete Version of Condition 3. By definition, the set of surfaces in G only contains surface patches which are consistently oriented. Therefore any surface in $\mathrm{surf}(G)$ satisfies Condition 3.

Definition 4. *Let* $\Gamma \in \{0,1\}^{|F|}$ *be a discrete surface in* G*, represented by its indicator vector. Then* Γ *is a* **discrete graph surface** *in* G *if*

$$\begin{pmatrix} \partial \\ \pi_X \\ \pi_Y \end{pmatrix} \cdot \Gamma = \begin{pmatrix} \mathbf{0} \\ \mathbf{1} \\ \mathbf{1} \end{pmatrix}. \tag{14}$$

3.3 Discrete Surface Energy

Now we introduce a discrete energy on the set of product triangles in G. For the membrane energy in (2) we adopt the term proposed by Delingette [26]. Given two triangles $T_1, T_2 \subset \mathbb{R}^3$, Delingette computes the stretch energy $E_{\text{mem}}(T_1 \to T_2)$ necessary for deforming T_1 in T_2. In our framework we associate with each product triangle $(a, b) \in F$ consisting of $a = (a_1, a_2, a_3)^\top \in \overline{F}_X$ and $b = (b_1, b_2, b_3)^\top \in \overline{F}_Y$ the membrane cost

$$E_{\text{mem}}(a, b) := E_{\text{mem}}\left(\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \to \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}\right) + E_{\text{mem}}\left(\begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \to \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}\right). \tag{15}$$

For the bending term we proceed similarly associating with each product triangle (a, b) the cost

$$E_{\text{bend}}(a, b) = \int_a (H_X - H_Y)^2 + \int_b (H_Y - H_X)^2. \tag{16}$$

In practice we discretize the mean curvature following [27].

Next, we extend the energy linearly from discrete surface patches to discrete surfaces in G. Identify a discrete surface with its indicator vector $\Gamma \in \{0, 1\}^{|F|}$. Define the vector $E \in \mathbb{R}^{|F|}$ whose f-th entry is

$$E_f = E_{\text{mem}}(f) + E_{\text{bend}}(f). \tag{17}$$

Then the **discrete energy** of Γ is given by the vector product

$$E^t \cdot \Gamma. \tag{18}$$

4 Linear Programming Solution

In the previous section we have introduced a discrete notion of graph surfaces (14) and a discrete deformation energy (18) for such graph surfaces. This enables us to state the discrete version of (5) in the form of a binary linear program:

$$\begin{aligned} \min_{\Gamma \in \{0,1\}^{|F|}} \quad & E^t \cdot \Gamma \\ \text{subject to} \quad & \begin{pmatrix} \partial \\ \pi_X \\ \pi_Y \end{pmatrix} \cdot \Gamma = \begin{pmatrix} \mathbf{0} \\ \mathbf{1} \\ \mathbf{1} \end{pmatrix}. \end{aligned} \tag{19}$$

For solving (19), we relax the binary constraints to $\Gamma \in [0, 1]^{|F|}$. This relaxed version can be solved globally optimally in polynomial time. We employed an alternating direction method developed by Eckstein et al. [29]. This algorithm is parallelizable which allowed us an efficient implementation on the GPU.

Since the constraint matrix of the relaxed problem is not totally unimodular, we are not guaranteed an integral solution. A simple thresholding scheme would

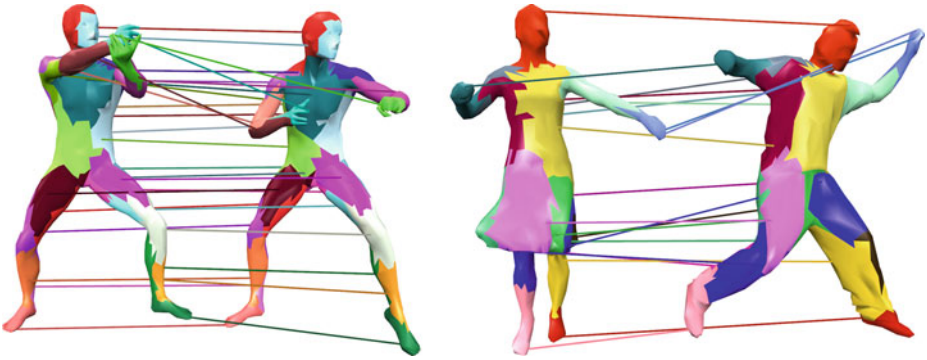


Fig. 4. SHREC 2011 benchmark and matching different objects. Left: The image illustrates the performance of the proposed method on the SHREC 2011 [28] dataset. **Right:** Matching of two different objects. While it is not well defined what a good matching between a skirt and trousers is, it is really remarkable how well the proposed algorithm finds a matching that apparently minimizes the deformation energy.

destroy the geometric consistency of the solution. Instead, we propose an iterative scheme: solve the relaxed version of (19), fix the variables with values above a threshold >0.5 to 1 and solve the relaxed version of (19) with these additional constraints. If there is no variable with value above the threshold fix only one variable with the highest value. In our experiments, this scheme typically converged to a binary solution after less than 10 iterations, in no experiment it took more than 20 iterations.

5 Multiresolution Framework

Because the number of product triangles grows quadratically with the number of triangles in both shapes the resulting Integer Linear Program (ILP) has a very high number of variables and constraints. Even the minimization of the relaxed Linear Program (LP) becomes impractical for state-of-the-art LP solvers, if the shapes have more than 250 triangles. In this section we present a multiresolution approach that overcomes this limitation and allows to match shapes of more than 2000 triangles.

The basic idea of the multiresolution approach is to solve the problem at a very coarse scale with the methods described in Sect. 4 and to recursively use the found solution to narrow the search space at the next finer level. To reduce the size of the search space we impose that a possible solution at a finer level must lie "near" an already found solution at the coarser scale.

For the definition of "near" we use a hierarchy of triangles across the resolution levels. Suppose that we obtain a triangle mesh X_i from a finer triangle mesh X_{i+1} by repeatedly merging triangles. In practice we use the quadric edge decimation algorithm [30] in its OpenMesh implementation [31]. Denote by

$\chi_i : F(X_{i+1}) \to F(X_i)$ the child-parent relation, mapping each triangle in $F_{X_{i+1}}$ to the triangle it is merged to on the coarser mesh X_i. These maps are extended to maps between the extended sets of triangles $\chi_i : \overline{F}_{X_{i+i}} \to \overline{F}_{X_i}$ (see Sect. 3.2).

Let now X and Y be two high-resolution meshes and let $X = X_n, X_{n-1}, \dots, X_0$ and $Y = Y_n, Y_{n-1}, \dots, Y_0$ be successive coarsenings with corresponding child-parent relations $\{\chi_i\}_{0 \le i \le n-1}$ and $\{\psi_i\}_{0 \le i \le n-1}$. We proceed as follows:

1. We compute a discrete graph surface Γ_0 (cf. Definition 4) inducing a matching of the coarsest meshes X_0 and Y_0. We use the methods described in Sect. 4 for this task.
2. Assuming inductively that we have found a discrete graph surface Γ_i which induces a matching of X_i and Y_i, we search for a discrete graph surface Γ_{i+1}. This surface has to lie in a search space which is reduced using the input of the already computed surface Γ_i. Rather than allowing Γ_{i+1} to be built of all product triangles F_{i+1} between X_{i+1} and Y_{i+1}, we only allow for product triangles whose parents or whose parents' neighbors are set in correspondence by Γ_i. Thus, Γ_{i+1} is searched as a subset of the reduced set of product triangles
$$F_{i+1}^{\text{red}} = \left\{ (f_a, f_b) \in F_{i+1} \;\middle|\; \begin{array}{l} \exists (f'_a, f'_b) \in \Gamma_i \subset F_i \text{ s.t.} \\ \chi_i(f_a) \in \mathcal{N}(f'_a) \text{ and} \\ \psi_i(f_a) \in \mathcal{N}(f'_b) \end{array} \right\}. \tag{20}$$
Here, for a triangle f on a triangle mesh we used the set of triangles in the one-ring or the two-ring of its vertices as neighborhood $\mathcal{N}(f)$.
Then we compute Γ_{i+1} by solving problem (19) over the reduced search space, that is $\Gamma_{i+1} \in \{0,1\}^{|F_{i+1}^{\text{red}}|}$.
3. We repeat Step 2 until a discrete graph surface Γ_n has been computed which induces a matching between X and Y.

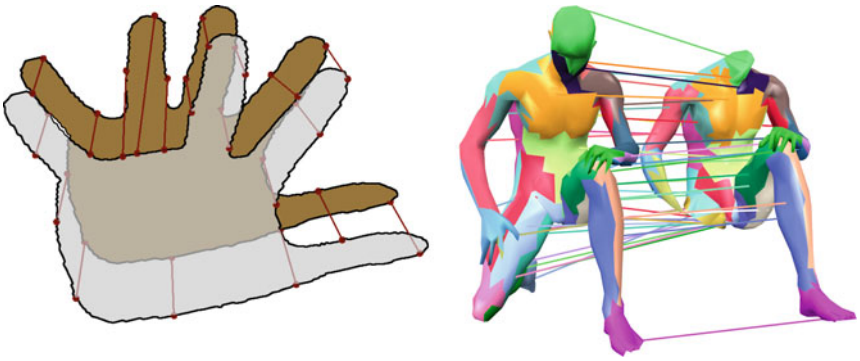
6 Experimental Results

We have introduced a framework for computing geometrically consistent elastic matchings between 3D shapes using LP relaxation. We evaluated the proposed method on several shapes taken from the SHREC 2011 benchmark [28] and a dataset by Vlasic et al. [32].

6.1 Matching of Articulated Shapes

A common problem in shape matching is that the same shape may undergo substantial deformation and articulation. Nevertheless, one would like to reliably identify corresponding structures. Figures 1 and 7 show the matchings computed for models of different articulations. Although the movement of arms and legs deform the shapes drastically the proposed method identifies the correct matchings. Since the proposed framework enforces geometric consistency matching errors occur only on a small spatial scale. In contrast to methods without spatial regularization strong outliers such as single points matched to the wrong leg do not arise.

Fig. 5. Multiresolution Framework. The images show matchings between two shapes at different resolutions. As described in Sect. 5 the computational cost is drastically reduced by first solving the matching problem at a lower resolution and then using the obtained solution to restrict the search space at the next higher resolution.



Elastic matching of planar shapes [3] Proposed elastic matching of 3D shapes

Fig. 6. 2D and 3D Shape Matching. While the elastic matching of planar shapes can be solved in polynomial time as a minimal cyclic path on a torus [3], the framework developed in this paper allows to compute an elastic matching of 3D shapes via linear programming relaxation. In both cases, missing parts can be accounted for due to the elasticity.

6.2 Partial Matching

The ability of the proposed method to model stretching and shrinking also allows to match shapes where large parts of the geometry are missing. The right image of Fig. 6 demonstrates this ability experimentally. The proposed algorithm matches the remaining parts of a human body missing a hand, a leg and the head to its original shape.

Fig. 7. Linear Interpolation. The images show a matching between the leftmost and rightmost models taken from the SHREC 2011 benchmark [28] by linearly interpolating between the triangle correspondences. This transition illustrates the geometric consistency property of the proposed method: At any time during the interpolation the model is a closed triangle mesh.

6.3 Quantitative Evaluation

We quantitatively evaluated the proposed method on 30 pairs of models from Vlasic et al. [32] by computing the mean geodesic error. One of the matchings is visualized in Fig. 1. Computing each of the matchings took about 2 hours. The results were compared to matchings generated by the GMDS method of Bronstein et al. [8] using their code.

Given two meshes X, Y and the available ground truth correspondences (x_i, y_i) we defined the mean geodesic error of a matching $\varphi : X \to Y$ by $\frac{1}{N}\sum_i d(\varphi(x_i), y_i)$, where d is the normalized geodesic distance on the manifold of mesh Y. The mean geodesic error produced by GMDS (using their code) was 0.079 while the proposed method had a mean geodesic error of 0.03.

Of course, this experiment does not pretend to be an exhaustive comparison against all methods in the literature. Nonetheless it shows, that the proposed method can compete with state-of-art matching algorithms in terms of accuracy while guaranteeing geometrically consistent solutions.

7 Conclusion

We proposed a new framework for finding a geometrically consistent matching of 3D shapes which minimizes an elastic deformation energy. The approach is

based on finding discrete minimal surfaces which set infinitesimal surface patches on both shapes into correspondence. In that sense the framework manages to generalize the planar shape matching to the more complex 3D shape matching. While the planar shape matching finds correspondences between infinitesimal line elements of two contours, the 3D shape matching sets infinitesimal surface elements in correspondence. We showed that a consistent discretization leads to an integer linear program. As a consequence, we can compute high-quality solutions to the matching problem which are independent of initialization by means of LP relaxation.

To improve the runtime and overcome possible memory limitations, we also introduced a multi-scale approach that improves iteratively matchings from a coarse level to its finest level. Experimental results confirm that the proposed method generates reliable dense correspondences for a variety of articulated real-world shapes.

Acknowledgments. The 3D shape data in Figs. 1, 4 (right) and 5 is courtesy of Vlasic et al. [32]. The 3D shape data in Figs. 4 (left), 6 and 7 is taken from the SHREC 2011 benchmark [28].

References

1. Windheuser, T., Schlickewei, U., Schmidt, F.R., Cremers, D.: Geometrically consistent elastic matching of 3D shapes: a linear programming solution. In: Proceedings of the International Conference on Computer Vision (2011)
2. Windheuser, T., Schlickewei, U., Schmidt, F.R., Cremers, D.: Large-scale integer linear programming for orientation-preserving 3D shape matching. In: Computer Graphics Forum (Proceedings of Eurographics Symposium Geometry Processing), Lausanne, Switzerland (2011)
3. Schmidt, F.R., Farin, D., Cremers, D.: Fast matching of planar shapes in sub-cubic runtime. In: Proceedings of the International Conference on Computer Vision, Rio de Janeiro, October 2007 (2007)
4. Sullivan, J.M.: A crystalline approximation theorem for hypersurfaces. Ph.D. thesis, Princeton University (1990)
5. Grady, L.: Minimal surfaces extend shortest path segmentation methods to 3D. IEEE Trans. Pattern Anal. Mach. Intell. **32**(2), 321–334 (2010)
6. Schoenemann, T., Kahl, F., Cremers, D.: Curvature regularity for region-based image segmentation and inpainting: a linear programming relaxation. In: Proceedings of the International Conference on Computer Vision, Kyoto (2009)
7. Mémoli, F., Sapiro, G.: A theoretical and computational framework for isometry invariant recognition of point cloud data. Found. Comput. Math. **5**(3), 313–347 (2005)
8. Bronstein, A., Bronstein, M., Kimmel, R.: Efficient computation of isometry-invariant distances between surfaces. SIAM J. Sci. Comput. **28**(5), 1812–1836 (2006). (electronic)
9. Bronstein, A., Bronstein, M.M., Kimmel, R., Mahmoudi, M., Sapiro, G.: A Gromov-Hausdorff framework with diffusion geometry for topologically-robust non-rigid shape matching. Int. J. Comput. Vis. **89**(2–3), 266–286 (2010)

10. Wang, Y., Gu, X., Hayashi, K.M., Chan, T.F., Thompson, P.M., Yau, S.-T.: Brain surface parameterization using Riemann surface structure. In: Duncan, J.S., Gerig, G. (eds.) MICCAI 2005. LNCS, vol. 3750, pp. 657–665. Springer, Heidelberg (2005)
11. Lui, L.M., Wong, T.W., Thompson, P., Chan, T., Gu, X., Yau, S.-T.: Shape-based diffeomorphic registration on hippocampal surfaces using beltrami holomorphic flow. In: Jiang, T., Navab, N., Pluim, J.P.W., Viergever, M.A. (eds.) MICCAI 2010, Part II. LNCS, vol. 6362, pp. 323–330. Springer, Heidelberg (2010)
12. Kurtek, S., Klassen, E., Ding, Z., Srivastava, A.: A novel Riemannian framework for shape analysis of 3D objects. In: CVPR, pp. 1625–1632 (2010)
13. Litke, N., Droske, M., Rumpf, M., Schröder, P.: An image processing approach to surface matching. In: Symposium on Geometry Processing, pp. 207–216 (2005)
14. Wirth, B., Bar, L., Rumpf, M., Sapiro, G.: Geodesics in shape space via variational time discretization. In: Cremers, D., Boykov, Y., Blake, A., Schmidt, F.R. (eds.) EMMCVPR 2009. LNCS, vol. 5681, pp. 288–302. Springer, Heidelberg (2009)
15. Zeng, Y., Wang, C., Wang, Y., Gu, X., Samaras, D., Paragios, N.: Dense non-rigid surface registration using high-order graph matching. In: CVPR, pp. 382–389 (2010)
16. Torresani, L., Kolmogorov, V., Rother, C.: Feature correspondence via graph matching: models and global optimization. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part II. LNCS, vol. 5303, pp. 596–609. Springer, Heidelberg (2008)
17. Lipman, Y., Daubechies, I.: Surface comparison with mass transportation (2009). (Preprint, arXiv:0912.3488)
18. Younes, L.: Shapes and Diffeomorphisms. Applied Mathematical Sciences, vol. 171. Springer, Heidelberg (2010)
19. Hilbert, D.: Über die stetige Abbildung einer Linie auf ein Flächenstück. Math. Ann. **38**, 459–460 (1891)
20. Ciarlet, P.: An Introduction to Differential Geometry with Applications to Elasticity. Springer, Dordrecht (2005). (Reprinted from J. Elasticity 78/79 (2005), no. 1–3)
21. Do Carmo, M.: Riemannian geometry. Birkhäuser, Boston (1992)
22. Koiter, W.: On the nonlinear theory of thin elastic shells. I, II, III. Nederl. Akad. Wetensch. Proc. Ser. B **69**, 1–17, 18–32, 33–54 (1966)
23. Giaquinta, M., Modica, G., Souček, J.: Cartesian Currents in the Calculus of Variations II: Variational Integrals. Ergebnisse der Mathematik und ihrer Grenzgebiete, vol. 38. Springer, Heidelberg (1998)
24. Tagare, H.: Shape-based nonrigid correspondence with application to heart motion analysis. IEEE Trans. Med. Imag. **18**(7), 570–579 (1999)
25. Desbrun, M., Hirani, A.N., Leok, M., Marsden, J.E.: Discrete exterior calculus (2005). (Preprint, arXiv:0508341v2)
26. Delingette, H.: Triangular springs for modeling nonlinear membranes. IEEE Trans. Vis. Comput. Graph. **14**(2), 329–341 (2008)
27. Meyer, M., Desbrun, M., Schröder, P., Barr, A.: Discrete differential-geometry operators for triangulated 2-manifolds. In: Hege, H.-C., Polthier, K. (eds.) Visualization and Mathematics III, pp. 35–57. Springer, Heidelberg (2002)
28. Boyer, E., Bronstein, A.M., Bronstein, M.M., Bustos, B., Darom, T., Horaud, R., Hotz, I., Keller, Y., Keustermans, J., Kovnatsky, A., Litman, R., Reininghaus, J., Sipiran, I., Smeets, D., Suetens, P., Vandermeulen, D., Zaharescu, A., Zobel, V.: SHREC 2011: robust feature detection and description benchmark, February 2011 (Preprint, arXiv:1102.4258)

29. Eckstein, J., Bertsekas, D.: An alternating direction method for linear programming. Technical report, Harvard Business School (1990)
30. Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics. In: SIGGRAPH, pp. 209–216 (1997)
31. Botsch, M., Steinberg, S., Bischoff, S., Kobbelt, L.: Openmesh - a generic and efficient polygon mesh data structure. In: OpenSG Symposium 2002 (2002)
32. Vlasic, D., Baran, I., Matusik, W., Popović, J.: Articulated mesh animation from multi-view silhouettes. In: ACM SIGGRAPH 2008 Papers, pp. 1–9. ACM (2008)

Fast Regularization of Matrix-Valued Images

Guy Rosman[1](✉), Yu Wang[1], Xue-Cheng Tai[2],
Ron Kimmel[1], and Alfred M. Bruckstein[1]

[1] Department of Computer Science, Technion, 32000 Haifa, Israel
{rosman,yuwang,ron,freddy}@cs.technion.ac.il
[2] Department of Mathematics, University of Bergen,
Johaness Brunsgate 12, 5007 Bergen, Norway
tai@mi.uib.no

Abstract. Regularization of matrix-valued data is important in many fields, such as medical imaging, motion analysis and scene understanding, where accurate estimation of diffusion tensors or rigid motions is crucial for higher-level computer vision tasks. In this chapter we describe a novel method for efficient regularization of matrix- and group-valued images. Using the augmented Lagrangian framework we separate the total-variation regularization of matrix-valued images into a regularization and projection steps, both of which are fast and parallelizable. Furthermore we extend our method to a high-order regularization scheme for matrix-valued functions. We demonstrate the effectiveness of our method for denoising of several group-valued image types, with data in $SO(n)$, $SE(n)$, and $SPD(n)$, and discuss its convergence properties.

Keywords: Regularization · Matrix-manifolds · Lie-groups · Total-variation · Segmentation

1 Introduction

Matrix-valued signals are an important part of computer vision and image processing. Specific fields where matrix-valued data is especially important include tracking and motion analysis [28,43,44,56], robotics [37,38,59,60], image processing and computer vision [10,40,42,63], as well as more general optimization research [63] and 3D reconstruction [10].

Developing efficient regularization schemes for matrix-valued images is an important aspect of analysis and processing in these fields. These images have been an integral part of various domains, such as image processing [4,11,15,25, 39,48,52,58,62], motion analysis [31,43,56], and surface segmentation [44].

We present an efficient method for augmented Lagrangian smoothing of maps from a Cartesian domain into matrix manifolds such as $SO(n)$, $SE(n)$ and

This research was supported by European Community's FP7-ERC program, grant agreement no. 267414.

A. Bruhn et al. (Eds.): Global Optimization Methods, LNCS 8293, pp. 19–43, 2014.
DOI: 10.1007/978-3-642-54774-4_2,

$SPD(n)$, the manifolds of special-orthogonal, special-Euclidean, and symmetric, positive-definite, matrices, respectively. Specifically, the data we regularize can be represented as matrices with constraints on their singular values or eigenvalues. The augmented Lagrangian technique allows us in such cases to separate the optimization process into a *total-variation* (TV, [45]) regularization step and an eigenvalue or singular value projection step, both of which are fast and easily parallelizable using consumer graphic processing units (GPUs).

Our method handles each constraint separately via an auxiliary variable. Optimization with respect the separate variables results in a closed-form solution obtained via shrinkage or matrix decomposition, and is efficient to compute. Specifically, the update rule associated with solving the Lie-group auxiliary variable is similar for the case of $SO(n)$, $SE(n)$ and $SPD(n)$, leading to a unified framework which we describe in Sects. 3, 4. We briefly discuss convergence properties of the suggested algorithms in Sect. 5. In Sect. 6 we demonstrate a few results of our method, including motion analysis from depth sensors, direction diffusion, and DTI denoising and reconstruction. Section 7 concludes the paper.

2 A Short Introduction to Lie-Groups

Lie-groups are differentiable manifolds endows with an algebraic group structure, with smooth generators. Lie-groups and their structure have been used extensively in computer vision, and have been the subject of intense research efforts, involving statistics of matrix-valued data [39], and regularization of matrix-valued images [20,53], as well as describing the evolution of differential processes with Lie-group data [12,24]. We give a short introduction to Lie-groups in this section and refer the reader to the literature for an in-depth discussion [21,50].

Because of the group nature of Lie-groups, elements can be mapped via multiplication with their inverse, to the origin. This provides us with a diffeomorphically mapping each points and its neighborhood onto a neighborhood of the origin element, by group action with their inverse, to the identity element of the group. The tangent space in the origin therefore defines a canonical way of parameterizing small changes of the manifold elements via a vector space. Such a vector space is known as the *Lie-algebra* corresponding to the Lie-group. Lie-algebras are equipped with an anti-symmetric bilinear operator, the *Lie-bracket*, that describes the non-commutative part of the group product. Lie-brackets are used in tracking [6], robotics, and computer vision [35], among other applications.

We deal with two Lie-groups, and two related matrix manifolds in this work. The Lie-groups mentioned are

The rotations group $SO(n)$ - The group $SO(n)$ describes all rotations of the n-dimensional Euclidean space. Elements of this group can be described in a matrix form

$$SO(n) = \left\{ \mathbf{R} \in \mathbb{R}_{n\times n}, \mathbf{R}^T\mathbf{R} = \mathbf{I}, \det(\mathbf{R}) = 1 \right\}, \tag{1}$$

with the group product being matrix multiplication. The Lie-algebra of this group is the space $so(n)$, which can be described by the set of skew-symmetric matrices,

$$so(n) = \left\{ \mathbf{A} \in \mathbb{R}_{n\times n}, \mathbf{A}^T = -\mathbf{A} \right\}. \tag{2}$$

Another manifolds which are of interest and are highly related to $SO(n)$ are its quotient manifolds, the Stiefel manifolds.

The special-Euclidean group $SE(n)$ - This group represents rigid transformations of the n-dimensional Euclidean space. This group can be thought of as the product manifold of $SO(n)$ and the manifold $\mathbb{R}^n$ describing all translations of the Euclidean space. In matrix form this group can be written as

$$SE(n) = \left\{ \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}, \mathbf{R} \in SO(n), \mathbf{t} \in \mathbb{R}^n \right\}, \tag{3}$$

with matrix multiplication as the group action.

The Lie-algebra of this group can be written as

$$se(n) = \left\{ \begin{pmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0} & 0 \end{pmatrix}, \mathbf{A} \in so(n), \mathbf{t} \in \mathbb{R}^n \right\}, \tag{4}$$

We note that these groups have trivially-defined embeddings into Euclidean spaces, and an easily computable projection operator from the embedding space onto the group. Also, the embedding space we relate to is equipped with a norm: $\|\cdot\|$ denote the Frobenius norm in this chapter. The inner product used in this chapter is also the inner product corresponding to the Frobenius norm – $\langle A, B\rangle = trace\{A^T B\}$. Matrix manifolds for which there exists a simple projector operator include

Symmetric positive definite matrices $SPD(n)$ - This matrix set has been studied extensively in control theory [18], as well as in the context of diffusion tensor images [39], where the matrices are used to describe the diffusion coefficients along each direction. By definition, this group is given in matrix form as

$$SPD(n) = \{\mathbf{A} \in \mathbb{R}_{n\times n}, \mathbf{A} \succeq 0\}, \tag{5}$$

Stiefel manifolds - The Stiefel manifold $\mathbf{V}_k(\mathbb{R}^n)$ is defined as the set of all k-frames in $\mathbb{R}^n$. This can be written as the set of all $n\times k$ matrices with orthonormal columns. This set, too, has a projection operator similar to $SO(n)$.

3 An Augmented Lagrangian Regularization Algorithm for Matrix-Valued Images

The optimization problem we consider is the equivalent of the total-variation regularization of a map from the image domain to the matrix-manifold or Lie-group, $\mathcal{G}$ [20],

$$\arg\min_{u\mathcal{G}} \int \|u^{-1}\nabla u\| + \lambda\|u - u_0\|^2 dx. \tag{6}$$

The function u represents an element in an embedding of $\mathcal{G}$ into Euclidean space, specifically for the manifolds $SO(n), SE(n), SPD(n), \mathbf{V}_k(\mathbb{R}^n)$. Elements of $SO(n)$ and $\mathbf{V}_k(\mathbb{R}^n)$ can be embedded into $\mathbb{R}^{n^2}$ and $\mathbb{R}^{nk}$, respectively. Elements of $SE(n)$ can similarly be embedded into $\mathbb{R}^{(n+1)^2}$, or more precisely, an $n(n+1)$-dimensional linear subspace of $\mathbb{R}^{(n+1)^2}$. The elements of $SPD(n)$ can be embedded into $\mathbb{R}^{n(n+1)/2}$. We note that different choice of effectively parametrizing the manifold are possible, simply by making the norm in Eq. 6 a weighted one. Specific choices of metric has been discussed in [37,60], but currently no single canonical choice prevails. Choosing an optimal parameterization is beyond the scope of this work.

We first describe our method in the context of $\mathcal{G} = SO(n)$, and then detail the differences required when $\mathcal{G} = SE(n)$ and $\mathcal{G} = SPD(n)$.

We use the same notation for representation of the manifold point, its matrix representation, and its embedding into the embedding space, as specified in each case we explore.

The term $\|u^{-1}\nabla u\|$ can be thought of as a regularization term placed on elements of the Lie-algebra about each pixel. In order to obtain a fast regularization scheme, we look instead at regularization of an embedding of the Lie-group elements into Euclidean space,

$$\underset{u\,:\,\Omega\to\mathcal{G}}{\arg\min}\int_{\Omega}\|\nabla u\|+\lambda\|u-u_0\|^2dx. \tag{7}$$

The rationale behind the different regularization term $\|\nabla u\|$ stems from the fact that $SO(n)$ and $SE(n)$ are isometries of Euclidean space. In fact, denote by u^j vectors in $\mathbb{R}^n$ representing the columns of the matrix $u(x)$. Since $u(x)$ is approximately an isometry of $\mathbb{R}^n$, let $\Delta\lambda(x)$ denote the maximal local perturbation of the singular values of $u^{-1}(x)$. We assume $\Delta\lambda < 1$. In this case,

$$\begin{aligned}&\left|\left\|\tfrac{\partial}{\partial x_i}u\right\|_F^2-\left\|u^{-1}\tfrac{\partial}{\partial x_i}u\right\|_F^2\right|\\ &=\left|\textstyle\sum_{j=1}^n\left\|u^j_{x_i}\right\|^2-\sum_{j=1}^n\left\|u^{-1}u^j_{x_i}\right\|^2\right|\\ &\le\Delta\lambda\textstyle\sum_{j=1}^n\left\|u^j_{x_i}\right\|^2=\Delta\lambda\left\|u_{x_i}\right\|_F^2\end{aligned} \tag{8}$$

Hence, as long as the constraint $u(x)\in\mathcal{G}\ \ \forall x\in\Omega$ is approximately fulfilled for an isometry group $\mathcal{G}$, $\|\nabla u\|_F^2\approx\|u^{-1}\nabla u\|_F^2$. Moreover, such a regularization is possible whenever the data consists of nonsingular and rectangular matrices, and has been used also for SPD matrices [57]. Next, instead of restricting u to $\mathcal{G}$, we add an auxiliary variable, v, at each point, such that $u=v$, and restrict v to $\mathcal{G}$. The equality constraint is enforced via augmented Lagrangian terms [22,41]. The suggested augmented Lagrangian optimization now reads

$$\begin{aligned}&\min_{v\in\mathcal{G},u\in\mathbb{R}^m}\max_{\mu}\mathcal{L}(u,v;\mu)\\ &=\min_{v\in\mathcal{G},u\in\mathbb{R}^m}\max_{\mu}\int\left[\begin{array}{l}\|\nabla u\|+\lambda\|u-u_0\|^2+\\ \frac{r}{2}\|v-u\|^2+\langle\mu,v-u\rangle\end{array}\right]dx.\end{aligned} \tag{9}$$

Given a fixed Lagrange multiplier μ, the minimization w.r.t. u, v can be split into alternating minimization steps as described in the following two subsections.

3.1 Minimization w.r.t. v

The minimization w.r.t. v is a projection problem per pixel,

$$\begin{aligned} &\arg\min_{v\in\mathcal{G}} \frac{r}{2}\|v-u\|^2 + \langle \mu, u-v\rangle \\ &= \arg\min_{v\in\mathcal{G}} \frac{r}{2}\left\|v-\left(\frac{\mu}{r}+u\right)\right\|^2 \qquad (10) \\ &= \mathrm{Proj}_{\mathcal{G}}\left(\frac{\mu}{r}+u\right), \qquad (11) \end{aligned}$$

where $\mathrm{Proj}_{\mathcal{G}}(\cdot)$ denotes a projection operator onto the specific matrix-group $\mathcal{G}$, and its concrete form for $SO(n)$,$SE(n)$ and $SPD(n)$ will be given later on.

3.2 Minimization w.r.t. u

Minimization with respect to u is a vectorial TV denoising problem

$$\arg\min_{u\in\mathbb{R}^m} \int \|\nabla u\| + \widetilde{\lambda}\left\|u-\widetilde{u}\left(u_0,v,\mu,r\right)\right\|^2 dx, \qquad (12)$$

with $\widetilde{u} = \frac{2\lambda u_0 + rv + 2\mu}{2\lambda + r}$. This problem can be solved via fast minimization techniques – specifically, we chose to use the augmented-Lagrangian TV denoising algorithm [51], as we now describe. In order to obtain fast optimization of the problem with respect to u, we add an auxiliary variable p, along with a constraint that $p = \nabla u$. Again, the constraint is enforced in an augmented Lagrangian manner. The optimal u now becomes a saddle point of the optimization problem

$$\begin{matrix}\min \\ u\in\mathbb{R}^m \\ p\in\mathbb{R}^{2m}\end{matrix} \quad \max_{\mu_2} \int \begin{bmatrix} \|p\| + \widetilde{\lambda}\left\|u-\widetilde{u}\left(u_0,v,\mu,r\right)\right\|^2 \\ +\mu_2^T(p-\nabla u) + \frac{r_2}{2}\|p-\nabla u\|^2 \end{bmatrix} dx, \qquad (13)$$

We solve for u using the Euler-Lagrange equation,

$$2\widetilde{\lambda}(u-\widetilde{u}) + (\operatorname{div}\mu_2 + r_2 \operatorname{div} p) + r_2 \Delta u = 0, \qquad (14)$$

for example, in the Fourier domain, or by Gauss-Seidel iterations.

The auxiliary field p is updated by rewriting the minimization w.r.t. p as

$$\arg\min_{p\in\mathbb{R}^{2m}} \int \|p\| + \mu_2^T p + \frac{r_2}{2}\|p-\nabla u\|^2, \qquad (15)$$

with the closed-form solution [51,61]

$$p = \frac{1}{r_2}\max\left(1-\frac{1}{\|w\|},0\right) w, w = r_2\nabla u - \mu_2. \qquad (16)$$

Hence, the main part of the proposed algorithm is to iteratively update v, u, and p respectively. Also, according to the optimality conditions, the Lagrange multipliers μ and μ_2 should be updated by taking

$$\begin{aligned} \mu^k &= \mu^{k-1} + r\left(v^k - u^k\right) \\ \mu_2^k &= \mu_2^{k-1} + r_2\left(p^k - \nabla u^k\right). \end{aligned} \tag{17}$$

Let

$$\mathcal{F}(u,v,p;\mu,\mu_2) = \int \left[\begin{array}{c} \lambda\left\|u-u_0\right\|^2 + \frac{r_2}{2}\|p-\nabla u\|^2 + \frac{r}{2}\|u-v\|^2 + \\ +\mu^T(u-v) + \mu_2^T(p-\nabla u) + \|p\| \end{array}\right] dx. \tag{18}$$

the constrained minimization problem in Eq. 7 becomes the following saddle-point problem

$$\begin{array}{l} \min\limits_{\substack{v\in\mathcal{G} \\ u\in\mathbb{R}^m \\ p\in\mathbb{R}^{2m}}} \max\limits_{\mu,\mu_2} \mathcal{F}(u,v,p;\mu,\mu_2) \end{array} \tag{19}$$

An algorithmic description is summarized as Algorithm 1, whose convergence properties are discussed in Sect. 5.

Algorithm 1 Fast TV regularization of matrix-valued data

1: **for** $k = 1, 2, \ldots,$ until convergence **do**
2: Update $u^k(x), p^k(x)$, according to Eqs. (14, 16).
3: Update $v^k(x)$, by projection onto the matrix group,

- For $SO(n)$ matrices, according to Eq. (20).
- For $SE(n)$ matrices, according to Eq. (21).
- For $SPD(n)$ matrices, according to Eq. (22).

4: Update $\mu^k(x), \mu_2^k(x)$, according to Eq. (17).
5: **end for**

3.3 Regularization of Maps onto $SO(n)$

In the case of $\mathcal{G} = SO(n)$, although the embedding of $SO(n)$ in Euclidean space is not a convex set, the projection onto the matrix manifold is easily achieved by means of the singular value decomposition [19]. Let $\mathbf{USV}^T = \left(\frac{\mu}{r} + u^k\right)$ be the SVD decomposition of $\frac{\mu}{r} + u^k$, we update v by

$$\begin{aligned} v^{k+1} &= \mathrm{Proj}_{SO(n)}\left(\frac{\mu}{r} + u^k\right) = \mathbf{U}(x)\mathbf{V}^T(x), \\ \mathbf{USV}^T &= \left(\frac{\mu}{r} + u^k\right). \end{aligned} \tag{20}$$

Other possibilities include using the Euler-Rodrigues formula, quaternions, or the polar decomposition [29]. We note that the non-convex domain $SO(n)$ prevents a global convergence proof of the type shown in Subsect. 5.2 for $SPD(n)$. Convergence properties of the algorithm, in the case of $\mathcal{G} = SO(n)$ and $\mathcal{G} = SE(n)$, are discussed in Subsect. 5.1.

We also note that the projection via SVD can be used to project matrices onto the Stiefel manifolds [33], themselves quotient groups of $SO(n)$ [55]. Thus, the same algorithm can be used for Stiefel manifolds as well.

3.4 Regularization of Maps onto $SE(n)$

In order to regularize images with values in $SE(n)$, we use an embedding into $\mathbb{R}^{n(n+1)}$ as our main optimization variable, u, per pixel.

The projection step w.r.t. v applies only for the n^2 elements of v describing the rotation matrix, leaving the translation component of $SE(n)$ unconstrained.

Specifically, let $v = (v_R, v_t), v_R \in \mathbb{R}^{n^2}, v_t \in \mathbb{R}^n$ denotes the rotation and translation parts of the current solution. Updating v in step 3 of algorithm 1 assumes the form

$$\begin{aligned} v_R^{k+1} = \mathbf{U}(x)\mathbf{V}^T(x), \quad \mathbf{U}\mathbf{S}\mathbf{V}^T = \left(\frac{\mu_R}{r} + u_R^k\right) \\ v_t^{k+1} = \left(\frac{\mu_t}{r} + u_t^k\right) \\ v^{k+1} = \mathrm{Proj}_{SE(n)}(v^k) = (v_R^{k+1}, v_t^{k+1}). \end{aligned} \tag{21}$$

3.5 Regularization of Maps onto $SPD(n)$

The technique described above can be used also for regularizing symmetric positive-definite matrices. A most prominent example for such matrices is that of diffusion tensor images [4,5,13,27,30,49,53]. This includes several attempts to define efficient and physically meaningful regularization techniques for DTI regularization [7,53,65]. Many papers dealing with the analysis of DTI rely on the eigenvalue decomposition of the tensor as well, i.e. for tractography [14], anisotropy measurements [64], and so forth. It is not surprising that the intuitive choice of projecting the eigenvalues of the matrices onto the positive half-space is shown to be optimal [9,23].

When using an augmented Lagrangian approach, the minimization problem w.r.t. v in step 3 of algorithm 1 is therefore solved by projection of eigenvalues,

$$\begin{aligned} v^{k+1} = \mathrm{Proj}_{SPD(n)}(v^k) = \mathbf{U}(x)\,\mathrm{diag}\left(\widetilde{\lambda}\right)\mathbf{U}^T(x), \\ \mathbf{U}\,\mathrm{diag}\,(\lambda)\,\mathbf{U}^T = \left(\frac{\mu}{r} + u^k\right), \left(\widetilde{\lambda}\right)_i = \max\left((\lambda)_i, 0\right), \end{aligned} \tag{22}$$

where the matrix U is a unitary one, representing the eigenvectors of the matrix, and the eigenvalues $\left(\widetilde{\lambda}\right)_i$ are the positive projection of the eigenvalues $(\lambda)_i$. Optimization w.r.t. u is done as in the previous cases, as described in Algorithm 1.

Furthermore, the optimization w.r.t. u, v is now over the domain $\mathbb{R}^m \times SPD(n)$, and the cost function is convex, resulting in a convex optimization problem. The convex domain of optimization allows us to formulate a convergence proof for the algorithm similar to the proof by Tseng [54]. This is discussed in Subsect. 5.2. An example of using the proposed method for DTI denoising and reconstruction is shown in Sect. 6.

3.6 A Higher-Order Prior for Group-Valued Images

We note that the scheme we describe is susceptible to the staircasing effect, since it minimizes the total variation of the map u. While one possibility to avoid such artifacts is to incorporate a linear diffusion term into the functional, there exists a much more elegant solution by incorporating a higher-order differential operator into the regularization term. One such possibile higher-order term generalizes the scheme presented by Wu and Tai [66], by replacing the per-element gradient operator with a Hessian operator. The resulting equivalent of Eq. 7 becomes

$$\underset{u \in \mathcal{G}}{\arg\min} \int \|Hu\| + \lambda \|u - u_0\|^2 dx, \tag{23}$$

where Hu is the per-channel Hessian operator, defined (on two-dimensional domains) by

$$\left(Hu^{(k)}\right)_{i,j} = \begin{pmatrix} \left(D_{xx}^{-+}u^{(k)}\right)_{i,j} & \left(D_{xy}^{++}u^{(k)}\right)_{i,j} \\ \left(D_{yx}^{++}u^{(k)}\right)_{i,j} & \left(D_{yy}^{-+}u^{(k)}\right)_{i,j} \end{pmatrix} \tag{24}$$

The numerical scheme solves the saddle-point problem

$$\begin{matrix} \min \\ u \in \mathbb{R}^m \\ p \in \mathbb{R}^{4m}, \\ v \in \mathcal{G} \end{matrix} \quad \max_{\mu_2} \int \begin{bmatrix} \|p\| + \widetilde{\lambda} \left\|u - \widetilde{u}\left(u_0, v, \mu, r\right)\right\|^2 \\ +\mu_2^T (p - Hu) + \frac{r_2}{2} \|p - Hu\|^2 \end{bmatrix} dx, \tag{25}$$

The update step w.r.t. u as in Eq. 14 is easy to modify, resulting in the Euler-Lagrange equation

$$2\widetilde{\lambda}(u - \widetilde{u}) - (H^* \mu_2 + r_2 H^* p) + r_2 H^* H u = 0, \tag{26}$$

where H^* is the adjoint operator of the Hessian,

$$H^* p^{(k)} = D_{xx}^{+-}\left(p^{(k)}\right)^{11} + D_{xy}^{--}\left(p^{(k)}\right)^{12} + D_{yx}^{--}\left(p^{(k)}\right)^{21} + D_{yy}^{+-}\left(p^{(k)}\right)^{22}. \tag{27}$$

The update step w.r.t. p remains similar to Eq. 16, and is given by

$$p = \frac{1}{r_2} \max\left(1 - \frac{1}{\|w\|}, 0\right) w, w = r_2 H u - \mu_2. \tag{28}$$

Updates of the variable v and the Lagrange multipliers μ, μ_2 remain the same as in Algorithm 1. As will be shown in Sect. 6, this regularization term prevents formation of staircasing effects where these are inappropriate.

4 Regularized DTI Reconstruction

There are several possibilities of using the proposed regularization scheme for DTI reconstruction from diffusion-weighted measurements. Instead of adding a fidelity term as in Eq. (7), we add a term for fitting the Stejskal-Tanner equations [49], based on a set of measurements describing the diffusion in specific directions, and reconstruct the full diffusion tensor at each voxel. The fitting term can be written as

$$\sum_i \left\| b_i \mathbf{g}_i^T u \mathbf{g}_i - \log\left(\frac{S_i}{S_0}\right) \right\|^2,$$

where b_i and $\mathbf{g}_i$ are the b-values and gradient vectors, u is the diffusion tensor reconstructed at each voxel, and $\frac{S_i}{S_0}$ define the relative signal ratio for each direction at each voxel. The complete minimization problem reads

$$\begin{aligned} \underset{\substack{v \in SPD(n) \\ u}}{\arg\min} \int \sum_i \left\| b_i \mathbf{g}_i^T u \mathbf{g}_i - \log\left(\tfrac{S_i}{S_0}\right) \right\|^2 + \lambda \|\nabla u\| \\ + \tfrac{r}{2}\|v-u\|^2 + \langle \mu, v-u \rangle dx. \end{aligned} \tag{29}$$

While the memory requirements seem less favorable for fast optimization, looking closely at the quadratic penalty data term, we see it can be expressed by looking at a fitting term for the Stejskal-Tanner equations,

$$\sum_i \left\| b_i \mathbf{g}_i^T u \mathbf{g}_i - \log\left(\frac{S_i}{S_0}\right) \right\|^2 = u^T \mathbf{A} u + \mathbf{b}^T u + c, \tag{30}$$

where $\mathbf{A}$ is a constant matrix over the whole volume,

$$\mathbf{A} = \sum_i b_i^2 \begin{pmatrix} g_1^4 & 2g_1^3 g_2 & 2g_1^3 g_3 & g_1^2 g_2^2 & 2g_1^2 g_2 g_3 & g_1^2 g_3^2 \\ 2g_1^3 g_2 & 4g_1^2 g_2^2 & 4g_1^2 g_2 g_3 & 2g_1 g_2^3 & 4g_1 g_2^2 g_3 & 2g_1 g_2 g_3^2 \\ 2g_1^3 g_3 & 4g_1^2 g_2 g_3 & 4g_1^2 g_3^2 & 2g_1 g_2^2 g_3 & 4g_1 g_2 g_3^2 & 2g_1 g_3^3 \\ g_1^2 g_2^2 & 2g_1 g_2^3 & 2g_1 g_2^2 g_3 & g_2^4 & 2g_2^3 g_3 & g_2^2 g_3^2 \\ 2g_1^2 g_2 g_3 & 4g_1 g_2^2 g_3 & 4g_1 g_2 g_3^2 & 2g_2^3 g_3 & 4g_2^2 g_3^2 & 2g_2 g_3^3 \\ g_1^2 g_3^2 & 2g_1 g_2 g_3^2 & 2g_1 g_3^3 & g_2^2 g_3^2 & 2g_2 g_3^3 & g_3^4 \end{pmatrix} \tag{31}$$

and $\mathbf{b}$ is the vector

$$\mathbf{b} = \sum_i b_i \log\left(\frac{S_i}{S_0}\right) \begin{pmatrix} 2g_1^2 & 4g_1 g_2 & 4g_1 g_3 & 2g_2^2 & 4g_2 g_3 & 2g_3^2 \end{pmatrix}^T, \tag{32}$$

and c is the scalar image

$$c = \sum_i \left(\log\left(\frac{S_i}{S_0}\right) \right)^2. \tag{33}$$

We note that, unlike the denoising case, in the reconstruction case it is the data term that couples together the elements of the tensor together. Care must be taken so as to handle this coupled data term.

Reconstruction with the new data term can be computed using several techniques.

- Freezing all elements of the tensor but one, we obtain from the Euler-Lagrange equations pertaining to Eq. 29 an update rule for the image, to be computed in the Fourier domain, or via Gauss-Seidel iterations. While the coupling between the tensor elements (expressed via the non-diagonal matrix $\mathbf{A}$) prevents us from treating each tensor element separately, the optimization w.r.t. each of the elements converges quite rapidly.
- Another possibility is to take a block Gauss-Seidel approach, and optimize each tensor separately, going over all the voxels one-by-one.
- Yet another possibility is to further decouple the TV and data term, using separate variables and constraining them using an augmented Lagrangian approach.

Of the above techniques, we have tried the first one. The reconstruction obtained is the spatially-regularized version of the *linear-least-squares* (LLS) method. One can incorporate a weighted least-squares (WLS, [47]), or nonlinear-least-squares (NLS) [27] data term instead. Combining such data terms and exploring the interaction between the regularization and nonlinear terms is beyond the scope of this work.

5 Convergence Properties of the Algorithm

We now turn to discuss the local convergence of Algorithm 1.

5.1 Local Convergence for $SO(n)$,$SE(n)$ Regularization

Looking at regularization of maps onto $SO(n)$,$SE(n)$, the non-convex nature of the optimization domain in Eq. 9 makes it difficult to prove global convergence. Furthermore, the nature of the projection operator into $SO(n)$ and $SE(n)$, makes it difficult to ascertain that at some point the sequence of iterants will converge. While showing there exists a converging subsequence of iterants is easy due to the boundedness of the sub-levelsets [54], the discontinuous nature of the projection unto non-convex spaces may cause the algorithm to oscillate, although this behaviour does not appear in practice. In order to avoid such a possibility and allow for an easy proof of convergence, we take a proximal step approach, and slightly modify our algorithm, as suggested by Attouch et al. [3], changing the first two steps of the algorithm into the minimization problems

$$\begin{aligned} u^k &= \arg\min_u \mathcal{F}(u, v^{k-1}, \mu) + \frac{1}{\theta_k}\|u - u^{k-1}\|^2 \\ v^k &= \arg\min_{v \in \mathcal{G}} \mathcal{F}(u^k, v, \mu) + \frac{1}{\theta_k}\|v - v^{k-1}\|^2. \end{aligned} \tag{34}$$

The proof of convergence becomes similar to the one shown by Attouch et al. [3]. Since $\mathcal{F}(u,v) > -\infty$ and $\{\mathcal{F}(u^k,v^k)\}$ is non-increasing, we have that $\mathcal{F}(u^k,v^k)$ converges to some finite value. Furthermore, by induction we can show that the residual converges to 0, providing us with a guarantee of the asympthotic behavior of the process.

The optimization steps in the modified algorithm remain a projection step and total-variation denoising, but with a change in their parameters. For example, the optimal update rule for v becomes

$$\begin{aligned}
&\arg\min_{v\in SO(n)} \frac{r}{2}\|v-u\|^2 + \langle \mu, v-u\rangle + \frac{1}{2\theta_k}\|v-v_{k-1}\|^2 = \\
&\arg\min_{v\in SO(n)} \left(\frac{r}{2}+\frac{1}{2\theta_k}\right)\|v\|^2 - \langle v, ru+\mu+\frac{v_{k-1}}{\theta_k}\rangle \\
&\qquad + \frac{r}{2}\|u\|^2 - \langle \mu, u\rangle + \frac{1}{2\theta_k}\|v_{k-1}\|^2 = \\
&\arg\min_{v\in SO(n)} \left(\frac{r}{2}+\frac{1}{2\theta_k}\right)\left\| v - \frac{ru+\mu+\frac{v_{k-1}}{\theta_k}}{r+\frac{1}{\theta_k}}\right\|^2,
\end{aligned}$$

where $\frac{1}{2\theta_k}$ denotes the coupling between each iterant and its previous value. We stress, however, that in practice the algorithm converges without the above modification quite well.

5.2 Global Convergence for $SPD(n)$ Regularization

For $SPD(n)$ regularization we basically do a coordinate descent on a convex domain [54] and therefore can show global convergence of our method. At each step of the inner iteration, we do a full minimization with respect to the selected variables block u, v and p. Using the notation provided by [54], we can rewrite our functional as

$$\mathcal{F}_{\mu,\mu_2}(u,v,p) = f_0(u,v,p) + f_1(u) + f_2(v) + f_3(p), \tag{35}$$

where

1. f_0 is a convex, smooth, function.

$$f_0(u,v,p) = \frac{r}{2}\|v-u\|^2 + \langle \mu, v-u\rangle + \frac{r_2}{2}\|p-\nabla u\|^2 + \langle \mu_2, p-\nabla u\rangle$$

2. f_1, f_2 and f_3 are convex, lower-semicontinuous, continuous in their effective domain,

$$f_1(u) = \|u-u_0\|^2 \tag{36}$$
$$f_2(v) = 0 \tag{37}$$
$$f_3(p) = \|p\|. \tag{38}$$

By [54, Proposition 1], it can be shown that the alternating minimization will converge to a minimizer of $\mathcal{F}_{\mu,\mu_2}(u,v,p)$. Along the same proof in [67], it can be proved the whole algorithm converges. For completeness we repeat the proof here. The following characterization for the minimizers of functional $\mathcal{F}(u,v,p;\mu,\mu_2)$ will be used. Assume that (u^*,v^*,p^*) is one of the minimizers, and for arbitrary (u',v',p') we have,

$$\begin{aligned}\lambda\|u^*-u_0\|^2-\lambda\|u'-u_0\|^2+r_2(p^*-\nabla u^*,-(\nabla u^*-\nabla u'))\\ +r(u^*-v^*,u^*-u')+(\mu^*,u^*-u')+(\mu_2^*,-(\nabla u^*-\nabla u'))\le 0 \qquad (39)\\ -r(u^*-v^*,v^*-v')-(\mu^*,v^*-v')\le 0 \qquad (40)\\ \|p^*\|-\|p'\|+r_2(p^*-\nabla u^*,p^*-p')+(\mu_2^*,p^*-p')\le 0 \qquad (41)\end{aligned}$$

(see [17], p.38 Proposition 2.2)

Theorem 51. *The sequence $(u^k,v^k,p^k;\mu^k,\mu_2^k)$ generated by Algorithm 1 converges to the saddle-point $(u^*,v^*,p^*;\mu^*,\mu_2^*)$ of the functional $\mathcal{F}(u,v,p;\mu,\mu_2)$*

Proof. Let $\bar{u}^k=u^*-u^k$,$\bar{v}^k=v^*-v^k$,$\bar{p}^k=p^*-p^k$, $\bar{\mu}^k=\mu^*-\mu^k$, and $\bar{\mu}_2^k=\mu_2^*-\mu_2^k$ Since $(u^*,v^*,p^*;\mu^*,\mu_2^*)$ is the saddle point of $\mathcal{F}(u,v,p;\mu,\mu_2)$, we have

$$\mathcal{F}(u^*,v^*,p^*;\mu^*,\mu_2^*)\le\mathcal{F}(u',v',p';\mu^*,\mu_2^*),\forall u,v,p \qquad (42)$$

In particular when $u'=u^k$ (39) still holds

$$\begin{aligned}\lambda\|u^*-u_0\|^2-\lambda\|u^k-u_0\|^2+r_2(p^*-\nabla u^*,-\nabla(u^*-u^k))\\ +r(u^*-v^*,u^*-u^k)+(\mu^*,u^*-u^k)+(\mu_2^*,-\nabla(u^*-u^k))\le 0 \qquad (43)\end{aligned}$$

On the other hand, since $(u^k,v^k,p^k;\mu^k,\mu_2^k)$ is the minimizer of $\mathcal{F}(u,v,p;\mu^k,\mu_2^k)$, u^k will also satisfy (39) and after substituting $u'=u^*$ we obtain

$$\begin{aligned}\lambda\|u^k-u_0\|^2-\lambda\|u^*-u_0\|^2+r_2(p^k-\nabla u^k,-\nabla(u^k-u^*))\\ +r(u^k-v^k,u^k-u^*)+(\mu^k,u^k-u^*)+(\mu_2^k,-\nabla(u^k-u^*))\le 0. \qquad (44)\end{aligned}$$

Adding the two inequalities yields

$$r_2(\bar{p}^k-\nabla\bar{u}^k,-\nabla\bar{u}^k)+r(\bar{u}^k-\bar{v}^k,\bar{u}^k)+(\bar{\mu}^k,\bar{u}^k)+(\bar{\mu_2}^k,-\nabla\bar{u}^k)\le 0 \qquad (45)$$

Similarly, w.r.t v^*,v^k using the same argument to (40) we have

$$\begin{aligned}-r(u^*-v^*,v^*-v^k)-(\mu^*,v^*-v^k)\le 0 \qquad (46)\\ -r(u^k-v^k,v^k-v^*)-(\mu^k,v^k-v^*)\le 0 \qquad (47)\end{aligned}$$

adding two inequalities yields

$$-r(\bar{u}^k-\bar{v}^k,\bar{v}^k)-(\bar{\mu}^k,\bar{v}^k)\le 0 \qquad (48)$$

w.r.t p^*,p^k, the same argument is applied to (41)

$$\begin{aligned}\|p^*\|-\|p^k\|+r_2(p^*-\nabla u^*,p^*-p^k)+(\mu_2^*,p^*-p^k)\le 0 \qquad (49)\\ \|p^k\|-\|p^*\|+r_2(p^k-\nabla u^k,p^k-p^*)+(\mu_2^k,p^k-p^*)\le 0 \qquad (50)\end{aligned}$$

thus

$$r_2(\bar{p}^k - \nabla\bar{u}^k, \bar{p}^k) + (\bar{\mu_2}^k, \bar{p}^k) \leq 0 \tag{51}$$

Adding (45), (48) and (51) we have

$$r_2\|\bar{p}^k - \nabla\bar{u}^k\|^2 + r\|\bar{u}^k - \bar{v}^k\|^2 + (\bar{\mu_2}^k, \bar{p}^k - \nabla\bar{u}^k) + (\bar{\mu}^k, \bar{u}^k - \bar{v}^k) \leq 0 \tag{52}$$

By the way of updating multipliers, also note that $u^* = v^*$ and $p^* = \nabla u^*$ we obtain

$$\bar{\mu}^{k+1} = \bar{\mu}^k + r(\bar{u}^k - \bar{v}^k) \tag{53}$$

$$\bar{\mu}_2^{k+1} = \bar{\mu}_2^k + r_2(\bar{p}^k - \nabla\bar{u}^k) \tag{54}$$

therefore by (52) we have

$$\begin{aligned} &r_2\|\bar{\mu}^{k+1}\|^2 + r\|\bar{\mu}_2^{k+1}\|^2 - r_2\|\bar{\mu}^k\|^2 - r\|\bar{\mu}_2^k\|^2 \\ &= 2rr_2(\bar{\mu}^k, \bar{u}^k - \bar{v}^k) + 2rr_2(\bar{\mu}_2^k, \bar{p}^k - \nabla\bar{u}^k) + r^2r_2\|\bar{u}^k - \bar{v}^k\|^2 + rr_2^2\|\bar{p}^k - \nabla\bar{u}^k\| \\ &\leq -r^2r_2\|\bar{u}^k - \bar{v}^k\|^2 - rr_2^2\|\bar{p}^k - \nabla\bar{u}^k\| \leq 0 \end{aligned} \tag{55}$$

This actually implies μ^k and μ_2^k are bounded, and

$$\lim_{k\to\infty} \|p^k - \nabla u^k\| = 0 \tag{56}$$

$$\lim_{k\to\infty} \|u^k - v^k\| = 0 \tag{57}$$

With this in mind, it is not hard to show that $(u^k, v^k, p^k; \mu^*, \mu_2^*)$ converge to the saddle-point of the functional

6 Numerical Results

As discussed above, the proposed algorithmic framework is quite general and is suitable for various applications. In this section, several examples from different applications are used to substantiate the effectiveness and efficiency of our algorithm.

6.1 Directions Regularization

Analysis of principal directions in an image or video is an important aspect of modern computer vision, in fields such as video surveillance [26,36, and references therein], vehicle control [16], crowd behaviour analysis [34], and other applications [40].

The input in this problem is a set of normalized/unnormalized direction vectors located throughout the image domain, either in a dense or sparse set of locations. The goal is to obtained a smoothed version of the underlying direction field. Since $SO(2)$ is isomorphic to S^1, the proposed regularization scheme can be

used for regularizing directions as well, as we demonstrate. A reasonable choice for a data term would try to align the rotated first coordinate axis with the motion directions in the area,

$$E_{PMD}(U) = \sum\nolimits_{(x_j,y_j)\in\mathcal{N}(i)} \left(U_{1,1} \left(v_j\right)_x + U_{1,2} \left(v_j\right)_y \right), \tag{58}$$

where $\left(x_j, y_j, \left(v_j\right)_x, \left(v_j\right)_y\right)$ represent a sampled motion particle [34] in the video sequence, and $U_{i,j}$ represent elements of the solution u.

In Fig. 1 we demonstrate two sparsely sampled, noisy, motion fields, and a dense reconstruction of the main direction of motion at each point. The data for the direction estimation was corrupted by adding component-wise Gaussian noise. In the first image, the motion field is comprised of 4 regions with a different motion direction at each region. The second image contains a sparse sampling of an expansion motion field of the form

$$\boldsymbol{v}(x,y) = \frac{(x,y)^T}{\| (x,y) \|}. \tag{59}$$

Such an expansion field is often observed by forward-moving vehicles. Note that despite the fact that a vanishing point of the flow is clearly not smooth in terms of the motion directions, the estimation of the motion field is still correct.

An example of the higher order regularization term is shown in Fig. 2, using the approach suggested in Subsect. 3.6. Note the smooth boundaries create due to the sparsely sampled data term – while the TV solution forces staircasing in the solution, the higher order regularization does not.

In Fig. 3 we used the algorithm to obtain a smooth field of principal motion directions over a traffic sequence taken from the UCF crowd flow database [2]. Direction cues are obtained by initializing correlation-based trackers from arbitrary times and positions in the sequence, and observing all of the tracks simultenaously. The result captures the main traffic lanes and shows the viability of our regularization for real data sequences.

Yet another application for direction diffusion is in denoising of directions in fingerprint images. An example for direction diffusion on a fingerprint image taken from the Fingerprint Verification Competition datasets [1] can be seen in Fig. 4. Adding a noise of $\sigma = 0.05$ to the image and estimating directions based on the structure tensor, we smoothed the direction field and compared it to the field obtained from the original image. We used our method with $\lambda = 3$, and the modified method based on Eq. 26 with $\epsilon = 10$, as well as the method suggested by Sochen et al. [46] with $\beta = 100, T = 425$. The resulting MSE values of the tensor field are 0.0317, 0.0270 and 0.0324, respectively, compared to an initial noisy field with $MSE = 0.0449$. These results demonstrate the effectiveness of our method for direction diffusion, even in cases where the staircasing effect may cause unwanted artifacts.

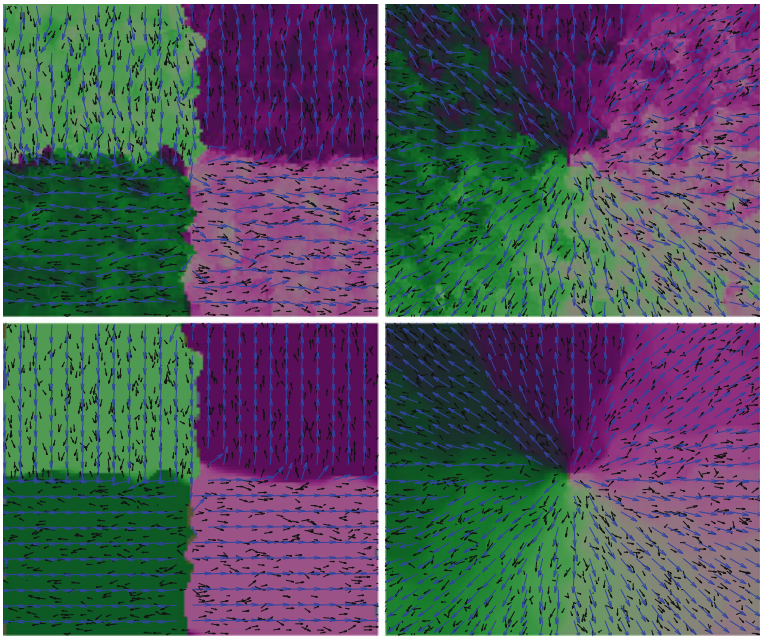


Fig. 1. TV regularization of $SO(n)$ data. Left-to-right, top-to-bottom: the initial estimated field for a 4-piece piecewise constant motion field, a concentric motion field, the denoised images for the piecewise constant field and the concentric motion field. Different colors mark different orientations of the initial/estimated dense field, black arrows signify the measured motion vectors, and blue arrows demonstrate the estimated field after sampling.

6.2 $SE(n)$ Regularization

We now demonstrate a smoothing of $SE(3)$ data obtained from locally matching between two range scans obtained from a Kinect device. For each small surface patch from the depth image we use an *iterative closest point* (ICP) algorithm [8] to match the surface from the previous frame. For each point in the foreground, an ICP algorithm is used to match the point's neighborhood from frame i to that of frame $i-1$. The background is segmented by simple thresholding. The results from this tracking process over raw range footage are an inherently noisy measurements set in $SE(3)$. We use our algorithm to smooth this $SE(3)$ image, as shown in Fig. 5. It can be seen that for a careful choice of the regularization parameter, total variation in the group elements is seen to significantly reduce rigid motion estimation errors. Furthermore, it allows us to discern the main rigidly moving parts in the sequence by producing a scale-space of rigid motions. Visualization is accomplished by projecting the embedded matrix

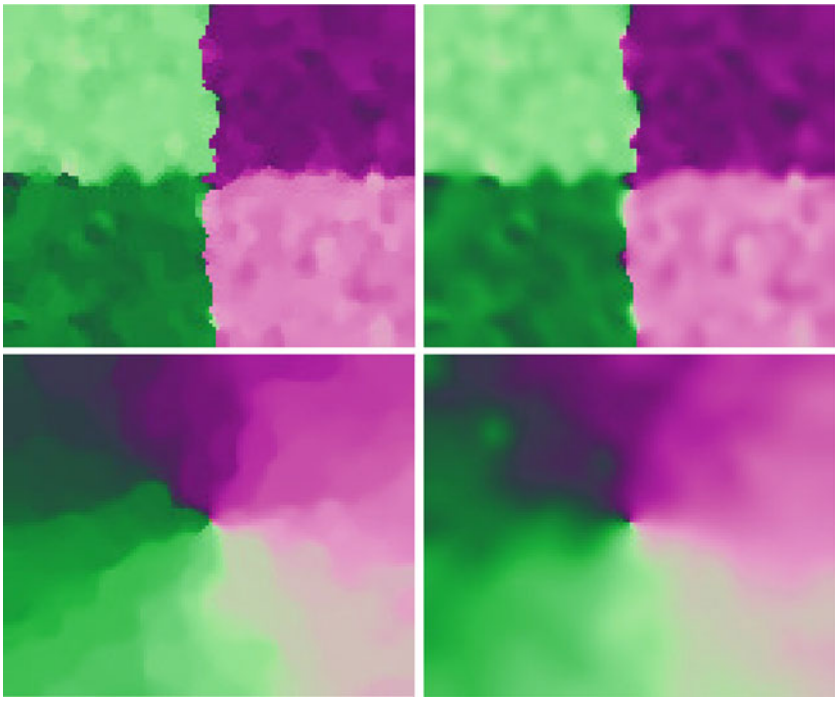


Fig. 2. TV regularization of $SO(n)$ data, based on the same data from Fig. 1, with a higher-order regularity term. Different color mark different orientations of the estimated motion field. Left: TV regularization result as demonstrated in Fig. 1. Right: regularization results based on Eq. 23. The parameter λ was chosen to be 2 for the upper example, and 0.2 for the lower example.

onto 3 different representative vectors in $\mathbb{R}^{12}$. The regularization is implemented using the CUDA framework, with computation times shown in Table 1. Using 15 outer iterations and 3 Gauss-Seidel iterations per inner iteration, practical convergence is achieved in 63 milliseconds on an NVIDIA GTX-580 card for QVGA-sized images, demonstrating the efficiency of our algorithm and its potential for real-time applications. This is especially important for applications such as gesture recognition where fast computation is important. A residual plot in the left sub-figure of Fig. 6 demonstrates convergence of our method.

Furthermore, since the main constraint for $SO(n)$ matrices (or the rotation part of $SE(n)$ matrices) is that of orthogonality, we measure during convergence

$$err_{\text{orth}}(u) = \left\| U^T U - I \right\|_F^2 \tag{60}$$

The plot of err_{orth} as a function of the iterations is shown in the right sub-figure of Fig. 6. The plot demonstrates the enforcement of the constraint

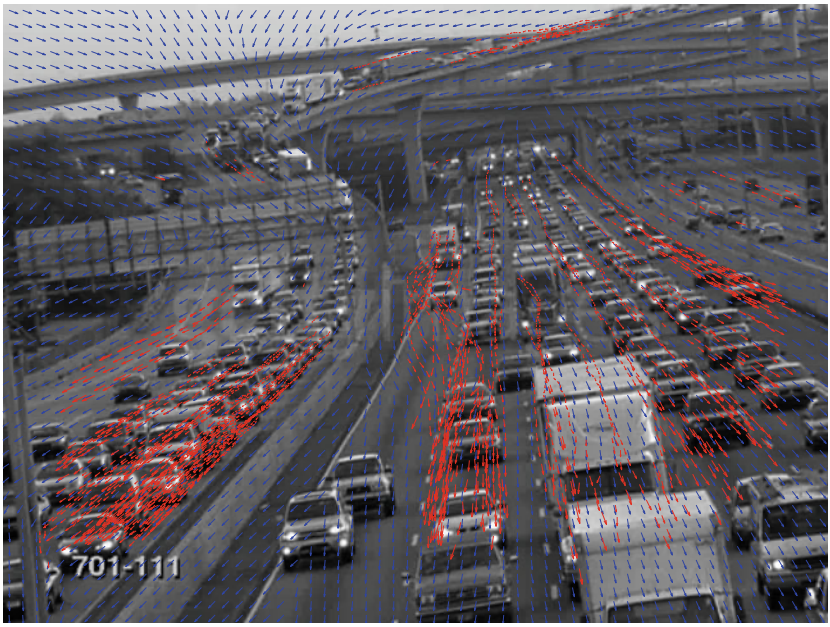


Fig. 3. Regularization of principal motion directions. The red arrows demonstrate measurements of motion cues based on a normalized cross-correlation tracker. Blue arrows demonstrate the regularized directions fields.

$u \in \mathcal{G}$ by the augmented Lagrangian scheme for most of the convergence. The close adherence to the isometry assumption validates in practice our usage of the regularization proposed in Eq. 7 for isometry groups.

6.3 DTI Regularization

In Fig. 7 we demonstrate a smoothing of DTI data from [32], based on the scheme suggested in Sect. 3.5, using the Slicer3D tool in order to visualize the tensors via ellipsoid glyphs. Figure 8 demonstrates the convergence rate for the regularization. MSE of the matrix representation was 0.0406 in the corrupted image and 0.0248 in the regularized image.

In Figs. 9, 10 we demonstrate reconstruction of the DTI tensors, again based data from Lundervold et al. [32], using a set of 30 directional measurements. The measure ratios $\log\left(\frac{S_i}{S_0}\right)$ were added a Gaussian additive noise of standard deviation 100. The reconstructed image obtained by regularized reconstruction with $\lambda = 1 \times 10^{-3}$ had an MSE of 2.1×10^{-4}, compared to 8.9×10^{-3} without regularization.

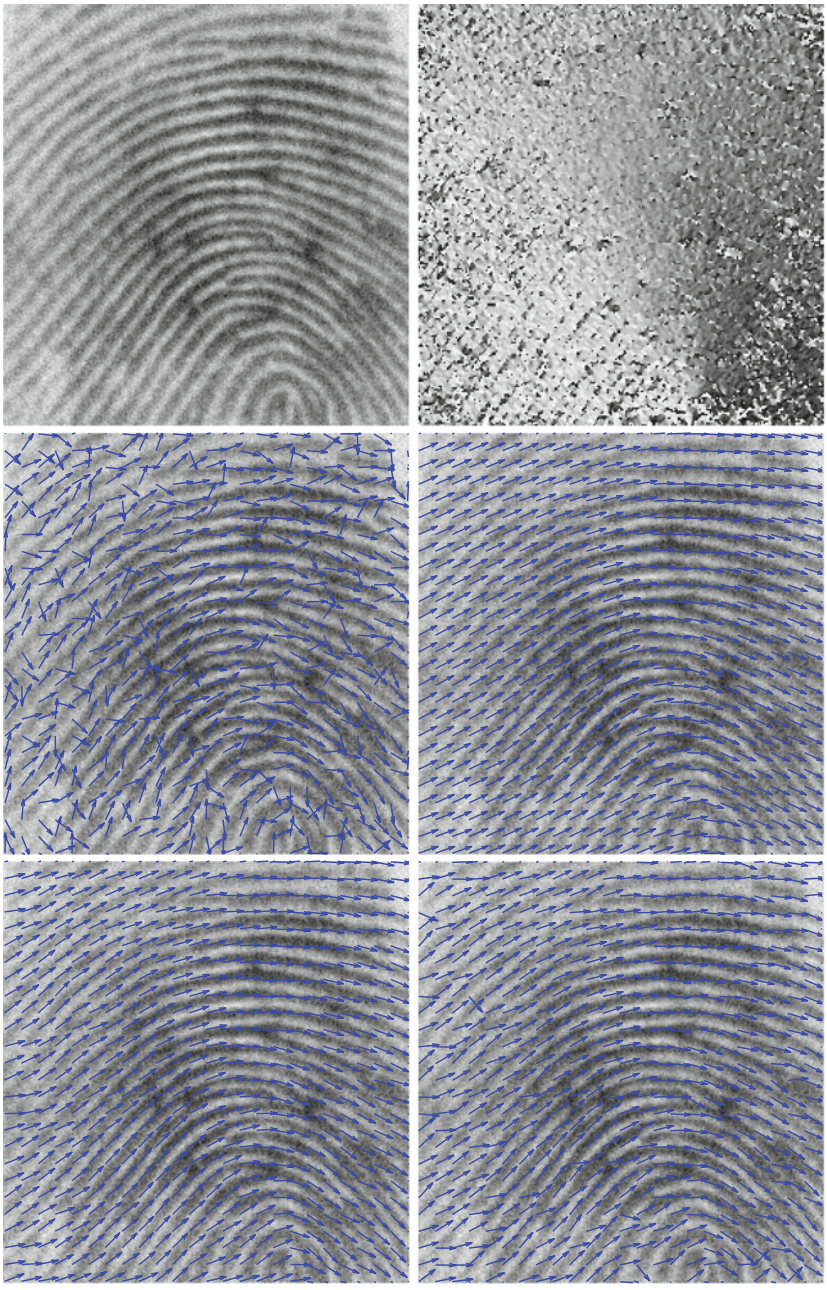


Fig. 4. TV regularization of $SO(2)$ data based on fingerprint direction estimation. Left-to-right, top-to-bottom: The fingerprint image with added Gaussian noise of $\sigma = 0.05$, the detected direction angles, the detected directions displayed as arrows, the detected directions after regularization with $\lambda = 3$, regularization results using Eq. 9, regularization results based on higher-order diffusion term with $\lambda = 6$, the regularization result by Sochen et al. [46].

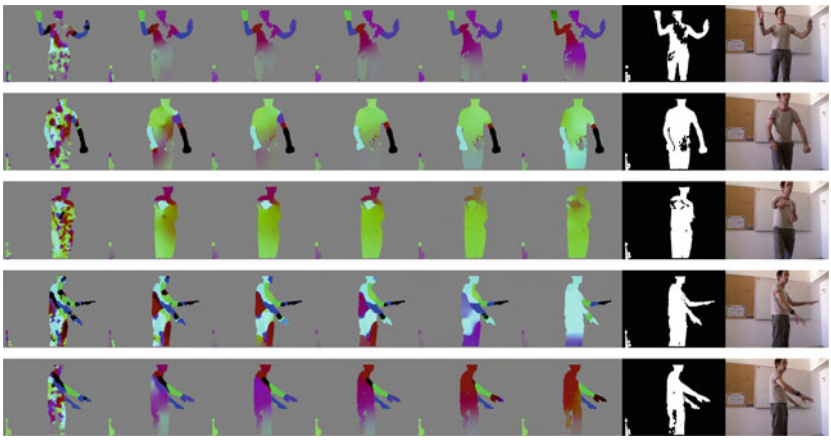


Fig. 5. Regularization of $SE(3)$ images obtained from local ICP matching of the surface patch between consecutive Kinect depth frames. Left-to-right: diffusion scale-space obtained by different values of λ: $1.5, 1.2, 0.7, 0.2, 0.1, 0.05$, the foreground segmentation based on the depth, and an intensity image of the scene.

Table 1. GPU processing times for various sizes of images, given in milliseconds.

Outer iterations	15	15	25	50
GS iterations	1	3	1	1
320×240	49	63	81	160
640×480	196	250	319	648
1920×1080	1745	2100	2960	5732

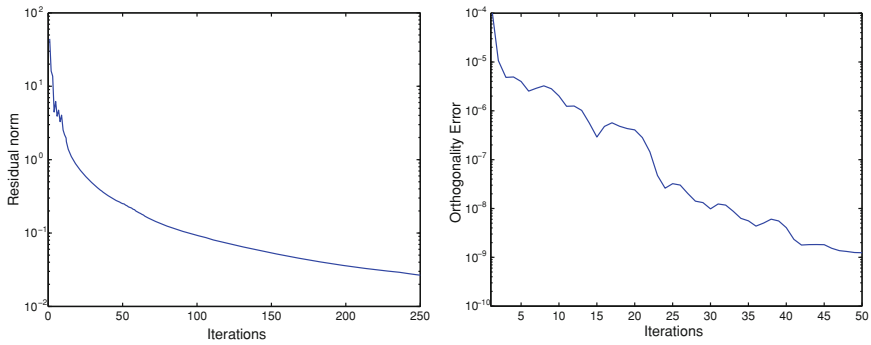


Fig. 6. A residual plot (left), and orthogonality error norm plot (right) for $SE(3)$ denoising as demonstrated in Fig. 5, for $\lambda = 0.2$.

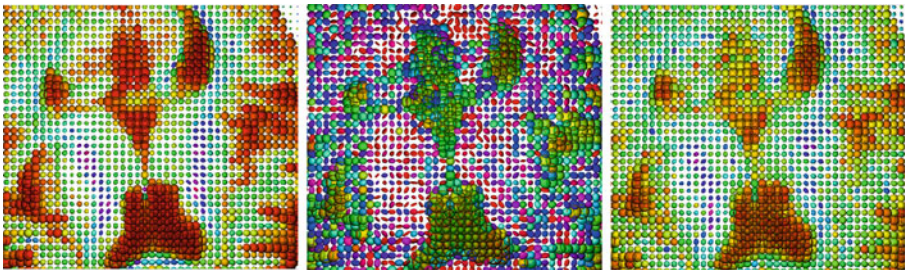


Fig. 7. TV denoising of images with diffusion tensor data. Left-to-right: the original image, an image with added component-wise Gaussian noise of $\sigma = 0.1$, and the denoised image with $\lambda = 30$.

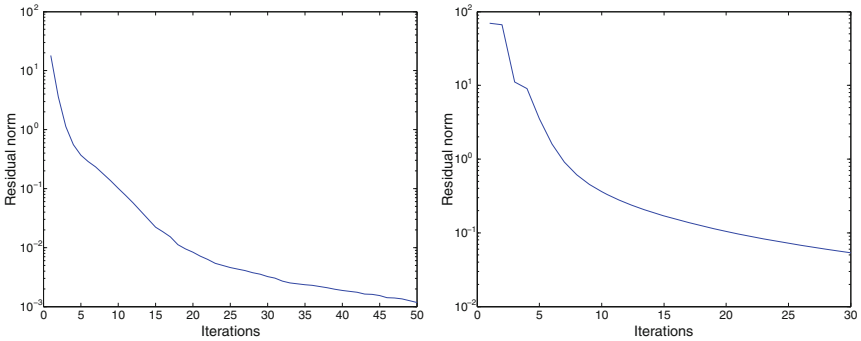


Fig. 8. A residual plot for DTI denoising (left) and reconstruction (right) as demonstrated in Figs. 7, 9, respectively.

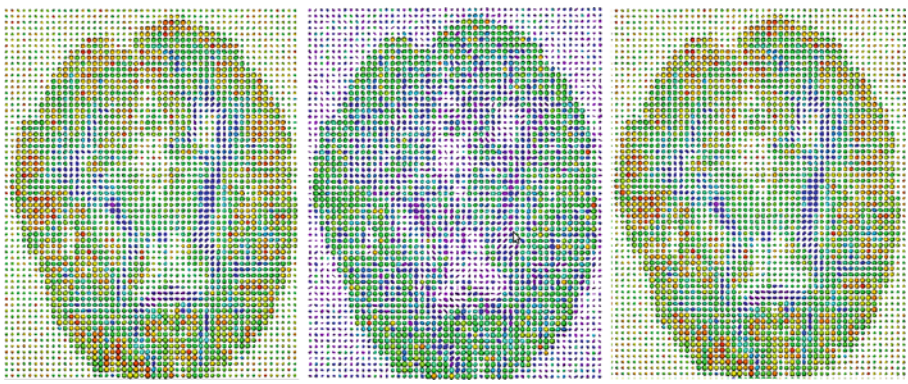


Fig. 9. TV-regularized reconstruction of images with diffusion tensor data. Left-to-right: the original image, an image with added component-wise Gaussian noise, and the denoised image. Noise was of standard deviation 100, $\lambda = 1 \times 10^{-3}$.

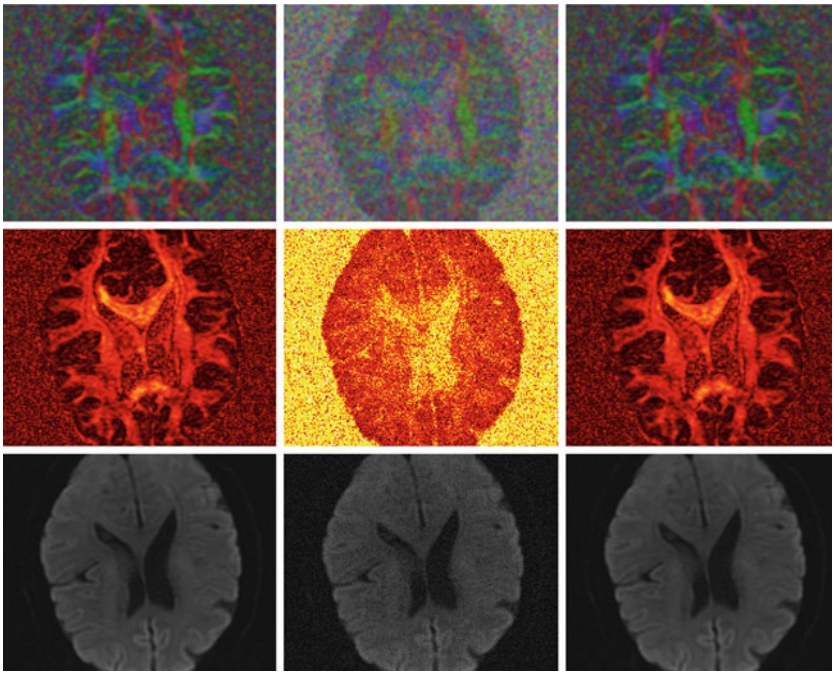


Fig. 10. TV-regularized reconstruction of diffusion tensor data. Left-to-right: the original reconstruction without noise, the noisy least-squares fitting solution (used as initialization), and the regularized reconstruction result. Top-to-bottom: a visualization of the principal directions, the fractional anisotropy, and the mean diffusivity. The noise added to the field ratio logarithm was of strength 100, $\lambda = 1 \times 10^{-3}$.

7 Conclusions

In this chapter we demonstrate thed effectiveness of augmented Lagrangian regularization of matrix-valued maps. Specifically, we have shown the efficiency and effectiveness of the resulting total-variation regularization of images with matrix-valued data taken from $SO(n)$, $SE(n)$, and $SPD(n)$. For the case of $SPD(n)$ we have shown the method's usefulness for denoising and regularized reconstruction of DTI data, as well as noted the convexity of the resulting optimization problem.

In future work we intend to explore the various ways of handling the matrix-valued regularization problem and the coupling between matrix elements, as well as extend our work into different data types and applications.

References

1. Fingerprints Verification Competition database
2. Ali, S., Shah, M.: A Lagrangian particle dynamics approach for crowd flow segmentation and stability analysis. In: Computer Vision and Pattern Recognition, pp. 1–6 (2007)

3. Attouch, H., Bolte, J., Redont, P., Soubeyran, A.: Proximal alternating minimization and projection methods for nonconvex problems: an approach based on the Kurdyka-Lojasiewicz inequality. Math. Oper. Res. **35**, 438–457 (2010)
4. Basser, P.J., Mattiello, J., LeBihan, D.: MR diffusion tensor spectroscopy and imaging. Biophys. J. **66**(1), 259–267 (1994)
5. Basu, S., Fletcher, T., Whitaker, R.T.: Rician noise removal in diffusion tensor MRI. In: Larsen, R., Nielsen, M., Sporring, J. (eds.) MICCAI 2006. LNCS, vol. 4190, pp. 117–125. Springer, Heidelberg (2006)
6. Bayro-Corrochano, E., Ortegón-Aguilar, J.: Lie algebra approach for tracking and 3D motion estimation using monocular vision. Image Vision Comput. **25**, 907–921 (2007)
7. Bergmann, Ø., Christiansen, O., Lie, J., Lundervold, A.: Shape-adaptive DCT for denoising of 3D scalar and tensor valued images. J. Digit. Imaging **22**(3), 297–308 (2009)
8. Besl, P.J., McKay, N.D.: A method for registration of 3D shapes. IEEE Trans. Pattern Anal. Mach. Intell. **14**(2), 239–256 (1992)
9. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
10. Del Bue, A., Xavier, J., Agapito, L., Paladini, M.: Bilinear factorization via augmented lagrange multipliers. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 283–296. Springer, Heidelberg (2010)
11. Burgeth, B., Didas, S., Florack, L.M.J., Weickert, J.: A generic approach to the filtering of matrix fields with singular PDEs. In: Sgallari, F., Murli, A., Paragios, N. (eds.) SSVM 2007. LNCS, vol. 4485, pp. 556–567. Springer, Heidelberg (2007)
12. Celledoni, E., Owren, B.: Lie group methods for rigid body dynamics and time integration on manifolds. Comput. Meth. Appl. Mech. Eng. **19**, 421–438 (1999)
13. Chen, B., Hsu, E.W.: Noise removal in magnetic resonance diffusion tensor imaging. Magn. Reson. Med.: Official J. Soc. Magn. Reson. Med./Soc. Magn. Reson. Med. **54**(2), 393–401 (2005)
14. Deriche, R., Tschumperle, D., Lenglet, C.: DT-MRI estimation, regularization and fiber tractography. In: ISBI, pp. 9–12 (2004)
15. Duits, R., Burgeth, B.: Scale spaces on lie groups. In: Sgallari, F., Murli, A., Paragios, N. (eds.) SSVM 2007. LNCS, vol. 4485, pp. 300–312. Springer, Heidelberg (2007)
16. Dumortier, Y., Herlin, I., Ducrot, A.: 4D tensor voting motion segmentation for obstacle detection in autonomous guided vehicle. In: IEEE Int. Vehicles Symp., pp. 379–384 (2008)
17. Ekeland, I., Temam, R.: Convex Analysis and Variational Problems. CMS Books in Mathematics. SIAM, Philadelphia (1999)
18. Fletcher, R.: Semi-definite matrix constraints in optimization. SIAM J. Cont. Optim. **23**(4), 493–513 (1985)
19. Gibson, W.: On the least-squares orthogonalization of an oblique transformation. Psychometrika **27**, 193–195 (1962)
20. Gur, Y., Sochen, N.A.: Regularizing flows over lie groups. J. Math. Imaging Vis. **33**(2), 195–208 (2009)
21. Hall, B.C.: Lie Groups, Lie Algebras, and Representations. An Elementary Introduction. Springer, New York (2004)
22. Hesteness, M.R.: Multipliers and gradient methods. J. Optim. Theor. Appl. **4**, 303–320 (1969)
23. Higham, N.J.: Matrix nearness problems and applications. In: Applications of Matrix Theory, pp. 1–27. Oxford University Press, Oxford (1989)

24. Iserles, A., Munthe-kaas, H.Z., Nørsett, S.P., Zanna, A.: Lie group methods. Acta Numerica **9**, 215–365 (2000)
25. Kimmel, R., Sochen, N.: Orientation diffusion or how to comb a porcupine. J. Vis. Commun. Image Representation **13**, 238–248 (2002). (special issue on PDEs in Image Processing, Computer Vision, and Computer Graphics)
26. Kiryati, N., Riklin-Raviv, T., Ivanchenko, Y., Rochel, S.: Real-time abnormal motion detection in surveillance video. In: ICPR, pp. 1–4 (2008)
27. Koay, C., Carew, J., Alexander, A., Basser, P., Meyerand, M.: Investigation of anomalous estimates of tensor-derived quantities in diffusion tensor imaging. Magn. Reson. Med. **55**, 930–936 (2006)
28. Kobilarov, M., Crane, K., Desbrun, M.: Lie group integrators for animation and control of vehicles. ACM Trans. Graph. **28**(2), 1–14 (2009)
29. Larochelle, P.M., Murray, A.P., Angeles, J.: SVD and PD based projection metrics on SE(N). In: Lenarčič, J., Galletti, C. (eds.) On Advances in Robot Kinematics, pp. 13–22. Kluwer, Dordrecht (2004)
30. Lenglet, C., Campbell, J.S.W., Descoteaux, M., Haro, G., Savadjiev, P., Wassermann, D., Anwander, A., Deriche, R., Pike, G.B., Sapiro, G.: Mathematical methods for diffusion MRI processing. Neuroimage **45**(1), S111–S122 (2009)
31. Lin, D., Grimson, W., Fisher, J.: Learning visual flows: a Lie algebraic approach. In: Computer Vision and Pattern Recognition, pp. 747–754 (2009)
32. Lundervold, A.: On consciousness, resting state fMRI, and neurodynamics. Nonlinear Biomed. Phys. **4**(Suppl. 1), S9–S18 (2010)
33. Manton, J.H.: Optimization algorithms exploiting unitary constraints. IEEE Trans. Signal Process. **50**(3), 635–650 (2002)
34. Mehran, R., Moore, B.E., Shah, M.: A streakline representation of flow in crowded scenes. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part III. LNCS, vol. 6313, pp. 439–452. Springer, Heidelberg (2010)
35. Moisan, L.: Perspective invariant movie analysis for depth recovery. Proc. SPIE **2567**, 84–94 (1995)
36. Nicolescu, M., Medioni, G.: A voting-based computational framework for visual motion analysis and interpretation. IEEE Trans. Pattern Anal. Mach. Intell. **27**, 739–752 (2005)
37. Park, F.C., Bobrow, J.E., Ploen, S.R.: A lie group formulation of robot dynamics. Int. J. Rob. Res. **14**, 609–618 (1995)
38. Park, W., Liu, Y., Zhou, Y., Moses, M., Chirikjian, G.S.: Kinematic state estimation and motion planning for stochastic nonholonomic systems using the exponential map. Robotica **26**, 419–434 (2008)
39. Pennec, X., Fillard, P., Ayache, N.: A riemannian framework for tensor computing. Int. J. Comput. Vis. **66**(1), 41–66 (2006)
40. Perona, P.: Orientation diffusions. IEEE Trans. Image Process. **7**(3), 457–467 (1998)
41. Powell, M.J.: A method for nonlinear constraints in minimization problems. In: Optimization, pp. 283–298. Academic Press (1969)
42. Rahman, I.U., Drori, I., Stodden, V.C., Donoho, D.L., Schroeder, P.: Multiscale representations of manifold-valued data. Technical report, Stanford (2005)
43. Raptis, M., Soatto, S.: Tracklet descriptors for action modeling and video analysis. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part I. LNCS, vol. 6311, pp. 577–590. Springer, Heidelberg (2010)
44. Rosman, G., Bronstein, M.M., Bronstein, A.M., Wolf, A., Kimmel, R.: Group-valued regularization framework for motion segmentation of dynamic non-rigid

shapes. In: Bronstein, A.M., Bronstein, M.M., Bruckstein, A.M., Haar Romeny, B.M. (eds.) SSVM 2011. LNCS, vol. 6667, pp. 725–736. Springer, Heidelberg (2012)
45. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Phys. D Lett. **60**, 259–268 (1992)
46. Sagiv, C., Sochen, N.A., Kimmel, R.: Stereographic combing a porcupine or studies on direction diffusion in image processing. SIAM J. Appl. Math. **64**(5), 1477–1508 (2004)
47. Salvador, R., Pena, A., Menon, D.K., Carpenter, T., Pickard, J., Bullmore, E.: Formal characterization and extension of the linearized diffusion tensor model. Hum. Brain Mapp. **24**(2), 144–155 (2005)
48. Steidl, G., Setzer, S., Popilka, B., Burgeth, B.: Restoration of matrix fields by second-order cone programming. Computing **81**(2–3), 161–178 (2007)
49. Stejskal, E.O., Tanner, J.E.: Spin diffusion measurements: spin echoes in the presence of a time-dependent field gradient. J. Chem. Phys. **42**, 288–292 (1965)
50. Stillwell, J.: Naive Lie Theory. Undergraduate texts in mathematics. Springer, New York (2008)
51. Tai, X.-C., Wu, C.: Augmented Lagrangian method, dual methods and split Bregman iteration for ROF model. In: Tai, X.-C., Mørken, K., Lysaker, M., Lie, K.-A. (eds.) SSVM 2009. LNCS, vol. 5567, pp. 502–513. Springer, Heidelberg (2009)
52. Tang, B., Sapiro, G., Caselles, V.: Diffusion of general data on non-flat manifolds viaharmonic maps theory: the direction diffusion case. Int. J. Comput. Vis. **36**, 149–161 (2000)
53. Tschumperlé, D., Deriche, R.: Vector-valued image regularization with pdes: a common framework for different applications. IEEE Trans. Pattern Anal. Mach. Intell. **27**, 506–517 (2005)
54. Tseng, P.: Coordinate ascent for maximizing nondifferentiable concave functions. LIDS-P 1940, MIT (1988)
55. Turaga, P., Veeraraghavan, A., Srivastava, A., Chellappa, R.: Statistical computations on grassmann and stiefel manifolds for image and video-based recognition. IEEE Trans. Pattern Anal. Mach. Intell. **33**, 2273–2286 (2011)
56. Tuzel, O., Porikli, F., Meer, P.: Learning on lie groups for invariant detection and tracking, In: Computer Vision and Pattern Recognition (2008)
57. Vemuri, B.C., Chen, Y., Rao, M., McGraw, T., Wang, Z., Mareci, T.: Fiber tract mapping from diffusion tensor MRI. In: Proceedings of the International Conference on Variational, Geometry and Level Sets Methods in Computer Vision, pp. 81–88. IEEE Computer Society (2001)
58. Vese, L.A., Osher, S.J.: Numerical methods for p-Harmonic flows and applications to image processing. SIAM J. Numer. Anal. **40**(6), 2085–2104 (2002)
59. Žefran, M., Kumar, V., Croke, C.: On the generation of smooth three-dimensional rigid body motions. IEEE Trans. Robot. Autom. **14**(4), 576–589 (1998)
60. Žefran, M., Kumar, V., Croke, C.: Metrics and connections for rigid-body kinematics. I. J. Robotic Res. **18**(2), 242 (1999)
61. Wang, Y., Yang, J., Yin, W., Zhang, Y.: A new alternating minimization algorithm for total variation image reconstruction. SIAM J. Imag. Sci. **1**(3), 248–272 (2008)
62. Weickert, J., Brox, T.: Diffusion and regularization of vector- and matrix-valued images. Inverse problems, image analysis, and medical imaging, vol. 313 (2002)
63. Wen, Z., Goldfarb, D., Yin, W.: Alternating direction augmented Lagrangian methods for semidefinite programming. CAAM TR09-42, Rice University (2009)
64. Westin, C.-F., Peled, S., Gudbjartsson, H., Kikinis, R., Jolesz, F.A.: Geometrical diffusion measures for MRI from tensor basis analysis. In: ISMRM (1997)

65. Wiest-Daesslé, N., Prima, S., Coupé, P., Morrissey, S.P., Barillot, Ch.: Non-local means variants for denoising of diffusion-weighted and diffusion tensor MRI. In: Ayache, N., Ourselin, S., Maeder, A. (eds.) MICCAI 2007, Part II. LNCS, vol. 4792, pp. 344–351. Springer, Heidelberg (2007)
66. Wu, C., Tai, X.-C.: Augmented lagrangian method, dual methods, and split Bregman iteration for ROF, vectorial TV, and high order models. SIAM J. Imaging Sci. **3**(3), 300–339 (2010)
67. Wu, C., Zhang, J., Tai, X.: Augmented lagrangian method for total variation restoration with non-quadratic fidelity. Inverse Prob. Imaging **5**(1), 237–261 (2011)

Recovering Piecewise Smooth Multichannel Images by Minimization of Convex Functionals with Total Generalized Variation Penalty

Kristian Bredies(✉)

Institute of Mathematics and Scientific Computing, University of Graz,
Heinrichstraße 36, 8010 Graz, Austria
kristian.bredies@uni-graz.at
http://www.uni-graz.at/~bredies

Abstract. We study and extend the recently introduced total generalized variation (TGV) functional for multichannel images. This functional has already been established to constitute a well-suited convex model for piecewise smooth scalar images. It comprises exactly the functions of bounded variation but is, unlike purely total-variation based functionals, also aware of higher-order smoothness. For the multichannel version which is developed in this paper, basic properties and existence of minimizers for associated variational problems regularized with second-order TGV is shown. Furthermore, we address the design of numerical solution methods for the minimization of functionals with TGV^2 penalty and present, in particular, a class of primal-dual algorithms. Finally, the concrete realization for various image processing problems, such as image denoising, deblurring, zooming, dequantization and compressive imaging, are discussed and numerical experiments are presented.

Keywords: Total generalized variation · Multichannel images · Primal-dual algorithms · Image denoising · Image deblurring · Zooming · Dequantization · Compressive imaging

1 Introduction

Many imaging problems are nowadays solved by variational methods, i.e., by finding a minimizer of a functional which models the problem in terms of encouraging potential solutions of the problems by low values and penalizing unsuitable images by high values. Typically, the variational problems are cast in the form

$$\min_u \; F(u) + \Psi(u)$$

where data term F models the fitness of the image u with respect to some given data and the regularization functional Ψ represents an underlying image model incorporating the essential features of the sought class of images. The latter functional is responsible for the qualitative properties of the solutions, it is therefore

A. Bruhn et al. (Eds.): Global Optimization Methods, LNCS 8293, pp. 44–77, 2014.
DOI: 10.1007/978-3-642-54774-4_3,

important to choose it appropriately. As images often possess multiple channels such as red, green, blue (RGB) or cyan, magenta, yellow, black (CYMK), such a model should also account for multichannel data. Moreover, regarding the efficient computation of numerical solutions, convexity of the objective functional is of great significance. In terms of algorithms, it is therefore favorable to consider convex models Ψ. This work is concerned with a multichannel version of the *total generalized variation* (TGV), which has been introduced, in its scalar form, in [4] and provides a well-suited convex model for piecewise smooth images. In particular, we study methods for the efficient global minimization of associated variational problems.

Let us discuss, along TGV, some existing regularization functionals for scalar images which are well-known and used in mathematical image processing. The most widely used is probably the *total variation* seminorm [28] which reads as

$$\mathrm{TV}(u) = \int_\Omega \mathrm{d}|\nabla u| = \sup\left\{ \int_\Omega u \operatorname{div} v \,\mathrm{d}x \;\middle|\; v \in \mathcal{C}_c^1(\Omega, \mathbf{R}^d),\ \|v\|_\infty \le 1 \right\}.$$

where $|\nabla u|$ denotes the variation-measure of the distributional derivative Du which is a vector-valued Radon measure. Its main feature is the incorporation of discontinuities along hypersurfaces making it a suitable model for images with edges. Indeed, solutions of variational problems with total-variation regularization admit many desirable properties, most notably the appearance of sharp edges. Unfortunately, one can also observe typical artifacts which are associated with the regularization with TV. The most prominent of these artifacts is the so-called *staircasing effect*, i.e., the undesired appearance of edges [21,36]. This is a side-effect of the model assumption that an image consists is piecewise constant up to a discontinuity set. Natural images are, however, often piecewise smooth due to shading, for instance. Several modified models have been suggested to overcome this limitation. The most famous is the Mumford-Shah model [20] which reads as

$$\Psi_{\mathrm{MS}}(u) = \int_{\Omega\setminus\Gamma} |\nabla u|^2 \,\mathrm{d}x + \beta \mathcal{H}^{d-1}(\Gamma)$$

and measures piecewise smoothness on Ω up to the discontinuity set Γ whose length (or surface measure) is also penalized. This functional can be well-defined on the set of special functions of bounded variation $\mathrm{SBV}(\Omega)$. However, Ψ_{MS} is non-convex which implies considerable analytical and practical effort when solving associated variational problems [1,24]. As we are interested in algorithms which can efficiently and globally solve variational imaging problems, we focus, in the following, on convex problems as their stationary points are always global minimizers. An illustration of the effect for most of these models applied to the image denoising problem with L^2-discrepancy term can be found in Fig. 1.

A first approach to reduce staircasing artifacts in a convex way is to smooth out the singularity at 0 for the penalization of the gradient [35]:

$$\mathrm{TV}_\varepsilon(u) = \int_\Omega \mathrm{d}\varphi_\varepsilon(\nabla u), \qquad \varphi_\varepsilon(t) = \sqrt{|t|^2 + \varepsilon^2} - \varepsilon.$$

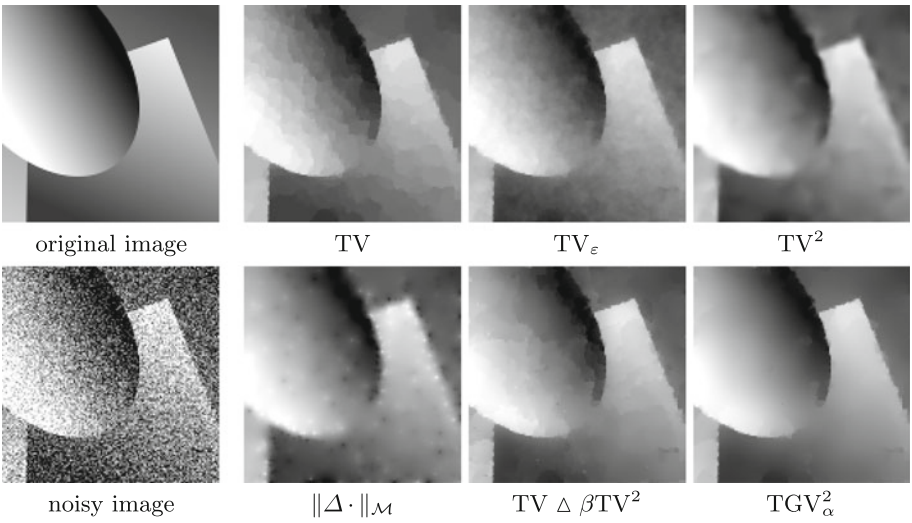


Fig. 1. Comparison of different first- and second-order image models for variational image denoising with L^2-discrepancy. Left column: The original image (top) and noisy input image (bottom). Columns 2–4: Results for variational denoising with different regularization terms. The parameters were optimized for best PSNR.

Here, $\varphi_\varepsilon(\mathrm{d}\nabla u)$ has to be interpreted in the sense of a function of a Radon measure. It can also be expressed as a dual functional:

$$\mathrm{TV}_\varepsilon(u) = \sup\Big\{ \int_\Omega u \operatorname{div} v - \varphi_\varepsilon^*(v)\ \mathrm{d}x \ \Big|\ v \in \mathcal{C}_\mathrm{c}^1(\Omega, \mathbf{R}^d) \Big\},$$

with

$$\varphi_\varepsilon^*(t) = \begin{cases} \varepsilon\big(1 - \sqrt{1 - |t|^2}\big) & \text{for } |t| < 1, \\ \infty & \text{else.} \end{cases}$$

This reduces the tendency towards piecewise constant solutions. However, as φ_ε still grows as fast as $|\cdot|$, discontinuities and consequently, the staircasing effect still appears. Such an observation can generally be made for first-order functionals penalizing the measure-valued gradient with linear growth at ∞.

One approach to overcome these defects is to incorporate higher-order derivatives into the image model. An obvious choice is taking the total variation of second order [14,19] which can also be expressed in a dual formulation using symmetric matrix fields $v : \Omega \to S^{d\times d}$:

$$\mathrm{TV}^2(u) = \int_\Omega \mathrm{d}|\nabla^2 u| = \sup\Big\{ \int_\Omega u \operatorname{div}^2 v\ \mathrm{d}x \ \Big|\ v \in \mathcal{C}_\mathrm{c}^2(\Omega, S^{d\times d}),\ \|v\|_\infty \le 1 \Big\}.$$

Here, the derivative ∇u is modeled to be piecewise constant which has the consequence that it is itself a regular function and can thus not have any discontinuities. Therefore, solutions of variational problems with TV^2 penalty cannot have jump discontinuities and object boundaries become inevitably blurry.

These effects cannot be overcome by changing the underlying second-order differentiation operator. For instance, taking the Laplacian

$$\|\Delta u\|_{\mathcal{M}} = \int_\Omega \mathrm{d}|\Delta u| = \sup\left\{ \int_\Omega u \Delta v \,\mathrm{d}x \;\middle|\; v \in \mathcal{C}_\mathrm{c}^2(\Omega),\ \|v\|_\infty \leq 1 \right\}$$

leads to a significantly weaker smoothness measure whose kernel is the set of harmonic functions on Ω which are arbitrarily smooth. Moreover, regularity theory for elliptic equations [30] tells us that each function $u \in L^1(\Omega)$ with $\Delta u \in \mathcal{M}(\Omega)$ also belongs to the Sobolev space $W^{1,q}_{\mathrm{loc}}(\Omega)$ for each $1 \leq q < d/(d-1)$ and hence, u can also not contain jump discontinuities.

One approach to incorporate smoothness information on different scales is to combine first- and second-order derivatives. This can, for instance, be realized by considering the weighted sum of TV and TV^2 [23]. Another possibility is to interpret an image $u = u^1 + u^2$ as the sum of a piecewise constant function u^1 and piecewise smooth function u^2. This results in *infimal convolution* models, for instance, with TV and TV^2:

$$(\mathrm{TV} \mathbin{\triangle} \beta\mathrm{TV}^2)(u) = \inf_{u=u^1+u^2} \int_\Omega \mathrm{d}|\nabla u^1| + \beta \int_\Omega \mathrm{d}|\nabla^2 u^2|.$$

Indeed, this model yields piecewise smooth functions [8]. However, plugging this functional into variational problems given solutions which still admit staircasing artifacts in situations where they also appear for TV-model. It seems that the tendency of TV^2 to incorporate the smoothness information of u is not strong enough such that the TV-term is still responsible for the overall impression of the solution. Again, changing the second-order term in the infimal convolution to $\|\Delta u\|_{\mathcal{M}}$, for instance [10], leads to results which are comparable to TV-TV^2 infimal convolution.

The *total generalized variation* model [4] can now be motivated by the dual formulation of $(\mathrm{TV} \mathbin{\triangle} \beta\mathrm{TV}^2)$ which reads as

$$\begin{aligned}(\mathrm{TV} \mathbin{\triangle} \beta\mathrm{TV}^2)(u) = \sup\Big\{ \int_\Omega uw \,\mathrm{d}x \;\Big|\;& v^1 \in \mathcal{C}_\mathrm{c}^1(\Omega, \mathbf{R}^d),\ \|v^1\|_\infty \leq 1,\\ & v^2 \in \mathcal{C}_\mathrm{c}^2(\Omega, S^{d\times d}),\ \|v^2\|_\infty \leq \beta,\\ & w = \operatorname{div} v^1 = \operatorname{div}^2 v^2 \Big\}.\end{aligned}$$

The total generalized variation of second order now arises from the introduction of the additional constraint $v^1 = \operatorname{div} v^2$:

$$\mathrm{TGV}^2_{(\beta,1)}(u) = \sup\left\{ \int_\Omega u \operatorname{div}^2 v \,\mathrm{d}x \middle| v \in \mathcal{C}_\mathrm{c}^2(\Omega, S^{d\times d}),\ \|v\|_\infty \leq \beta,\ \|\operatorname{div} v\|_\infty \leq 1 \right\}.$$

It is a special case of the total generalized variation of order k and positive weights $\alpha = (\alpha_0, \ldots, \alpha_{k-1})$ which is defined as follows:

$$\begin{aligned}\mathrm{TGV}^k_\alpha(u) = \sup\Big\{ \int_\Omega u \operatorname{div}^k v \,\mathrm{d}x \;\Big|\;& v \in \mathcal{C}_\mathrm{c}^k(\Omega, \mathrm{Sym}^k(\mathbf{R}^d)),\\ & \|\operatorname{div}^\kappa v\|_\infty \leq \alpha_\kappa,\ \kappa = 0, \ldots, k-1 \Big\}. \quad (1)\end{aligned}$$

For a detailed description of the notions utilized here, we ask for the reader's patience until Sect. 2. This functional can be interpreted to measure smooth regions as well as jump discontinuities in a convex manner. In particular, it leads to solutions which desirable properties when used as a regularization term for variational imaging problems.

The models presented above also extend to the multichannel case. Among the several choices which are possible, the most obvious is just summing up the respective regularization functionals over each channel. In this situation, the qualitative properties of the solutions of variational problems are comparable to the scalar versions. However, there are other choices which realize coupling between the channels, e.g. for TV, see [7,37]. Nevertheless, these models again lead to results which are similar to the scalar case in particular, the typical staircasing artifacts are also present. We can therefore expect that the qualitative properties of scalar models are generally reflected in respective multichannel versions. This motivates to define the total generalized variation functional also for multichannel data.

The aim of the present paper is, on the one hand, to shortly review as well as to extend the notion of total generalized variation to multichannel images and thus to provide a framework for color images. This is done in Sect. 2. Moreover, we present and discuss, in Sect. 3 a class of numerical methods which are easy to implement and suitable to solve general convex variational imaging problems with TGV^2_α-penalty. These are applied in Sect. 4 to a variety of imaging problems: denoising, deblurring, zooming, dequantization and compressive imaging. This includes in particular a specific numerical algorithm for each of these problems. Finally, conclusions are drawn in Sect. 5.

2 Total Generalized Variation

2.1 General Theory for Scalar Functions

Let us first review the concept of total generalized variation (1) for the scalar case as introduced [4], starting with a more detailed explanation of the notions involved in its definition.

Throughout this section, we assume that $d \in \mathbf{N}$, $d \geq 1$ is a fixed space dimension, usually, for images, we have $d = 2$. Moreover, let $\Omega \subset \mathbf{R}^d$ be a domain, i.e., a non-empty, open and connected set. We need the space of *symmetric tensors* on $\mathbf{R}^d$, denoted by $\mathrm{Sym}^k(\mathbf{R}^d)$. The latter is defined, for each $k \in \mathbf{N}$, as

$$\mathrm{Sym}^k(\mathbf{R}^d) = \{\xi : \underbrace{\mathbf{R}^d \times \cdots \times \mathbf{R}^d}_{k \text{ times}} \to \mathbf{R} \;\big|\; \xi \text{ multilinear and symmetric}\}.$$

It is, however, convenient to identify elements $\xi \in \mathrm{Sym}^k(\mathbf{R}^d)$ with its coefficients $\{\xi_\beta\}_{\beta \in M_k}$ where

$$M_k = \Big\{\beta \in \mathbf{N}^d \;\Big|\; |\beta| = \sum_{i=1}^{d} \beta_i = k\Big\},$$

is the set of multiindices of order k. This allows to define the spaces of compactly supported symmetric tensor fields $\mathcal{C}_c^m(\Omega, \mathrm{Sym}^k(\mathbf{R}^d))$ for $m, k \in \mathbf{N}$. For symmetric k-tensor fields which are smooth enough, iterated divergence operators are defined componentwise by

$$(\mathrm{div}^\kappa v)_\beta = \sum_{\gamma \in M_\kappa} \frac{\kappa!}{\gamma!} \frac{\partial^\kappa v_{\beta+\gamma}}{\partial x^\gamma} \qquad \text{for each component } \beta \in M_{k-\kappa}.$$

Here, $\gamma!$ denotes the factorial for multiindices, i.e., $\gamma! = \prod_{i=1}^d \gamma_i!$. Moreover, we define the supremum norm of a compactly supported continuous symmetric tensor field $v \in \mathcal{C}_c(\Omega, \mathrm{Sym}^k(\mathbf{R}^d))$ as

$$\|v\|_\infty = \sup_{x \in \Omega} \Big\{ \Big(\sum_{\beta \in M_k} \frac{k!}{\beta!} v_\beta(x)^2 \Big)^{1/2} \Big\}$$

which corresponds to the ∞-norm with respect to the pointwise Frobenius norm for tensors.

With these prerequisites, TGV_α^k according to (1) makes sense for any scalar function $u \in L^1_{\mathrm{loc}}(\Omega)$, any order $k \in \mathbf{N}$, $k \geq 1$ and any set of weights $\alpha = (\alpha_0, \ldots, \alpha_{k-1})$ satisfying $\alpha_\kappa > 0$ for $\kappa = 0, \ldots, k-1$.

As we will later focus on total generalized variation of second order, i.e., $k = 2$, let us elaborate on the above notions in this specific case. It turns out that TGV_α^2 can equally be written as

$$\mathrm{TGV}_\alpha^2(u) = \sup \Big\{ \int_\Omega u \, \mathrm{div}^2 v \, \mathrm{d}x \,\Big|\, v \in \mathcal{C}_c^2(\Omega, S^{d\times d}),\ \|v\|_\infty \leq \alpha_0,\ \|\mathrm{div}\, v\|_\infty \leq \alpha_1 \Big\}$$

with $S^{d\times d}$ denoting the space of symmetric $d \times d$ matrices. The first and second divergences of a symmetric matrix field are then vector and scalar fields, respectively, given by

$$(\mathrm{div}\, v)_i = \sum_{j=1}^d \frac{\partial v_{ij}}{\partial x_j}, \qquad (\mathrm{div}^2 v) = \sum_{i=1}^d \frac{\partial v_{ii}}{\partial x_i^2} + 2 \sum_{i=1}^2 \sum_{j<i} \frac{\partial v_{ij}}{\partial x_i \partial x_j}.$$

Likewise, the ∞-norms of matrix and vector fields v and w, respectively, used here are given by

$$\|v\|_\infty = \sup_{x \in \Omega} \Big\{ \Big(\sum_{i=1}^d v_{ii}(x)^2 + 2 \sum_{i=1}^d \sum_{j<i} v_{ij}(x)^2 \Big)^{1/2} \Big\},$$

$$\|w\|_\infty = \sup_{x \in \Omega} \Big\{ \Big(\sum_{i=1}^d w_i(x)^2 \Big)^{1/2} \Big\}.$$

Let us now summarize some basic properties of the total generalized variation and discuss solvability of inverse problems with TGV-regularization for the second-order case.

First, observe that for $k = 1$, it follows from the definition (1) that the total generalized variation coincides, up to a factor, with the total variation, i.e., we have $\mathrm{TGV}_\alpha^k = \alpha_0 \mathrm{TV}$. Hence, one can indeed speak of a generalization of the total variation.

Having defined TGV_α^k according to (1), one can show that it constitutes a proper, convex and lower semi-continuous functional on each $L^p(\Omega)$ space $(1 \leq p < \infty)$ which is moreover translation and rotation invariant. The space

$$\mathrm{BGV}^k(\Omega) = \{u \in L^1(\Omega) \mid \mathrm{TGV}_\alpha^k(u) < \infty\}, \qquad \|u\|_{\mathrm{BGV}^k} = \|u\|_1 + \mathrm{TGV}_\alpha^k(u)$$

is a Banach space which is independent of the weights $\alpha_0, \ldots, \alpha_{k-1}$ chosen in the definition of TGV_α^k. On this space, TGV_α^k is a semi-norm which vanishes exactly on $\mathcal{P}_{k-1}(\Omega)$, the space of polynomials of degree less than or equal to $k-1$. It can be interpreted as a model for piecewise smooth functions as follows. Let u be piecewise polynomials of maximal degree $k-1$, i.e.,

$$u = \sum_{i=1}^n \chi_{\Omega_i} q_i$$

where $\Omega_1, \ldots, \Omega_n \subset \Omega$ are disjoint Lipschitz subdomains such that $\overline{\Omega} = \bigcup_{i=1}^n \overline{\Omega_i}$ and $q_i \in \mathcal{P}_{k-1}(\Omega)$ for $i = 1, \ldots, n$. Then, we have that TGV_α^k is finite and measures the jump of the derivatives, from the zeroth to the $(k-1)$-st order, of these polynomials only at the interfaces $\Gamma_{i,j} = \partial\Omega_i \cap \partial\Omega_j \cap \Omega$:

$$\mathrm{TGV}_\alpha^k(u) \leq \frac{1}{2} \sum_{i,j=1}^n \int_{\Gamma_{i,j}} \sum_{\kappa=0}^{k-1} \left| \| \left(\nabla^{k-1-\kappa}(q_i - q_j) \otimes \nu_i \right) \right| \, \mathrm{d}\mathcal{H}^{d-1}(x)$$

where $\|\|$ denotes the symmetrization of a tensor and ν_i the outer normal of Ω_i. In some cases, the estimate can be proven to be sharp. Again, see [4] for the proofs and more details.

For the second-order total generalized variation it has been shown in [5] that for $u \in L^1(\Omega)$,

$$\mathrm{TGV}_\alpha^2(u) = \min_{p \in \mathrm{BD}(\Omega)} \alpha_1 \|\nabla u - p\|_{\mathcal{M}} + \alpha_0 \|\mathcal{E}(p)\|_{\mathcal{M}}.$$

Here, $\mathrm{BD}(\Omega)$ denotes the space of vector fields of bounded deformation [33], i.e., the set of vector fields whose weak symmetrized derivative $\mathcal{E}(p) = \frac{1}{2}(\nabla p + \nabla p^{\mathrm{T}})$ is a matrix-valued Radon measure. Moreover, $\|\cdot\|_{\mathcal{M}}$ denotes the Radon norm for vector-valued and matrix-valued Radon measures, respectively.

Furthermore, for bounded Lipschitz domains $\Omega \subset \mathbf{R}^d$, $\mathrm{BGV}^2(\Omega)$ coincides with $\mathrm{BV}(\Omega)$, the space of functions of bounded variation, in the topological sense, i.e., there exist $0 < c < C < \infty$ such that for each $u \in \mathrm{BV}(\Omega)$,

$$c\big(\|u\|_1 + \mathrm{TGV}_\alpha^2(u)\big) \leq \|u\|_1 + \mathrm{TV}(u) \leq C\big(\|u\|_1 + \mathrm{TGV}_\alpha^2(u)\big).$$

We therefore have also the usual embeddings $\mathrm{BGV}^2(\Omega) \hookrightarrow L^p(\Omega)$ for $1 \leq p \leq d/(d-1)$ which are compact for $p < d/(d-1)$. Finally, there is also a variant of the

Poincaré-Friedrichs inequality available which states that for a linear projection $P : L^{d/(d-1)}(\Omega) \to \mathcal{P}_1(\Omega)$ onto $\mathcal{P}_1(\Omega)$, we can find a constant $C > 0$, only depending on Ω, P and α such that

$$\|u - Pu\|_{d/(d-1)} \leq C\mathrm{TGV}^2_\alpha(u) \qquad \text{for all } u \in \mathrm{BV}(\Omega). \tag{2}$$

This can be used to solve the linear inverse problem

$$Ku = f$$

with TGV^2_α-regularization. Indeed, existence of solutions for the Tikhonov functional

$$\min_{u \in L^p(\Omega)} \frac{\|Ku - f\|_H^2}{2} + \mathrm{TGV}^2_\alpha(u)$$

where $p \in (1, \infty)$, $p \leq d/(d-1)$, $K : L^p(\Omega) \to H$ is linear, continuous and H is a Hilbert space can be shown under the assumption that K is injective on $\ker(\mathrm{TGV}^2_\alpha) = \mathcal{P}_1(\Omega)$. Proofs and more details regarding these results can be found in [5].

2.2 Extension to Multichannel Images

In order to deal with, e.g., color images, it is convenient to have a notion of total generalized variation also for vector-valued images $u : \Omega \to \mathbf{R}^L$ for some $L \geq 1$. Here, we assume that color information can be encoded by a linear space, for instance $\mathbf{R}^3$ for the RGB or YUV color space and $\mathbf{R}^4$ for the CYMK color space. The space $\mathbf{R}^L$ is then equipped with a norm $|\cdot|_\circ$ which provides a way to compare colors allowing to incorporate the characteristics of different color channels.

In order to define TGV^k_α for vector-valued images, we need to know the dual norm of $|\cdot|_\circ$ which is given, for $y \in \mathbf{R}^L$, by

$$|y|_* = \sup_{x \in \mathbf{R}^L,\ |x|_\circ \leq 1} x \cdot y.$$

This dual norm can now be extended to L-tuples of symmetric tensors $\xi \in \mathrm{Sym}^k(\mathbf{R}^d)^L$ by setting

$$|\xi|_{*,k} = \big||\xi|\big|_*, \qquad \text{where} \qquad |\xi| \in \mathbf{R}^L, \quad |\xi|_l = \Big(\sum_{\beta \in M_k} \frac{k!}{\beta!}\xi_\beta^2\Big)^{1/2}. \tag{3}$$

Consequently, the ∞-norms for compactly supported symmetric tensor fields of order k read as

$$v \in \mathcal{C}_\mathrm{c}(\Omega, \mathrm{Sym}^k(\mathbf{R}^d)^L) : \qquad \|v\|_{\infty,*,k} = \sup_{x \in \Omega} |v(x)|_{*,k}.$$

A vector-valued version of TGV^k_α for a $u \in L^1_{\mathrm{loc}}(\Omega, \mathbf{R}^L)$ can then be defined as

$$\mathrm{TGV}^k_\alpha(u) = \sup \Big\{ \int_\Omega \sum_{l=1}^{L} u_l(\mathrm{div}^k v_l)\, \mathrm{d}x \;\Big|\; v \in \mathcal{C}^k_\mathrm{c}(\Omega, \mathrm{Sym}^k(\mathbf{R}^d)^L), \\ \|\mathrm{div}^\kappa v\|_{\infty,*,\kappa} \leq \alpha_\kappa,\ \kappa = 0, \dots, k-1 \Big\}. \tag{4}$$

Slightly abusing notation and since in the scalar case, it coincides with (1) up to a positive constant, TGV^k_α will in the following always refer to this definition unless it is explicitly stated otherwise.

Example 1. An obvious choice for $|\cdot|_\circ$ is, of course, the Euclidean norm $|x|_\circ = \left(\sum_{l=1}^L x_l^2\right)^{1/2}$. It is dual to itself, so

$$\xi \in \mathrm{Sym}^k(\mathbf{R}^d)^L : \qquad |\xi|_{*,k} = \Big(\sum_{l=1}^L \sum_{\beta\in M_k} \frac{k!}{\beta!}\xi_{l,\beta}^2\Big)^{1/2}$$

which is a generalization of the Frobenius norm to $\mathrm{Sym}^k(\mathbf{R}^d)^L$ and therefore making this space a Hilbert space. The associated TGV^k_α involves, consequently, only Hilbert norms which can be exploited in numerical computations.

With more or less effort we can see that this functional possesses basically the same properties as TGV^k_α for the scalar case. As we will use them in the sequel, we highlight some of these properties. We start with basic observations.

Proposition 1. *The functional* TGV^k_α *is non-negative on* $L^1_{\mathrm{loc}}(\Omega, \mathbf{R}^L)$. *For each* $1 \leq p \leq \infty$, TGV^k_α *restricted to* $L^p(\Omega, \mathbf{R}^L)$ *is proper, convex and lower semi-continuous.*

Proof. Note that for each $v \in \mathcal{C}^k_{\mathrm{c}}(\Omega, \mathrm{Sym}^k(\mathbf{R}^d)^L)$ which satisfies the constraints $\|\mathrm{div}^\kappa\|_{\infty,\kappa} \leq \alpha_\kappa$, we also have that $-v$ satisfies the same constraints. Plugging in both v and $-v$, we see that we can replace $\int_\Omega \sum_{l=1}^L u_l \mathrm{div}^k v_l \, \mathrm{d}x$ by its absolute value in (4) without changing the supremum. Hence, $\mathrm{TGV}^k_\alpha(u) \geq 0$.

To see that TGV^k_α is proper, observe that $\mathrm{TGV}^k_\alpha(0) = 0$. Finally, for each $v \in \mathcal{C}^k_{\mathrm{c}}(\Omega, \mathrm{Sym}^k(\mathbf{R}^d)^L)$ which satisfies the constraints in (4), we have $\mathrm{div}^k v \in \mathcal{C}_{\mathrm{c}}(\Omega, \mathbf{R}^L)$ and hence, the mapping

$$u \mapsto \int_\Omega \sum_{l=1}^L u_l \, \mathrm{div}^k v_l \, \mathrm{d}x$$

is in the dual space of $L^p(\Omega, \mathbf{R}^L)$. Consequently, TGV^k_α is a pointwise supremum of convex and continuous functionals on $L^p(\Omega, \mathbf{R}^L)$ which implies the convexity and lower semi-continuity. □

The next is the observation that each of the vector-valued TGV^k_α are equivalent in the following sense:

Proposition 2. *There are constants* $0 < c < C < \infty$ *such that for each* $u \in L^1_{\mathrm{loc}}(\Omega, \mathbf{R}^L)$, *we have*

$$c \sum_{l=1}^L \mathrm{TGV}^k_\alpha(u_l) \leq \mathrm{TGV}^k_\alpha(u) \leq C \sum_{l=1}^L \mathrm{TGV}^k_\alpha(u_l). \tag{5}$$

Here, TGV^k_α *on the left- and right-hand side has to be understood in the sense of (1).*

Proof. Denote by $|x|_1 = \sum_{l=1}^{L} |x_l|$ the 1-norm on $\mathbf{R}^L$ whose dual is the ∞-norm $|y|_\infty = \max_{l=1,\dots,L} |y_l|$. Note that by (3),

$$\|\xi\|_{\infty,k} = \max_{l=1,\dots,L} |\xi|_l \qquad \text{for all} \qquad \xi \in \mathrm{Sym}^k(\mathbf{R}^d)^L. \tag{6}$$

For a fixed k and an arbitrary norm $|\cdot|_\circ$ in $\mathbf{R}^L$, the corresponding tensor norms $|\cdot|_{*,\kappa}$ are equivalent to $|\cdot|_{\infty,\kappa}$, i.e., there exist $0 < c < C < \infty$ such that

$$\kappa = 0, \dots, k-1 \ \text{ and } \ \xi \in \mathrm{Sym}^\kappa(\mathbf{R}^d)^L : \qquad C^{-1}|\xi|_{\infty,\kappa} \le |\xi|_{*,\kappa} \le c^{-1}|\xi|_{\infty,\kappa}.$$

This implies for $v \in \mathcal{C}_c^k(\Omega, \mathrm{Sym}^k(\mathbf{R}^d)^L)$ that

$$\begin{aligned} \|\mathrm{div}^\kappa v\|_{\infty,*,\kappa} \le \alpha_\kappa \qquad &\Rightarrow \qquad \|\mathrm{div}^\kappa v\|_{\infty,\infty,\kappa} \le C\alpha_\kappa, \\ \|\mathrm{div}^\kappa v\|_{\infty,\infty,\kappa} \le c\alpha_\kappa \qquad &\Rightarrow \qquad \|\mathrm{div}^\kappa v\|_{\infty,*,\kappa} \le \alpha_\kappa. \end{aligned}$$

Denoting

$$\begin{aligned} K_\alpha^k &= \{v \in \mathcal{C}_c^k(\Omega, \mathrm{Sym}^k(\mathbf{R}^d)) \;\big|\; \|\mathrm{div}^\kappa v\|_\infty \le \alpha_\kappa \text{ for } \kappa = 0, \dots, k-1\}, \\ K_{\alpha,*}^k &= \{v \in \mathcal{C}_c^k(\Omega, \mathrm{Sym}^k(\mathbf{R}^d)^L) \;\big|\; \|\mathrm{div}^\kappa v\|_{\infty,*,\kappa} \le \alpha_\kappa \text{ for } \kappa = 0, \dots, k-1\} \end{aligned}$$

and $K_{\alpha,\infty}^k$ analogously to $K_{\alpha,*}^k$, the latter leads to $cK_{\alpha,\infty}^k \subset K_{\alpha,*}^k \subset CK_{\alpha,\infty}^k$ and, consequently,

$$c \sup_{v \in K_{\alpha,\infty}^k} \int_\Omega \sum_{l=1}^{L} u_l \mathrm{div}^k v_l \,\mathrm{d}x \le \mathrm{TGV}_\alpha^k(u) \le C \sup_{v \in K_{\alpha,\infty}^k} \int_\Omega \sum_{l=1}^{L} u_l \mathrm{div}^k v_l \,\mathrm{d}x. \tag{7}$$

From (6) now follows that $K_{\alpha,\infty}^k = (K_\alpha^k)^L$, hence

$$\sup_{v \in K_{\alpha,\infty}^k} \int_\Omega \sum_{l=1}^{L} u_l (\mathrm{div}^k v_l) \,\mathrm{d}x = \sum_{l=1}^{L} \sup_{v_l \in K_\alpha^k} \int_\Omega u_l \mathrm{div}^k v_l \,\mathrm{d}x = \sum_{l=1}^{L} \mathrm{TGV}_\alpha^k(u_l).$$

Together with (7), this gives (5). □

Corollary 1. *The kernel of* TGV_α^k *for multichannel data reads as*

$$\begin{aligned} \ker(\mathrm{TGV}_\alpha^k) &= \{u \in L^1_{\mathrm{loc}}(\Omega)^L \;\big|\; \mathrm{TGV}_\alpha^k(u) = 0\} \\ &= \Big\{ u(x) = \sum_{|\alpha| \le k-1} a_\alpha x^\alpha \ \textit{a.e. in } \Omega \;\Big|\; a_\alpha \in \mathbf{R}^L \textit{ for } \alpha \in \mathbf{N}^d,\ |\alpha| \le k-1 \Big\} \end{aligned}$$

Proof. Observe that the norm equivalence (5) gives that $\mathrm{TGV}_\alpha^k(u) = 0$ if and only if $\mathrm{TGV}_\alpha^k(u_l) = 0$ for $l = 1, \dots, L$ in the sense of (1). This is in turn equivalent to $u_l(x) = \sum_{|\alpha| \le k-1} a_{\alpha,l} x^\alpha$ a.e. in Ω for $a_{\alpha,l} \in \mathbf{R}$ and each $\alpha \in \mathbf{N}^d$ with $|\alpha| \le k-1$ and each $l = 1, \dots, L$. Arranging each $\{a_{\alpha,l}\}$ to a vector $a_\alpha \in \mathbf{R}^L$ yields the result. □

Remark 1. For the case $k = 2$, the statement of Corollary 1 reads as

$$\ker(\mathrm{TGV}_\alpha^2) = \{u(x) = Ax + b \text{ a.e. in } \Omega \;\big|\; A \in \mathbf{R}^{L\times d},\ b \in \mathbf{R}^L\}. \tag{8}$$

Remark 2. An even more general definition for TGV for multichannel data would arise from choosing, for each $k = 0, 1, \ldots$, a norm $|\cdot|_{\circ,k}$ on $\mathrm{Sym}^k(\mathbf{R}^d)^L$, setting $|\cdot|_{*,k}$ its dual norm and defining again TGV_α^k as in (4), utilizing the supremum norms associated to $|\cdot|_{*,k}$. However, this requires, besides a model how to measure the distance of colors in the respective representation in $\mathbf{R}^L$, also a model for each $\mathrm{Sym}^k(\mathbf{R}^d)^L$. In this view, setting the dual norm as in (3) seems quite natural.

Nevertheless, such a model is, for example, given by choosing $|\cdot|_\circ$ as above and defining the dual norm $|\cdot|_{*,k}$ for $\mathrm{Sym}^k(\mathbf{R}^d)^L$ as

$$|\xi|_{*,k} = \sup_{|x|_\circ \le 1,\ |y|_* \le 1} x \cdot \xi(\underbrace{y, \ldots, y}_{k \text{ times}})$$

where each $\xi_l \in \mathrm{Sym}^k(\mathbf{R}^d)$ is interpreted as a multilinear mapping. This can be interpreted as a generalization of the spectral norm for tensors of arbitrary order (which is also known as the least cross norm). It is, however, not clear how to treat these norms in concrete numerical implementations.

2.3 Solution of Variational Problems

Let us now consider variational problems for recovering multichannel images supported on the bounded Lipschitz domain $\Omega \subset \mathbf{R}^d$ which are regularized with a TGV_α^2 penalty. The general problem reads as

$$\min_{u \in L^p(\Omega,\mathbf{R}^L)} F(u) + \mathrm{TGV}_\alpha^2(u) \tag{9}$$

where $p \in (1, \infty)$, $p \le d/(d-1)$ and $F : L^p(\Omega, \mathbf{R}^L) \to (-\infty, \infty]$ is a proper, convex and lower semi-continuous functional which is also bounded from below.

For the following, we choose projection operators which project on the kernel of TGV_α^2 which is given as follows (recall (8)):

$$\{u : \Omega \to \mathbf{R}^L \;\big|\; u(x) = Ax + b \text{ a.e. for some } A \in \mathbf{R}^{L\times d}, b \in \mathbf{R}^L\}.$$

A projection operator onto the kernel is then a mapping P which satisfies

$$P : L^p(\Omega, \mathbf{R}^L) \to \ker(\mathrm{TGV}_\alpha^2) \ \text{ linear}, \qquad P|_{\ker(\mathrm{TGV}_\alpha^2)} = \mathrm{id}, \qquad P^2 = P.$$

To obtain existence of solutions for (9), the following coercivity assumption on F is made: For any sequence $\{u^n\}$ in $L^p(\Omega, \mathbf{R}^L)$ it follows that

$$\|Pu^n\|_p \to \infty \quad \text{and} \quad \{\|(\mathrm{id} - P)u^n\|_p\} \ \text{ bounded} \quad \Rightarrow \quad F(u^n) \to \infty. \tag{10}$$

Under these prerequisites, there exists at least one minimizer.

Theorem 1. *If (10) is satisfied, then (9) possesses a minimizer.*

Proof. The case where $F + \mathrm{TGV}_\alpha^2$ is constant ∞ is trivial, so let $F + \mathrm{TGV}_\alpha^2$ be proper. Let $\{u^n\}$ be a minimizing sequence in $L^p(\Omega)^L$, i.e., $\lim_{n\to\infty}(F + \mathrm{TGV}_\alpha^2)(u^n) = \inf\,(F + \mathrm{TGV}_\alpha^2)$. Then, $\{\mathrm{TGV}_\alpha^2(u^n)\}$ has to be bounded as $\{F(u^n)\}$ is bounded from below.

Now, for each $l = 1, \ldots, L$, we have that $P_l : u \mapsto (Pu)_l$ is a projection onto $\mathcal{P}_1(\Omega)$, so there exists a constant $C_1 > 0$ such that $\|u_l - P_l u\|_{d/(d-1)} \leq C_1 \mathrm{TGV}_\alpha^k(u_l)$, for each $u \in L^{d/(d-1)}(\Omega, \mathbf{R}^L)$ see (2), hence, in view of Proposition 2 it follows that

$$\begin{aligned}\|u^n - Pu^n\|_{d/(d-1)} &\leq C_2 \sum_{l=1}^{L} \|u_l^n - P_l u^n\|_{d/(d-1)} \\ &\leq C_1 C_2 \sum_{l=1}^{L} \mathrm{TGV}_\alpha^2(u_l^n) \leq c^{-1} C_1 C_2 \mathrm{TGV}_\alpha^2(u^n).\end{aligned}$$

This implies, by continuous embedding $L^{d/(d-1)}(\Omega, \mathbf{R}^L) \hookrightarrow L^p(\Omega, \mathbf{R}^L)$, that $\{\|(\mathrm{id} - P)u^n\|_p\}$ is bounded. In addition, we can exclude that $\|Pu^n\|_p$ is unbounded: If there is an unbounded subsequence, then by restricting to that subsequence without relabeling, we can achieve that $\{u^n\}$ is still a minimizing sequence with $\|Pu^n\|_p \to \infty$ and $\|(\mathrm{id} - P)u^n\|_p$ is bounded. By assumption, $F(u^n) \to \infty$ and $(F + \mathrm{TGV}_\alpha^2)(u^n) \to \infty$ which is a contradiction to $\{u^n\}$ being a minimizing sequence. Hence, $\|Pu^n\|_p$ is bounded. As we have $u^n = Pu^n + (\mathrm{id} - P)u^n$ for each n and each summand on the right-hand side gives a bounded sequence, the boundedness of $\{u^n\}$ follows. By reflexivity of $L^p(\Omega, \mathbf{R}^L)$, a subsequence of $\{u^n\}$ converges weakly to some $u^* \in L^p(\Omega, \mathbf{R}^L)$. The sum $F + \mathrm{TGV}_\alpha^2$ is convex and lower semi-continuous as its summands are, according to the assumptions as well as Proposition 1, which implies weak sequential lower semi-continuity and finally

$$(F + \mathrm{TGV}_\alpha^2)(u^*) \leq \liminf_{n\to\infty} (F + \mathrm{TGV}_\alpha^2)(u^n) = \inf_{u \in L^p(\Omega, \mathbf{R}^L)} (F + \mathrm{TGV}_\alpha^2)(u).$$

Therefore, u^* is the sought minimizer. □

3 Numerical Approximation and Minimization

3.1 Discretization as a Convex-Concave Saddle-Point Problem

For the numerical solution, we first discretize the problem (9). For simplicity, we assume that $\Omega = (0, N_1) \times (0, N_2) \subset \mathbf{R}^2$ for some positive $N_1, N_2 \in \mathbf{N}$ as it is easy to generalize to arbitrary domains and dimensions.

Following essentially the presentation in [4], we first replace Ω by the discretized grid

$$\Omega_h = \{(i, j) \mid i, j \in \mathbf{N},\ 1 \leq i \leq N_1,\ 1 \leq j \leq N_2\}.$$

The TGV^2_α functional will be discretized by finite differences where we also choose step-size 1, again for the sake of simplicity. For this purpose, we introduce the respective forward and backward operators which are, up to the factor -1, adjoint to each other:

$$(\partial_x^+ u)_{i,j} = \begin{cases} u_{i+1,j} - u_{i,j} & \text{for } 1 \leq i < N_1, \\ 0 & \text{for } i = N_1, \end{cases}$$
$$(\partial_y^+ u)_{i,j} = \begin{cases} u_{i,j+1} - u_{i,j} & \text{for } 1 \leq j < N_2, \\ 0 & \text{for } j = N_2, \end{cases}$$

as well as

$$(\partial_x^- u)_{i,j} = \begin{cases} u_{1,j} & \text{if } i = 1 \\ u_{i,j} - u_{i-1,j} & \text{for } 1 < i < N_1, \\ -u_{N_1-1,j} & \text{for } i = N_1, \end{cases}$$
$$(\partial_y^- u)_{i,j} = \begin{cases} u_{i,1} & \text{for } j = 1, \\ u_{i,j} - u_{i,j-1} & \text{for } 1 < j < N_2, \\ -u_{i,N_2-1} & \text{for } j = N_2. \end{cases}$$

Let us further introduce the appropriate vector spaces of functions, vector and tensor fields. For $L \geq 1$, define

$$U = \{u : \Omega_h \to \mathbf{R}\}^L, \quad V = \{u : \Omega_h \to \mathbf{R}^2\}^L, \quad W = \{u : \Omega_h \to \mathrm{Sym}^2(\mathbf{R}^2)\}^L.$$

We will denote $v = (v^l, \ldots, v^l) \in V$ and its components $(v^l)^1$ and $(v^l)^2$. Likewise the components of $w = (w^1, \ldots, w^L) \in W$ are $(w^l)^{11}, (w^l)^{12}$ and $(w^l)^{22}$. For convenience, we introduce

$$a, b : \Omega_h \to \mathbf{R} : \qquad \langle a, b \rangle = \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} a_{i,j} b_{i,j}$$

The spaces U, V and W will be interpreted as Hilbert spaces with the scalar products

$$u, r \in U : \ \langle u, r \rangle_U = \sum_{l=1}^{L} \langle u^l, r^l \rangle,$$
$$v, p \in V : \ \langle v, p \rangle_V = \sum_{l=1}^{L} \langle (v^l)^1, (p^l)^1 \rangle + \langle (v^l)^2, (p^l)^2 \rangle,$$
$$w, q \in W : \ \langle w, q \rangle_W = \sum_{l=1}^{L} \langle (w^l)^{11}, (q^l)^{11} \rangle + \langle (w^l)^{22}, (q^l)^{22} \rangle + 2\langle (w^l)^{12}, (q^l)^{12} \rangle.$$

The gradient, symmetrized gradient as well as the divergence operator for vector and tensor fields can then be expressed as

$$\nabla_h : U \to V, \qquad (\nabla_h u)^l = \begin{pmatrix} \partial_x^+ u^l \\ \partial_y^+ u^l \end{pmatrix},$$
$$\mathcal{E}_h : V \to W, \qquad (\mathcal{E}_h(v))^l = \begin{pmatrix} \partial_x^-(v^l)^1 & \frac{1}{2}\big(\partial_y^-(v^l)^1 + \partial_x^-(v^l)^2\big) \\ \frac{1}{2}\big(\partial_y^-(v^l)^1 + \partial_x^-(v^l)^2\big) & \partial_y^-(v^l)^2 \end{pmatrix},$$

and

$$\begin{aligned}\operatorname{div}_h : V \to U, \qquad & (\operatorname{div}_h v)^l = \partial_x^-(v^l)^1 + \partial_y^-(v^l)^2,\\ \operatorname{div}_h : W \to V, \qquad & (\operatorname{div}_h w)^l = \begin{pmatrix} \partial_x^+(w^l)^{11} + \partial_y^+(w^l)^{12} \\ \partial_x^+(w^l)^{12} + \partial_y^+(w^l)^{22} \end{pmatrix}.\end{aligned}$$

Note that with the scalar products introduced above, it holds that $(\nabla_h)^* = -\operatorname{div}_h$ as well as $(\mathcal{E}_h)^* = -\operatorname{div}_h$. In order to define a discrete version of TGV^2_α, we still need the norms

$$\begin{aligned} v \in V: \qquad & \|v\|_\infty = \max_{(i,j)\in\Omega_h} \Big(\sum_{l=1}^{L} \big((v^l)^1_{i,j}\big)^2 + \big((v^l)^2_{i,j}\big)^2\Big)^{1/2},\\ w \in W: \qquad & \|w\|_\infty = \max_{(i,j)\in\Omega_h} \Big(\sum_{l=1}^{L} \big((w^l)^{11}_{i,j}\big)^2 + \big((w^l)^{22}_{i,j}\big)^2 + 2\big((w^l)^{12}_{i,j}\big)^2\Big)^{1/2}.\end{aligned}$$

These norms correspond to discrete ∞-norms with respect to the norms according to (3) where $|\cdot|_\circ$ is the Euclidean norm on $\mathbf{R}^L$, also see Example 1. With the constraint $\operatorname{div}_h w = v$, we can now deduce a discrete version of TGV^2_α:

$$\begin{aligned}\mathrm{TGV}^2_\alpha(u) = \max \{\langle u, \operatorname{div}_h v\rangle_U \,\big|\, (v,w) \in V \times W,\ \operatorname{div}_h w = v,\\ \|w\|_\infty \le \alpha_0,\ \|v\|_\infty \le \alpha_1\}.\end{aligned}$$

Introducing indicator functionals, i.e.,

$$\mathcal{I}_K(x) = \begin{cases} 0 & \text{if } x \in K,\\ \infty & \text{else}\end{cases}$$

and observing that

$$-\mathcal{I}_{\{0\}}(\operatorname{div}_h w - v) = \min_{p\in V} \langle p, \operatorname{div}_h w - v\rangle_V,$$

the discrete functional can be rewritten to

$$\begin{aligned}\mathrm{TGV}^2_\alpha(u) = \max_{(v,w)\in V\times W} \min_{p\in V} \ & \langle u, \operatorname{div}_h v\rangle_U + \langle p, \operatorname{div}_h w - v\rangle_V\\ & -\mathcal{I}_{\{\|\cdot\|_\infty \le \alpha_0\}}(w) - \mathcal{I}_{\{\|\cdot\|_\infty \le \alpha_1\}}(v).\end{aligned}$$

One can show that the maximum and minimum can be interchanged. Moreover, the constraints are symmetric around 0, so the above can be rewritten to

$$\begin{aligned}\mathrm{TGV}^2_\alpha(u) = \min_{p\in V} \max_{(v,w)\in V\times W} \ & \langle \nabla_h u - p, v\rangle_V + \langle \mathcal{E}_h(p), w\rangle\\ & -\mathcal{I}_{\{\|\cdot\|_\infty \le \alpha_0\}}(w) - \mathcal{I}_{\{\|\cdot\|_\infty \le \alpha_1\}}(v). \quad (11)\end{aligned}$$

Next, assume that $F_h : \Omega_h \to (-\infty, \infty]$ is proper, convex and lower semi-continuous and corresponds to a discretized version of the data term F in (9).

Then, a discretization of the variational problem (9) is given by the saddle-point problem

$$\min_{(u,p)\in U\times V}\ \max_{(v,w)\in V\times W}\ \langle \nabla_h u - p,\, v\rangle_V + \langle \mathcal{E}_h(p),\, w\rangle_W + F_h(u) - \mathcal{I}_{\{\|\cdot\|_\infty\leq\alpha_0\}}(w) - \mathcal{I}_{\{\|\cdot\|_\infty\leq\alpha_1\}}(v). \tag{12}$$

We also like to consider the situation where F_h is also given by the solution of a certain maximization problem:

$$F_h(u) = \max_{\lambda\in\Lambda}\ \langle Ku,\, \lambda\rangle_\Lambda + \widetilde{F}_h(u) - G_h(\lambda) \tag{13}$$

where Λ is a finite-dimensional Hilbert space, $K : U \to \Lambda$ a linear mapping and $\widetilde{F}_h : U \to (-\infty, \infty]$, $G_h : \Lambda \to (-\infty, \infty]$ are proper, convex and lower semi-continuous functionals. In this case, we like to solve the saddle-point problem

$$\min_{(u,p)\in U\times V}\ \max_{(v,w,\lambda)\in V\times W\times\Lambda}\ \langle \nabla_h u - p,\, v\rangle_V + \langle \mathcal{E}_h(p),\, w\rangle_W + \langle Ku,\, \lambda\rangle_\Lambda + \widetilde{F}_h(u) - \mathcal{I}_{\{\|\cdot\|_\infty\leq\alpha_0\}}(w) - \mathcal{I}_{\{\|\cdot\|_\infty\leq\alpha_1\}}(v) - G_h(\lambda). \tag{14}$$

3.2 A Numerical Algorithm

For the solution of (12) and (14), any kind of numerical algorithm for the solution of convex-concave saddle point problems can be used. Here, we chose to employ the primal-dual ascent-descent method with primal extragradient according to [9]. The main reason is its applicability for a wide range of problems as we will see in Sect. 4. However, for the solution of specialized problems, other algorithms might be suited and efficient as well. Basically, every convergent method which finds a zero of a maximal monotone operator or the sum to two maximally monotone operators may work [13,18,27]. In its general form, the method finds a saddle point for the problem

$$\min_{x\in\mathcal{X}}\ \max_{y\in\mathcal{Y}}\ \langle \mathcal{K}x,\, y\rangle_\mathcal{Y} + \mathcal{F}(x) - \mathcal{G}(y) \tag{15}$$

where $\mathcal{X}$, $\mathcal{Y}$ are Hilbert spaces, $\mathcal{K} : \mathcal{X} \to \mathcal{Y}$ is a linear and continuous mapping, and $\mathcal{F} : \mathcal{X} \to (-\infty, \infty]$, $\mathcal{G} : \mathcal{Y} \to (-\infty, \infty]$ are proper, convex and lower semi-continuous functionals.

In order to state the algorithm, we need the notion of resolvent operators $(\mathrm{id} + \tau\partial\mathcal{F})^{-1}$ and $(\mathrm{id} + \sigma\partial\mathcal{G})^{-1}$ for the subgradients of $\mathcal{F}$ and $\mathcal{G}$, respectively. They can be characterized as the solutions of

$$x^* = (\mathrm{id} + \tau\partial\mathcal{F})^{-1}(\bar{x}) \quad\Leftrightarrow\quad x^* = \arg\min_{x\in\mathcal{X}} \frac{\|x-\bar{x}\|_\mathcal{X}^2}{2} + \tau\mathcal{F}(x),$$

$$y^* = (\mathrm{id} + \sigma\partial\mathcal{G})^{-1}(\bar{y}) \quad\Leftrightarrow\quad y^* = \arg\min_{y\in\mathcal{Y}} \frac{\|y-\bar{y}\|_\mathcal{Y}^2}{2} + \sigma\mathcal{G}(y)$$

where $\sigma, \tau > 0$. These resolvent operators are assumed to be computationally accessible. We will discuss some examples in Sect. 4.

The iteration procedure for the solution of (15) can be written as follows. Choose $\sigma, \tau > 0$ such that $\sigma\tau\|\mathcal{K}\|^2 < 1$. For initial values $(x^0, y^0) \in \mathcal{X} \times \mathcal{Y}$ and $\bar{x}^0 = x^0$, the iteration reads as

$$\begin{cases} y^{n+1} = (\mathrm{id} + \sigma\partial\mathcal{G})^{-1}(y^n + \sigma\mathcal{K}\bar{x}^n), \\ x^{n+1} = (\mathrm{id} + \tau\partial\mathcal{F})^{-1}(x^n - \tau\mathcal{K}^* y^{n+1}), \\ \bar{x}^{n+1} = 2x^{n+1} - x^n. \end{cases} \tag{16}$$

If $\mathcal{X}$ and $\mathcal{Y}$ are finite dimensional, this algorithm is known [9] to converge to a saddle point (x^*, y^*) of (15) provided that a saddle point exists.

We like to apply this algorithm for the solution of (12) and (14). First, let us address the problem (12) which admits the structure (15) if one chooses

$$\mathcal{X} = U \times V, \qquad \mathcal{Y} = V \times W, \qquad \mathcal{K} = \begin{bmatrix} \nabla_h & -\mathrm{id} \\ 0 & \mathcal{E}_h \end{bmatrix} \quad \Rightarrow \quad \mathcal{K}^* = \begin{bmatrix} -\mathrm{div}_h & 0 \\ -\mathrm{id} & -\mathrm{div}_h \end{bmatrix}$$

as well as

$$\begin{aligned} \mathcal{F}(x) &= \mathcal{F}(u, p) = F_h(u), \\ \mathcal{G}(y) &= \mathcal{G}(v, w) = \mathcal{I}_{\{\|\cdot\|_\infty \le \alpha_1\}}(v) + \mathcal{I}_{\{\|\cdot\|_\infty \le \alpha_0\}}(w). \end{aligned}$$

As the functionals $\mathcal{F}$ and $\mathcal{G}$ are the sum of functionals which only depend on one component of x and y, respectively, the resolvent operators decouple meaning that they can be performed componentwise. For $\mathcal{G}$, they correspond to projection operators on the respective constraint sets. They can be seen to correspond to

$$v^* = \arg\min_{\|v\|_\infty \le \alpha_1} \frac{\|v - \bar{v}\|_V^2}{2} \quad \Leftrightarrow \quad v^* = \mathcal{P}_{\alpha_1}(\bar{v}) = \frac{\bar{v}}{\max\left(1, \frac{|\bar{v}|}{\alpha_1}\right)},$$

$$w^* = \arg\min_{\|w\|_\infty \le \alpha_0} \frac{\|w - \bar{w}\|_V^2}{2} \quad \Leftrightarrow \quad w^* = \mathcal{P}_{\alpha_0}(\bar{w}) = \frac{\bar{w}}{\max\left(1, \frac{|\bar{w}|}{\alpha_0}\right)}$$

where the operations on the right-hand side have to be interpreted in the pointwise sense with $|\bar{v}|$ and $|\bar{w}|$ according to

$$\bar{v} \in V: \qquad |\bar{v}|_{i,j} = \Big(\sum_{l=1}^{L} \big((\bar{v}^l)^1_{i,j}\big)^2 + \big((\bar{v}^l)^2_{i,j}\big)^2\Big)^{1/2},$$

$$\bar{w} \in W: \qquad |\bar{w}|_{i,j} = \Big(\sum_{l=1}^{L} \big((\bar{w}^l)^{11}_{i,j}\big)^2 + \big((\bar{w}^l)^{22}_{i,j}\big)^2 + 2\big((\bar{w}^l)^{12}_{i,j}\big)^2\Big)^{1/2}.$$

For $\mathcal{F}$, the componentwise resolvents just correspond to $(\mathrm{id} + \sigma\partial F_h)^{-1}$ and, as the functional is independent of p, to the identity on V, respectively.

Finally, in order to choose the step sizes σ and τ, we need an estimate for the norm of $\mathcal{K}$. One can see that $\|\nabla_h\|^2 < 8$ and $\|\mathcal{E}_h\|^2 < 8$ which leads, after some computations, to the estimate

$$\|\mathcal{K}\|^2 < \frac{17 + \sqrt{33}}{2} < 12.$$

Algorithm 1 Solve $\min_{u\in U}\ F_h(u) + \mathrm{TGV}^2_\alpha(u)$

1. Choose $\sigma > 0$, $\tau > 0$ such that $\sigma\tau\frac{1}{2}(17+\sqrt{33}) \leq 1$.
2. Choose $(u^0, p^0) \in U \times V$, $(v^0, w^0) \in V \times W$ and set $\bar{u}^0 = u^0$, $\bar{p}^0 = p^0$.
3. For $n = 0, 1, 2, \ldots$ iterate according to

$$\begin{cases} v^{n+1} = \mathcal{P}_{\alpha_1}\big(v^n + \sigma(\nabla_h \bar{u}^n - \bar{p}^n)\big), \\ w^{n+1} = \mathcal{P}_{\alpha_0}\big(w^n + \sigma\mathcal{E}_h(\bar{p}^n)\big), \\ u^{n+1} = (\mathrm{id} + \tau\partial F_h)^{-1}(u^n + \tau\mathrm{div}_h v^{n+1}), \\ p^{n+1} = p^n + \tau(v^{n+1} + \mathrm{div}_h w^{n+1}), \\ \bar{u}^{n+1} = 2u^{n+1} - u^n, \\ \bar{p}^{n+1} = 2p^{n+1} - p^n. \end{cases}$$

4. Return u^N for some large N.

Hence, the primal-dual method for the saddle-point problem (12) reads as follows.

Note that the problem of choosing N, i.e., finding an appropriate stopping criterion, remains. However, as our major goal is to demonstrate the applicability and efficiency of algorithms suitable for the minimization of TGV^2_α, we do not discuss this issue here. Let us nevertheless remark that it is possible to compute estimates for the primal-dual gap for the underlying saddle-point problem which allows to estimate the distance of the current iterate to the minimizer in terms of the functional values. These estimates could be used to implement a stopping criterion, see, for instance [3].

It can be seen in Algorithm 1 that the resolvent $(\mathrm{id} + \tau\partial F_h)^{-1}$ is needed in order to perform the computational procedure. In some cases, this resolvent is not computationally accessible or expensive to compute. It might, however, be possible to write F_h in terms of (13) where the resolvents $(\mathrm{id} + \tau\partial\widetilde{F}_h)^{-1}$ and $(\mathrm{id}+\sigma\partial G_h)^{-1}$ are easy to compute. In such a case, the algorithm can be modified in order to accommodate for this situation: Indeed, the associated saddle-point problem (14) can be represented by (15) if one chooses

$$\mathcal{X} = U \times V, \quad \mathcal{Y} = V \times W \times \Lambda, \quad \mathcal{K} = \begin{bmatrix} \nabla_h & -\mathrm{id} \\ 0 & \mathcal{E}_h \\ K & 0 \end{bmatrix} \ \Rightarrow\ \mathcal{K}^* = \begin{bmatrix} -\mathrm{div}_h & 0 & K^* \\ -\mathrm{id} & -\mathrm{div}_h & 0 \end{bmatrix}$$

as well as

$$\begin{aligned} \mathcal{F}(x) &= \mathcal{F}(u,p) = \widetilde{F}_h(u), \\ \mathcal{G}(y) &= \mathcal{G}(v,w,\lambda) = \mathcal{I}_{\{\|\cdot\|_\infty \leq \alpha_1\}}(v) + \mathcal{I}_{\{\|\cdot\|_\infty \leq \alpha_0\}}(w) + G_h(\lambda). \end{aligned}$$

As in the previous case, all the resolvents decouple and each component of x and y can be updated individually. However, one has be more careful when choosing σ and τ as the norm of $\mathcal{K}$ can only be guaranteed to obey

$$\|\mathcal{K}\|^2 < \frac{\sqrt{(\|K\|^2-1)^2 + 32} + \|K\|^2 + 17}{2} < \|K\|^2 + 12.$$

The primal-dual method then corresponds to the following.

Algorithm 2 Solve $\min_{u\in U}\ F_h(u) + \mathrm{TGV}^2_\alpha(u)$
with $F_h(u) = \max_{\lambda\in\Lambda}\ \langle Ku,\, \lambda\rangle_\Lambda + \widetilde{F}_h(u) - G_h(\lambda)$

1. Choose $\sigma > 0$, $\tau > 0$ such that $\sigma\tau\frac{1}{2}(\sqrt{(\|K\|^2-1)^2+32} + \|K\|^2 + 17) \le 1$.
2. Choose $(u^0, p^0) \in U \times V$, $(v^0, w^0, \lambda^0) \in V \times W \times \Lambda$.
   Set $\bar{u}^0 = u^0$, $\bar{p}^0 = p^0$.
3. For $n = 0, 1, 2, \ldots$ iterate according to

$$\begin{cases} v^{n+1} = \mathcal{P}_{\alpha_1}\big(v^n + \sigma(\nabla_h \bar{u}^n - \bar{p}^n)\big), \\ w^{n+1} = \mathcal{P}_{\alpha_0}\big(w^n + \sigma\mathcal{E}_h(\bar{p}^n)\big), \\ \lambda^{n+1} = (\mathrm{id} + \sigma\partial G_h)^{-1}(\lambda^n + \sigma K\bar{u}^n) \\ u^{n+1} = (\mathrm{id} + \tau\partial\widetilde{F}_h)^{-1}\big(u^n + \tau(\mathrm{div}_h v^{n+1} - K^*\lambda^{n+1})\big), \\ p^{n+1} = p^n + \tau(v^{n+1} + \mathrm{div}_h w^{n+1}), \\ \bar{u}^{n+1} = 2u^{n+1} - u^n, \\ \bar{p}^{n+1} = 2p^{n+1} - p^n. \end{cases}$$

4. Return u^N for some large N.

Again, the procedure converges to a saddle-point, so u^N for N large enough is close to a solution of the original problem.

4 Application to Mathematical Imaging Problems

Now, we aim at applying the total generalized variation model to some well-known variational problems. One the one hand, we show how existence in the continuous setting can be ensured using the results of Sect. 2. On the other hand, it is also discussed how the algorithms in Sect. 3 can be realized and how they perform in numerical experiments.

4.1 Denoising

We first look at the TGV^2_α-regularized multichannel denoising problem for a noisy image $f \in L^q(\Omega, \mathbf{R}^L)$ where $q \in [1, \infty)$. Let the norm in $L^q(\Omega, \mathbf{R}^L)$ be based on the vector norm $|\cdot|_\circ$. Then, the variational denoising problem with L^q-data term, is to solve

$$\min_{u\in L^q(\Omega,\mathbf{R}^L)} F(u) + \mathrm{TGV}^2_\alpha(u), \qquad F(u) = \frac{1}{q}\int_\Omega |u-f|^q_\circ \,\mathrm{d}x = \frac{\|u-f\|_q^q}{q}. \quad (17)$$

We like to verify existence of a minimizer in $L^p(\Omega, \mathbf{R}^L)$ for some $p \in (1, \infty)$ with $p \le d/(d-1)$. For this purpose, observe that it is easy to see that F is non-negative, proper, convex and lower semi-continuous on $L^p(\Omega, \mathbf{R}^L)$, the latter

with the help of Fatou's lemma. To establish property (10), set $r = \min(p, q)$ and choose $P : L^r(\Omega, \mathbf{R}^L) \to \ker(\mathrm{TGV}^2_\alpha)$ as a linear and continuous projection. If, for a sequence $\{u^n\}$ in $L^p(\Omega, \mathbf{R}^L)$ it holds that $\|Pu^n\|_p \to \infty$, then also $\|Pu^n\|_q \to \infty$ since all norms are equivalent on the finite-dimensional space $\ker(\mathrm{TGV}^2_\alpha)$. Consequently, as P is continuous on $L^q(\Omega, \mathbf{R}^L)$,

$$\|u^n - f\|_q \geq \|u^n\|_q - \|f\|_q \geq c\|Pu^n\|_q - M$$

for some constants $c > 0$ and $M > 0$. Hence, $F(u^n) \to \infty$ and (10) is satisfied. By Theorem 1, there exists a minimizer.

Let us now discretize (17) according to Sect. 3. For this purpose, we choose F_h for some data $f \in U$ according to

$$F_h(u) = \frac{1}{q} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} |u_{i,j} - f_{i,j}|_\circ^q$$

which is in accordance with a discretization step-size of 1. We like to use Algorithm 1 which needs the resolvent $(\mathrm{id} + \tau \partial F_h)^{-1}$. For $|\cdot|_\circ = |\cdot|$ the Euclidean norm on $\mathbf{R}^L$ and $q \in \{1, 2\}$, this operator can be computed:

$$u^* = (\mathrm{id} + \tau \partial F_h)^{-1}(\bar{u}) \quad \Leftrightarrow \quad u^* = \begin{cases} \dfrac{\bar{u} + \tau f}{1 + \tau} & \text{if } q = 2, \\ f + \mathcal{S}_\tau(\bar{u} - f) & \text{if } q = 1. \end{cases}$$

where $\mathcal{S}_\tau$ is the pointwise shrinkage operator

$$\mathcal{S}_\tau(u) = \frac{u}{|u|} \max(0, |u| - \tau)$$

where we agree to set $u/|u| = 0$ where $u = 0$. This leads to the following iteration for $\{u^n\}$ in Algorithm 1:

$$u^{n+1} = \begin{cases} \dfrac{u^n + \tau(\mathrm{div}_h v^{n+1} + f)}{1 + \tau} & \text{if } q = 2, \\ f + \mathcal{S}_\tau(u^n + \tau \mathrm{div}_h v^n - f) & \text{if } q = 1. \end{cases}$$

The resulting algorithm was implemented in Python [25] using Scientific Tools for Python (SciPy) [34] and graphics-processing unit (GPU) acceleration based on NIVIDA's CUDA™ Toolkit [22] via the Python interface PyCUDA [16]. Computations where performed on a AMD Phenom™ 9950 Quad-Core Processor with a NVIDIA GeForce GTX 280 GPU with 1 Gigabyte of memory. The outcome of the TGV^2-based denoising procedure, for the L^2-norm and L^1-norm as well as a comparison to the standard TV-based counterparts, are depicted in the Figs. 2 and 3, respectively. In order to compare the models, the parameters were chosen to give the best peak signal-to-noise ratio (PSNR). The actual values as well as the image size, noise level, iteration count and computation time can be found in the respective captions. One can observe that the multichannel TGV^2 image model is able to recover smooth regions as well as discontinuities

at object boundaries. In particular, artifacts which are typical for TV-based denoising, do not appear when TGV^2 is used. The improved image quality is also reflected by a slightly higher PSNR value. However, since this distance measure is essentially based on pointwise comparison and is not incorporating neighborhood information, more accurate recovery of smooth regions does not lead to an significantly higher PSNR, although the differences can noticeably be visually perceived. Of course, usage of the TGV^2 image model come with higher computational effort. In the case of denoising, TV and TGV^2 roughly need the same number of iterations such that TGV^2 needs about 2 to 3 times more computation time. The absolute computation time is, however, still quite low thanks to the parallelization provided by the GPU.

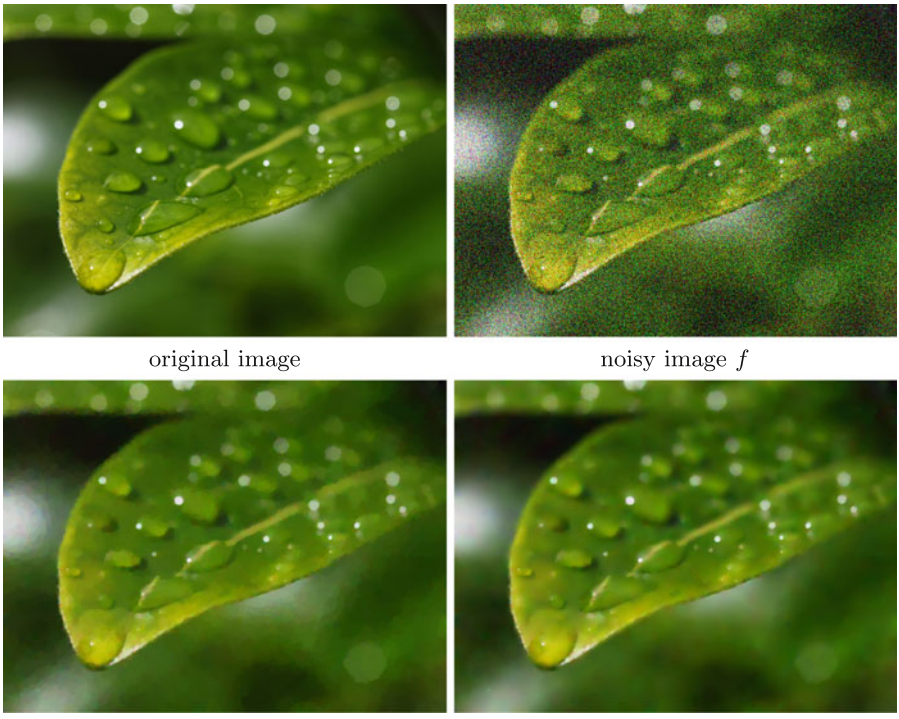


Fig. 2. Example for variational denoising according to (17) with L^2 data fitting term and TV/TGV^2 image model. Top: The original image [31] (left, 640×480 pixels, RGB) and a noise-perturbed version (right, additive Gaussian noise, standard deviation $\sigma = 0.2$). Bottom: Result of TV-based denoising (left, PSNR=31.91 dB, 500 iterations, computation time: 0.48 s), and TGV^2-based denoising (right, PSNR=32.29 dB, 500 iterations, computation time: 1.29 s). Images licenced under CreativeCommons-by-sa-2.0 (http://creativecommons.org/licenses/by-sa/2.0/).

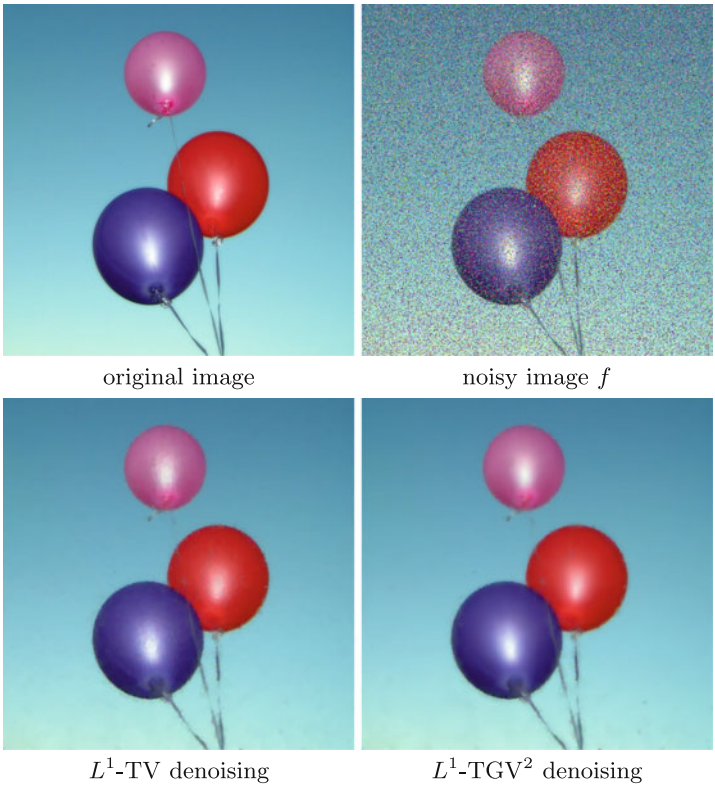


Fig. 3. Example for variational denoising with L^1 data fitting term and TV/TGV^2 image model. Top: The original image [29] (left, 512×512 pixels, RGB) and a noise-perturbed version (right, 33.3 % of the pixels replaced by random values). Bottom: Result of TV-based denoising (left, PSNR=33.23 dB, 1000 iterations, computation time: 0.98 s), and TGV^2-based denoising (right, PSNR=33.77 dB, 1000 iterations, computation time: 2.46 s).

Additionally, the proposed primal-dual algorithm admits the convergence behavior which is typical for first-order methods. In terms of the discrete energy which arises by maximizing the saddle-point functional in (12), i.e.,

$$J(u,p) = \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} \frac{1}{q}|u_{i,j} - f_{i,j}|^q + \alpha_1 |\nabla_h u - p_{i,j}|_{i,j} + \alpha_0 |\mathcal{E}_h(p)|_{i,j},$$

one can observe a quick decrease within the first 100 iterations which then becomes slower (see Fig. 4). Note that the method is non-monotone which can also seen in the behavior of J. However, in the denoising examples, this was only noticeable in the first few iterations.

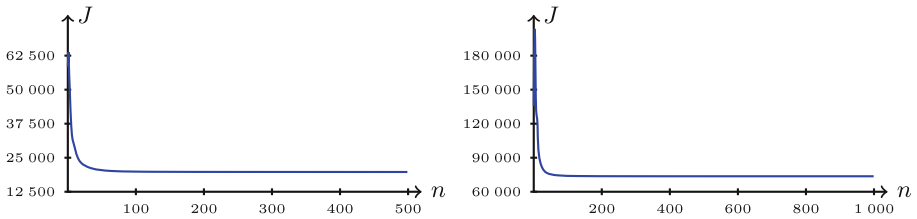


Fig. 4. Convergence behavior for the L^2-TGV^2 (left) and the L^1-TGV^2 (right) denoising examples of Figs. 2 and 3, respectively.

Remark 3. In order to preserve more details, one approach is to introduce a spatially dependent parameter which serves as a weight for the L^2-discrepancy, i.e.,

$$F(u) = \frac{1}{2}\int_\Omega \lambda|u-f|^2 \,\mathrm{d}x, \qquad \lambda \in L^\infty(\Omega).$$

An appropriate choice of λ then leads, in conjunction with TV-regularization, to denoised images which indeed admit more details [11]. Recently, this framework has been extended to TGV^2_α-regularization which yields further improvements, see [2] for details.

4.2 Deblurring

Next, let us discuss the deblurring of a multichannel image. We model this problem as the general linear inverse problem of finding a solution to

$$Ku = f$$

where, for some $p \in (1,\infty)$, $p \le d/(d-1)$ the operator $K : L^p(\Omega, \mathbf{R}^L) \to H$ is a linear and continuous mapping into a Hilbert space H in which the data $f \in H$ is also given. We like to regularize this generally ill-posed problem with TGV^2_α and solve the associated Tikhonov minimization problem, i.e.,

$$\min_{u\in L^p(\Omega,\mathbf{R}^L)} F(u) + \mathrm{TGV}^2_\alpha(u), \qquad F(u) = \frac{\|Ku-f\|_H^2}{2}. \tag{18}$$

This problem turns out to have a solution as soon as K is injective on $\ker(\mathrm{TGV}^2_\alpha)$: Let us assume that

$$Ku = 0 \quad \text{for some} \quad u \in \ker(\mathrm{TGV}^2_\alpha) \qquad \Rightarrow \qquad u = 0,$$

which is, as $\ker(\mathrm{TGV}^2_\alpha)$ is finite-dimensional, equivalent to the existence of a $c > 0$ such that $\|Ku\|_H \ge c\|u\|_p$ for each $u \in \ker(\mathrm{TGV}^2_\alpha)$. For an arbitrary projection operator $P : L^p(\Omega, \mathbf{R}^L) \to \ker(\mathrm{TGV}^2_\alpha)$ and a sequence $\{u^n\}$ in $L^p(\Omega, \mathbf{R}^L)$ such that $\|Pu^n\|_p \to \infty$ and $\{\|(\mathrm{id}-P)u^n\|_p\}$ is bounded, we then have

$$\|Ku-f\|_H \ge \|KPu^n\|_H - \|K(\mathrm{id}-P)u^n\|_H - \|f\|_H \ge c\|Pu^n\|_H - M$$

for some $M > 0$ and the above $c > 0$. Hence $F(u^n) \to \infty$ as $n \to \infty$. Together with the observation that F is non-negative, proper, convex, continuous (and hence, lower semi-continuous), this implies by Theorem 1 that there is always a solution to (18).

In the concrete case of deblurring, we model the forward operator K as the convolution with a kernel $k \in L^\infty(\mathbf{R}^d)$ with compact support Ω_0. The data space is $H = L^2(\Omega', \mathbf{R}^L)$ where Ω' is a non-empty open set which models the region on which the blurred image is measured. We assume that only data on Ω is convolved which is satisfied if

$$\Omega' - \Omega_0 = \{x - y \;\big|\; x \in \Omega',\ y \in \Omega_0\} \subset \Omega.$$

Furthermore, let $k \geq 0$ almost everywhere and such that $\int_{\Omega_0} k \,\mathrm{d}x = 1$. The operator K is then given by

$$Ku = u * k, \qquad (u * k)(x) = \int_{\Omega_0} u(x - y)k(y) \,\mathrm{d}y \quad \text{for} \quad x \in \Omega'. \tag{19}$$

Note that $\|u * k\|_\infty \leq \|u\|_1 \|k\|_\infty$, so K is in particular continuous between $L^p(\Omega, \mathbf{R}^L)$ and H. It remains to verify that K is injective on $\ker(\mathrm{TGV}^2_\alpha)$. For this purpose, let $u(x) = Ax + b$ for $A \in \mathbf{R}^{L \times d}$, $b \in \mathbf{R}^L$ such that $Ku = 0$. Then, for $m \in \mathbf{R}^d$, $m_i = \int_{\Omega_0} y_i \,\mathrm{d}y$, we have

$$\int_{\Omega_0} \big(A(x - y) + b\big)k(y) \,\mathrm{d}y = Ax + (b - Am) \qquad \text{for all} \qquad x \in \Omega'$$

meaning that Ku is an affine linear function on Ω'. As Ω' contains a non-empty open set, $Ku = 0$ is only possible if $A = 0$ and $b - Am = b = 0$, implying that $u = 0$. This shows the injectivity, hence (18) can always be solved for the blurring operator according to (19).

Let us now discuss the numerical realization of the solution of (18) in the framework of Sect. 3. Regarding the general problem, we assume that K can be discretized to a $K_h : U \to \Lambda$ where the Hilbert space Λ corresponds to the discretized data space H. The discrepancy functional for discrete data $f \in \Lambda$ then reads as

$$F_h(u) = \frac{\|K_h u - f\|^2_\Lambda}{2}.$$

To describe Λ and K_h for the blurring operator (19), let $k \in \mathbf{R}^{(2M+1)\times(2M+1)}$ a discrete convolution kernel which is indexed through $-M, \ldots, M$. The data can then be measured on the set

$$\Omega'_h = \{(i, j) \;\big|\; i, j \in \mathbf{N}, M + 1 \leq i \leq N_1 - M,\ M + 1 \leq j \leq N_2 - M\}.$$

Consequently, we let $\Lambda = \{\Omega_h \to \mathbf{R}\}^L$ such that the discrete convolution operator becomes

$$K_h u = u * k, \qquad (u * k)^l_{i,j} = \sum_{i'=-M}^{M} \sum_{j'=-M}^{M} u^l_{i-i',j-j'} k_{i',j'} \quad \text{for} \quad (i, j) \in \Omega'_h.$$

One can easily see that if k is normalized, i.e., $k \geq 0$ componentwise and $\sum_{i=-M}^{M} \sum_{j=-M}^{M} k_{i,j} = 1$, then $\|K_h\| \leq 1$.

There is now the choice whether to take Algorithm 1 or 2 for the numerical solution. Let us shortly discuss Algorithm 1. Here, one has again to evaluate the resolvent operator which corresponds to

$$u^* = (\mathrm{id} + \tau \partial F_h)^{-1}(\bar{u}) \qquad \Leftrightarrow \qquad u^* = (\mathrm{id} + \tau K_h^* K_h)^{-1}(\bar{u} + \tau K_h^* f).$$

Hence, the iteration step for u^{n+1} reads as

$$u^{n+1} = (\mathrm{id} + \tau K_h^* K_h)^{-1}\big(u^n + \tau(\mathrm{div}_h v^{n+1} + K_h^* f)\big)$$

which involves the solution of a linear equation. As this might be quite costly, in particular if it has to be done iteratively and the evaluation of K_h or K_h^* is expensive, we also discuss Algorithm 2 which, as it turns out, does not involve such an inversion step.

It bases on the observation that F_h can be written as

$$F_h(u) = \max_{\lambda \in \Lambda} \langle K_h u, \lambda \rangle_\Lambda - \Big(\frac{\|\lambda\|_\Lambda^2}{2} + \langle f, \lambda \rangle_\Lambda \Big)$$

which is of the form (13) with $\widetilde{F}_h(u) = 0$ and $G_h(\lambda) = \frac{1}{2}\|\lambda\|_\Lambda^2 + \langle f, \lambda \rangle_\Lambda$. The resolvent associated with the subgradient of $\widetilde{F}_h$ again turns out to be the identity while

$$\lambda^* = (\mathrm{id} + \sigma \partial G_h)^{-1}(\bar{\lambda}) \qquad \Leftrightarrow \qquad \lambda^* = \frac{\bar{\lambda} - \sigma f}{1 + \sigma}.$$

Hence, the iteration steps for λ and u in Algorithm 2 read as

$$\begin{cases} \lambda^{n+1} = \dfrac{\lambda^n + \sigma(K_h \bar{u}^n - f)}{1 + \sigma}, \\ u^{n+1} = u^n + \tau(\mathrm{div}_h v^{n+1} - K_h^* \lambda^{n+1}). \end{cases}$$

This variant provides an alternative in which only one evaluation of K_h and K_h^* is necessary in each iteration step. However, one needs to have an estimate for $\|K_h\|$ in order to choose step-sizes σ and τ such that convergence can be ensured. In the case of the discrete convolution operator introduced above, one obtains again the estimate $\|K_h\| \leq 1$ for normalized kernels.

As in Subsect. 4.1, this method has again been implemented in Python using PyCUDA. Computations have been performed to deconvolve a blurred image which has additionally been contaminated by noise. The same configuration as for the denoising experiments has been used, the outcome as well as the details are shown in Fig. 5. Again, one can observe the ability of the TGV^2-model to nicely resolve smooth regions as well as sharp edges. One can also observe that the computation time is only slightly higher compared to TV-based deblurring. This is due to the fact that most of the time is spent in evaluating the forward and adjoint operator K_h and K_h^*, respectively.

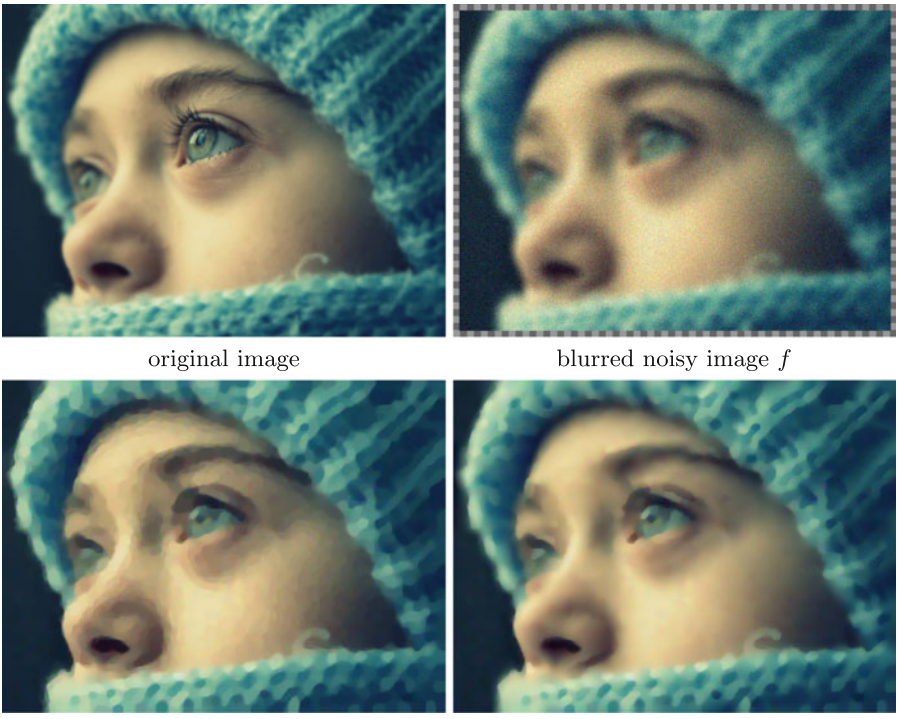


Fig. 5. Example for variational deblurring according to (18) with L^2 data fitting term and TV/TGV^2 image model. Top: The original image [15] (left, 512×384 pixels, RGB) and a blurred, noise-perturbed version (right, out-of-focus kernel with 15 pixels diameter, additive Gaussian noise, standard deviation $\sigma = 0.05$). Bottom: Result of TV-based deblurring (left, PSNR=31.45 dB, 1000 iterations, computation time: 45.63 s), and TGV^2-based deblurring (right, PSNR=32.05 dB, 1000 iterations, computation time: 46.72 s).

4.3 Zooming

The following deals with the problem of recovering an image from a projected version which can be interpreted as a zooming problem. In order to describe the setting, let $p \in (1, \infty)$ with $p \leq d/(d-1)$ and $Z \subset L^p(\Omega, \mathbf{R}^L)$ be a closed subspace which is modelling images at a low-resolution in which a low-resolution image $f \in Z$ is given. Furthermore, let the operator $P_Z : L^p(\Omega, \mathbf{R}^L) \to Z$ be a continuous projection onto Z which is modelling the way the resolution is reduced. The corresponding zooming problem then reads as

$$\min_{u \in L^p(\Omega, \mathbf{R}^L)} F(u) + \mathrm{TGV}^2_\alpha(u), \qquad F(u) = \mathcal{I}_{\{0\}}(P_Z u - f). \tag{20}$$

Let us discuss existence of solutions. As P_Z is continuous and $\mathrm{rg}(P_Z) = Z$, it is obvious that the indicator functional F is non-negative, proper, convex and

lower semi-continuous. To establish the desired coercivity property, we need the assumption

$$\ker(P_Z) \cap \ker(\mathrm{TGV}_\alpha^2) = \{0\}.$$

Note that this implies that P_Z is injective on $\ker(\mathrm{TGV}_\alpha^2)$ as $P_Z u = 0$ and $u \in \ker(\mathrm{TGV}_\alpha^2)$ implies $u = 0$. Now, if $P : L^p(\Omega, \mathbf{R}^L) \to \ker(\mathrm{TGV}_\alpha^2)$ is a linear, continuous projection, then there is a constant $c > 0$ such that $\|P_Z P u\|_p \geq c\|Pu\|_p$ for each $u \in L^p(\Omega, \mathbf{R}^L)$. Thus, for each sequence $\{u^n\}$ in $L^p(\Omega, \mathbf{R}^L)$ such that $\|Pu^n\|_p \to \infty$ and $\{\|(\mathrm{id} - P)u^n\|_p\}$ is bounded, it is impossible that $P_Z u^n - f = 0$ for each n: If this is the case, then

$$P_Z P u^n + P_Z(\mathrm{id} - P)u^n = P_Z u^n = f$$

and consequently,

$$c\|Pu^n\|_p \leq \|P_Z P u^n\|_p = \|P_Z(P - \mathrm{id})u^n + f\|_p \leq C + \|f\|_p$$

which implies that $\|Pu^n\|_p$ is bounded, a contradiction. Hence, $F(u^n) \to \infty$ as this argumentation also applies to each subsequence. This establishes (10) and, by Theorem 1, existence of a minimizer.

Example 2. Let $\Omega = (0, M_1) \times (0, M_2)$ with $M_1, M_2 \geq 2$. Denote by $Q_{i,j} = (i-1, i) \times (j-1, j)$ and set

$$Z = \Big\{ \sum_{i=1}^{M_1} \sum_{j=1}^{M_2} c_{i,j} \chi_{Q_{i,j}} \;\Big|\; c_{i,j} \in \mathbf{R}^L \Big\}$$

which is modelling $N \times M$ pixel images. A projection onto Z is then given by

$$P_Z u = \sum_{i=1}^{M_1} \sum_{j=1}^{M_2} \Big(\int_{Q_{i,j}} u \, \mathrm{d}x \Big) \chi_{Q_{i,j}}.$$

A $u \in \ker(\mathrm{TGV}_\alpha^2)$ can be expressed by $u(x) = a_1 x_1 + a_2 x_2 + b$ where $a_1, a_2, b \in \mathbf{R}^L$. We then see that

$$c_{i,j} = \int_{Q_{i,j}} u \, \mathrm{d}x = \frac{2i-1}{2} a_1 + \frac{2j-1}{2} a_2 + b$$

so $a_1 = c_{2,1} - c_{1,1}$, $a_2 = c_{1,2} - c_{1,1}$ and $b = 2c_{1,1} - \frac{1}{2}(c_{1,2} + c_{2,1})$. Hence, if $P_Z u = 0$, then $c_{1,1} = c_{1,2} = c_{2,2} = 0$ and consequently, $u = 0$. This shows $\ker(P_Z) \cap \ker(\mathrm{TGV}_\alpha^2) = \{0\}$, thus a solution for the zooming problem (20) with averaging over squares exists.

Example 3. Let $\Omega = (0, \pi)^2$ and $M_1, M_2 \geq 2$ and denote by

$$z_{i,j}(x) = \zeta_i(x_1)\zeta_j(x_2), \qquad \zeta_i(x) = \begin{cases} \sqrt{\dfrac{1}{\pi}} & \text{if } i = 0, \\ \sqrt{\dfrac{2}{\pi}} \cos(ix) & \text{if } i > 0, \end{cases}$$

which is corresponding to the cosine orthonormal basis of $L^2(\Omega)$. The space Z and a projection P_Z is then given by

$$Z = \Big\{ \sum_{i=0}^{M_1-1} \sum_{j=0}^{M_2-1} c_{i,j} z_{i,j} \;\Big|\; c_{i,j} \in \mathbf{R}^L \Big\}, \qquad P_Z u = \sum_{i=0}^{M_1-1} \sum_{j=0}^{M_2-1} \Big(\int_\Omega z_{i,j} u \,\mathrm{d}x \Big) z_{i,j}.$$

For a $u \in \ker(\mathrm{TGV}_\alpha^2)$, i.e., $u(x) = a_1 x_1 + a_2 x_2 + b$, $a_1, a_2, b \in \mathbf{R}^L$ we see that

$$c_{0,0} = \frac{\pi^2}{2}(a_1 + a_2) + \pi b, \qquad c_{1,0} = -4a_1, \qquad c_{0,1} = -4a_2$$

which implies that if $P_Z u = 0$, then also $u = 0$. Again, we thus have $\ker(P_Z) \cap \ker(\mathrm{TGV}_\alpha^2) = \{0\}$ and consequently, a minimizer for the zooming problem (20) with cosine low-pass filter exists.

For a numerical realization, we have to discretize the space Z. In case of Example 3, a good choice is the corresponding two-dimensional discrete cosine basis. For a discrete low-pass image $f \in (\mathbf{R}^{M_1 \times M_2})^L$, $2 \le M_1 \le N_1$, $2 \le M_2 \le N_2$, and with DCT denoting the associated parallel discrete cosine transform operator, the discrete functional F_h reads as

$$F_h(u) = \begin{cases} 0 & \text{if } \mathrm{DCT}(u)_{i,j} = f_{i,j} \text{ for } 0 \le i \le M_1 - 1,\ 0 \le j \le M_2 - 1, \\ \infty & \text{else.} \end{cases}$$

Consequently, since DCT is an orthonormal mapping, the resolvent reads as

$$(\mathrm{id} + \tau \partial F_h)^{-1}(u) = \mathrm{DCT}^{-1}(\widetilde{c}),$$
$$\widetilde{c}_{i,j} = \begin{cases} f_{i,j} & \text{if } 0 \le i \le M_1 - 1,\ 0 \le j \le M_2 - 1, \\ \mathrm{DCT}(u)_{i,j} & \text{else.} \end{cases}$$

The iteration step which computes u^{n+1} in Algorithm 1 then reads as

$$\begin{cases} c^{n+1} = \mathrm{DCT}(u^n + \tau \mathrm{div}_h v^{n+1}), \\ \widetilde{c}_{i,j}^{n+1} = \begin{cases} f_{i,j} \text{ for } 0 \le i \le M_1 - 1,\ 0 \le j \le M_2 - 1, \\ c_{i,j}^{n+1} \text{ else,} \end{cases} \\ u^{n+1} = \mathrm{DCT}^{-1}(\widetilde{c}^{n+1}). \end{cases}$$

This procedure was again implemented in Python/PyCUDA and tested on the same machine as for the experiments in the previous subsections. The outcome of a zooming experiment with the factor 8 can be seen in Fig. 6. Compared to TV-based zooming, it is interesting to observe that neither models lead to pronounced staircasing artifacts. However, one sees that the TGV^2 model nevertheless leads to a solution which appears less blocky.

4.4 Dequantization

Although images are often modelled as functions which admit continuous values, their digital representation is often quantized to a finite number of bins. The most

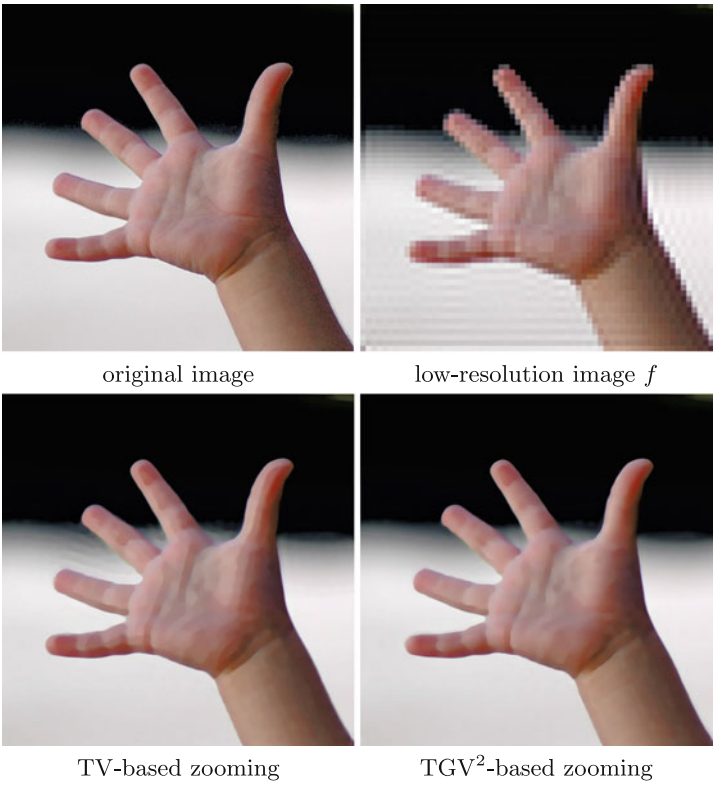


Fig. 6. Example for variational zooming with the TV/TGV^2 image model according to 4.3. Top: The original image [17] (left, 512×512 pixels, RGB) and a low-resolution representation (right, 64×64 pixels, DCT low-pass coefficients). Bottom: Result of TV-based zooming (left, PSNR=36.60 dB, 2000 iterations, computation time: 4.44 s), and TGV^2-based zooming (right, PSNR=37.30 dB, 2000 iterations, computation time: 7.46 s). Images licenced under CreativeCommons-by-sa-2.0 (http://creativecommons.org/licenses/by-sa/2.0/).

common case is the 256-level representation which corresponds to 8 bits per pixel and color channel. In the case where significantly less bins are available, the image representing the pixelwise centers of the respective bins appears blocky and unnatural. Hence, one is interested in restoring a more natural image from a quantized image. We assume that each bin has the form

$$[a, b] = \{c \in \mathbf{R}^L \mid a_l \leq c_l \leq b_l,\ l = 1, \ldots, L\} \qquad \text{for some } a, b \in \mathbf{R}^L.$$

The given data can then be represented by the respective lower and upper bound functions $f_{\text{lower}}, f_{\text{upper}} \in L^p(\Omega, \mathbf{R}^L)$ for some $p \in (1, \infty)$, $p \leq d/(d-1)$ which have to satisfy $f_{\text{lower}} \leq f_{\text{upper}}$ (componentwise) almost everywhere in Ω. The feasible images are then given by $u \in L^p(\Omega, \mathbf{R}^L)$ such that $f_{\text{lower}} \leq u \leq f_{\text{upper}}$ almost everywhere in Ω. As the standard dequantization $\widetilde{f} = \frac{1}{2}(f_{\text{lower}} + f_{\text{upper}})$ is

the image which is most probable if for almost every pixel $x \in \Omega$, the values are uniformly distributed in $[f_{\text{lower}}(x), f_{\text{upper}}(x)]$, we like to minimize TGV_α^2 under the above constraint penalizing also the distance to the standard dequantization. This results in the minimization problem

$$\begin{cases} \min\limits_{u \in L^p(\Omega, \mathbf{R}^L)} F(u) + \text{TGV}_\alpha^2(u), \\ F(u) = \mathcal{I}_{\{f_{\text{lower}} \leq u \leq f_{\text{upper}} \text{ a.e.}\}}(u) + \dfrac{1}{p} \displaystyle\int_\Omega |u - \widetilde{f}|_\circ^p \, \mathrm{d}x. \end{cases} \tag{21}$$

The functional $F : L^p(\Omega, \mathbf{R}^L) \to (-\infty, \infty]$ can easily seen to be non-negative, proper, convex and lower semi-continuous, see also Subsect. 4.4. Moreover, as we have $F(u) \geq \frac{1}{p}\|u - \widetilde{f}\|_p^p$ for each u, the proof of (10) in Subsect. 4.1 directly leads to (10) for the above F and, consequently, to the existence of minimizers by virtue of Theorem 1.

The functional F can be discretized in a straightforward way:

$$F_h(u) = \begin{cases} \dfrac{1}{p} \displaystyle\sum_{i=1}^{N_1} \sum_{j=1}^{N_2} |u_{i,j} - \widetilde{f}_{i,j}|_\circ^p & \text{if } (f_{\text{lower}}) \leq u \leq (f_{\text{upper}}), \\ \infty & \text{else.} \end{cases}$$

The associated resolvent operator is explicitly computable in case $|\cdot|_\circ$ is the Euclidean norm and $q = 2$:

$$(\mathrm{id} + \tau \partial F_h)^{-1}(\bar{u}) = \min\Big(f_{\text{upper}}, \max\Big(f_{\text{lower}}, \frac{\bar{u} + \tau \widetilde{f}}{1 + \tau}\Big)\Big),$$

hence, Algorithm 1 can be employed yielding an iteration step for u^{n+1} which reads as

$$u^{n+1} = \min\Big(f_{\text{upper}}, \max\Big(f_{\text{lower}}, \frac{\bar{u} + \tau(\text{div}_h v^{n+1} + \widetilde{f})}{1 + \tau}\Big)\Big).$$

This algorithm has again been implemented. A numerical test on an image showing an oak leaf is depicted in Fig. 7. One observes that the TV-based dequantization coincides in essential parts with either f_{lower} or f_{upper} which cause the result to appear blocky. In contrast to that, the TGV_α^2-based model yields smooth transitions where necessary. In both cases, however, some of the small details in the leaf are lost.

Remark 4. Let us remark that the need for appropriate dequantization also arises in artifact-free JPEG decompression. In this situation, the blockwise DCT-coefficients are only given in a quantized form. This problem can also be solved with TV and TGV_α^2 regularization, see [3,6] for details.

4.5 Compressive Imaging

The last problem we like to discuss is the compressive imaging or 'single-pixel camera' reconstruction of a one-channel image [12], a problem which is already

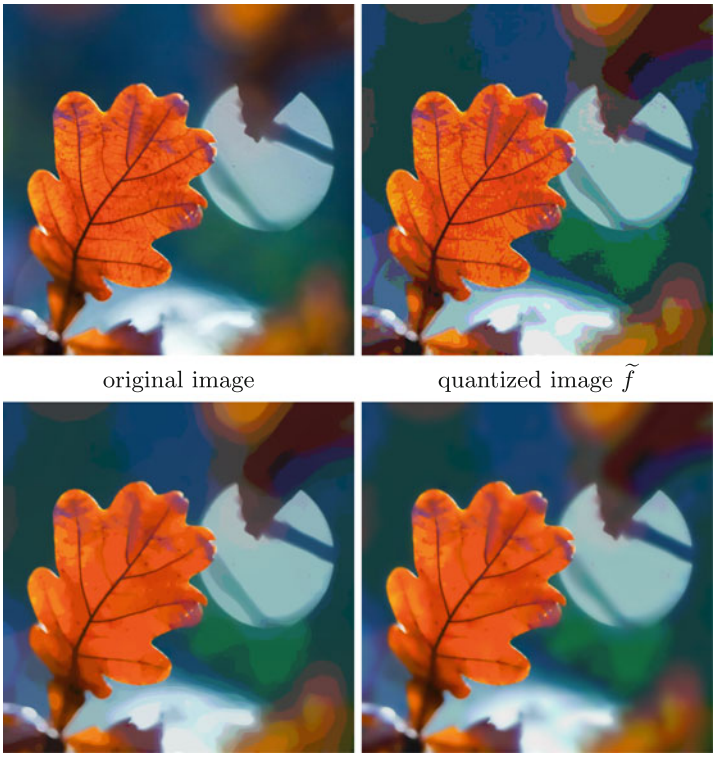


Fig. 7. Example for variational dequantization with the TV/TGV^2 image model according to (21). Top: The original image [32] (left, 512 × 512 pixels, RGB) and a quantized representation (right, 6 bins per color channel). Bottom: Result of TV-based dequantization (left, PSNR=29.59 dB, 2000 iterations, computation time: 1.88 s), and TGV^2-based dequantization (right, PSNR=29.87 dB, 2000 iterations, computation time: 4.76 s). Images licenced under CreativeCommons-by-sa-2.0 (http://creativecommons.org/licenses/by-sa/2.0/).

set in finite dimensions. Here, an image is not observed directly but only the accumulated gray values over random pixel patterns are measured. One essential point is that the number of measurements is significantly smaller than the number of pixels. For a discrete image of size $N_1 \times N_2$, let $M \ll N_1 N_2$ and $f \in \mathbf{R}^M$ represent the data. For each $1 \le m \le M$, let $k^m \in \{0,1\}^{N_1 \times N_2}$ be the random pixel pattern which is associated with the m-th measurement. The sought image $u \in U$ then obeys

$$Ku = f, \qquad \text{where} \qquad (Ku)_m = \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} k_{i,j}^m u_{i,j}, \qquad 1 \le m \le M.$$

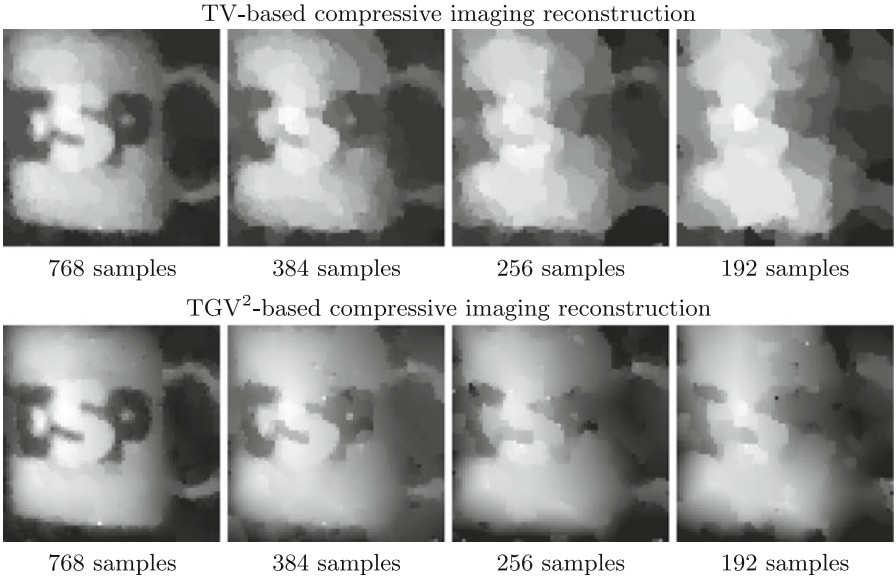


Fig. 8. Example for TV/TGV^2 compressive imaging reconstruction. Top: TV-based reconstruction of a 64×64 image from 18.75 %, 9.375 %, 6.25 % and 4.6875 % of the data (from left to right). Bottom: TGV^2-based reconstruction obtained from the same data.

As this constraint is highly ambiguous, the compressive imaging approach assumes that the image u is sparse in a certain representation which is usually translated into the discrete total variation $\mathrm{TV}(u)$ being small. A way to reconstruct u from f is then to solve

$$\min_{u \in U} \ \mathrm{TV}(u) + F_h(u), \qquad F_h(u) = \mathcal{I}_{\{0\}}(Ku - f).$$

We like to test the TGV^2_α-model for this application and propose to solve, numerically, the problem

$$\min_{u \in U} \ \mathrm{TGV}^2_\alpha(u) + F_h(u), \qquad F_h(u) = \mathcal{I}_{\{0\}}(Ku - f). \tag{22}$$

For this purpose, we rewrite F_h according to (13):

$$F_h(u) = \sup_{\lambda \in \Lambda} \ \langle Ku, \lambda \rangle_\Lambda - G_h(\lambda), \qquad G_h(u) = \langle f, \lambda \rangle_\Lambda$$

where $\Lambda = \mathbf{R}^M$. Hence, one can employ Algorithm 2 where the iteration steps for λ^{n+1} and u^{n+1} reads as

$$\begin{cases} \lambda^{n+1} = \lambda^n + \sigma(K\bar{u}^n - f), \\ u^{n+1} = u^n + \tau(\mathrm{div}_h v^{n+1} - K^*\lambda^{n+1}). \end{cases}$$

The norm of K can be estimated in terms of the Frobenius norm

$$\|K\| \leq \|K\|_{\mathrm{F}} = \Big(\sum_{m=1}^{M} |k^m|_2{}^2 \Big)^{1/2}$$

which is easily computable from the given random pixel patterns k^m, $1 \leq m \leq M$.

This method has also been implemented and tested. The utilized test data was the compressed sensing camera 'mug' data set from Rice's single-pixel camera project [26]. In Fig. 8, the outcome of the computations is depicted. Comparing the results for TV and TGV^2, one can see a noticeable increase in visual quality as smooth regions are resolved better. Even a low number of samples, rough features of the object is still perceptible in the TGV^2_α reconstruction. One has to note, however, that the current method is only suitable to study the effect of the TGV^2-model as it takes an extremely large number of iteration and, consequently, much computation time, in order to obtain the results. The reason seems to lie in the ill-posedness of the inversion of K, for which Algorithm 2 only performs a Landweber-type iteration ('perturbed' by the TGV^2_α regularization). In this case, it might be more efficient to utilize Algorithm 1.

5 Conclusions

The framework of total generalized variation, which already constitutes a convex model for piecewise smooth functions, can easily be extended to the multichannel case. Compared to the scalar case, the same results hold with respect to basic properties and existence of solutions for associated convex variational problems. Furthermore, these problems can be discretized and solved numerically in a unified way by realizing a primal-dual method for associated convex-concave saddle point problems. This numerical framework is general enough to devise efficient algorithms for the solution of common low-level image processing tasks such as denoising, deblurring, zooming and dequantization. These algorithms can be implemented, without greater effort, in parallel architecture such as GPUs. Numerical experiments confirm that the multichannel version of TGV^2_α is a suitable model for natural-looking color images and that incorporating it in variational problems leads to visually improved results in comparison to the total-variation counterpart. Moreover, the proof-of-concept application of the primal-dual algorithm to the single-pixel camera compressive imaging framework indicates that TGV^2_α might also be a suitable model for compressed sensing. However, due to the ill-posedness in this situation, it seems that more effort has to be made in order to solve the reconstruction problem efficiently. This could be a topic of further studies.

Acknowledgements. Support by the Austrian Science Fund (FWF) under grant SFB F32 (SFB "Mathematical Optimization and Applications in Biomedical Sciences") is gratefully acknowledged.

References

1. Alberti, G., Bouchitté, G., Maso, D.D.: The calibration method for the Mumford-Shah functional and free-discontinuity problems. Calc. Var. Partial Differ. Equ. **16**(3), 299–333 (2003)
2. Bredies, K., Dong, Y., Hintermüller, M.: Spatially dependent regularization parameter selection in total generalized variation models for image restoration. Int. J. Comput. Math. **90**(1), 109–123 (2013)
3. Bredies, K., Holler, M.: A total variation-based JPEG decompression model. SIAM J. Imaging Sci. **5**(1), 366–393 (2012)
4. Bredies, K., Kunisch, K., Pock, T.: Total generalized variation. SIAM J. Imaging Sci. **3**, 492–526 (2011)
5. Bredies, K., Valkonen, T.: Inverse problems with second-order total generalized variation constraints. In: Proceedings of SampTA 2011, 9th International Conference on Sampling Theory and Applications (2011)
6. Bredies, K., Holler, M.: Artifact-free JPEG decompression with total generalized variation. In: VISAPP 2012: Proceedings of the International Conference on Computer Vision Theory and Applications (2012)
7. Bresson, X., Chan, T.F.C.: Fast dual minimization of the vectorial total variation norm and applications to color image processing. Inverse Probl. Imaging **2**(4), 455–484 (2008)
8. Chambolle, A., Lions, P.L.: Image recovery via total variation minimization and related problems. Numer. Math. **76**, 167–188 (1997)
9. Chambolle, A., Pock, T.: A first-order primal-dual algorithm for convex problems with applications to imaging. J. Math. Imaging Vis. **40**, 120–145 (2011)
10. Chan, T.F., Esedoglu, S., Park, F.E.: Image decomposition combining staircase reduction and texture extraction. J. Visual Commun. Image Represent. **18**(6), 464–486 (2007)
11. Dong, Y., Hintermüller, M., Rincon-Camacho, M.: Automated regularization parameter selection in multi-scale total variation models for image restoration. J. Math. Imaging Vis. **40**, 82–104 (2011)
12. Duarte, M.F., Davenport, M.A., Takhar, D., Laska, J., Sun, T., Kelly, K., Baraniuk, R.G.: Single-pixel imaging via compressive sampling. IEEE Signal Proc. Mag. **25**(2), 83–91 (2008)
13. Eckstein, J., Bertsekas, D.P.: On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. Math. Program. **55**, 293–318 (1992)
14. Hinterberger, W., Scherzer, O.: Variational methods on the space of functions of bounded Hessian for convexification and denoising. Computing **76**, 109–133 (2006)
15. ˆ @ ˆ ina@Flickr: Alina's eye. Licenced under CreativeCommons-by-2.0 (http://creativecommons.org/licenses/by/2.0/). http://www.flickr.com/photos/angel_ina/3201337190 (2009)
16. Klöckner, A.: PyCUDA 2011.2.2. http://mathema.tician.de/software/pycuda
17. Kubina, J.: Bubble popper. Licenced under CreativeCommons-by-sa-2.0 (http://creativecommons.org/licenses/by-sa/2.0/). http://www.flickr.com/photos/kubina/42275122 (2005)
18. Lions, P.L., Mercier, B.: Splitting algorithms for the sum of two nonlinear operators. SIAM J. Numer. Anal. **16**(6), 964–979 (1979)
19. Lysaker, M., Lundervold, A., Tai, X.C.: Noise removal using fourth-order partial differential equation with applications to medical magnetic resonance images in space and time. IEEE Trans. Image Process. **12**, 1579–1590 (2003)

20. Mumford, D., Shah, J.: Optimal approximation by piecewise smooth functions and associated variational problems. Commun. Pure Appl. Math. **42**, 577–685 (1989)
21. Nikolova, M.: Local strong homogeneity of a regularized estimator. SIAM J. Appl. Math. **61**(2), 633–658 (2000)
22. NIVIDA Corporation: CUDA toolkit 4.0. http://www.nvidia.com/getcuda
23. Papafitsoros, K., Schönlieb, C.B.: A combined first and second order variational approach for image reconstruction. ArXiv e-print 1202.6341 (2012)
24. Pock, T., Cremers, D., Bischof, H., Chambolle, A.: An algorithm for minimizing the Mumford-Shah functional. In: IEEE 12th International Conference on Computer Vision (ICCV). pp. 1133–1140 (2009)
25. Python Software Foundation: Python programming language 2.6. http://www.python.org/
26. Rice Single-Pixel Camera Project: CS camera data. http://dsp.rice.edu/cscamera
27. Rockafellar, R.T.: Monotone operators and the proximal point algorithm. SIAM J. Control Optim. **14**(5), 877–898 (1976)
28. Rudin, L., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Physica D **60**, 259–268 (1992)
29. Schristia@Flickr: 1, 2, 3. Licenced under CreativeCommons-by-2.0 (http://creativecommons.org/licenses/by/2.0/). http://www.flickr.com/photos/schristia/4057490235 (2009)
30. Stampacchia, G.: Le problème de Dirichlet pour les équations elliptiques du second ordre à coefficients discontinus. Ann. Inst. Fourier (Grenoble) **15**(1), 189–258 (1965)
31. Steve-h@Flickr: Wet green smiles. Licenced under CreativeCommons-by-sa-2.0 (http://creativecommons.org/licenses/by-sa/2.0/). http://www.flickr.com/photos/sbh/3859041020 (2009)
32. Steve-h@Flickr: Oak leaves and bokeh. Licenced under CreativeCommons-by-sa-2.0 (http://creativecommons.org/licenses/by-sa/2.0/). http://www.flickr.com/photos/sbh/6802942537 (2012)
33. Temam, R.: Mathematical Problems in Plasticity. Bordas, Paris (1985)
34. The Scipy Community: Scientific tools for Python. http://www.scipy.org/
35. Vese, L.: A study in the BV space of a denoising-deblurring variational problem. Appl. Math. Opt. **44**, 131–161 (2001)
36. Ring, W.: Structural properties of solutions to total variation regularization problems. ESAIM: Math. Model. Num. **34**(4), 799–810 (2000)
37. Yang, J., Yin, W., Zhang, Y., Wang, Y.: A fast algorithm for edge-preserving variational multichannel image restoration. SIAM J. Imaging Sci. **2**(2), 569–592 (2009)

Half-Quadratic Algorithm for ℓ_p-ℓ_q Problems with Applications to TV-ℓ_1 Image Restoration and Compressive Sensing

Raymond H. Chan[1(✉)] and Hai-Xia Liang[2(✉)]

[1] Department of Mathematics, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong, People's Republic of China
rchan@math.cuhk.edu.hk

[2] Mathematics and Physics Teaching Centre, Xi'an Jiaotong-Liverpool University, No. 111 Ren'ai Road, Suzhou Industrial Park, Suzhou, Jiangsu Province, People's Republic of China
haixia.liang@xjtlu.edu.cn

Abstract. In this paper, we consider the ℓ_p-ℓ_q minimization problem with $0 < p, q \leq 2$. The problem has been studied extensively in image restoration and compressive sensing. In the paper, we first extend the half-quadratic algorithm from ℓ_1-norm to ℓ_p-norm with $0 < p < 2$. Based on this, we develop a half-quadratic algorithm to solve the ℓ_p-ℓ_q problem. We prove that our algorithm is indeed a majorize-minimize approach. From that we derive some convergence results of our algorithm, e.g. the objective function value is monotonically decreasing and convergent. We apply the proposed approach to TV-ℓ_1 image restoration and compressive sensing in magnetic resonance (MR) imaging applications. The numerical results show that our algorithm is fast and efficient in restoring blurred images that are corrupted by impulse noise, and also in reconstructing MR images from very few k-space data.

Keywords: ℓ_p regularization · Half-quadratic · Majorize-minimize algorithm · Impulse noise · Compressive sensing · Magnetic resonance imaging

1 Introduction

In this paper, we consider the following ℓ_p-ℓ_q minimization problem

$$\min_{\mathbf{u}} \left\{ \frac{\lambda}{p} \|\Psi \mathbf{u}\|_p^p + \frac{1}{q} \|A\mathbf{u} - \mathbf{f}\|_q^q \right\} \tag{1}$$

where $0 < p, q \leq 2$ and $\mathbf{u} \in \mathbb{R}^n$ is an image represented by a vector by concatenating the columns. Here, Ψ can be a sparsifying operator such as a wavelet transform or a regularization operator such as the discrete gradient operator;

The research was supported in part by HKRGC Grant CUHK 400510 and CUHK DAG 2060408.

A. Bruhn et al. (Eds.): Global Optimization Methods, LNCS 8293, pp. 78–103, 2014.
DOI: 10.1007/978-3-642-54774-4_4,

and A can be a sampling operator or a blurring operator. Problem (1) has been studied extensively in image processing and compressive sensing. For example, if $p = 1$, $q = 2$, and Ψ is the discrete gradient operator, then (1) is the TV-ℓ_2 minimization problem. It has been successfully applied to image restoration [20,37,39] because of its good property in preserving edges. TV-ℓ_1 model (i.e. $q = 1$) has also been successfully applied to applications such as impulse noise removal [26,41], image cartoon-texture decomposition [45], feature selection [45], multiscale decomposition [46], and computer vision [10].

When A is a sampling operator, model (1) with $0 \leq p \leq 1$ and $q = 2$ has received a lot of attention lately because of the introduction of compressive sensing techniques ($\|\mathbf{u}\|_0$ is defined to be the number of nonzero entries in $\mathbf{u}$). The techniques allow high resolution images and signals to be reconstructed from a small amount of data [6,7,17]. There, the linear constrained minimization problems are considered. Unfortunately, as $p = 0$, the constrained minimization problem is NP-hard [1]. For this reason, different approaches are used to approximate the ℓ_0-norm [3,14,23,29,30,34], or alternatively one solves the ℓ_1-norm [5,20,32,47] or the non-convex ℓ_p-norm [12–14] problem with $0 < p < 1$. The application of compressive sensing with $p = 1$ and $q = 2$ to magnetic resonance (MR) image reconstruction can be found in [7,27]. There it was shown that perfect reconstruction of the Shepp-Logam phantom is possible from 22 radial lines or 9 % of the k-space data. For real images which are less sparse than the synthetic phantoms, one can obtain improved results by having both a wavelet transform and a discrete gradient in the objective function. However, the ℓ_1-norm regularized model can not get good results from fewer k-space data. See [27].

Problem (1) with $0 < p < 1$ is a non-convex optimization problem. Theoretical work [15,38] has justified the non-convex approach as it guarantees perfect reconstruction under a less restrictive condition than that would be needed by ℓ_1 minimization. There are quite a few algorithms for solving the non-convex problem, see [4,12–14,35]. The numerical results in [12,13] show that the perfect MR image can be recovered from 10 radial lines (i.e. 3.8 % of the k-space data) for some $0 < p < 1$. In [12], a fast algorithm based on the p-shrinkage reduces the number of lines further to 9. For more details on p-shrinkage, one may consult [43] where the 1/2-theresholding algorithm was studied. In [42], the author analyzed the effectiveness of problem (1) in recovering sparse signals. The results showed that if $p \in [1/2, 1)$, then the smaller the p is, the sparser the ℓ_p-norm regularization solution will be; and if $p \in (0, 1/2]$, there are no significant differences in the sparsity of the solution.

In this paper, we propose a half-quadratic algorithm (HQA) to solve (1) for $0 < p, q \leq 2$. We prove that our algorithm is indeed a majorize-minimize algorithm [11,24,25] for solving (1) and from that some convergence results can be obtained immediately. For example, we show that the objective function value is monotonically decreasing and convergent. We also give the convergence rate of the method for $1 \leq p, q \leq 2$. We test our algorithms on two applications: (i) TV-ℓ_1 minimization problem, and (ii) non-convex ℓ_p-ℓ_2 compressive sensing. Problem

(i) is for restoring blurred images that are corrupted by impulse noise, and our algorithm can reach high SNR value in less CPU time than the augmented-Lagrangian method (ALM) in [41] and the fast total variation deconvolution method (FTVDM) in [44]. Problem (ii) is for reconstructing MR images from few k-space data, and our algorithm can get better results with less computational cost than the p-shrinkage algorithm in [12].

The outline of the paper is as follows: In Sect. 2, we first derive our HQA for model (1) and then adapt it to solve the TV-ℓ_1 minimization problem and non-convex ℓ_p-ℓ_2 compressive sensing problem. In Sect. 3, we prove that our HQA is indeed a majorize-minimize algorithm, and hence we derive some convergence results for the algorithm. Comparison with the ALM, FTVDM and the p-shrinkage algorithm [12] are given in Sect. 4. Finally Sect. 5 is on some concluding remarks.

2 The Half-Quadratic Approach for ℓ_p-ℓ_q Minimization Problem

The half-quadratic regularization approach has been used in image processing [11,19]. In [31], the authors showed the equivalence of the HQ minimization and the gradient linearization iterations. The HQ technique is based on the fact that, if $0 \neq t \in \mathbb{R}$, then

$$|t| = \min_{v>0}\{vt^2 + \frac{1}{4v}\} \tag{2}$$

and the minimum value is reached at $v = \frac{1}{2|t|}$. Note that the function $vt^2+1/(4v)$ is quadratic in t but not in v and hence the name *half-quadratic*. In this section, we first study the general form of (2) for $0 < p < 2$. Then we derive our HQA to solve (1) and adapt it to the TV-ℓ_1 minimization problem and compressive sensing.

2.1 The Half-Quadratic Algorithm

The following lemma shows us the corresponding formula of (2) for $|\cdot|^p$ with $0 < p < 2$.

Lemma 1. *For any $0 < p < 2$, if $0 \neq t \in \mathbb{R}$, then there exist positive constants α and ξ such that*

$$|t|^p = \min_{v>0}\left\{vt^2 + \frac{1}{\xi v^\alpha}\right\}. \tag{3}$$

Proof: Let us first assume that $\alpha, \xi > 0$ and define

$$f(v,t) := vt^2 + \frac{1}{\xi v^\alpha}. \tag{4}$$

Then $f(v,t)$ is convex with respect to $v > 0$ for any fixed $t \neq 0$. In addition, $f(v,t) \to \infty$ as either $v \to 0$ or $v \to \infty$. The minimizer of $f(v,t)$ with respect to v is therefore given by solving $f'_v(v,t) = 0$ and is equal to

$$v^* = \left(\frac{\alpha}{\xi t^2}\right)^{\frac{1}{1+\alpha}}. \tag{5}$$

Substituting (5) into (4), the minimum value is

$$f(v^*,t) = \Big[\left(\frac{\alpha}{\xi}\right)^{\frac{1}{1+\alpha}} + \frac{1}{\xi}\left(\frac{\xi}{\alpha}\right)^{\frac{\alpha}{1+\alpha}}\Big] t^{\frac{2\alpha}{1+\alpha}}.$$

Since we want the minimum value to be $|t|^p$ for any $t \neq 0$, we set

$$\begin{cases} 2\alpha/(1+\alpha) & = p, \\ \left(\frac{\alpha}{\xi}\right)^{\frac{1}{1+\alpha}} + \frac{1}{\xi}\left(\frac{\xi}{\alpha}\right)^{\frac{\alpha}{1+\alpha}} & = 1. \end{cases}$$

Solving the system for α and ξ, we have

$$\alpha = \frac{p}{2-p} \quad \text{and} \quad \xi = \frac{2^{\frac{2}{2-p}}}{(2-p)\cdot p^{\frac{p}{2-p}}}. \tag{6}$$

Clearly both α and ξ are positive for any fixed $0 < p < 2$. □

Remarks

(a) As an example, for $p = 1/2$, we have $\xi = 2^{8/3}/3$ and $\alpha = 1/3$. For $p = 1$, we have $\alpha = 1$ and $\xi = 4$, and hence (3) reduces to (2). Note that we would like to have $\alpha > 0$ so that the functional $f(v,t)$ is convex with respect to v for any fixed $t \neq 0$. By (5) and (6), we have

$$v^* = \frac{p}{2}|t|^{p-2}. \tag{7}$$

(b) From the above lemma, we know that for fixed $\alpha, \xi > 0$ of (6) and any $t \neq 0$, the minimum of (3) is reached at the stationary point of $f(v,t)$ w.r.t v, which is an interior point of the open, convex, feasible set $\mathbb{R}^+$.

Next we apply (3) to solve (1). To simplify the discussions, we first consider the case where $0 < p, q < 2$, and leave the case where p and/or $q = 2$ later.

Case 1: $0 < p, q < 2$. Notice that Lemma 1 holds only for $t \neq 0$ as negative power of $|t|$ appears in v^*, see (7). Hence in order to apply (3), we need to smooth (1) first. In the following, we denote $|\rho|_\epsilon := \sqrt{\rho^2 + \epsilon}$ for any $\rho \in \mathbb{R}$ and $\epsilon > 0$. The smoothed ℓ_p-ℓ_q problem of (1) is

$$\min_{\mathbf{u}} \left\{ \frac{\lambda}{p}\|\Psi\mathbf{u}\|_{p,\beta}^p + \frac{1}{q}\|A\mathbf{u} - \mathbf{f}\|_{q,\gamma}^q \right\} =: \min_{\mathbf{u}}\{\Phi_{\beta,\gamma}(\mathbf{u})\} \tag{8}$$

where $\|\Psi\mathbf{u}\|_{p,\beta}^p := \sum_{i=1}^n |\Psi_i\mathbf{u}|_\beta^p$ and $\|A\mathbf{u}-\mathbf{f}\|_{q,\gamma}^q =: \sum_{i=1}^n |A_i\mathbf{u}-f_i|_\gamma^q$, with β and γ being small positive numbers, and Ψ_i and A_i are the ith rows of Ψ and A respectively. Applying (3) to (8), problem (8) becomes

$$\begin{aligned}
&\min_{\mathbf{u}}\left\{\sum_{i=1}^n\left[\frac{\lambda}{p}\min_{v_i>0}\left(v_i|\Psi_i\mathbf{u}|_\beta^2+\frac{1}{\xi_p v_i^{\alpha_p}}\right)+\frac{1}{q}\min_{w_i>0}\left(w_i|A_i\mathbf{u}-f_i|_\gamma^2+\frac{1}{\xi_q w_i^{\alpha_q}}\right)\right]\right\}\\
&=\min_{\mathbf{u},\mathbf{v}>\mathbf{0},\mathbf{w}>\mathbf{0}}\left\{\sum_{i=1}^n\left[\frac{\lambda}{p}\left(v_i|\Psi_i\mathbf{u}|_\beta^2+\frac{1}{\xi_p v_i^{\alpha_p}}\right)+\frac{1}{q}\left(w_i|A_i\mathbf{u}-f_i|_\gamma^2+\frac{1}{\xi_q w_i^{\alpha_q}}\right)\right]\right\}\\
&=:\min_{\mathbf{u},\mathbf{v}>\mathbf{0},\mathbf{w}>\mathbf{0}}\{\mathcal{L}(\mathbf{u},\mathbf{v},\mathbf{w})\},
\end{aligned}\tag{9}$$

where $\mathbf{v},\mathbf{w}>0$ mean that all the components of $\mathbf{v},\mathbf{w}$ are greater than 0. Here ξ_i and α_i, $i=p,q$ are scalars given by (6).

To solve (9), we apply the alternating minimization procedure, namely,

$$\mathbf{v}^{k+1}=\arg\min_{\mathbf{v}>0}\mathcal{L}(\mathbf{u}^k,\mathbf{v},\mathbf{w}^k),\tag{10}$$

$$\mathbf{w}^{k+1}=\arg\min_{\mathbf{w}>0}\mathcal{L}(\mathbf{u}^k,\mathbf{v}^{k+1},\mathbf{w}),\tag{11}$$

$$\mathbf{u}^{k+1}=\arg\min_{\mathbf{u}}\mathcal{L}(\mathbf{u},\mathbf{v}^{k+1},\mathbf{w}^{k+1}).\tag{12}$$

By (7), we know that (10) and (11) have explicit component minimizers

$$v_i^{k+1}=\frac{p}{2}|\Psi_i\mathbf{u}^k|_\beta^{p-2}\quad\text{and}\quad w_i^{k+1}=\frac{q}{2}|A_i\mathbf{u}^k-f_i|_\gamma^{q-2}.\tag{13}$$

Note that $\mathcal{L}(\mathbf{u},\mathbf{v}^{k+1},\mathbf{w}^{k+1})$ is continuously differentiable in $\mathbf{u}$. Hence $\mathbf{u}^{k+1}$ in (12) is the solution of

$$0=\nabla_{\mathbf{u}}\mathcal{L}(\mathbf{u},\mathbf{v}^{k+1},\mathbf{w}^{k+1})=\lambda\Psi^\top D_\beta(\mathbf{u}^k)\Psi\mathbf{u}+A^\top D_\gamma(\mathbf{u}^k)(A\mathbf{u}-\mathbf{f}),\tag{14}$$

where $D_\beta(\mathbf{u}^k)$ and $D_\gamma(\mathbf{u}^k)$ are diagonal matrices with their ith diagonal entries being

$$\frac{2}{p}v_i^{k+1}=|\Psi_i\mathbf{u}^k|_\beta^{p-2}\quad\text{and}\quad\frac{2}{q}w_i^{k+1}=|A_i\mathbf{u}^k-f_i|_\gamma^{q-2}\tag{15}$$

respectively. Equation (14) provides us an iterative scheme for finding the minimizer of (8).

Case 2: p and/or $q=2$. In that case, the corresponding term in (1) is quadratic and differentiable. So there is no need to apply the half-quadratic technique (3) to the term. However, one can easily check by differentiation of (1) that (14) and (15) are still valid. More precisely, if $p=2$, then differentiation of the first term in (1) gives $\lambda\Psi^\top\Psi\mathbf{u}$. But by (15), $D_\beta(\mathbf{u}^k)\equiv I$, the identity matrix and hence the first term of (14) also reduces to $\lambda\Psi^\top\Psi\mathbf{u}$. Similarly, if $q=2$, then $D_\gamma(\mathbf{u}^k)\equiv I$ and (14) still holds. In particular, if $p=q=2$, then (14) reduces to the least-squares problem as expected, and the minimizer $\mathbf{u}$ can be obtained in one iteration. In the following discussions, we will exclude this trivial case $p=q=2$.

Thus combining Case 1 and Case 2, we see that (14) holds for $0 < p, q \leq 2$. We summarize our half-quadratic algorithm (HQA) for the smoothed ℓ_p-ℓ_q problem (8) below:

(1) Initialize $\mathbf{u}^0$;
(2) For $l = 0, 1, \cdots$ until convergence, find $\mathbf{u}^{k+1}$ by solving (cf (14))

$$\Big(\lambda \Psi^\top D_\beta(\mathbf{u}^k)\Psi + A^\top D_\gamma(\mathbf{u}^k)A\Big)\mathbf{u} = A^\top D_\gamma(\mathbf{u}^k)\mathbf{f}, \tag{16}$$

where $D_\beta(\cdot)$ and $D_\gamma(\cdot)$ are diagonal matrices given in (15).

To find the solution to the ℓ_p-ℓ_q problem (1), we can use a continuation method and apply HQA to a sequence of $\{\beta^l, \gamma^l\}$ going to zero. We will discuss the implementation in more details in the section on numerical tests, see **Algorithm 1** in Sect. 4.

2.2 Half-Quadratic Algorithm for TV-ℓ_1 and Compressive Sensing

Let us consider HQA (16) for two specific examples: TV-ℓ_1 image restoration and compressive sensing. The TV-ℓ_1 minimization problem is of the form:

$$\min_{\mathbf{u}} \Big\{\lambda\|\nabla\mathbf{u}\|_1 + \|A\mathbf{u} - \mathbf{f}\|_1\Big\}, \tag{17}$$

where $\|\nabla\mathbf{u}\|_1 := \sum_{i=1}^n \sqrt{[(G_1)_i\mathbf{u}]^2 + [(G_2)_i\mathbf{u}]^2}$ with $(G_j)_i$ representing the ith row of the finite difference operator in the x_j-coordinate. The smoothed version of (17) is

$$\min_{\mathbf{u}} \left\{\sum_{i=1}^n [\lambda|\nabla u_i|_\beta + |A_i\mathbf{u} - f_i|_\gamma]\right\}, \tag{18}$$

where $|\nabla u_i|_\beta := \sqrt{[(G_1)_i\mathbf{u}]^2 + [(G_2)_i\mathbf{u}]^2 + \beta}$ and $\beta, \gamma \to 0$. Letting $p = q = 1$, $\Psi = G$ in (15) and (16), we see that (16) should be replaced by:

$$\Big\{\lambda \sum_{j=1}^2 [G_j^\top D_\beta(\mathbf{u}^k)G_j] + A^\top D_\gamma(\mathbf{u}^k)A\Big\}\mathbf{u} = A^\top D_\gamma(\mathbf{u}^k)\mathbf{f}, \tag{19}$$

where $D_\beta(\mathbf{u}^k)$ and $D_\gamma(\mathbf{u}^k)$ are diagonal matrices with their ith diagonal entries being $|\nabla u_i^k|_\beta^{-1}$ and $|A_i\mathbf{u}^k - f_i|_\gamma^{-1}$ respectively.

Next we consider HQA for MR image reconstruction problem. In [27], a regularization term combining a discrete gradient ∇ and an orthogonal wavelet W [8] is considered for $0 < p \leq 1$:

$$\min_{\mathbf{u}} \Big\{\|\nabla\mathbf{u}\|_p^p + \delta\|W\mathbf{u}\|_p^p : RF\mathbf{u} = \mathbf{f}\Big\}. \tag{20}$$

Here R is a selection matrix (a diagonal matrix) and F is the Fourier transform. As mentioned in [12], it is sufficient to use $\delta = 1$ in (20); and for gradient-sparse

images, we can simply take $\delta = 0$. Problem (20) is related to the minimization problem

$$\min_{\mathbf{u}} \Big\{ \frac{\lambda}{p} \Big(\|\nabla \mathbf{u}\|_p^p + \delta \|W\mathbf{u}\|_p^p \Big) + \frac{1}{2} \|RF\mathbf{u} - \mathbf{f}\|_2^2 \Big\}, \tag{21}$$

where λ is the Lagrange multiplier. As before, since the data fitting term is quadratic, we keep it intact, and we smooth only the p-norm terms. Hence we have the following smoothed problem:

$$\min_{\mathbf{u}} \Big\{ \frac{\lambda}{p} \sum_{i=1}^{n} \Big(|\nabla u_i|_\beta^p + \delta |W_i \mathbf{u}|_\gamma^p \Big) + \frac{1}{2} \|RF\mathbf{u} - \mathbf{f}\|_2^2 \Big\},$$

with $\beta, \gamma \to 0$. Correspondingly, Eq. (16) should be replaced by:

$$\Big[\lambda \Big(\sum_{j=1}^{2} [G_j^\top D_\beta(\mathbf{u}^k) G_j] + \delta W^\top D_\gamma(\mathbf{u}^k) W \Big) + F^* RF \Big] \mathbf{u} = F^* R \mathbf{f}, \tag{22}$$

where F^* is inverse Fourier transform, $D_\beta(\mathbf{u}^k)$ and $D_\gamma(\mathbf{u}^k)$ are diagonal matrices with their ith diagonal entries being $|\nabla u_i^k|_\beta^{p-2}$ and $|W_i \mathbf{u}^k|_\gamma^{p-2}$ respectively.

3 Convergence Analysis

In this section, we analyze the convergence of the HQA (16) based on the convergence property of the majorize-minimize algorithm (MMA) in [11,24,25,40] for fixed β, γ. We first show that $\Phi_{\beta,\gamma}(\mathbf{u}^k)$ is monotonically decreasing and convergent for $0 < p, q \leq 2$. Then we show that $\mathbf{u}^k$ is convergent and linearly convergent for $1 \leq p, q \leq 2$.

3.1 Convergence of $\Phi_{\beta,\gamma}(\mathbf{u}^k)$ for $0 < p, q \leq 2$

The MM optimization technique [11,24,25] is to solve a minimization problem $\min_{\mathbf{u}} \Phi_{\beta,\gamma}(\mathbf{u})$ by

$$\mathbf{u}^{k+1} = \arg\min_{\mathbf{u}} \{ Q(\mathbf{u}, \mathbf{u}^k) \}, \tag{23}$$

where $Q(\mathbf{u}, \mathbf{u}^k)$, called a *tangent majorant function* of $\Phi_{\beta,\gamma}(\mathbf{u})$ at $\mathbf{u}^k$, must satisfy

$$Q(\mathbf{u}, \mathbf{u}^k) \geq \Phi_{\beta,\gamma}(\mathbf{u}), \qquad \forall\ \mathbf{u} \in \mathbb{R}^n, \tag{24}$$

$$Q(\mathbf{u}, \mathbf{u}^k) = \Phi_{\beta,\gamma}(\mathbf{u}), \qquad \text{at } \mathbf{u} = \mathbf{u}^k, \tag{25}$$

$$\nabla_1 Q(\mathbf{u}, \mathbf{u}^k) = \nabla \Phi_{\beta,\gamma}(\mathbf{u}), \qquad \text{at } \mathbf{u} = \mathbf{u}^k. \tag{26}$$

Here, $\nabla_1 Q(\mathbf{u}, \mathbf{u}^k)$ denotes the partial derivative with respect to the first vector variable. Convergence analysis of the MMA can be found in [11,24,25].

For the smoothed ℓ_p-ℓ_q problem (8), if we define $Q(\mathbf{u}, \mathbf{u}^k) := \mathcal{L}(\mathbf{u}, \mathbf{v}^{k+1}, \mathbf{w}^{k+1})$, then our HQA (12) can be written as

$$\mathbf{u}^{k+1} = \arg\min_{\mathbf{u}} \mathcal{L}(\mathbf{u}, \mathbf{v}^{k+1}, \mathbf{w}^{k+1}) = \arg\min_{\mathbf{u}} \{ Q(\mathbf{u}, \mathbf{u}^k) \}, \tag{27}$$

which is of the same form as (23). For $0 < p, q < 2$, substituting (13) into (27), we obtain the explicit form of $Q(\mathbf{u}, \mathbf{u}^k)$:

$$Q(\mathbf{u}, \mathbf{u}^k) = \sum_{i=1}^{n} \left[\frac{\lambda}{p} \left(\frac{p}{2} |\Psi_i \mathbf{u}^k|_\beta^{p-2} |\Psi_i \mathbf{u}|_\beta^2 + \frac{2-p}{2} |\Psi_i \mathbf{u}^k|_\beta^p \right) + \frac{1}{q} \left(\frac{q}{2} |A_i \mathbf{u}^k - f_i|_\gamma^{q-2} |A_i \mathbf{u} - f_i|_\gamma^2 + \frac{2-q}{2} |A_i \mathbf{u}^k - f_i|_\gamma^p \right) \right]. \quad (28)$$

We recall that when p or q is equal to 2, there is no need to smooth the corresponding term as it is already differentiable. Hence if we use the convention that $\beta = 0$ (or respectively $\gamma = 0$) whenever $p = 2$ (or respectively $q = 2$), then (28) holds for all $0 < p, q \leq 2$.

Lemma 2. *Let $0 < p, q \leq 2$. For any fixed $\beta, \gamma > 0$, the HQA for the smoothed ℓ_p-ℓ_q problem (8) is the same as an MMA with tangent majorant function $Q(\mathbf{u}, \mathbf{u}^k)$ defined by (28).*

Proof: Since our HQA (12) is rewritten as (27) with $Q(\mathbf{u}, \mathbf{u}^k)$ given by (28), we only need to prove that (24)–(26) holds for such Q. Substituting $\mathbf{u} = \mathbf{u}^k$ in (28) and using the definition of $\Phi_{\beta,\gamma}(\mathbf{u})$ in (8), we see that $Q(\mathbf{u}^k, \mathbf{u}^k) = \Phi_{\beta,\gamma}(\mathbf{u}^k)$, which is (25). To prove that Q satisfies (24), we use the Young inequality, which states that $(x^a/a + y^b/b) \geq xy$ for all $x, y \geq 0$, $a, b \geq 1$ and $1/a + 1/b = 1$. Let us consider the case where $0 < p < 2$ first, and set

$$x = |\Psi_i \mathbf{u}^k|_\beta^{\frac{(p-2)p}{2}} |\Psi_i \mathbf{u}|_\beta^p, \quad y = |\Psi_i \mathbf{u}^k|_\beta^{\frac{(2-p)p}{2}}, \quad a = \frac{2}{p}, \quad b = \frac{2}{2-p}.$$

Then Young's inequality implies that

$$\frac{p}{2} |\Psi_i \mathbf{u}^k|_\beta^{p-2} |\Psi_i \mathbf{u}|_\beta^2 + \frac{2-p}{2} |\Psi_i \mathbf{u}^k|_\beta^p \geq |\Psi_i \mathbf{u}|_\beta^p.$$

Clearly, the inequality becomes a trivial equality when $p = 2$. Similarly, we can show that

$$\frac{q}{2} |A_i \mathbf{u}^k - f_i|_\gamma^{q-2} |A_i \mathbf{u} - f_i|_\gamma^2 + \frac{2-q}{2} |A_i \mathbf{u}^k - f_i|_\gamma^p \geq |A_i \mathbf{u} - f_i|_\gamma^q,$$

for all $0 < q \leq 2$. Then by taking the summation, we immediately have $Q(\mathbf{u}, \mathbf{u}^k) \geq \Phi_{\beta,\gamma}(\mathbf{u})$; and hence (24) holds. Finally by taking the derivatives of $\Phi_{\beta,\gamma}(\mathbf{u})$ and $Q(\mathbf{u}, \mathbf{u}^k)$ with respect to $\mathbf{u}$, we have

$$\nabla \Phi_{\beta,\gamma}(\mathbf{u}) = \lambda \Psi^\top D_\beta(\mathbf{u}) \Psi \mathbf{u} + A^\top D_\gamma(\mathbf{u})(A\mathbf{u} - \mathbf{f}), \quad (29)$$

$$\nabla_1 Q(\mathbf{u}, \mathbf{u}^k) = \lambda \Psi D_\beta(\mathbf{u}^k) \Psi \mathbf{u} + A^\top D_\gamma(\mathbf{u}^k)(A\mathbf{u} - \mathbf{f}), \quad (30)$$

where $D_\beta(\cdot)$ and $D_\gamma(\cdot)$ are defined as in (14). Substituting $\mathbf{u} = \mathbf{u}^k$ into (29) and (30), we immediately have (26). □

Based on Lemma 2, we can derive the following fundamental convergence theorem.

Theorem 1. *Let $0 < p, q \leq 2$. For the sequence $\{\mathbf{u}^k\}$ generated by the HQA, we have that $\{\Phi_{\beta,\gamma}(\mathbf{u}^k)\}$ is monotonically decreasing and convergent.*

Proof: The theorem is a direct result of Lemma 2. First, $\{\Phi_{\beta,\gamma}(\mathbf{u}^k)\}$ is bounded from below by 0. In Lemma 2, we have shown that the HQA is an MMA, which implies that

$$\Phi_{\beta,\gamma}(\mathbf{u}^{k+1}) \leq Q(\mathbf{u}^{k+1}, \mathbf{u}^k) \leq Q(\mathbf{u}^k, \mathbf{u}^k) = \Phi_{\beta,\gamma}(\mathbf{u}^k). \tag{31}$$

Here the first inequality and the last equality follow from (24) and (25), while the second inequality holds because $\mathbf{u}^{k+1}$ is a minimizer of $Q(\mathbf{u}, \mathbf{u}^k)$. □

3.2 Convergence of $\mathbf{u}^k$ for $1 \leq p, q \leq 2$

Note that if $0 < p, q < 1$, the ℓ_p-ℓ_q minimization problem is non-convex. Hence, in the following, we discuss the convergence of $\mathbf{u}^k$ for $1 \leq p, q \leq 2$ only. In order that $\mathbf{u}^k$ is solvable from (16) and hence our HQA will not break down, we need the following assumption:

$$\ker(\Psi^\top \Psi) \cap \ker(A^\top A) = \{\mathbf{0}\}. \tag{32}$$

We remark that this assumption is very general and usually satisfied. For example, in regularization problems, Ψ is usually a difference operator, and hence is a high-pass filter; whereas A is a blurring operator, and hence is a low-pass filter. Therefore, (32) holds. For compressive sensing, Ψ is usually taken to be an orthogonal transform and we have $\Psi^\top \Psi = I$. Hence, $\ker(\Psi^\top \Psi) = \{\mathbf{0}\}$ which implies (32) holds for any A.

In [11,40], the authors gave the convergence proof of general MMAs when the objective function Φ and its corresponding tangent majorant function Q satisfy Hypotheses 4.1 and 4.2 there. The convergence proof for our HQA will follow closely the proofs there. More precisely, we will show that our $\Phi_{\beta,\gamma}$ defined in (8) and our Q defined in (27) do satisfy the hypotheses, and hence the convergence follows immediately. Let us write out the hypotheses below.

Hypothesis 1. *[Hypothesis 4.1 in [11]]*

1. *Φ is twice continuously differentiable and strictly convex.*
2. *Φ is coercive, i.e., $\lim_{\|\mathbf{u}\|_2 \to \infty} \Phi(\mathbf{u}) = \infty$.*
3. *Φ is bounded from below.*

Hypothesis 2. *[Hypothesis 4.2 in [11]]*

(a) *There exists a properly defined function $C : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ such that*
   (i) *$Q(\mathbf{u}, \mathbf{v}) = \Phi(\mathbf{v}) + (\mathbf{u} - \mathbf{v})^\top \nabla \Phi(\mathbf{v}) + \frac{1}{2}(\mathbf{u} - \mathbf{v})^\top C(\mathbf{v})(\mathbf{u} - \mathbf{v})$ for all $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$.*
   (ii) *C is continuous.*

(iii) *There exists a constant* η *such that, for the smallest eigenvalue* $\lambda_{\min}(C(\mathbf{v}))$ *of* $C(\mathbf{v})$*, the following inequality holds:* $\lambda_{\min}(C(\mathbf{v})) \geq \eta > 0$*, for all* $\mathbf{v} \in \mathbb{R}^n$.

(b) $\Phi(\mathbf{u}) \leq Q(\mathbf{u}, \mathbf{v})$ *for all* $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$.

Lemma 3. *Let* $1 \leq p, q \leq 2$ *and* $\ker(\Psi^\top \Psi) \cap \ker(A^\top A) = \{\mathbf{0}\}$*. Then* $\Phi_{\beta,\gamma}(\mathbf{u})$ *defined in (8) satisfies Hypothesis 1. In particular,* $\Phi_{\beta,\gamma}(\mathbf{u})$ *has a unique minimizer.*

Proof: By the definition of $\Phi_{\beta,\gamma}$ in (8), it is obvious that $\Phi_{\beta,\gamma}$ is twice continuously differentiable and bounded from below by 0. We thus only need to prove the strict convexity and coercivity.

We start with the strict convexity. Taking derivatives on both sides of (29), we have

$$\nabla^2 \Phi_{\beta,\gamma}(\mathbf{u}) = \lambda \Psi^\top P_\beta(\mathbf{u}) \Psi + A^\top P_\gamma(\mathbf{u}) A,$$

where $P_\beta(\mathbf{u})$ and $P_\gamma(\mathbf{u}) \in \mathbb{R}^{n \times n}$ are the diagonal matrices with their ith diagonal entries being $|\Psi_i \mathbf{u}|_\beta^{p-4}(\beta + (p-1)|\Psi_i \mathbf{u}|^2)$ and $|A_i \mathbf{u} - f_i|_\gamma^{q-4}(\gamma + (q-1)|A_i \mathbf{u} - f_i|^2)$ respectively. Here recall our convention that when p or $q = 2$, the corresponding β or γ should be set to 0 because there is no need to smooth the term. Bounding each diagonal entry from below, we have

$$\nabla^2 \Phi_{\beta,\gamma}(\mathbf{u}) \succeq c_1 \Psi^\top \Psi + c_2 A^\top A,$$

where

$$c_1 =: \begin{cases} \frac{\lambda\beta}{\|\Psi\|_\infty \|\mathbf{u}\|_\infty |_\beta^{4-p}}, & 1 \leq p < 2, \\ \lambda, & p = 2, \end{cases} \qquad c_2 =: \begin{cases} \frac{\gamma}{\|\|A\|_\infty \|\mathbf{u}\|_\infty + \|\mathbf{f}\|_\infty |_\gamma^{4-q}}, & 1 \leq q < 2, \\ 1, & q = 2. \end{cases}$$

By the assumption (32), we have $\nabla^2 \Phi_{\beta,\gamma}(\mathbf{u}) \succ 0$, and the strict convexity of $\Phi_{\beta,\gamma}(\mathbf{u})$ is proven.

Next we show the coercivity. Note that $f(\cdot) = |\cdot|^p$ is convex for $p \geq 1$, which implies that

$$\frac{1}{n} \sum_{i=1}^n f(x_i) \geq f\left(\frac{1}{n} \sum_{i=1}^n x_i \right). \tag{33}$$

We rewrite $\Phi_{\beta,\gamma}$ in (8) into a summation form and then apply (33). Then we have

$$\begin{aligned} \Phi_{\beta,\gamma}(\mathbf{u}) &= \frac{\lambda}{p} \sum_{i=1}^n |\Psi_i \mathbf{u}|_\beta^p + \frac{1}{q} \sum_{i+1}^n |A_i \mathbf{u} - f_i|_\gamma^q \\ &\geq \frac{\lambda n}{p} \Big(\frac{1}{n} \sum_{i=1}^n |\Psi_i \mathbf{u}|_\beta \Big)^p + \frac{n}{q} \Big(\frac{1}{n} \sum_{i=1}^n |A_i \mathbf{u} - f_i|_\gamma \Big)^q. \end{aligned} \tag{34}$$

Next we apply the inequality $\sum_i |a_i| \geq \sqrt{\sum_i |a_i|^2}$ to (34), then we have

$$\begin{aligned}\Phi_{\beta,\gamma}(\mathbf{u}) &\geq \frac{\lambda n^{1-p}}{p}\left(\sqrt{\mathbf{u}^\top \Psi^\top \Psi \mathbf{u} + \beta n}\right)^p + \frac{n^{1-q}}{q}\left(\sqrt{(A\mathbf{u}-\mathbf{f})^\top (A\mathbf{u}-\mathbf{f}) + \gamma n}\right)^q \\ &\geq c_3 \left[\left(\sqrt{\mathbf{u}^\top \Psi^\top \Psi \mathbf{u} + \beta n}\right)^p + \left(\sqrt{(A\mathbf{u}-\mathbf{f})^\top (A\mathbf{u}-\mathbf{f}) + \gamma n}\right)^q\right], \quad (35)\end{aligned}$$

where $c_3 = \min\{\lambda n^{1-p}/p, n^{1-q}/q\}$. Now, we prove the coercivity of $\Phi_{\beta,\gamma}(\mathbf{u})$ by contradiction. Define

$$\phi_{\beta,\gamma}(\mathbf{u}) = \mathbf{u}^\top (\Psi^T \Psi + A^\top A)\mathbf{u} - 2\mathbf{f}^\top A\mathbf{u} + \|\mathbf{f}\|_2^2 + (\beta+\gamma)n.$$

Since $\Psi^\top \Psi + A^\top A \succ 0, \sigma^2 =: \lambda_{\min}(\Psi^\top \Psi + A^\top A) > 0$. Thus, if $\|\mathbf{u}\|_2 \to \infty$, we see that $\lim_{\|\mathbf{u}\|_2 \to \infty} \frac{\phi_{\beta,\gamma}(\mathbf{u})}{\|\mathbf{u}\|_2^2} \geq \sigma^2$. Hence, $\phi_{\beta,\gamma}(\mathbf{u})$ is coercive, i.e. for any $M_0 > 0$, there exists $M_1 > 0$, for any $\|\mathbf{u}\|_2 \geq M_1$, then we have $\phi_{\beta,\gamma}(\mathbf{u}) > M_0$. Suppose that $\Phi_{\beta,\gamma}(\mathbf{u})$ is non coercive, i.e.

$$\lim_{\|\mathbf{u}\|_2 \to \infty} \Phi_{\beta,\gamma}(\mathbf{u}) \neq \infty.$$

Thus, for the above M_0, for any $M_2 \geq M_1$, there exists $\|\mathbf{u}_0\|_2 \geq M_2$, but yet $\Phi_{\beta,\gamma}(\mathbf{u}_0) \leq c_3 \min\{(M_0/2)^p, (M_0/2)^q\}$. Together with (35), we have

$$\begin{aligned}\mathbf{u}_0^\top \Psi^\top \Psi \mathbf{u}_0 + \beta n &\leq \frac{M_0}{2}, \\ (A\mathbf{u}_0 - \mathbf{f})^\top (A\mathbf{u}_0 - \mathbf{f}) + \gamma n &\leq \frac{M_0}{2}.\end{aligned}$$

Summing these two inequalities up, we have $\phi_{\beta,\gamma}(\mathbf{u}_0) \leq M_0$, which is a contradiction to the coercivity of $\phi_{\beta,\gamma}(\mathbf{u})$. □

Regarding Hypothesis 2, in fact, we cannot show that Hypothesis 2 holds for arbitrary vectors $\mathbf{v}$. We can only show that it holds for $\mathbf{v} = \mathbf{u}^k$, the sequence generated by HQA. However, as we will see later in Theorem 2, it will be enough for us to prove the convergence of HQA.

Lemma 4. *Let $1 \leq p, q \leq 2$ and $\ker(\Psi^\top \Psi) \cap \ker(A^\top A) = \{\mathbf{0}\}$. Then $\Phi_{\beta,\gamma}(\mathbf{u})$ defined in (8) and $Q(\mathbf{u}, \mathbf{u}^k)$ defined in (27) satisfy Hypothesis 2 at $\mathbf{v} = \mathbf{u}^k$. In particular, the coefficient matrix of the linear system (16) is invertible.*

Proof: By definition of $Q(\mathbf{u}, \mathbf{u}^k)$ in (28), $Q(\mathbf{u}, \mathbf{u}^k)$ is quadratic in $\mathbf{u}$ and its Hessian matrix is given by

$$\nabla_1^2 Q(\mathbf{u}, \mathbf{u}^k) = \lambda \Psi^\top D_\beta(\mathbf{u}^k)\Psi + A^\top D_\gamma(\mathbf{u}^k)A, \quad (36)$$

which is independent of $\mathbf{u}$. Taking the Taylor expansion for $Q(\mathbf{u}, \mathbf{u}^k)$ at $\mathbf{u}^k$, we have

$$Q(\mathbf{u}, \mathbf{u}^k) = Q(\mathbf{u}^k, \mathbf{u}^k) + \langle \nabla_1 Q(\mathbf{u}^k, \mathbf{u}^k), \mathbf{u} - \mathbf{u}^k \rangle + \frac{1}{2}(\mathbf{u} - \mathbf{u}^k)^\top \nabla_1^2 Q(\mathbf{u}^k, \mathbf{u}^k)(\mathbf{u} - \mathbf{u}^k).$$

Since we have proven that the HQA is indeed an MMA in Lemma 2, we can replace $Q(\mathbf{u}^k, \mathbf{u}^k)$ and $\nabla_1 Q(\mathbf{u}^k, \mathbf{u}^k)$ by $\Phi_{\beta,\gamma}(\mathbf{u}^k)$ and $\nabla\Phi_{\beta,\gamma}(\mathbf{u}^k)$ respectively in the equality above and then we obtain

$$Q(\mathbf{u}, \mathbf{u}^k) = \Phi_{\beta,\gamma}(\mathbf{u}^k) + \langle \nabla\Phi_{\beta,\gamma}(\mathbf{u}^k), \mathbf{u} - \mathbf{u}^k \rangle + \frac{1}{2}(\mathbf{u} - \mathbf{u}^k)^\top \nabla_1^2 Q(\mathbf{u}^k, \mathbf{u}^k)(\mathbf{u} - \mathbf{u}^k).$$

Notice that $\{\Phi_{\beta,\gamma}(\mathbf{u}^k)\}$ is bounded from below by 0 from the definition in (8). In addition, recalling that $\{\Phi_{\beta,\gamma}(\mathbf{u}^k)\}$ is monotonically decreasing and bounded from above by $\Phi_{\beta,\gamma}(\mathbf{u}^0)$ by (31). Therefore, by coercivity, see Hypothesis 1(b), $\{\|\mathbf{u}^k\|_2\}$ must be bounded from above. Denote the bound by M. Recalling the definition of $D_\beta(\mathbf{u}^k), D_\gamma(\mathbf{u}^k)$ in (16), we have

$$\lambda_{\min}\left(\nabla_1^2 Q_1(\mathbf{u}^k, \mathbf{u}^k)\right) \geq \lambda_{\min}\left(c_4 \Psi^\top \Psi + c_5 A^\top A\right) := \eta, \tag{37}$$

where

$$c_4 = \begin{cases} \frac{\lambda}{|||\Psi||_\infty M|_\beta}, & 1 \leq p < 2, \\ \lambda, & p = 2, \end{cases} \qquad c_5 = \begin{cases} \frac{1}{|M\|A\|_\infty + \|\mathbf{f}\|_\infty|_\gamma}, & 1 \leq q < 2, \\ 1, & q = 2. \end{cases}$$

By (32), $\eta > 0$. Hypothesis 2(a)(iii) holds.

Hypothesis 2(b) is just (24), and hence is true. Finally notice that the coefficient matrix of the linear system in (16) is precisely $\nabla_1^2 Q(\mathbf{u}^k, \mathbf{u}^k)$ in (36) and hence by (37), it is invertible. □

Since Hypothesis 2 is only valid for $\mathbf{u}^k$ and not for arbitrary vectors $\mathbf{v}$, we cannot directly apply the convergence theorems in [11]. However, the proof in [11] can easily be adapted to prove the following two convergence theorems for HQA.

Theorem 2. *Let* $1 \leq p, q \leq 2$ *and* $\ker(\Psi^\top \Psi) \cap \ker(A^\top A) = \{\mathbf{0}\}$. *For the sequence* $\{\mathbf{u}^k\}$ *generated by HQA, we have*

(a) $\lim_{k\to\infty} \|\mathbf{u}^k - \mathbf{u}^{k+1}\|_2 = 0$;

(b) $\{\mathbf{u}^k\}$ *converges to the unique minimizer* $\mathbf{u}^*$ *of* $\Phi_{\beta,\gamma}(\mathbf{u})$ *from any initial guess* $\mathbf{u}^0$.

Proof:

(a) We see from (36) that $Q(\mathbf{u}, \mathbf{u}^k)$ is quadratic in $\mathbf{u}$. Taking Taylor expansion of $Q(\mathbf{u}, \mathbf{u}^k)$ at $\mathbf{u}^{k+1}$, we have

$$\begin{aligned} Q(\mathbf{u}, \mathbf{u}^k) = Q(\mathbf{u}^{k+1}, \mathbf{u}^k) &+ \langle \nabla_1 Q(\mathbf{u}^{k+1}, \mathbf{u}^k), \mathbf{u} - \mathbf{u}^{k+1} \rangle \\ &+ \frac{1}{2}(\mathbf{u} - \mathbf{u}^{k+1})^\top \nabla_1^2 Q(\mathbf{u}^k, \mathbf{u}^k)(\mathbf{u} - \mathbf{u}^{k+1}). \end{aligned} \tag{38}$$

By (23), we have $\nabla_1 Q(\mathbf{u}^{k+1}, \mathbf{u}^k) = 0$. By taking $\mathbf{u} = \mathbf{u}^k$ in (38) and using (37), we thus have

$$Q(\mathbf{u}^k, \mathbf{u}^k) \geq Q(\mathbf{u}^{k+1}, \mathbf{u}^k) + \frac{\eta}{2}\|\mathbf{u}^k - \mathbf{u}^{k+1}\|_2^2,$$

where $\eta > 0$. Together with (31), we obtain that

$$\Phi_{\beta,\gamma}(\mathbf{u}^k) - \Phi_{\beta,\gamma}(\mathbf{u}^{k+1}) \geq Q(\mathbf{u}^k, \mathbf{u}^k) - Q(\mathbf{u}^{k+1}, \mathbf{u}^k) \geq \frac{\eta}{2}\|\mathbf{u}^k - \mathbf{u}^{k+1}\|_2^2. \quad (39)$$

By Theorem 1, the convergence of $\{\Phi_{\beta,\gamma}(\mathbf{u})\}$ implies that

$$\lim_{k\to 0} \Phi_{\beta,\gamma}(\mathbf{u}^k) - \Phi_{\beta,\gamma}(\mathbf{u}^{k+1}) = 0$$

Together with (39) and $\eta > 0$, we have $\lim_{k\to\infty} \|\mathbf{u}^k - \mathbf{u}^{k+1}\|_2 = 0$.

(b) By the proof for Lemma 4, we know that the sequence $\{\|\mathbf{u}^k\|_2\}$ is bounded from above. Hence it converges to the unique minimizer $\mathbf{u}^*$ if and only if all convergent subsequences of $\{\mathbf{u}^k\}$ converge to $\mathbf{u}^*$. Let $\{\mathbf{u}^{k_j}\}$ be an arbitrary convergence subsequence of $\{\mathbf{u}^k\}$ that converges to $\bar{\mathbf{u}}$. To finish the proof for the theorem, we only need to prove that $\bar{\mathbf{u}} = \mathbf{u}^*$. Since $Q(\mathbf{u}, \mathbf{u}^{k_j})$ is quadratic in $\mathbf{u}$, we have

$$\begin{aligned} Q(\mathbf{u}, \mathbf{u}^{k_j}) = Q(\mathbf{u}^{k_j}, \mathbf{u}^{k_j}) &+ \langle \nabla_1 Q(\mathbf{u}^{k_j}, \mathbf{u}^{k_j}), \mathbf{u} - \mathbf{u}^{k_j} \rangle \\ &+ \frac{1}{2}(\mathbf{u} - \mathbf{u}^{k_j})^\top \nabla_1^2 Q(\mathbf{u}^{k_j}, \mathbf{u}^{k_j})(\mathbf{u} - \mathbf{u}^{k_j}). \end{aligned}$$

By taking the partial derivative with respect to $\mathbf{u}$ and substituting (26), we then have

$$\nabla_1 Q(\mathbf{u}, \mathbf{u}^{k_j}) = \nabla \Phi_{\beta,\gamma}(\mathbf{u}^{k_j}) + \nabla_1^2 Q(\mathbf{u}^{k_j}, \mathbf{u}^{k_j})(\mathbf{u} - \mathbf{u}^{k_j}).$$

$\nabla_1 Q(\mathbf{u}, \mathbf{v})$ is continuous since $\Phi_{\beta,\gamma}$ is twice continuously differentiable by Hypothesis 1(a), and $C(\mathbf{v}) = \nabla_1^2 Q(\mathbf{v}, \mathbf{v})$ is continuous by Hypothesis 2(a)(ii). Letting $\mathbf{u} = \mathbf{u}^{k_j+1}$ and using (23), we then have

$$0 = \nabla_1 Q(\mathbf{u}^{k_j+1}, \mathbf{u}^{k_j}) = \nabla \Phi_{\beta,\gamma}(\mathbf{u}^{k_j}) + \nabla_1^2 Q(\mathbf{u}^{k_j}, \mathbf{u}^{k_j})(\mathbf{u}^{k_j+1} - \mathbf{u}^{k_j}). \quad (40)$$

By (a), we know that $\lim_{j\to\infty} \|\mathbf{u}^{k_j+1} - \mathbf{u}^{k_j}\|_2 = 0$. This implies that $\lim_{j\to\infty} \mathbf{u}^{k_j+1} = \bar{\mathbf{u}}$. Taking limits to the both sides of (40), we obtain

$$\begin{aligned} 0 &= \lim_{j\to\infty} \nabla_1 Q(\mathbf{u}^{k_j+1}, \mathbf{u}^{k_j}) \\ &= \nabla_1 Q(\lim_{j\to\infty} \mathbf{u}^{k_j+1}, \lim_{j\to\infty} \mathbf{u}^{k_j}) \\ &= \nabla_1 Q(\bar{\mathbf{u}}, \bar{\mathbf{u}}) = \nabla \Phi_{\beta,\gamma}(\bar{\mathbf{u}}) + \nabla_1^2 Q(\bar{\mathbf{u}}, \bar{\mathbf{u}})(\bar{\mathbf{u}} - \bar{\mathbf{u}}) = \nabla \Phi_{\beta,\gamma}(\bar{\mathbf{u}}). \end{aligned}$$

By the uniqueness of the minimizer, see Lemma 3, we can conclude that $\bar{\mathbf{u}} = \mathbf{u}^*$. □

Theorem 3. *Let* $1 \leq p, q \leq 2$ *and* $\ker(\Psi^\top \Psi) \cap \ker(A^\top A) = \{\mathbf{0}\}$. *Let* $\mathbf{u}^*$ *be the unique minimizer of* $\Phi_{\beta,\gamma}(\mathbf{u})$ *and*

$$\Lambda := 1 - \lambda_{\min}\Big(\nabla_1^2 Q(\mathbf{u}^*, \mathbf{u}^*)^{-1} \nabla^2 \Phi_{\beta,\gamma}(\mathbf{u}^*)\Big).$$

Then $\Lambda < 1$ *and the sequence* $\{\Phi_{\beta,\gamma}(\mathbf{u}^k)\}$ *has a linear convergence rate of at most* Λ *while the sequence* $\{\mathbf{u}^k\}$ *is r-linearly convergent with a convergence rate of at most* $\sqrt{\Lambda}$.

To prove Theorem 3, we can follow the proof of Theorem 6.1 in [11].

4 Numerical Results

In this section, we test our algorithm on deblurring images corrupted by impulse noise and restoring MR images from few k-space data. Recall that the HQA given in (16) is for solving the smoothed ℓ_p-ℓ_q problem (8) for a fixed pair of smoothing parameters β and γ. To solve the original ℓ_p-ℓ_q problem (1), we apply the idea of continuation method on HQA for a sequence of $\{\beta^l, \gamma^l\}$ going to zero. We note that continuation methods has been used in solving TV problems before, see [9,44]. We summarize the HQA for (1) in **Algorithm 1**.

Algorithm 1 The HQA for solving (1) with $0 < p, q \leq 2$:

(1) Initialize $\beta^0, \gamma^0, \mathbf{u}^0$;
(2) For $l = 0, 1, \cdots$ until stopping criteria are met, do:
  (a) For $k = 0, 1, \cdots$ until stopping criteria are met, do:
    (i) Initialize $\mathbf{u}^{l,0} = \mathbf{u}^{l-1}$;
    (ii) Get $\mathbf{u}^{l,k+1}$ by solving

$$\Big(\lambda \Psi^\top D_{\beta^l}(\mathbf{u}^{l,k})\Psi + A^\top D_{\gamma^l}(\mathbf{u}^{l,k})A\Big)\mathbf{u} = A^\top D_{\gamma^l}(\mathbf{u}^{l,k})\mathbf{f}, \tag{41}$$

    where $D_{\beta^l}(\cdot)$ and $D_{\gamma^l}(\cdot)$ are diagonal matrices given as in (14) with $\beta = \beta^l, \gamma = \gamma^l$.
  (b) Set $\mathbf{u}^l$ to be the final solution from part (a).
  (c) Update β^l, γ^l to $\beta^{l+1}, \gamma^{l+1}$.

In all the following tests, we take the stopping criteria for the inner loop as

$$\tau := \|\nabla \Phi_{\beta^l, \gamma^l}(\mathbf{u}^{l,k})\|_2 > 0.2,$$

where $\nabla \Phi_{\beta,\gamma}(\mathbf{u})$ has been given in (29).

4.1 Numerical Results on the TV-ℓ_1 Image Restoration

In this section, we apply **Algorithm 1** to deblur images that are corrupted by impulse noise. Here, (41) is replaced by (19) with $\mathbf{u}^{l,k}$ replacing $\mathbf{u}^k$. The deblurring problem has been discussed recently in many papers, see for examples [16,41,44]. Among all these methods, the FTVDM and the ALM are the most efficient linear algorithms; and according to the numerical results in [41], ALM is the fastest one. Hence in this paper, we compare our HQA with FTVDM and ALM only. The FTVDM and ALM codes we used here are provided by the authors in [41] and we use the same parameters as in [41]. For more details on the algorithms and the parameters, please consult [41,44].

We test three 256×256 images: Barbara, Bridge and Goldhill. The matrix A is the blurring matrix corresponding to the Gaussian blur generated by the

MATLAB command

```
fspecial('Gaussian', [7 , 7], 5).
```

Then salt-and-pepper noise is added to the blurred image to obtain the observed image $\mathbf{f}$. The noise levels are taken to be 30 %, 40 %, 50 %, 60 %. For all methods, the regularization parameter λ is set to $1/13, 1/10, 1/8, 1/4$ for noise level 30 %, 40 %, 50 %, 60 % respectively. In our algorithm, we initialize $\mathbf{u}_0 = \text{rand}(\text{size}(\mathbf{f}))$. As in the FTVDM, to speed up the convergence and improve the resolution quality, we take large β, γ at the beginning and reduce them gradually to smaller ones respectively. We set β^l to be $10^{-3}, 10^{-4}, \cdots, 10^{-16}$ and $\gamma^l = (\beta^l)^2$. Equation (19) is solved by the conjugate gradient (CG) method. Considering that more iterations for CG are needed with the decreasing of β, therefore we fix the iteration number in the inner loop to be $10 \times l$ at β^l. In all tests, we consider periodic boundary condition for the difference matrix A, as it is the boundary conditions used in the tests in [41]. We compare the accuracy of the methods by the signal-to-noise ratio (SNR) used in [41]. It is defined by

$$\text{SNR} := 10 \log_{10} \frac{\|\mathbf{u} - E(\mathbf{u})\|_2^2}{\|\hat{\mathbf{u}} - \mathbf{u}\|_2^2} (\text{dB}).$$

Here $\mathbf{u}$ and $\hat{\mathbf{u}}$ denote the original image and the restored image respectively, and $E(\mathbf{u})$ is the mean gray-level value of the original image.

First we compare the speed of the three methods. Figures 1, 2 and 3 show the timing comparison of the three algorithms. Each point in the figures show the accumulated CPU time until that iteration and the corresponding SNR. The results show that our method is the fastest amongst the three methods. It is also the most accurate one.

Table 1. SNR of the restored images.

Image	Method	Salt-and-pepper noise			
		30 %	40 %	50 %	60 %
Barbara	ALM	13.93	13.35	12.45	11.37
	HQA	14.24	13.59	12.83	11.72
Bridge	ALM	11.85	10.95	10.13	8.52
	HQA	12.03	11.12	10.27	9.00
Goldhill	ALM	16.08	15.03	13.78	12.05
	HQA	16.50	15.32	14.10	12.72

From Figs. 1, 2 and 3, it is clear that FTVDM is the slowest amongst the three. In order to compare the accuracy of the two faster methods ALM and HQA more precisely, we list in Table 1 the average SNR of the recovered images in five trials by the two methods. To compare the timing fairly, we first run ALM until its stopping criteria [41] is satisfied, say with t_0 CPU seconds. Then we let

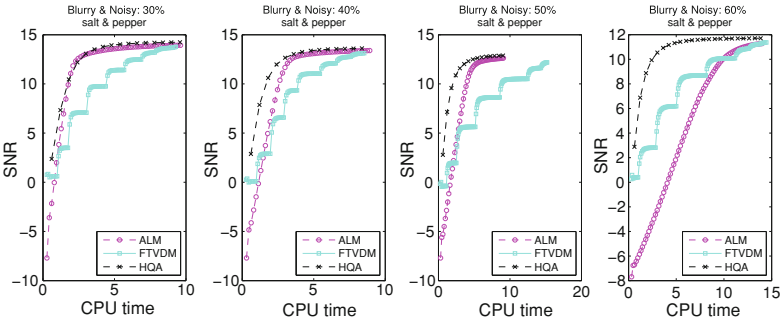


Fig. 1. SNR versus CPU time in seconds for "Barbara" with salt-and-pepper noise.

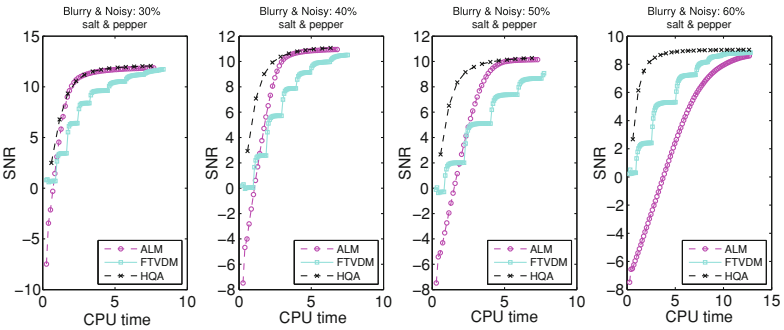


Fig. 2. SNR versus CPU time in seconds for "Bridge" with salt-and-pepper noise.

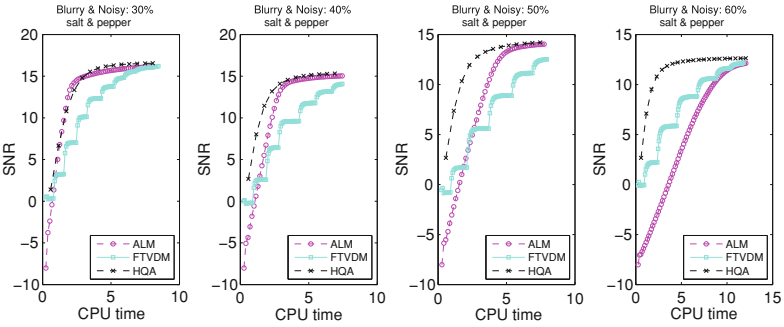


Fig. 3. SNR versus CPU time in seconds for "Goldhill" with salt-and-pepper noise.

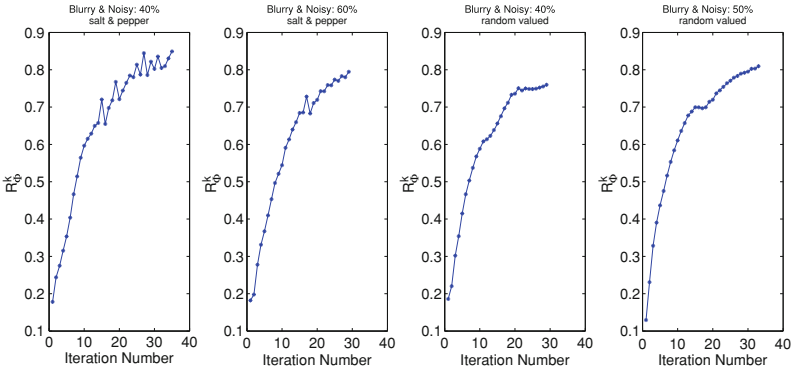


Fig. 4. The ratio $R_\Phi^k := \frac{\Phi_{\beta,\gamma}(\mathbf{u}^{k+1})-\Phi_{\beta,\gamma}(\mathbf{u}^k)}{\Phi_{\beta,\gamma}(\mathbf{u}^k)-\Phi_{\beta,\gamma}(\mathbf{u}^{k-1})}$ versus iteration number for "Barbara". The ratios are less than 1, illustrating the linear convergence of $\{\Phi_{\beta,\gamma}(\mathbf{u}^k)\}$.

HQA run until the CPU time of the kth iteration is just greater than t_0. Then we record the SNR of the $(k-1)$th iteration as our result for HQA. We see from Table 1 that HQA is more accurate than ALM. Recovered images taking Barbara for example are shown in Fig. 7 for "eyeball" illustration.

We illustrate the convergence rate of $\{\Phi_{\beta,\gamma}(\mathbf{u}^k)\}$ and $\{\mathbf{u}^k\}$ in our HQA as mentioned in Theorem 3. We use the Barbara image as example. Since we do not have the true minimizer, we use

$$R_\Phi^k =: [\Phi_{\beta,\gamma}(\mathbf{u}^{k+1}) - \Phi_{\beta,\gamma}(\mathbf{u}^k)]/[\Phi_{\beta,\gamma}(\mathbf{u}^k) - \Phi_{\beta,\gamma}(\mathbf{u}^{k-1})]$$

and

$$R_\mathbf{u}^k =: [\|\mathbf{u}^{k+1} - \mathbf{u}^k\|_2/\|\mathbf{u}^k - \mathbf{u}^{k-1}\|_2]$$

to estimate the convergence rate for $\{\Phi_{\beta,\gamma}(\mathbf{u}^k\}$ and $\{\mathbf{u}^k\}$ respectively. In Fig. 4, we plot the ratio R_Φ^k against the iteration number. We see that the ratios are

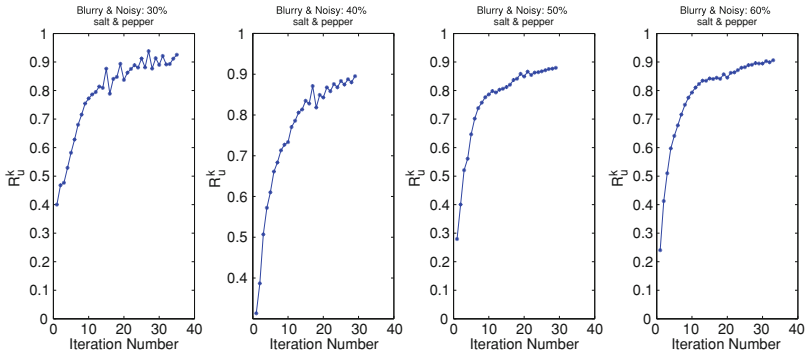


Fig. 5. The ratio $R_\mathbf{u}^k = \frac{\|\mathbf{u}^{k+1}-\mathbf{u}^k\|_2}{\|\mathbf{u}^k-\mathbf{u}^{k-1}\|_2}$ versus iteration number for "Barbara". The ratios are less than 1, illustrating the linear convergence of $\{\mathbf{u}^k\}$.

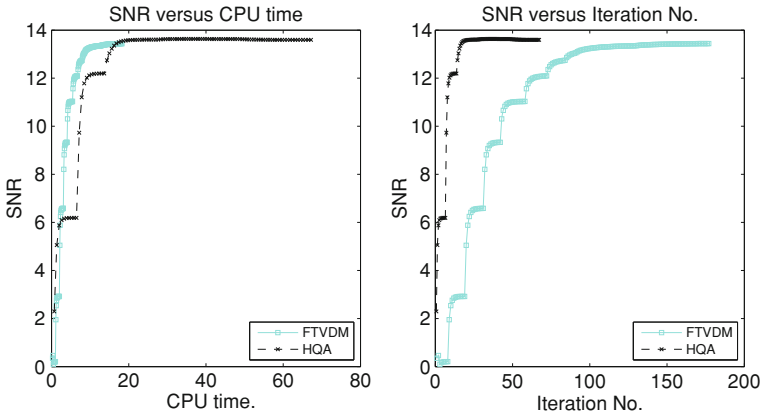


Fig. 6. The figure shows the comparison results of the FTVDM and the HQA for Barbara image with salt-and-pepper noise removal at noise level 40 %. For HQA, in (18), $\beta = 10^{-1}, 10^{-2}, \cdots, 10^{-17}$, and correspondingly $\gamma = \beta^2$. For FTVDM, $\theta_{\mathbf{w}} = 1, 2^{2/3}, \cdots, 2^{10}$, correspondingly, $\theta_{\mathbf{z}} = 1, 2, \cdots, 2^{15}$. At each jump, $\beta_k, \gamma_k, \theta_{\mathbf{w}}^k, \theta_{\mathbf{z}}^k$ jump to $\beta_{k+1}, \gamma_{k+1}, \theta_{\mathbf{w}}^{k+1}, \theta_{\mathbf{z}}^{k+1}$ in HQA and FTVDM.

all less than 1, indicating that $\{\Phi_{\beta,\gamma}(\mathbf{u}^k)\}$ is linearly convergent as stated in Theorem 3. In Fig. 5, we plot $R_{\mathbf{u}}^k$ against the iteration number. We see that $R_{\mathbf{u}}^k < c$ (c is a positive constant less than 1), indicating that $\{\mathbf{u}^k\}$ indeed is linearly convergent.

In [44], the original energy functional (17) is a non-differentiable functional of $\mathbf{u}$, hence auxiliary variables $\mathbf{w}, \mathbf{z}$ and regularization parameters $\theta_{\mathbf{w}}$, $\theta_{\mathbf{z}}$ are taken to approximate (17). The approximated problem is

$$\min_{\mathbf{w},\mathbf{z},\mathbf{u}} \{\lambda(\||\mathbf{w}|\|_1 + \frac{\theta_{\mathbf{w}}}{2}\|\mathbf{w} - \nabla\mathbf{u}\|_2^2) + \|\mathbf{z}\|_1 + \frac{\theta_{\mathbf{z}}}{2}\|\mathbf{z} - (B\mathbf{u} - \mathbf{f})\|_2^2\}$$

The parameters $\theta_{\mathbf{w}}$ and $\theta_{\mathbf{z}}$ are upper limited to be $\theta_{\mathbf{w}} = 2^{10}, \theta_{\mathbf{z}} = 2^{15}$ in the approximate problem. To speed up the convergence, $\theta_{\mathbf{w}}$ and $\theta_{\mathbf{z}}$ are both implemented in a continuous scheme; that is, let $\theta_{\mathbf{w}}$ and $\theta_{\mathbf{z}}$ take small values at the beginning and gradually increase their values to 2^{10} and 2^{15} respectively. Specially, a $\theta_{\mathbf{w}}$-sequence $2^0, 2^{2/3}, 2^{4/3}, \cdots, 2^{10}$ is tested. Accordingly, $\theta_{\mathbf{z}}$ is set to be $2^0, 2^1, 2^2, \cdots, 2^{15}$.

A similar continuation scheme is also taken in our HQA. We take $\beta = 10^{-1}, 10^{-2}, \cdots, 10^{-17}$, correspondingly, $\gamma = \beta^2$ and compare the FTVDM and the HQA. Figure 6 shows their comparison results on the SNR versus CPU time and SNR versus iteration number. The jump shows the improvements in SNR as $\theta_{\mathbf{w}}^k, \theta_{\mathbf{z}}^k, \beta^k, \gamma^k$ change to $\theta_{\mathbf{w}}^{k+1}, \theta_{\mathbf{z}}^{k+1}, \beta^{k+1}, \gamma^{k+1}$.

Fig. 7. Restored images.

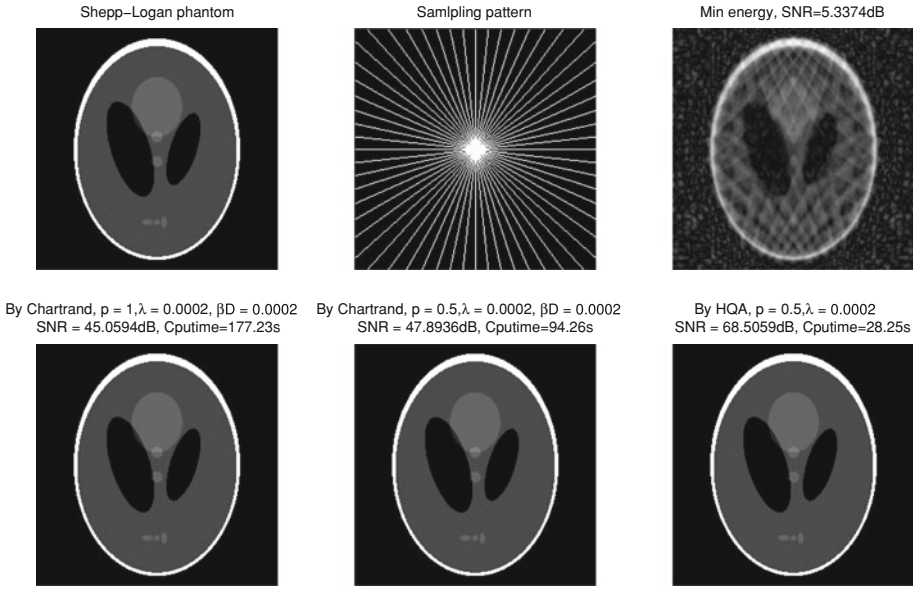


Fig. 8. The figure shows the reconstruction for the 256×256 Shepp-Logan phontom. Top (left): the original Shepp-Logan image; Top (middle): the 30 radial lines on the white pixels (11.32 % sampling); Top (right): the backprojection reconstruction with 30 views, which is poor; Bottom (left): the reconstruction from 30 views by p-shrinkage algorithm with $p = 1$; Bottom (middle): the reconstruction by the p-shrinkage algorithm with $p = 1/2$; Bottom (right): the reconstruction by HQA with $p = 1/2$.

4.2 Numerical Results on the MR Image Reconstruction

In this section, we apply **Algorithm 1** to reconstruct MR image from few k-space data. Here (41) is replaced by (22) with $\mathbf{u}^{l,k}$ replacing $\mathbf{u}^k$. To test the efficiency of our algorithm, we compare our algorithm and the p-shrinkage algorithm by Chartrand in [12]. As in [12], we take $p = 1/2$. In addition, we also give the numerical results by ℓ_1-norm regularized model for comparison. We test our algorithm on the two images: 256×256 Shepp-Logan phantom and 224×224 Brain image. In all the tests, we set β to be $10^{-4}, 10^{-5}, \cdots, 10^{-14}$, and $\gamma = \beta$ correspondingly. Moreover, we just use the simple CG method to solve the corresponding linear system (22).

We begin with the Shepp-Logan phantom. As in [12], because the phantom has a very sparse gradient, we do not use the wavelet regularization, and let $\delta = 0$ in (20). We show the comparison results on the MR image reconstruction from 10 radial lines (3.85 % sampling), 22 radial lines (8.36 % sampling), and 30 radial lines (11.32 % sampling) respectively. In all the three tests, we take $\lambda = 0.0002$. When $p = 1$, the p-shrinkage [12] is actually the soft-thresholding. The results are shown in Figs. 8, 9 and 10. In all three figures, we see that our HQA can reach better reconstruction (at least 13–16 dB better) using less

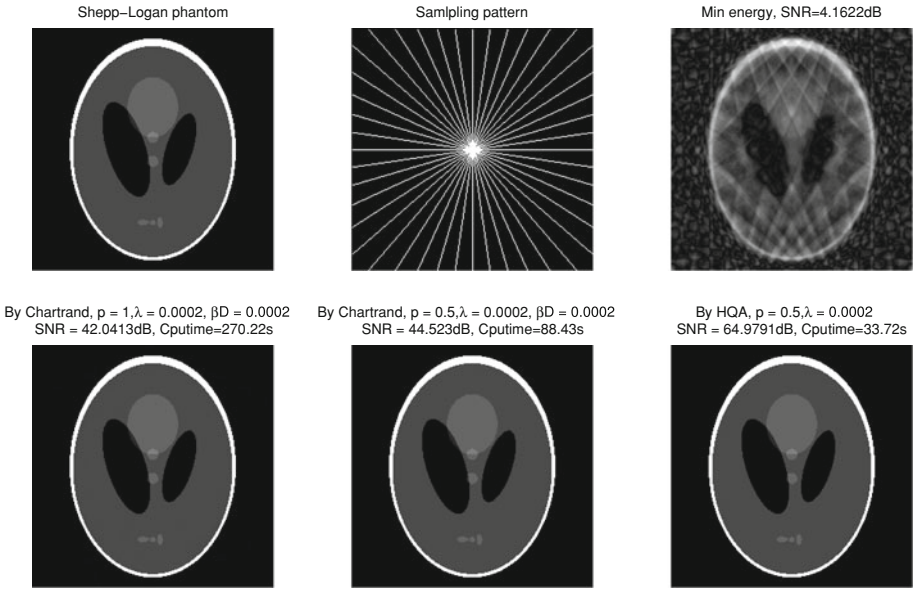


Fig. 9. The figure shows the reconstruction for the 256×256 Shepp-Logan phontom. Top (left): the original Shepp-Logan image; Top (middle): the 22 radial lines on the white pixels (8.36 % sampling); Top (right): the backprojection reconstruction with 22 views, which is poor; Bottom (left): the reconstruction from 22 views by p-shrinkage algorithm with $p = 1$; Bottom (middle): the reconstruction by the p-shrinkage algorithm with $p = 1/2$; Bottom (right): the reconstruction by HQA with $p = 1/2$.

computational time (at least 1/2 of the time) than the p-shrinkage algorithm. Among all the results, the ℓ_1-norm regularization model takes the most time to obtain a suitable reconstruction image, especially when the k-space data are very few.

Next, we apply our algorithm to recover the MR brain image in the presence of noise. We set that the image is corrupted by the white Gaussian noise with noise level $\sigma = 5$. Here, by error and trials, we take $\lambda = 0.002$. Our results show that the recovered images have higher quality by the $\|\nabla \mathbf{u}\|_p^p$ regularization model than by the $\|\nabla \mathbf{u}\|_p^p + \|W\mathbf{u}\|_p^p$. Hence, here, we show the recovered results from (22) with $\delta = 0$. The comparison results are shown in Fig. 11. For the brain image, we take 40 views (16.97 % sampling). The results show that our HQA can reach the best reconstruction with the clearest background in the least amount of time.

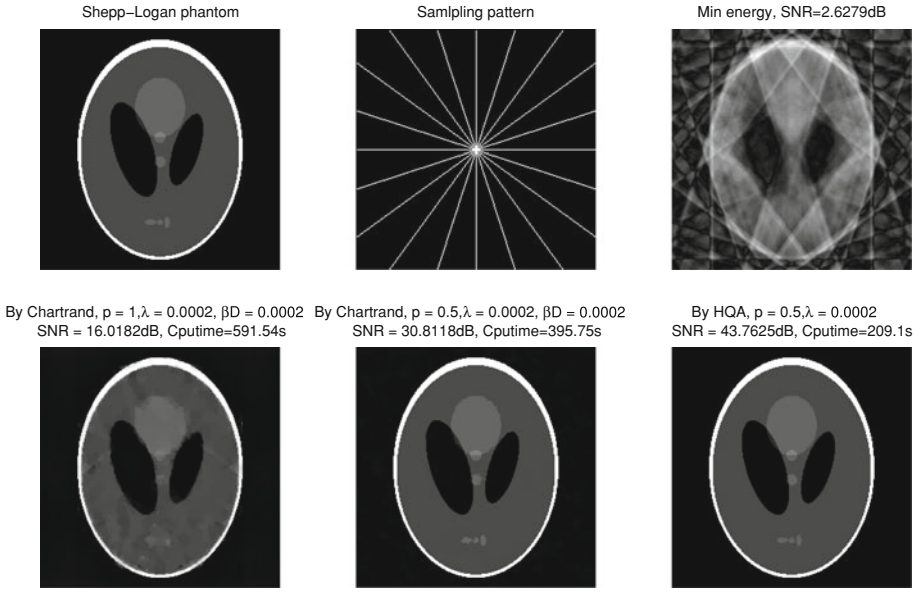


Fig. 10. The figure shows the reconstruction for the 256×256 Shepp-Logan phontom. Top (left): the original Shepp-Logan image; Top (middle): the 10 radial lines on the white pixels (3.85 % sampling); Top (right): the backprojection reconstruction with 10 views, which is poor; Bottom (left): the reconstruction from 10 views by p-shrinkage algorithm with $p = 1$; Bottom (middle): the reconstruction by the p-shrinkage algorithm with $p = 1/2$; Bottom (right): the reconstruction by HQA with with $p = 1/2$. From the results, we find that it will take more time to reach a good reconstruction from fewer k-space data. By ℓ_1-norm regularized model, it is still difficult to obtain a good result even with much time, while the ℓ_p-norm regularized models ($0 < p < 1$) do.

5 Conclusion

In this paper, we study the half-quadratic technique for ℓ_p-norm and propose an algorithm for solving ℓ_p-ℓ_q ($0 < p, q \leq 2$) minimization problem. We show that the algorithm for the related minimization problem with regularization smoothing parameters β and γ is equivalent to a majorize-minimize algorithm. Weak convergence result for $0 < p$ or $q < 1$, and linear convergence rate for $1 \leq p, q \leq 2$ are obtained immediately. We will consider the convergence of the original non-smooth problems in our future work. We compare our algorithm with standard ones in the TV-ℓ_1 minimization problem and the MR image reconstruction. The results show that our algorithm can reach better reconstruction results with less computational cost.

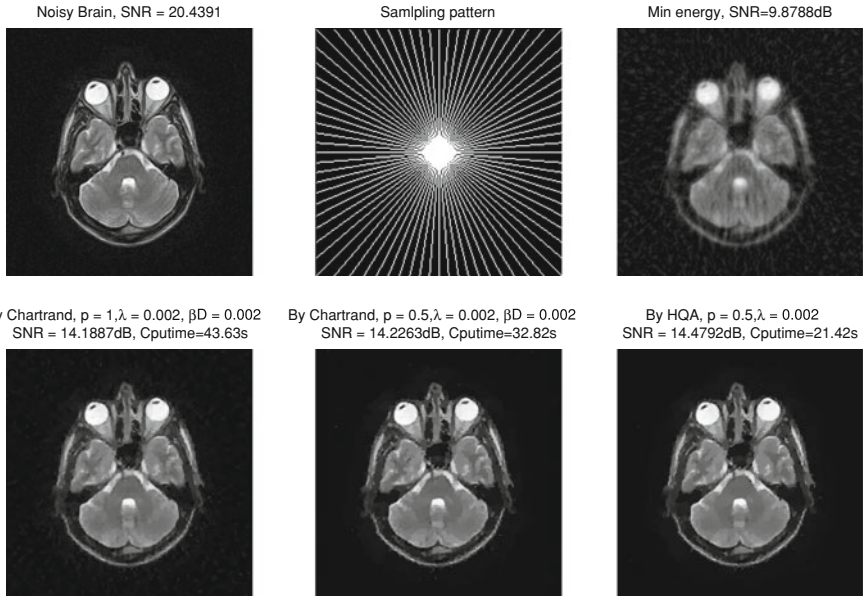


Fig. 11. The figure shows the reconstruction for the 224×224 real brain image. Top (left): the noisy brain image with noise level $\sigma = 5$, which is generated with the Matlab command: "$\texttt{imnoise}(\mathbf{x}, '\,\texttt{gaussian}', 0, \sigma^2)$"; Top (middle): the 40 radial lines on the white pixels (16.97 % sampling); Top (right): the backprojection reconstruction with 40 views, which is poor; Bottom (left): the reconstruction from 40 views by p-shrinkage algorithm with $p = 1$; Bottom (middle): the reconstruction by the p-shrinkage algorithm with $p = 1/2$; Bottom (right): the reconstruction by HQA with $p = 1/2$.

Acknowledgement. The authors would like to thank the financial support of project in Nanyang Technological University.

References

1. Amaldi, E., Kann, V.: On the approximability of minimizing nonzero variables or unsatisfied relations in linear system. Theoretical Comp. Sci. **209**, 237–260 (1998)
2. Baraniuk, R., Steeghs, P.: Compressive radar imaging. Radar Conference, 2007 IEEE, pp. 128–133 (2007)
3. Berger, C.R., Areta, J., Pattipati, K., Willett, P.: Compressed sensing - a look beyond linear programming. In: 33rd International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 3857–3860 (2008)
4. Blake, A., Zisserman, A.: Visual reconstruction and the GNC algorithm. In: Parallel Architectures and Computer Vision, pp. 33–48 (1988)
5. Cai, J.F., Osher, S., Shen, Z.: Linearized Bregman iterations for compressed sensing. In: UCLA CAM Report, 08-06 (2008)
6. Candè, E.J., Romberg, J.: Signal recovery from random projections. Proc. SPIE Comput. Imaging III **5674**, 76–86 (2005)

7. Candè, E.J., Romberg, J., Tao, T.: Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. IEEE Trans. Inform. Theory **52**, 489–509 (2006)
8. Chan, R.H., Chan, T., Shen, L., Shen, Z.: Wavelet algorithms for high-resolution image reconstruction. SIAM J. Sci. Comput. **24**, 1408–1432 (2003)
9. Chan, R.H., Chan, T.F., Zhou, H.M.: Continuation method for total variation denoising problems. In: UCLA CAM Report 95-18 (1995)
10. Chan, T., Esedoglu, S.: Aspects of total variation regularized L^1 function approximation. SIAM J. Appl. Math. **65**, 1817–1837 (2005)
11. Chan, T.F., Mulet, P.: On the convergence of the lagged diffusivity fixed point method in total variation image restoration. SIAM J. Numer. Anal. **36**, 354–367 (1999)
12. Chartrand, R.: Fast algorithms for non-convex compressive sensing: MRI reconstruction from very few data. In: IEEE International Symposium on Biomedical Imaging (ISBI) (2009)
13. Chartrand, R.: Exact reconstruction of sparse signals via non-convex minimization. IEEE Signal Process. Lett. **14**, 707–710 (2007)
14. Chartrand, R., Yin, W.: Iteratively reweighted algorithms for compressive sensing. In: 33rd International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 3869–3872 (2008)
15. Chartrand, R., Staneva, V.: Restricted isometry properties and non-convex compressive sensing. Inverse Problems **24**, 1–14 (2008)
16. Dong, Y.Q., Hintermüller, M., Neri, M.: An efficient primal-dual method for L^1-TV image restoration. SIAM J. Imaging Sci. **2**, 1168–1189 (2009)
17. Donoho, D.: Compressed sensing. IEEE Trans. Inf. Theory **52**, 1289–1306 (2006)
18. Ekerland, I., Témam, R.: Convex Analysis and Variational Problems. SIAM, Philadelphia (1999)
19. Geman, D., Yang, C.: Nonlinear image recovery with half-quadratic regularization and FFTs. IEEE Trans. Imaging Proc. **4**, 932–946 (1995)
20. Goldstein, T., Osher, S.: The split Bregman L1 regularized problems. SIAM J. Imaging Sci. **2**, 323–343 (2009)
21. He, L., Chang, T.C., Osher, S..: MR image reconstruction from sparse radial samples by using iterative refinement procedures. In: Proceeding of the 13th Annual Meeting of ISMRM, p. 696 (2006)
22. Hestenes, M.R.: Multiplier and gradient methods. J. Optim. Theor. Appl. **4**, 303–320 (1969)
23. Hyder, M., Mahata, K.: An approximate L0 norm minimization algorithm for compressed sensing. In: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3365–3368 (2009)
24. Jacobson, M., Fessler, J.: An expanded theoretical treatment of iteration-dependent majorize-minimize algorithms. IEEE Trans. Image Proc. **16**, 2411–2422 (2007)
25. Jacobson, M., Fessler, J.: Properties of MM algorithms on convex feasible sets: extended version. Technical Report 353, Communications and Signal Processing Laboratory, Department of EECS, University of Michigan, Ann Arbor, MI, 48109–2122 (2004)
26. Koko, J., Jehan-Besson, S.: An augmented Lagrangian method for TVg+L^1-norm minimization. Research Report LIMOS/RR-09-07 (2009)
27. Lusting, M., Donoho, D., Pauly, J.M.: Sparse MRI: the application of compressed sensing for rapid MR imaging. Magn. Reson. Med. **58**, 1182–1195 (2007)

28. Lusting, M., Lee, J.H., Donoho, D.L., Pauly, J.M.: Faster imaging with randomly perturbed undersampled spirals and L1 reconstruction. In: Proceedings of the ISMRM (2005)
29. Mohimani, G.H., Babaic-Zadeh, M., Jutten, C.: Fast sparse representation based on smoothed ℓ^0 norm. In: Davies, M.E., James, C.J., Abdallah, S.A., Plumbley, M.D. (eds.) ICA 2007. LNCS, vol. 4666, pp. 389–396. Springer, Heidelberg (2007)
30. Mancera, L., Portilla, J.: L0-norm-based sparse representation through alternate projections. In: International Conference on Image Processing, pp. 2089–2092. IEEE (2006)
31. Nikolova, M., Chan, R.H.: The equivalence of the half-quadratic minimization and the gradient linearization iteration. IEEE Trans. Image Proc. **16**, 5–18 (2007)
32. Osher, S., Mao, Y., Dong, B., Yin, W.: Fast linearized Bregman iterations for compressed sensing and sparse denoising. In: UCLA CAM Report, 08–37 (2008)
33. Powell, M.J.D.: A method for nonlinear constraints in minimization problems. In: Fletcher, R. (ed.) Optimization, pp. 283–298. Academic Press, New York (1972)
34. Qu, X.B., Cao, X., Guo, D., Hu, C., Chen, Z.: Compressed sensing MRI with combined sparsifying transforms and smoothed L0 norm minimization. In: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 626–629 (2010)
35. Rao, B.D., Kreutz-Delgado, K.: An affine scaling methodology for best basis selection. IEEE Trans. Signal Process **47**, 187–200 (1999)
36. Rockafellar, R.T.: A dual approach to solving nonlinear programming problems by unconstrained optimization. Math. Program. **5**, 354–373 (1973)
37. Rudin, L., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Physica D. **60**, 259–268 (1992)
38. Saab, R., Chartrand, R., Yilmaz, Ö.: Stable sparse approximations via non-convex optimization. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (2008)
39. Vogel, C.R., Oman, M.E.: Fast, robust total variation-based reconstruction of noisy blurred images. IEEE Trans. Image Proc. **7**, 813–824 (1998)
40. Weiszfeld, E.: Sur le point pour lequel la somme des distances de n points donnés est minimum. Tôhoku Math. J. **43**, 355–386 (1937)
41. Wu, C.L., Zhang, J.Y., Tai, X.C.: Augmented Lagrangian method for total variation restoration with non-quadratic fidelity. Inverse Prob. Imaging **5**, 237–261 (2010)
42. Xu, Z.B.: Data modeling: visual psychology approach and $l_{1/2}$ regularization theory. In: Proceedings of the International Congress of Mathematicians, Hyderabad, India (2010)
43. Xu, Z.B., Chang, X.Y., Xu, F.M., Zhang, H.: $L_{1/2}$ regularization: an iterative half thresholding algorithm. Technique Report (2010)
44. Yang, J.F., Zhang, Y., Yin, W.T.: An efficient TVL1 algorithm for deblurring multichannel images corrupted by impulsive noise. SIAM J. Sci. Comput. **31**, 2842–2865 (2009)
45. Yin, W.T., Goldfarb, D., Osher, S.: Image cartoon-texture decomposition and feature selection using the total variation regularized L^1 functional. In: Paragios, N., Faugeras, O., Chan, T., Schnörr, C. (eds.) VLSM 2005. LNCS, vol. 3752, pp. 73–84. Springer, Heidelberg (2005)

46. Yin, W.T., Goldfarb, D., Osher, S.: The total variation regularized L^1 model for multiscale decomposition. Multis. Model. Simul. **6**, 190–211 (2006)
47. Yin, W.T., Osher, S., Goldfarb, D., Darbon, J.: Bregman iteration algorithms for l1-minimization with applications to compressed sensing. SIAM J. Imaging Sci. **1**, 142–168 (2008)

A Fast Algorithm for a Mean Curvature Based Image Denoising Model Using Augmented Lagrangian Method

Wei Zhu[1], Xue-Cheng Tai[2(✉)], and Tony Chan[3]

[1] Department of Mathematics, University of Alabama, 870350, Tuscaloosa, AL 35487, USA
wzhu7@bama.ua.edu

[2] Department of Mathematics, University of Bergen, 5007, Bergen, Norway
tai@math.uib.no

[3] Office of the President, Hong Kong University of Science and Technology (HKUST), Clear Water Bay, Kowlon, Hong Kong
tonyfchan@ust.hk

Abstract. Recently, many variational models using high order derivatives have been proposed to accomplish advanced tasks in image processing. Even though these models are effective in fulfilling those tasks, it is very challenging to minimize the associated high order functionals. In [33], we focused on a recently proposed mean curvature based image denoising model and developed an efficient algorithm to minimize it using augmented Lagrangian method, where minimizers of the original high order functional can be obtained by solving several low order functionals. Specifically, these low order functionals either have closed form solutions or can be solved using FFT. Since FFT yields exact solutions to the associated equations, in this work, we consider to use only approximations to replace these exact solutions in order to reduce the computational cost. We thus employ the Gauss-Seidel method to solve those equations and observe that the new strategy produces almost the same results as the previous one but needs less computational time, and the reduction of the computational time becomes salient for images of large sizes.

1 Introduction

Image denoising is to remove noise while keeping meaningful vision information such as object edges and boundaries. It is a crucial step in image processing with a wide range of applications in medical image analysis, video monitoring, and others. During the last three decades, numerous models have been proposed to deal with this problem [3,4,7,19–21,23–25,28]. One of the most popular variational models was proposed by Rudin, Osher, and Fatemi in their seminal work (ROF model) [25], where the cleaned image corresponds to the minimizer of the following functional

$$E(u) = \lambda \int_{\Omega} |\nabla u| + \int_{\Omega} (f-u)^2, \tag{1}$$

A. Bruhn et al. (Eds.): Global Optimization Methods, LNCS 8293, pp. 104–118, 2014.
DOI: 10.1007/978-3-642-54774-4_5,

where $f : \Omega \to \mathbb{R}$ is a given noisy image defined on Ω (always a rectangle in $\mathbb{R}^2$) and $\lambda > 0$ is a positive tuning parameter controlling how much noise will be removed. The remarkable feature of the ROF model lies in its effectiveness in preserving object edges while removing noise. This is due to the total variation based regularizer. In fact, the total variation has been widely employed in accomplishing other image tasks such as deblurring, segmentation, and registration. However, as pointed out in [6], the ROF model has several unfavorable features. The main caveat is the stair case effect, that is, the resulting clean image would present blocks even though the desired image could be smooth, such as human face. Other undesirable properties include corner smearing and loss of image contrast. To remedy these drawbacks, quite a few high order variational models have been proposed [1,2,9,16–18,31]. Despite of the effectiveness of these models in removing the staircase effect, it is often a challenging issue to minimize the corresponding functionals. Note that the models contain second order derivatives, the related Euler-Lagrange equations are fourth-order, which raises a nontrivial problem of developing effective and efficient algorithms to solve them. Indeed, as more and more high order models were used in image processing [5,7,8,12–15,22,27,30,32], it is an imperative need to explore efficient numerical algorithms for these models.

Recently, augmented Lagrangian methods have been successfully employed in the minimization of nondifferentiable or high order functionals [26,29]. For instance, the minimization of the ROF model suffers from the presence of its nonlinear and nondifferentiable term. In [29], the authors proposed an efficient and accurate algorithm using augmented Lagrangian method to minimize the ROF functional. In [26], the technique was extended to functionals related to Euler's elastica with applications in image denoising, inpainting, and zooming [1,2,8,17,18], where the original minimization problem was converted to several subproblems of low order functionals. Therefore, augmented Lagrangian method becomes a suitable technique to handle curvature related functionals.

Inspired by these works, in [33], we constructed an augmented Lagrangian method based fast algorithm for the mean curvature based image denoising model [31], whose functional can be expressed as follows:

$$E(u) = \lambda \int \left| \nabla \cdot \left(\frac{\nabla u}{\sqrt{1 + |\nabla u|^2}} \right) \right| + \frac{1}{2} \int (f - u)^2, \tag{2}$$

where λ is a tuning parameter and the term $\nabla \cdot \left(\frac{\nabla u}{\sqrt{1+|\nabla u|^2}} \right)$ is the mean curvature of the surface $\phi(x, y, z) = u(x, y) - z = 0$ (see [10]). The model tries to fit the given noisy image surface $(x, y, f(x, y))$ with a surface $(x, y, u(x, y))$ that bears small magnitude of mean curvature. As demonstrated in [31], the model is able to sweep noise while keeping object edges, and it also ameliorates the staircase effect. More importantly, the model is also capable of preserving image contrasts as well as geometry of object shapes, especially object corners. As detailed in [33], by using augmented Lagrangian method, the original minimization problem was reformulated as a constrained optimization, and with the

specially designed constraints, the pursuit of saddle points of the optimization problem amounts to minimizing several functionals alternatively, some of which have closed form solutions while the other ones can be solved using FFT. Note that FFT yields exact solutions to those equations, in this work, we want to check whether these exact solutions in each iteration can be replaced by some approximations in order to further reduce the computational cost.

This paper is organized as follows. In Sect. 2, we present a short review of the mean curvature denoising model and recall the augmented Lagrangian method developed for this model [33]. Section 3 presents the details of the minimization of the associated subproblems. In Sect. 4, numerical results obtained using the two methods as well as the efficiency will be compared, which is then followed by a conclusion in Sect. 5.

2 The Image Denoising Model and the Augmented Lagrangian Method

In this section, we first review the mean curvature based image denoising model and then sketch the augmented Lagrangian method developed for this model [33].

2.1 The Mean Curvature Based Image Denoising Model

For a given image $f : \Omega \to \mathbb{R}$ defined on a domain $\Omega \subset \mathbb{R}^2$, one can regard it as a surface $(x, y, f(x, y))$ in $\mathbb{R}^3$. The denoising task amounts to finding a piecewisely smooth surface that approximates that noisy surface while also keeping its sharp gradients, since these sharp transitions determine important vision clues such as edges and corners. To single out those piecewisely smooth surfaces, one needs to choose an appropriate regularizer. In [31], the L^1-norm of mean curvature of an image surface was employed as the regularizer.

Specifically, let $u : \Omega \to \mathbb{R}$ be a function. Its image surface is just the zero level set of the function $\phi(x, y, z) = u(x, y) - z$. Then the mean curvature of this surface can be expressed as $\nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|} \right) = \nabla \cdot \left(\frac{\nabla u}{\sqrt{1+|\nabla u|^2}} \right)$, denoted by κ_u. Using the L^1-norm of mean curvature as the regularizer, the mean curvature denoising model can be written as the minimization of the following functional:

$$E(u) = \lambda \int |\kappa_u| + \frac{1}{2} \int (f - u)^2, \tag{3}$$

which gives the Eq. (2).

Due to the non-differentiable term in this functional, numerically, one often considers its regularized version [22,32]

$$E(u) = \lambda \int \Phi(\kappa_u) + \frac{1}{2} \int (f - u)^2, \tag{4}$$

with

$$\Phi(x) = \begin{cases} x^2, & |x| \leq 1 \\ |x|, & |x| > 1, \end{cases} \tag{5}$$

and derives the following fourth order Euler-Lagrange equation:

$$\lambda\nabla\cdot[\frac{1}{\sqrt{1+|\nabla u|^2}}(\mathbf{I}-\mathbf{P})\nabla\Phi'(\kappa_u)]-(f-u)=0, \tag{6}$$

where $\mathbf{I},\mathbf{P}:\mathbb{R}^2\to\mathbb{R}^2$ with $\mathbf{I}(\mathbf{x})=\mathbf{x}$ and $\mathbf{P}(\mathbf{x})=\left(\mathbf{x}\cdot\frac{\nabla u}{\sqrt{1+|\nabla u|^2}}\right)\frac{\nabla u}{\sqrt{1+|\nabla u|^2}}$. This equation is often solved by considering the steady state of the following time-dependent one:

$$\frac{\partial u}{\partial t}=-\lambda\nabla\cdot\left[\frac{1}{\sqrt{1+|\nabla u|^2}}(\mathbf{I}-\mathbf{P})\nabla\Phi'(\kappa_u)\right]+(f-u), \tag{7}$$

with time t being an evolution parameter.

The above modification surely affects the outcome of the model. However, as shown later, by using augmented Lagrangian method, the functional (2) can be exactly treated. This is one of the most important merits of augmented Lagrangian method, which has been shown in the treatment of the non-differentiable total variation norm of the ROF model in [29].

2.2 Augmented Lagrangian Method

In [33], following a similar idea for treating Euler's elastica based functionals [26], we converted the minimization of functional (Eq. 2) to be the following constrained problem;

$$\begin{aligned}&\min_{u,q,\mathbf{n},\mathbf{p}}\left[\lambda\int_\Omega|q|+\frac{1}{2}\int(f-u)^2\right],\\&\text{with } q=\nabla\cdot\mathbf{n},\ \mathbf{n}=\frac{\mathbf{p}}{|\mathbf{p}|},\ \mathbf{p}=\langle\nabla u,1\rangle,\end{aligned} \tag{8}$$

and developed the associated augmented Lagrangian functional:

$$\begin{aligned}\mathcal{L}(u,q,\mathbf{p},\mathbf{n},\mathbf{m};\lambda_1,\boldsymbol{\lambda}_2,\lambda_3,\boldsymbol{\lambda}_4)=\lambda\int|q|+\frac{1}{2}\int(f-u)^2\\+r_1\int(|\mathbf{p}|-\mathbf{p}\cdot\mathbf{m})+\int\lambda_1(|\mathbf{p}|-\mathbf{p}\cdot\mathbf{m})\\+\frac{r_2}{2}\int|\mathbf{p}-\langle\nabla u,1\rangle|^2+\int\boldsymbol{\lambda}_2\cdot(\mathbf{p}-\langle\nabla u,1\rangle)\\+\frac{r_3}{2}\int(q-\partial_x n_1-\partial_y n_2)^2+\int\lambda_3(q-\partial_x n_1-\partial_y n_2)\\+\frac{r_4}{2}\int|\mathbf{n}-\mathbf{m}|^2+\int\boldsymbol{\lambda}_4\cdot(\mathbf{n}-\mathbf{m})+\delta_{\mathcal{R}}(\mathbf{m}),\end{aligned} \tag{9}$$

where $\mathbf{n},\mathbf{m},\mathbf{p}\in\mathbb{R}^3$ are auxiliary vectors and $\lambda_1,\lambda_3\in\mathbb{R}$, $\boldsymbol{\lambda}_2,\boldsymbol{\lambda}_4\in\mathbb{R}^3$ are Lagrange multipliers. Note that in this Lagrangian functional, we used $\mathbf{p}=\langle\nabla u,1\rangle$ instead of $\mathbf{p}=\nabla u$ as in [26]. This substitution was chosen purposely to

treat the mean curvature term since it is nonhomogeneous in u. Just as in [26], the introduction of the variable $\mathbf{m}$ is to relax the variable $\mathbf{n}$ that is supposed to connect with the variable $\mathbf{p}$ in terms of $\mathbf{n} = \mathbf{p}/|\mathbf{p}|$, and the variable $\mathbf{m}$ is required to lie in the set $\mathcal{R} = \{\mathbf{m} \in L^2(\Omega) : |\mathbf{m}| \leq 1 \text{ a.e. in } \Omega\}$ through the characteristic function $\delta_{\mathcal{R}}(\cdot)$ defined as follows:

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

so that the term $|\mathbf{p}| - \mathbf{p} \cdot \mathbf{m}$ is always non-negative. The benefit of this non-negativeness is that the L^2 penalization is unnecessary and we just use $|\mathbf{p}| - \mathbf{p} \cdot \mathbf{m}$ as a penalization, which simplifies the associated subproblems when finding saddle points of the above Lagrangian functional (Eq. 9).

Note that saddle points of the functional (9) correspond to minimizers of the constrained minimization problem (8), and equivalently, minimizers of the mean curvature model (2), one just needs to find saddle points of (9). To this end, as in [26], we apply an iterative algorithm. Specifically, for each variable in (9), we fix all the other variables and seek a critical point of the induced functional to update this variable. Once all the variables are updated, the Lagrange multipliers will then be advanced accordingly. Then we repeat this process until the variables converge to steady state.

To find saddle points of the functional (9), we consider the following subproblems and need to obtain minimizers for all of them.

$$\varepsilon_1(u) = \frac{1}{2}\int (f-u)^2 + \frac{r_2}{2}\int |\mathbf{p} - \langle \nabla u, 1\rangle|^2 + \int \boldsymbol{\lambda}_2 \cdot (\mathbf{p} - \langle \nabla u, 1\rangle), \tag{10}$$

$$\varepsilon_2(q) = \lambda \int |q| + \frac{r_3}{2}\int (q - \partial_x n_1 - \partial_y n_2)^2 + \int \lambda_3 (q - \partial_x n_1 - \partial_y n_2), \tag{11}$$

$$\begin{aligned} \varepsilon_3(\mathbf{p}) = {} & r_1 \int (|\mathbf{p}| - \mathbf{p}\cdot\mathbf{m}) + \int \lambda_1 (|\mathbf{p}| - \mathbf{p}\cdot\mathbf{m}) + \frac{r_2}{2}\int |\mathbf{p} - \langle \nabla u, 1\rangle|^2 \\ & + \int \boldsymbol{\lambda}_2 \cdot (\mathbf{p} - \langle \nabla u, 1\rangle), \end{aligned} \tag{12}$$

$$\begin{aligned} \varepsilon_4(\mathbf{n}) = {} & \frac{r_3}{2}\int (q - \partial_x n_1 - \partial_y n_2)^2 + \int \lambda_3 (q - \partial_x n_1 - \partial_y n_2) + \frac{r_4}{2}\int |\mathbf{n} - \mathbf{m}|^2 \\ & + \int \boldsymbol{\lambda}_4 \cdot (\mathbf{n} - \mathbf{m}), \end{aligned} \tag{13}$$

$$\begin{aligned} \varepsilon_5(\mathbf{m}) = {} & r_1 \int (|\mathbf{p}| - \mathbf{p}\cdot\mathbf{m}) + \int \lambda_1 (|\mathbf{p}| - \mathbf{p}\cdot\mathbf{m}) + \frac{r_4}{2}\int |\mathbf{n} - \mathbf{m}|^2 \\ & + \int \boldsymbol{\lambda}_4 \cdot (\mathbf{n} - \mathbf{m}) + \delta_{\mathcal{R}}(\mathbf{m}). \end{aligned} \tag{14}$$

The functionals $\varepsilon_2(q)$, $\varepsilon_3(\mathbf{p})$, and $\varepsilon_5(\mathbf{m})$ have closed-form solutions for their minimizers, while the minimizers of the functionals $\varepsilon_1(u)$ and $\varepsilon_4(\mathbf{n})$ are determined by the associated Euler-Lagrange equations. Specifically, as discussed

in [33], the minimizers of $\varepsilon_2(q)$, $\varepsilon_3(\mathbf{p})$, and $\varepsilon_5(\mathbf{m})$ read

$$\text{Argmin}_q \varepsilon_2(q) = \max\left\{0, 1 - \frac{\lambda}{r_3|\widetilde{q}|}\right\}\widetilde{q}, \quad \widetilde{q} = \partial_x n_1 + \partial_y n_2 - \frac{\lambda_3}{r_3}, \tag{15}$$

$$\text{Argmin}_{\mathbf{p}} \varepsilon_3(\mathbf{p}) = \max\left\{0, 1 - \frac{r_1+\lambda_1}{r_2|\widetilde{\mathbf{p}}|}\right\}\widetilde{\mathbf{p}}, \quad \widetilde{\mathbf{p}} = \langle \nabla u, 1\rangle - \frac{\boldsymbol{\lambda}_2}{r_2} + \frac{(r_1+\lambda_1)\mathbf{m}}{r_2}, \tag{16}$$

$$\text{Argmin}_{\mathbf{m}} \varepsilon_5(\mathbf{m}) = \begin{cases} \widetilde{\mathbf{m}}, & |\widetilde{\mathbf{m}}| \le 1; \\ \widetilde{\mathbf{m}}/|\widetilde{\mathbf{m}}|, & |\widetilde{\mathbf{m}}| > 1. \end{cases} \quad \widetilde{\mathbf{m}} = \mathbf{n} + \frac{\boldsymbol{\lambda}_4}{r_4} + \frac{(r_1+\lambda_1)\mathbf{p}}{r_4}, \tag{17}$$

and the Euler-Lagrange equations associated with $\varepsilon_1(u)$ and $\varepsilon_4(\mathbf{n})$ are given as follows:

$$-r_2\Delta u + u = f - \partial_x(r_2 p_1 + \lambda_{21}) - \partial_y(r_2 p_2 + \lambda_{22}), \tag{18}$$

and

$$\begin{aligned} -r_3\partial_x(\partial_x n_1 + \partial_y n_2) + r_4 n_1 &= r_4 m_1 - \lambda_{41} - (r_3 q + \lambda_3)_x, \\ -r_3\partial_y(\partial_x n_1 + \partial_y n_2) + r_4 n_2 &= r_4 m_2 - \lambda_{42} - (r_3 q + \lambda_3)_y, \\ n_3 &= m_3 - \lambda_{43}/r_4, \end{aligned} \tag{19}$$

where $\mathbf{p} = \langle p_1, p_2, p_3\rangle$, $\mathbf{m} = \langle m_1, m_2, m_3\rangle$, $\mathbf{n} = \langle n_1, n_2, n_3\rangle$, $\boldsymbol{\lambda}_2 = \langle \lambda_{21}, \lambda_{22}, \lambda_{23}\rangle$, and $\boldsymbol{\lambda}_4 = \langle \lambda_{41}, \lambda_{42}, \lambda_{43}\rangle$. To update the variables u and $\mathbf{n}$, one needs to solve these Euler-Lagrange equations. In the subsequent section, we discuss how to solve them using Gauss-Seidel method. Indeed, in [33], we employed FFT and thus got the exact numerical solutions of the two equations for each iteration. This is often expensive but may be unnecessary since it is enough to have some good approximations of u and $\mathbf{n}$ for each iteration. To this end, in this work, we apply the Gauss-Seidel method for the two equations. The later experiments demonstrate that this numerical method yield very similar results but with much less computational cost, especially for large size images. A related work can also be found in [11].

Besides the above variables, the Lagrange multipliers $\lambda_1, \boldsymbol{\lambda}_2, \lambda_3, \boldsymbol{\lambda}_4$ also need updates for each iteration:

$$\begin{aligned} \lambda_1^{new} &= \lambda_1^{old} + r_1(|\mathbf{p}| - \mathbf{p}\cdot\mathbf{m}), \\ \boldsymbol{\lambda}_2^{new} &= \boldsymbol{\lambda}_2^{old} + r_2(\mathbf{p} - \langle \nabla u, 1\rangle), \\ \lambda_3^{new} &= \lambda_3^{old} + r_3(q - \partial_x n_1 - \partial_y n_2), \\ \boldsymbol{\lambda}_4^{new} &= \boldsymbol{\lambda}_4^{old} + r_4(\mathbf{n} - \mathbf{m}), \end{aligned} \tag{20}$$

where λ_1^{old} denotes the value of λ_1 at the previous iteration while λ_1^{new} represents that of the current one.

3 Numerical Implementation

In this section, we present the details of solving Eqs. (18) and (19) using one sweeping of the Gauss-Seidel method as well as updating the variables q, $\mathbf{p}$, and $\mathbf{m}$ for each iteration.

As discussed in [33], we emphasize that the spatial mesh size is needed when considering the discretization of derivatives. This is because the mean curvature is not homogeneous in u, which is one of the most important features that distinguishes this mean curvature denoising model from other ones such as the ROF model [25] and the Euler's elastica based model [1,2].

Let $\Omega = \{(i,j)|1 \le i \le M, 1 \le j \le N\}$ be the discretized image domain and each point (i,j) is called a pixel point. All the variables are defined on these pixel points. We then introduce the discrete backward and forward differential operators with periodic boundary condition and the spatial mesh size h as follows:

$$\partial_1^- u(i,j) = \begin{cases} (u(i,j) - u(i-1,j))/h, & 1 < i \le M; \\ (u(1,j) - u(M,j))/h, & i = 1. \end{cases}$$

$$\partial_1^+ u(i,j) = \begin{cases} (u(i+1,j) - u(i,j))/h, & 1 \le i < M-1; \\ (u(1,j) - u(M,j))/h, & i = M. \end{cases}$$

$$\partial_2^- u(i,j) = \begin{cases} (u(i,j) - u(i,j-1))/h, & 1 < j \le N; \\ (u(i,1) - u(i,N))/h, & j = 1. \end{cases}$$

$$\partial_2^+ u(i,j) = \begin{cases} (u(i,j+1) - u(i,j))/h, & 1 \le j < N; \\ (u(i,1) - u(i,N))/h, & j = N. \end{cases}$$

For Eq. (18), a detailed discussion on how to solve it using FFT can be found in [33], and we here employ one sweeping of the Gauss-Seidel method and get

$$\left(1 + \frac{4r_2}{h^2}\right) u^{new}(i,j) =$$

$$g(i,j) + \frac{r_2}{h^2}[u^{new}(i-1,j) + u^{old}(i+1,j) + u^{new}(i,j-1) + u^{old}(i,j+1)], \quad (21)$$

where $u^{new}(i,j)$ denotes the updated value of u and $g(i,j)$ represents the value of the right-hand side of Eq. (18) at the pixel point (i,j). In the experiments, we just use one sweeping of the Gauss-Seidel method. Moreover, periodic condition is imposed. This boundary condition is often employed for the image denoising problem. In fact, this condition won't affect too much the result obtained by the denoising model under consideration since it is able to preserve jumps and image contrasts.

Similarly, for Eq. (19), one gets the following

$$\left(r_4 + \frac{2r_3}{h^2}\right) n_1^{new}(i,j) = g_1(i,j) + \frac{r_3}{h^2}[n_1^{new}(i-1,j) + n_1^{old}(i+1,j)], \quad (22)$$

$$\left(r_4 + \frac{2r_3}{h^2}\right) n_2^{new}(i,j) = g_2(i,j) + \frac{r_3}{h^2}[n_2^{new}(i,j-1) + n_2^{old}(i,j+1)], \quad (23)$$

where g_1 and g_2 denote the right-hand side of Eq. (19).

We then discuss the update of the variables q, $\mathbf{p}$, $\mathbf{m}$ as well as the Lagrange multipliers. As q is a scalar defined on the pixel point (i,j), based on the formulation (15), one gets

$$q(i,j) = \max\left\{0, 1 - \frac{\lambda}{r_3|\widetilde{q}(i,j)|}\right\} \widetilde{q}(i,j),$$

with $\widetilde{q}_{i,j} = \partial_1^- n_1(i,j) + \partial_2^- n_2(i,j) - \lambda_3/r_3$.

As for the variable $\mathbf{p}$, we first calculate the three components of $\widetilde{\mathbf{p}}$, the length of $\widetilde{\mathbf{p}}$, and then the updated $\mathbf{p}$. Specifically,

$$\widetilde{\mathbf{p}}(i,j) = \langle \partial_1^+ u, \partial_2^+ u, 1\rangle(i,j) - \frac{\langle \lambda_{21}, \lambda_{22}, \lambda_{23}\rangle + (r_1+\lambda_1)\langle m_1, m_2, m_3\rangle}{r_2}(i,j),$$

and based on the formulation (16)

$$\mathbf{p}(i,j) = \max\left\{0, 1 - \frac{r_1+\lambda_1(i,j)}{r_2|\widetilde{\mathbf{p}}(i,j)|}\right\}\widetilde{\mathbf{p}}(i,j).$$

Similarly, we calculate

$$\widetilde{\mathbf{m}}(i,j) = \mathbf{n}(i,j) + \frac{\boldsymbol{\lambda}_4(i,j)}{r_4} + \frac{(r_1+\lambda_1(i,j))\mathbf{p}(i,j)}{r_4},$$

and get the new $\mathbf{m}(i,j)$ using the formulation (17).

Moreover, based on the formulations (20), we may update all the Lagrange multipliers:

$$\lambda_1^{new}(i,j) = \lambda_1^{old}(i,j) + r_1(|\mathbf{p}|(i,j) - \mathbf{p}(i,j)\cdot\mathbf{m}(i,j)),$$

with $|\mathbf{p}|(i,j) = \sqrt{p_1^2(i,j) + p_2^2(i,j) + p_3^2(i,j)}$, and

$$\begin{aligned}
\lambda_{21}^{new}(i,j) &= \lambda_{21}^{old}(i,j) + r_2(p_1(i,j) - \partial_1^- u(i,j)),\\
\lambda_{22}^{new}(i,j) &= \lambda_{22}^{old}(i,j) + r_2(p_2(i,j) - \partial_2^- u(i,j)),\\
\lambda_{23}^{new}(i,j) &= \lambda_{23}^{old}(i,j) + r_2(p_3(i,j) - 1),\\
\lambda_3^{new}(i,j) &= \lambda_3^{old}(i,j) + r_3(q(i,j) - \partial_1^- n_1(i,j) - \partial_2^- n_2(i,j)),\\
\lambda_{41}^{new}(i,j) &= \lambda_{41}^{old}(i,j) + r_4(n_1(i,j) - m_1(i,j)),\\
\lambda_{42}^{new}(i,j) &= \lambda_{42}^{old}(i,j) + r_4(n_2(i,j) - m_2(i,j)),\\
\lambda_{43}^{new}(i,j) &= \lambda_{43}^{old}(i,j) + r_4(n_3(i,j) - m_3(i,j)).
\end{aligned}$$

4 Numerical Experiments

In this section, we present numerical experiments to compare the results obtained using Gauss-Seidel method and FFT respectively and also to illustrate the efficiency of the proposed algorithm.

For each experiment, as in [33], we monitor the following relative residuals in order to check whether the iteration converges to a saddle point:

$$(R_1^k, R_2^k, R_3^k, R_4^k) = \frac{1}{|\Omega|}(\|\widetilde{R}_1^k\|_{L^1}, \|\widetilde{R}_2^k\|_{L^1}, \|\widetilde{R}_3^k\|_{L^1}, \|\widetilde{R}_4^k\|_{L^1}), \tag{24}$$

with

$$\begin{aligned}
\widetilde{R}_1^k &= |\mathbf{p}^k| - \mathbf{p}^k\cdot\mathbf{m}^k,\\
\widetilde{R}_2^k &= \mathbf{p}^k - \langle\nabla u^k, 1\rangle,\\
\widetilde{R}_3^k &= q^k - \partial_x n_1^k - \partial_y n_2^k,\\
\widetilde{R}_4^k &= \mathbf{n}^k - \mathbf{m}^k,
\end{aligned}$$

and $|\Omega|$ is the domain area. We use these relative residuals (24) as the stopping criterion, that is, for a given threshold ϵ_r, once $R_i^k < \epsilon_r$ for $i = 1, ..., 4$ and some k, the iteration process will be terminated. To determine the convergence of the iteration process, we also check the relative errors of Lagrange multipliers:

$$(L_1^k, L_2^k, L_3^k, L_4^k) = \left(\frac{\|\lambda_1^k - \lambda_1^{k-1}\|_{L^1}}{\|\lambda_1^{k-1}\|_{L^1}}, \frac{\|\boldsymbol{\lambda}_2^k - \boldsymbol{\lambda}_2^{k-1}\|_{L^1}}{\|\boldsymbol{\lambda}_2^{k-1}\|_{L^1}}, \frac{\|\lambda_3^k - \lambda_3^{k-1}\|_{L^1}}{\|\lambda_3^{k-1}\|_{L^1}}, \frac{\|\boldsymbol{\lambda}_4^k - \boldsymbol{\lambda}_4^{k-1}\|_{L^1}}{\|\boldsymbol{\lambda}_4^{k-1}\|_{L^1}} \right), \quad (25)$$

and the relative error of the solution u^k

$$\frac{\|u^k - u^{k-1}\|_{L^1}}{\|u^{k-1}\|_{L^1}}. \quad (26)$$

We also observe how the energy (2) is evolving during the iteration by tracking the amount $E(u^k)$. For the presentation purpose, all the above quantities are shown in *log*-scale. Moreover, to illustrate what signals are removed as noise, we present the associated residual image $f - u$, besides the given noisy image f and the cleaned one u.

To compare the results using FFT and Gauss-Seidel method respectively, we calculate the quantity $\|u_{GS} - u_{FFT}\|_{L^1}/|\Omega|$ and also present the iteration numbers needed for a given threshold ϵ_r for these two methods.

Two numerical experiments are considered to test the effectiveness of the proposed algorithm. The first example is a synthetic image inside which there are several shapes with straight or curvy edges as well as sharp corners; the second one is the real image "Lenna". The original noisy images, denoised images, and their differences $f - u$ are presented in Fig. 4. The results for the synthetic image demonstrate that besides removing noise and keeping edges, the model also preserves sharp corners and image contrasts, which can be visualized from the difference image in which there is almost no meaningful signals. For the real image "Lenna", the noise part is effectively removed while edges are kept. An important feature worthy of emphasizing is the improvement of the staircase effect. The cleaned image, especially at the face and shoulder, illustrates that the model produces smooth patches instead of blocky ones.

Figure 4 lists the plots of the relative residuals (Eq. 24), relative errors of the Lagrange multipliers (Eq. 25), relative error of the iterative u^k (Eq. 26), and the energy $E(u^k)$ versus iterations for the synthetic image "shape". The first row

Table 1. The comparison of the iteration numbers needed for $\epsilon_r = 2.5 \times 10^{-3}$ using Gauss-Seidel and FFT respectively and the difference quantity $\|u_{GS} - u_{FFT}\|_{L^1}/|\Omega|$.

Image	Size	Number of iterations (GS)	Number of iterations (FFT)	$\|u_{GS} - u_{FFT}\|_{L^1}/\|\Omega\|$
Figure 4-(a)	256×256	334	314	0.035
Figure 4-(b)	256×256	761	754	0.134

Table 2. The image sizes, the SNRs, the iteration numbers needed for the given threshold ϵ_r, and the computational times using Gauss-Seidel and FFT for the experiments.

Image	Size	SNR	ϵ_r	Number of iterations	Time (sec) using GS	Time (sec) using FFT
Figure 4-(a)	256×256	10.75	2.5×10^{-3}	334	16.30	20.81
Figure 4-(b)	256×256	13.65	2.5×10^{-3}	761	37.70	47.82
Figure 4-(a)	256×400	10.70	4.0×10^{-3}	786	65.05	84.80
Figure 4-(b)	256×320	8.97	2.0×10^{-3}	466	33.38	41.71
Figure 4-(c)	320×320	11.88	4.0×10^{-3}	700	57.56	75.02

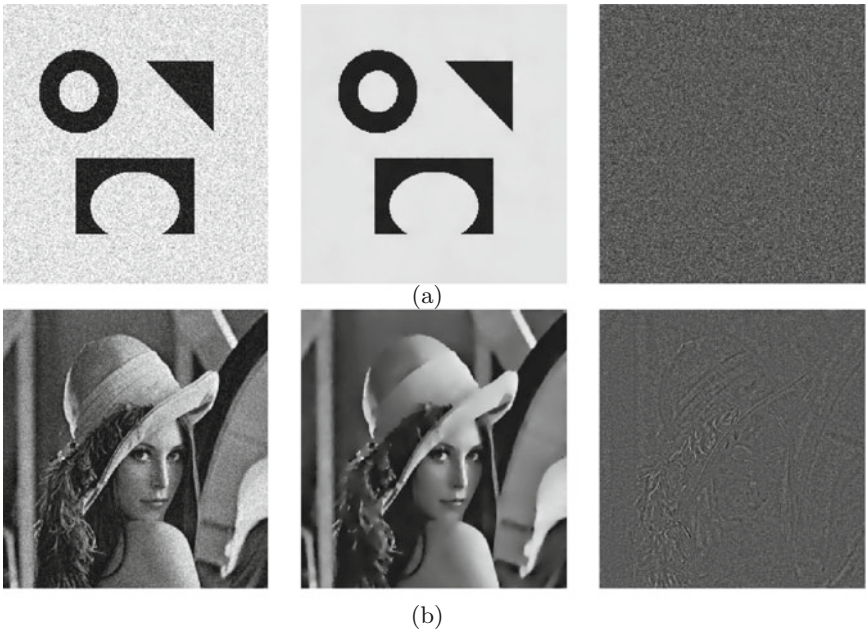


Fig. 1. The denoising results for a synthetic image and a real image "Lenna". The noisy, denoised, and residual images are listed from the first row to the third row respectively. For the synthetic image, we set the spatial step size $h = 5.0$, and choose $\lambda = 2 \times 10^3$, $r_1 = 40$, $r_2 = 40$, $r_3 = 10^5$, $r_4 = 2.0 \times 10^5$, and $\epsilon_r = 2.5 \times 10^{-3}$. For the real image, we set the spatial step size $h = 5.0$, choose $\lambda = 10^3$, $r_1 = 40$, $r_2 = 40$, $r_3 = 10^5$, $r_4 = 10^5$, and $\epsilon_r = 2.5 \times 10^{-3}$.

presents the plots obtained using Gauss-Seidel method while the second row shows those ones obtained using FFT. Firstly, these plots demonstrate the convergence of iteration and therefore a saddle point of the augmented Lagrangian functional (9) and thus a minimizer of the functional (2) is achieved for each case. Secondly, the plots also illustrate the efficiency of the proposed augmented Lagrangian method. In general, it only needs a few hundred of iterations to

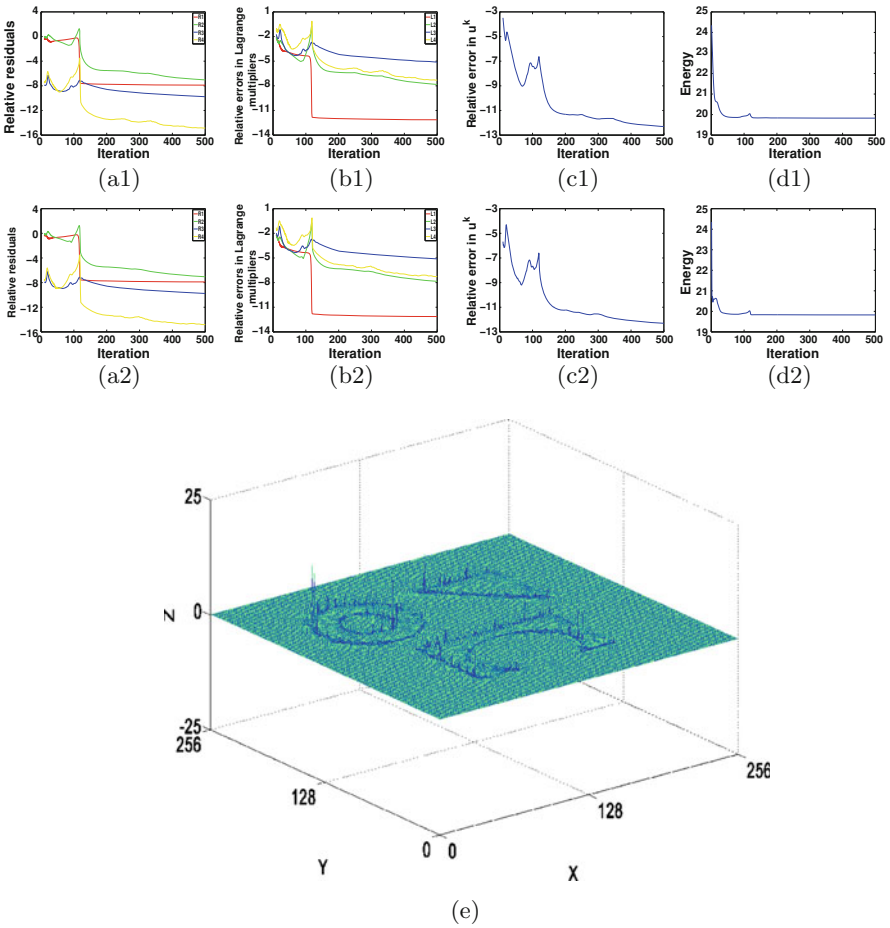


Fig. 2. The first row lists the plots of relative residuals (a1), relative errors in Lagrange multipliers (b1), relative error in u^k (c1), and energy (d1) versus iterations for the example "shape" using Gauss-Seidel method; the second row presents the corresponding plots when using FFT. The plot (e) shows the difference between the two cleaned images u_{GS} and u_{FFT}.

approach minimizers with reasonable accuracy. In fact, in the experiments, for each iteration, one needs to use two times of Gauss-Seidel sweeping and three times of the application of closed-form solutions. Thirdly, the plots in these two rows present almost no difference for each corresponding quantity versus iteration, indicating that the substitution of FFT with one sweeping of Gauss-Seidel for solving the equations of u and $\mathbf{n}$ is feasible. To further compare these two methods, we present in the third row with the difference $u_{GS} - u_{FFT}$, each of which is obtained with the same iteration number. This plot illustrates that the two cleaned images only differ in small magnitudes around some parts on the edges of the shapes.

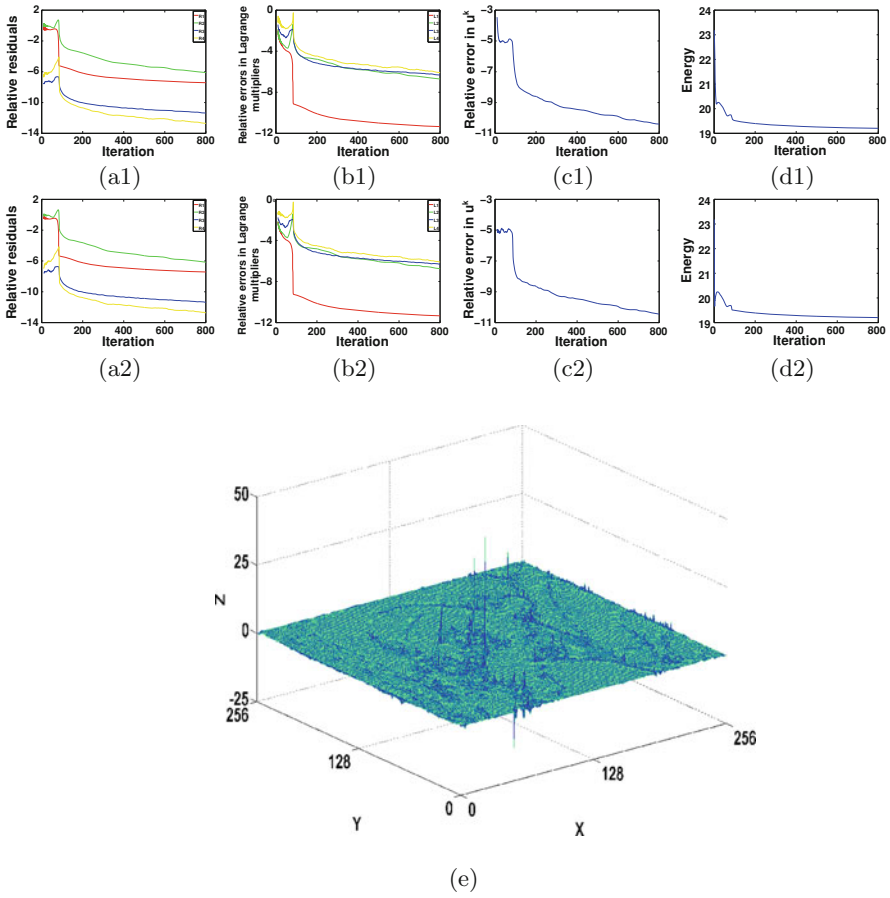


Fig. 3. The first row lists the plots of relative residuals (a1), relative errors in Lagrange multipliers (b1), relative error in u^k (c1), and energy (d1) versus iterations for the example "Lenna" using Gauss-Seidel method; the second row presents the corresponding plots when using FFT. The plot (e) shows the difference between the two cleaned images u_{GS} and u_{FFT}.

Besides these, the iteration numbers needed for the stopping threshold and the difference quantity $\|u_{GS} - u_{FFT}\|_{L^1}/|\Omega|$ are also compared for the two methods in Table 1. From this table, when the threshold ϵ_r is achieved, less iterations are needed using FFT than Gauss-Seidel method, which is reasonable since FFT gives exact solutions to those equations. However, as shown in Table 2, this cannot save too much the computational cost for the utilization of FFT.

Figure 4 presents the comparison of the two methods for the example "Lenna". In summary, these two examples demonstrate that replacing FFT by Gauss-Seidel method won't affect the final results considerably and thus this replacement is applicable when using the mean curvature denoising model.

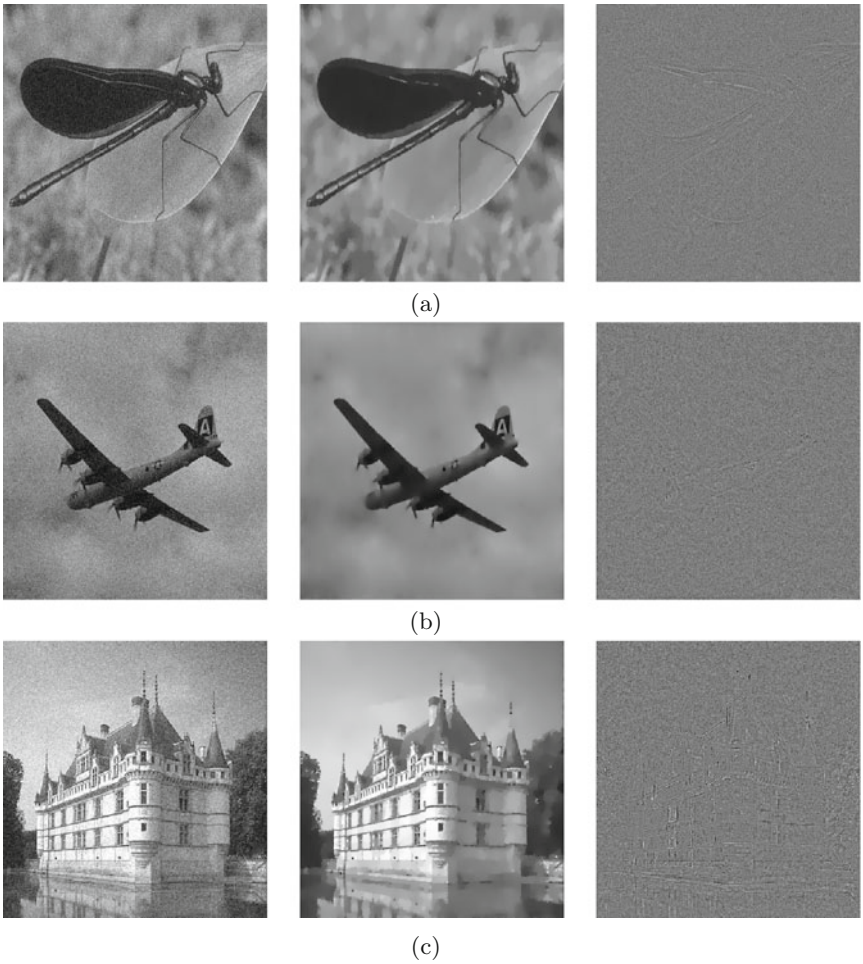


Fig. 4. The denoising results for real images. Each row presents the original image, the cleaned one, and the difference image from the left to the right.

In Fig. 4, we consider more experiments for real images using Gauss-Seidel method. These examples again demonstrate the features of the mean curvature based denoising model, including the amelioration of the staircase effect and the preservation of image contrasts. To show the efficiency of the proposed algorithm as well as the comparison of the Gauss-Seidel method and FFT that are used for solving Eqs. (18) and (19), we present in Table 2 for all the experiments with the image sizes, the SNRs, the numbers of total iteration needed to satisfy the given threshold ϵ_r, and the computational times using Gauss-Seidel and FFT respectively. From this table, when compared with FFT, the utilization of Gauss-Seidel method reduces the computational cost, and the larger the image size, the more this reduction will be. Here, we omit the comparison of the proposed algorithm with the standard gradient descent method that is used to solve Eq. (7), which

can found in [33]. Moreover, all computations were done using Matlab R2011a on an Intel Core I5 machine (Figs. 1, 2 and 3).

5 Conclusion

Recently, the augmented Lagrangian method has been successfully used to minimize the classical ROF functional [29] and Euler's elastica based functionals [26]. In [33], inspired by the idea in [26], we developed a special technique to treat the mean curvature based image denoising model [31] using augmented Lagrangian method, and thus converted the original challenging minimization problem to be several tractable ones, some of which have closed form solutions while the other of which can be solved using FFT. In this work, we consider to use Gauss-Seidel method to substitute FFT for solving those equations in order to further reduce the computational cost. The numerical experiments demonstrate that the substitution is feasible and saves considerable computational effort especially for images of large sizes.

Acknowledgments. The authors thank the anonymous referees for their valuable comments and suggestions. The work was supported by NSF contract DMS-1016504.

References

1. Ambrosio, L., Masnou, S.: A direct variational approach to a problem arising in image reconstruction. Interfaces Free Bound. **5**, 63–81 (2003)
2. Ambrosio, L., Masnou, S.: On a Variational Problem Arising in Image Reconstruction, vol. 147. Birkhauser, Basel (2004)
3. Alvarez, L., Guichard, F., Lions, P.L., Morel, J.M.: Axioms and fundamental equations of image-processing. Arch. Ration. Mech. Anal. **123**, 199–257 (1993)
4. Aubert, G., Vese, L.: A variational method in image recovery. SIAM J. Numer. Anal. **34**(5), 1948–1979 (1997)
5. Bertozzi, A.L., Greer, J.B.: Low curvature image simplifiers: global regularity of smooth solutions and Laplacian limiting schemes. Comm. Pure Appl. Math. **57**, 764–790 (2004)
6. Chan, T., Esedoḡlu, S., Park, F., Yip, M.H.: Recent developments in total variation image restoration. In: Paragios, N., Chen, Y., Faugeras, O. (eds.) Handbook of Mathematical Models in Computer Vision, pp. 17–31. Springer, Heidelberg (2006)
7. Chambolle, A., Lions, P.L.: Image recovery via total variation minimization and related problems. Numer. Math. **76**, 167–188 (1997)
8. Chan, T., Kang, S.H., Shen, J.H.: Euler's elastica and curvature based inpaintings. SIAM J. Appl. Math **63**(2), 564–592 (2002)
9. Chan, T., Marquina, A., Mulet, P.: High-order total variation-based image restoration. SIAM J. Sci. Comput. **22**(2), 503–516 (2000)
10. do Carmo, M.P.: Differential Geometry of Curves and Surfaces. Prentice-Hall Inc., Englewood (1976)
11. Duan, Y., Wang, Y., Tai, X.-C., Hahn, J.: A fast augmented lagrangian method for Euler's elastica model. In: Bronstein, A.M., Bronstein, M.M., Bruckstein, A.M., Haar Romeny, B.M. (eds.) SSVM 2011. LNCS, pp. 144–156. Springer, Heidelberg (2012)

12. Elsey, M., Esedoḡlu, S.: Analogue of the total variation denoising model in the context of geometry processing. SIAM J. Multiscale Model. Simul. **7**(4), 1549–1573 (2009)
13. Esedoḡlu, S., Shen, J.: Digital inpainting based on the Mumford-Shah-Euler image model. Eur. J. Appl. Math **13**, 353–370 (2002)
14. Greer, J.B., Bertozzi, A.L.: Traveling wave solutions of fourth order PDEs for image processing. SIAM J. Math. Anal. **36**(1), 38–68 (2004)
15. Greer, J.B., Bertozzi, A.L., Sapiro, G.: Fourth order partial differential equations on general geometries. J. Comp. Phys. **216**(1), 216–246 (2006)
16. Lysaker, M., Lundervold, A., Tai, X.C.: Noise removal using fourth-order partial differential equation with applications to medical magnetic resonance images in space and time. IEEE. Trans. Image Process. **12**, 1579–1590 (2003)
17. Masnou, S.: Disocclusion: a variational approach using level lines. IEEE Trans. Image Process. **11**, 68–76 (2002)
18. Masnou, S., Morel, J.M.: Level lines based disocclusion. In: Proceedings of IEEE International Conference on Image Processing, Chicago, IL, pp. 259–263 (1998)
19. Meyer, Y.: Oscillating Patterns in Image Processing and Nonlinear Evolution Equations. University Lecture Series, vol. 22. American Mathematical Society, Providence (2001)
20. Mumford, D., Shah, J.: Optimal approximation by piecewise smooth functions and associated variational problems. Comm. Pure Appl. Math. **42**, 577–685 (1989)
21. Morel, J.M., Solimini, S.: Variational Methods in Image Segmentation. Birkhauser, Boston (1995)
22. Nitzberg, M., Shiota, T., Mumford, D.: Filtering, Segmentation and Depth. LNCS, vol. 662. Springer, Heidelberg (1993)
23. Osher, S., Sole, A., Vese, L.: Image decomposition and restoration using total variation minimization and the $H^{-1}norm$. SIAM Multiscale Model. Simul. **1**, 349–370 (2003)
24. Perona, P., Malik, J.: Scale-space and edge-detection using anisotropic diffusion. IEEE Trans. Pattern Anal. Mach. Intell. **12**(7), 629–639 (1990)
25. Rudin, L., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithm. Physica D **60**, 259–268 (1992)
26. Tai, X.C., Hahn, J., Chung, G.J.: A fast algorithm for Euler's Elastica model using augmented Lagrangian method. SIAM J. Imag. Sci. **4**(1), 313–344 (2011)
27. Tasdizen, T., Whitaker, R., Burchard, P., Osher, S.: Geometric surface processing via normal maps. ACM Trans. Graph. **22**, 1012–1033 (2003)
28. Vese, L., Osher, S.: Modeling textures with total variation minimization and oscillatory patterns im image processing. SINUM **40**, 2085–2104 (2003)
29. Wu, C., Tai, X.C.: Augmented Lagrangian method, dual methods, and split Bregman iteration for ROF, Vectorial TV, and high order models. SIAM J. Imag. Sci. **3**(3), 300–339 (2010)
30. Zhu, W., Chan, T.: A variational model for capturing illusory contours using curvature. J. Math. Imag. Vis. **27**, 29–40 (2007)
31. Zhu, W., Chan, T.: Image denoising using mean curvature of image surface. SIAM J. Imag. Sci. **5**, 1–32 (2012)
32. Zhu, W., Chan, T., Esedoḡlu, S.: Segmentation with depth: a level set approach. SIAM J. Sci. Comput. **28**, 1957–1973 (2006)
33. Zhu, W., Tai, X.C., Chan, T.: Augmented Lagrangian method for a mean curvature based image denoising model, Accepted by Inverse Problems and Imaging (2013)

A Smoothing Descent Method for Nonconvex TV^q-Models

Michael Hintermüller[1(✉)] and Tao Wu[2]

[1] Department of Mathematics, Humboldt-University of Berlin, Unter den Linden 6, 10099 Berlin, Germany
hint@math.hu-berlin.de

[2] Institute for Mathematics and Scientific Computing, University of Graz, Heinrichstrasse 36, 8010 Graz, Austria
tao.wu@edu.uni-graz.at

Abstract. A novel class of variational models with nonconvex ℓ_q-norm-type regularizations ($0 < q < 1$) is considered, which typically outperforms popular models with convex regularizations in restoring sparse images. Due to the fact that the objective function is nonconvex and non-Lipschitz, such models are very challenging from an analytical as well as numerical point of view. In this work a smoothing descent method with provable convergence properties is proposed for computing stationary points of the underlying variational problem. Numerical experiments are reported to illustrate the effectiveness of the new method.

1 Introduction

In signal or image recovery from sparse data, it has been observed that models based on nonconvex regularization typically outperform convex ones. In particular, variational models relying on *ℓ_q-norm-type* regularization with $q \in (0, 1)$ are of interest [3,9,16]. Due to the particular choice of q, the associated functional $\|v\|_q^q := \sum_i |v_i|^q$ turns out to be nonconvex, nondifferentiable and not even locally Lipschitz continuous, and it is not a norm in the usual sense. These properties imply several challenges from an analytical as well as numerical point of view in the treatment of such problems. In fact, analytically generalized derivative concepts are challenged [7] and, thus, the first-order optimality description becomes an issue. Concerning the numerical solution of general nonsmooth minimization problems, we mention that while convex problems are rather well understood [13,14,19] nonconvex ones are still challenging [1,14]. For nonconvex and non-locally-Lipschitz problems, the literature on methods based on generalized derivatives is rather scarce. However, taking the particular format of the underlying nonsmoothness into account and possibly applying tailored (vanishing) smoothing concepts, in [4,6,9,12,15,16] solvers of the associated minimization problems in sparse signal or image recovery were developed recently.

This research was supported by the Austrian Science Fund (FWF) through START project Y305 "Interfaces and Free Boundaries" and through SFB project F3204 "Mathematical Optimization and Applications in Biomedical Sciences".

A. Bruhn et al. (Eds.): Global Optimization Methods, LNCS 8293, pp. 119–133, 2014.
DOI: 10.1007/978-3-642-54774-4_6,

In view of these references, we note that [6,15] requires conditions for the successful analysis which ultimately rule out the ℓ_q-norm-type regularization, [9] needs a sparsity assumption, and [4] provides a method based on Bergman iterations and specific shrinkage-procedures, but does not include a convergence analysis. In [12] a regularized nonsmooth Newton technique is proposed which relies on some kind of local smoothing.

Motivated by smoothing methods (see for example [6]), in this work, for solving the problem (1) below, we present a descent algorithm combining a Huber-type smoothing (which we call *Huberization* in the sequel) with elements of the nonsmooth Newton solver from [12]. In fact, the smoothing method provides a mechanism which allows us to drive the Huber-type smoothing of the ℓ_q-norm to zero, thus, genuinely approaching, along a subsequence, a stationary point of the ℓ_q-norm-type problem. From the gradient of the *Huberized* objective, a suitable descent direction for that objective is computed by the so-called R-regularized nonsmooth Newton method [12]. By this procedure, a variable-metric-type scaling is applied to the steepest descent direction, thus, improving the convergence behavior of the overall method. Moreover, convergence of the algorithmic scheme towards a stationary point of (1) is established.

The rest of the paper is organized as follows. In Sect. 2, we introduce the nonconvex TV^q-model in finite dimension and establish the existence of a solution and the necessary optimality condition. A smoothing descent method and its convergence analysis are presented in Sect. 3. Finally, the proposed method is implemented for various image processing tasks and the results are reported in Sect. 4.

2 Nonconvex TV^q-Model

The nonconvex TV^q-model considered in this paper is formulated as follows:

$$\min_{u\in\mathbb{R}^{|\Omega|}} f(u) := \sum_{(i,j)\in\Omega} \left(\frac{\mu}{2}|(\nabla u)_{ij}|^2 + \frac{\alpha}{q}|(\nabla u)_{ij}|^q + \frac{1}{2}|(Ku-z)_{ij}|^2 \right). \quad (1)$$

Here, Ω is a two-dimensional discrete image domain with $|\Omega|$ being the number of pixels in Ω, $u \in \mathbb{R}^{|\Omega|}$ represents the digital image which is to be reconstructed from observed data $z \in \mathbb{R}^{|\Omega|}$, $\nabla \in \mathbb{R}^{2|\Omega|\times|\Omega|}$ is the gradient operator, $|(\nabla u)_{ij}|$ denotes the Euclidean norm of $(\nabla u)_{ij}$ in $\mathbb{R}^2$, $K \in \mathbb{R}^{|\Omega|\times|\Omega|}$ is a continuous linear operator (which, for instance, might describe blurring), and $\alpha > 0$, $0 < q < 1$, $0 < \mu \ll \alpha$ are user-chosen regularization parameters. In particular, if $q = 1$ and $\mu = 0$, then (1) corresponds to the conventional (convex) total variation (TV-) model according to Rudin, Osher and Fatemi [18].

Throughout this paper, we shall assume that

$$\mathrm{Ker}\nabla \cap \mathrm{Ker}K = \{0\}. \quad (2)$$

Under the condition (2), the existence of a solution of (1) is guaranteed by the following theorem. Its proof can be found in [12].

Theorem 1 (Existence of Solution). *There exists a global minimizer for the variational problem (1).*

Note that due to the presence of the power-q term the objective f in (1) is not even locally Lipschitz continuous. This causes difficulties when using the Clarke generalized gradient of f [7] for describing stationarity of a point u. However, distinguishing smooth and nonsmooth regions of f through the norm of the gradient of u, stationary points of (1) can still be characterized by the following Euler-Lagrange-type system:

$$\begin{cases} -\mu\Delta u + K^\top(Ku - z) + \alpha\nabla^\top(|\nabla u|^{q-2}\nabla u) = 0, & \text{if } (\nabla u)_{ij} \neq 0; \\ \nabla u = 0, & \text{otherwise.} \end{cases} \tag{3}$$

The disjunctive nature of the above system, which is due to the nonsmoothness of f, causes severe difficulties in the design of solution algorithms. In the following, we propose a smoothing descent method, which generates a sequence that has an accumulation point satisfying (3).

3 Smoothing Descent Method

The smoothing descent method proceeds iteratively as follows. In each iteration, the TV^q-objective is smoothed locally around the nondifferentiability by a Huber function which is controlled through the parameter $\gamma > 0$. The Huber function reads

$$\varphi_\gamma(s) := \begin{cases} \frac{1}{q}s^q - (\frac{1}{q} - \frac{1}{2})\gamma^q, & \text{if } s > \gamma, \\ \frac{1}{2}\gamma^{q-2}s^2, & \text{if } 0 \leq s \leq \gamma, \end{cases}$$

and the associated Huberized version of (1) is given by

$$\min_{u\in\mathbb{R}^{|\Omega|}} f_\gamma(u) := \sum_{(i,j)\in\Omega} \left(\frac{\mu}{2}|(\nabla u)_{ij}|^2 + \alpha\varphi_\gamma(|(\nabla u)_{ij}|) + \frac{1}{2}|(Ku - z)_{ij}|^2 \right). \tag{4}$$

The corresponding Huberized Euler-Lagrange equation is

$$\nabla f_\gamma(u) = -\mu\Delta u + K^\top(Ku - z) + \alpha\nabla^\top\left(\max(|\nabla u|, \gamma)^{q-2}\nabla u\right) = 0. \tag{5}$$

Here the max-operation is understood in a componentwise sense.

Clearly, the Huberized objective f_γ is continuously differentiable and bounded from below. Therefore, (4) can be solved by well-known standard solvers for smooth minimization problems; see, e.g., [17]. In this work, however, we utilize a tailored approach by employing the so-called *R-regularized Newton scheme* which was very recently proposed in [12]. In order to globalize the Newton iteration, a backtracking Armijo line search is used which is particularly tuned to the structure of the problem of interest.

For completeness, we provide a self-contained introduction of the R-regularized Newton method in the appendix. Essentially, this method is a generalization of the lagged-diffusivity fixed-point iteration [2,5,20], which alleviates diffusion nonlinearity by using information from the previous iterate. Moreover, with the aid of the infeasible Newton technique [11], a sufficient condition can be derived for obtaining a descent direction in each iteration; see Theorem 5 in the appendix. The descent property is important for the Armijo-based line search globalization; see [17] and the references therein for a general account of Armijo's line search rule.

When (4) is solved with sufficient accuracy, for instance with respect to the residual norm $\|f_{\gamma^k}(u^{k+1})\|$, then the Huber parameter is reduced and the current solution serves as the initial guess for solving the next Huberized problem. The resulting overall algorithmic scheme is specified next.

Algorithm 1. Smoothing descent method.

Choose $0 < \tau < 1,\ 0 < \theta < 1,\ \rho > 0,\ \bar{\gamma} > 0$. Then iterate as follows:

1. Compute a descent direction δu^k for f_{γ^k} at u^k, i.e. $\nabla f_{\gamma^k}(u^k)^\top \delta u^k < 0$, for the Huberized problem (4) by the R-regularized Newton method..
2. Perform an Armijo line search, i.e. determine the size $a^k > 0$ such that

$$f_{\gamma^k}(u^k + a^k \delta u^k) \le f_{\gamma^k}(u^k) + \tau a^k \nabla f_{\gamma^k}(u^k)^\top \delta u^k, \tag{6}$$

   and set $u^{k+1} := u^k + a^k \delta u^k$.
3. If $\|\nabla f_{\gamma^k}(u^{k+1})\| \ge \rho \gamma^k$, then set $\gamma^{k+1} := \gamma^k$; otherwise, set $\gamma^{k+1} := \theta \gamma^k$.
4. If $\gamma^k \ge \bar{\gamma}$, then set $k := k + 1$ and return to step 1; otherwise stop.

In our experiments, we shall always fix the parameters $\tau = 0.1,\ \theta = 0.8,\ \rho = 0.1$.

Next we present the convergence analysis for the smoothing descent method. For this purpose, we take $\bar{\gamma} = 0$.

Lemma 2. *The sequence generated by Algorithm 3 satisfies*

$$\lim_{k\to\infty} \gamma^k = 0, \quad \text{and} \quad \liminf_{k\to\infty} \|\nabla f_{\gamma^k}(u^{k+1})\| = 0.$$

Proof. Define the index set

$$\mathcal{K} := \{k : \gamma^{k+1} = \theta \gamma^k\}. \tag{7}$$

If $\mathcal{K}$ is finite, then there exists some $\bar{k}$ such that for all $k > \bar{k}$ we have $\gamma^k = \gamma^{\bar{k}}$ and $\|\nabla f_{\gamma^k}(u^{k+1})\| \ge \rho \gamma^{\bar{k}}$. This contradicts the fact that the first-order method with Armijo line search for solving the smooth problem $\min_u f_{\gamma^{\bar{k}}}(u)$ generates a sequence (u^k) such that $\liminf_{k\to\infty} \|\nabla f_{\gamma^{\bar{k}}}(u^k)\| = 0$, cf. [10]. Thus, $\mathcal{K}$ is infinite and $\lim_{k\to\infty} \gamma^k = 0$. Moreover, let $\mathcal{K} = (k^l)_{l=1}^\infty$ with $k^1 < k^2 < \ldots$, then we have $\|\nabla f_{\gamma^{k^l}}(u^{k^l+1})\| \le \rho \gamma^{k^l} \to 0$ as $l \to \infty$. Hence, $\liminf_{k\to\infty} \|\nabla f_{\gamma^k}(u^{k+1})\| = 0$.

Theorem 3. *Assume that the sequence* (u^k) *generated by Algorithm 3 is bounded. Then there exists an accumulation point* u^* *of* (u^k) *such that* $u^* \in \mathbb{R}^{|\Omega|}$ *satisfies the Euler-Lagrange Eq. (3).*

Proof. In view of the result in Lemma 2, there exists a subsequence of (u^k), say $(u^{k'})$, such that $\lim_{k'\to\infty} \|\nabla f_{\gamma^{k'-1}}(u^{k'})\| = 0$. Since $(u^{k'})$ is bounded, by compactness there exists a subsequence of $(u^{k'})$, say $(u^{k''})$, such that $(u^{k''})$ converges to some u^* as $k'' \to \infty$. We show that u^* is a solution to (3). On the set $\{(i,j) \in \Omega : (\nabla u^*)_{ij} = 0\}$, the conclusion follows automatically. On the set $\{(i,j) \in \Omega : (\nabla u^*)_{ij} \neq 0\}$, we have that $\max(|\nabla u^{k''}|, \gamma^{k''-1}) \to |\nabla u^*| > 0$ as $k'' \to \infty$. Therefore, it follows from

$$|\nabla f(u^*)| \leq |\nabla f_{\gamma^{k''-1}}(u^{k''}) - \nabla f(u^*)| + |\nabla f_{\gamma^{k''-1}}(u^{k''})| \to 0, \tag{8}$$

that u^* satisfies (3).

4 Numerical Experiments

In this section, we report on numerical results obtained by our algorithm for various tasks in TV^q-model based image restoration. The experiments are performed under MATLAB R2009b on a 2.66 GHz Intel Core Laptop with 4 GB RAM.

4.1 Denoising

We first test our algorithm on denoising the "Two Circles" image; see Fig. 1. This image, shown in plot (a), is degraded by zero-mean white Gaussian noise of 7 % and 14 % standard deviation respectively; see plots (b) and (c). The parameters $q = 0.75,\ \mu = 0$ are used in this example. The restored images of two different noisy images are given in plots (d) and (e). In the following, we use the data set in (b) to investigate the numerical behavior of Algorithm 3 in details.

Robustness to Initialization. Note that our algorithm is intended to find a stationary point of the TV^q-model (which is often a local minimizer in our numerical experiments). It is worthwhile to check the quality of such local solutions. In Fig. 2, we implement the algorithm starting from three different choices of initializations; see the first row. The corresponding restored images are shown in the second row, which are visually indistinguishable. The energy values of the restorations in (e), (f), (g), (h) are equal to 29.4488, 29.4499, 29.4497, 29.4594, respectively, which also indicates that the restorations have small differences in quality. We remark that choosing a relatively large initial Huber parameter γ^0 in general strengthens the robustness of the algorithm against poor initializations.

Choice of Stopping Criteria. As the stopping criteria (step 4) of Algorithm 3 depends on $\bar{\gamma}$, here we suggest an empirical way based on the *histogram* for choosing a proper $\bar{\gamma}$ in an a posteriori way. As we know, the TV^q-model tends

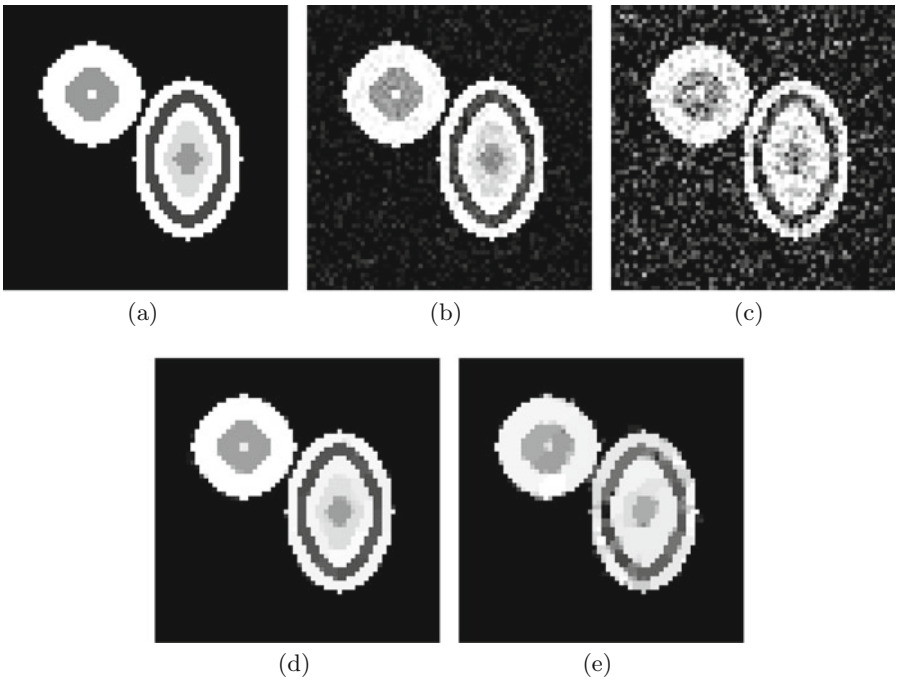


Fig. 1. Denoising: (a) "Two Circles" image. (b) Corrupted with 7 % Gaussian noise. (c) Corrupted with 14 % Gaussian noise. (d) Restoration of (b) with $\alpha = 0.05$. (e) Restoration of (c) with $\alpha = 0.12$.

to promote a solution with very sparse histogram. This is numerically confirmed by the histogram plots in Fig. 3. Therefore, it is reasonable to terminate the algorithm once there is no longer significant change in the sparsity pattern of the histogram. In our particular example, this suggests that $\bar{\gamma} = 10^{-4}$ is a proper choice. The same choice of $\bar{\gamma}$ will be used in all the following experiments.

Miscellaneous Numerical Behavior. We further demonstrate the numerical behavior of the algorithm in Fig. 4. All data points in the plots are taken from those iterations with $k \in \mathcal{K}$; see (7) for the definition of $\mathcal{K}$. As γ^k decreases, our algorithm is well behaved in terms of objective value, PSNR, and residual norm. Qualitatively a similar numerical behavior is observed in the experiments that follow.

4.2 Deblurring

In Fig. 5 we test our algorithm in the context of deblurring the 256-by-256 phantom image depicted in plot (a). The original image is blurred by a two-dimensional truncated Gaussian kernel yielding

$$(Ku)_{ij} = \sum_{|i'|\le 3,\ |j'|\le 3} \exp\left(-\frac{|i'|^2+|j'|^2}{2|\sigma_K|^2}\right) u_{i-i',j-j'}.$$

Then white Gaussian noise of zero mean and 0.05 standard deviation is added to the blurred image; see (b). We apply our algorithm with $q = 0.75,\ \alpha = 0.01,\ \mu = 10^{-6}$, and $u^0 = z$. In plot (c) the restored image is shown. Its PSNR-value is 27.6672.

4.3 Compressed Sensing

We also apply our algorithm to a k-space compressed sensing problem; see Fig. 6. The observed data z is constructed as follows: $z = SFu_{true}$, where F is the 2D discrete Fourier transform and S is a 20 % k-space random sampling matrix. We reconstruct the image by solving our TV^q-model with $q = 0.75,\ \alpha = 0.001,\ \mu = 10^{-6}$, and $u^0 = 0$. The corresponding restored image is shown in (e). This result is compared with the reconstruction obtained from the inverse Fourier transform in (c) and the reconstruction obtained from the TV-model in (d) with $q = 1,\ \alpha = 0.02,\ \mu = 10^{-6}$. In our implementation of the TV-model, α is chosen in order to obtain an image with optimal PSNR. The (convex) TV-model, here as well as in Sect. 4.4, is solved by a primal-dual Newton method [11] with Huber parameter $\bar{\gamma}$. We remark that many other algorithms in the literature may also work efficiently in the same context. In the left part of Table 1, we provide the comparison of the three candidate methods with respect to PSNR and CPU time. It is observed

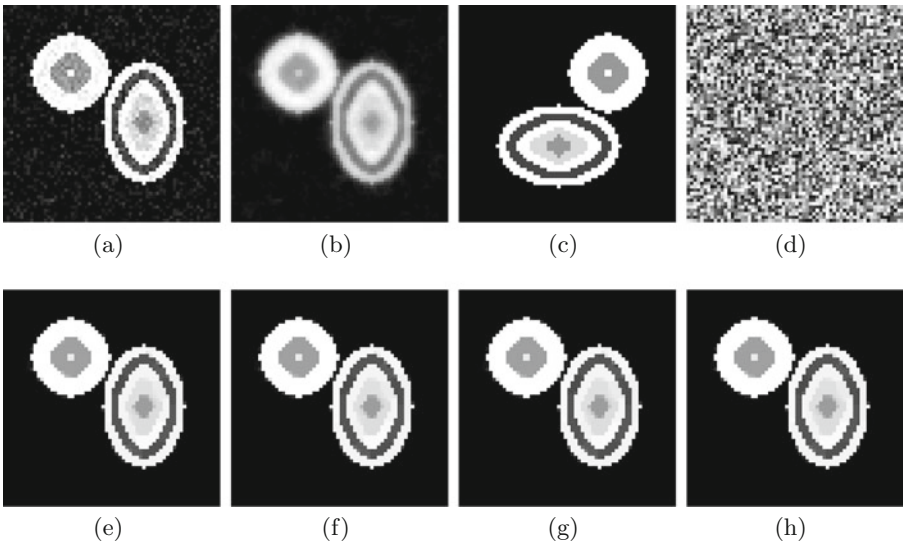
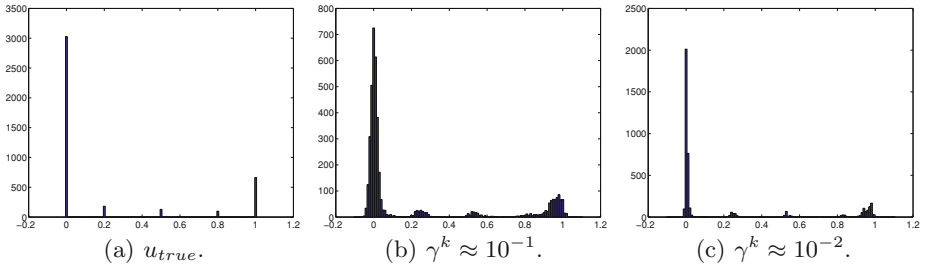


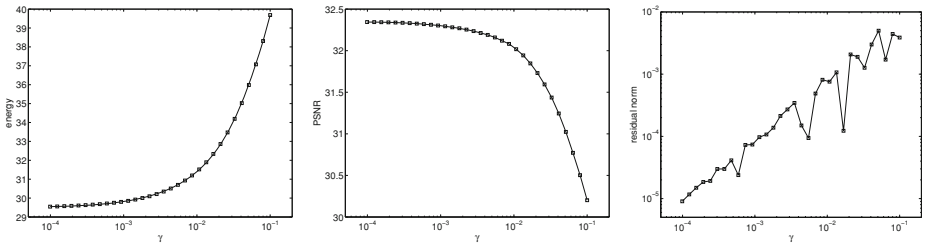
Fig. 2. Initialization test: (a) Observed image as initial guess. (b) Tikhonov regularized image as initial guess. (c) Rotated image as initial guess (d) Random initial guess. (e), (f), (g), and (h) are the restorations from (a), (b), (c), and (d) respectively.



(a) u_{true}. (b) $\gamma^k \approx 10^{-1}$. (c) $\gamma^k \approx 10^{-2}$.

(d) $\gamma^k \approx 10^{-3}$. (e) $\gamma^k \approx 10^{-4}$. (f) $\gamma^k \approx 10^{-5}$.

Fig. 3. Histogram of u^{k+1} $(k \in \mathcal{K})$.



(a) (b) (c)

Fig. 4. Numerical behavior $(k \in \mathcal{K})$: (a) TV^q-energy $f(u^{k+1})$. (b) PSNR(u^{k+1}). (c) Residual norm $\|\nabla f_{\gamma^k}(u^{k+1})\|$.

that the inverse Fourier transform is computationally cheap but only yields a poor result. The TV method takes about 6 seconds but still cannot recover the image to fine details. Our TV^q method takes about double the CPU time of TV and provides an almost perfect reconstruction.

4.4 Integral-Geometry Tomography

In Fig. 7, we apply our algorithm to integral-geometry tomography. The given data z in (b), also known as the *sinogram*, is constructed by taking the 2D Radon transform of the underlying image every 15 degrees (out of 180 degrees). The matrix K in this example is a discrete Radon transform of size 1235-by-4096.

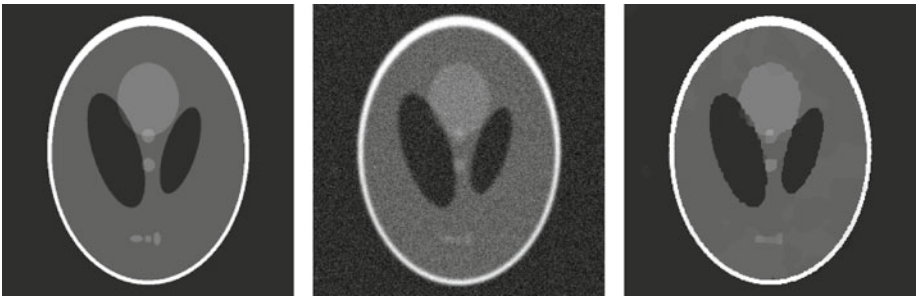


Fig. 5. Deblurring: (a) 256-by-256 Phantom. (b) Noisy blurred image; PSNR=21.7276. (c) Restored image; PSNR=27.6672.

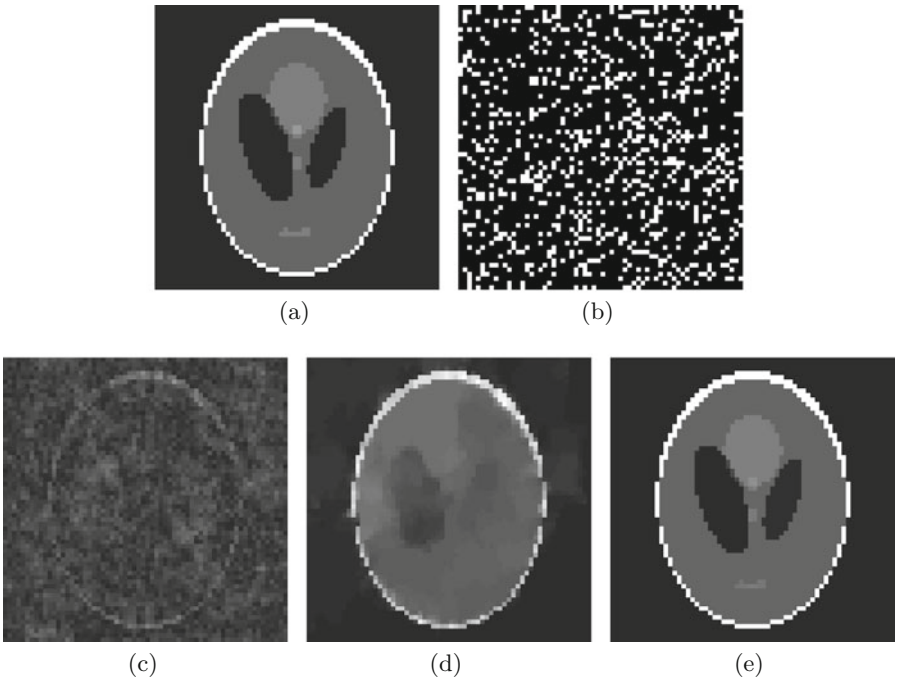


Fig. 6. Compressed sensing: (a) 64-by-64 Phantom. (b) 20 % k-space random sampling. (c) Direct reconstruction by FFT. (d) Reconstruction by TV-model. (e) Reconstruction by TV^q-model.

We utilize our TV^q-model with $q = 0.75,\ \alpha = 0.001,\ \mu = 10^{-6}$, and $u^0 = 0$. The restoration is shown in plot (e). This result clearly is superior to the one shown in plot (c), which is obtained by filtered backprojection, and the one shown in plot (d), which is obtained from the TV-model with $q = 1,\ \alpha = 0.02$,

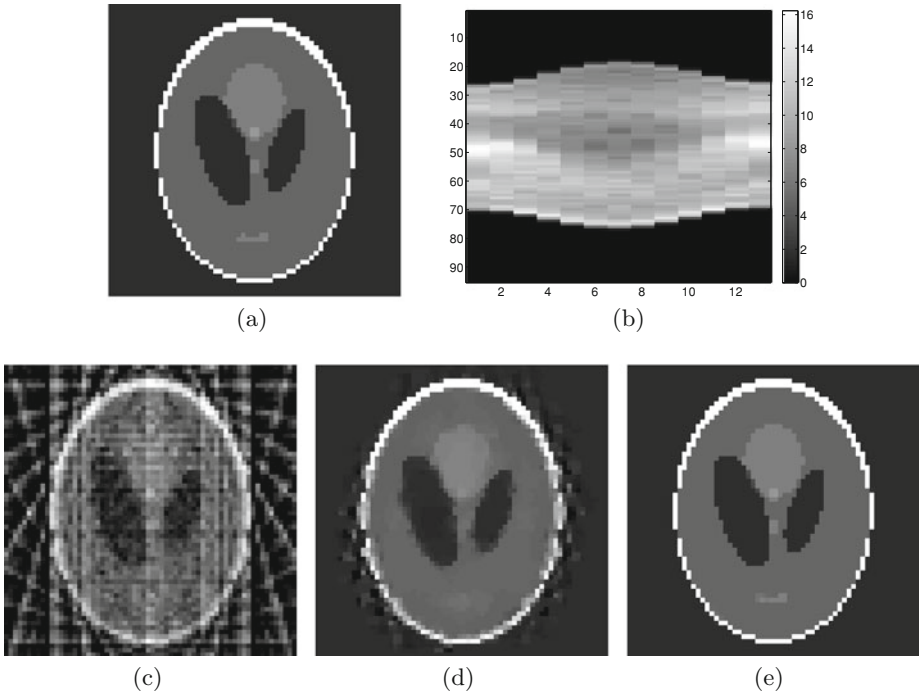


Fig. 7. Integral-geometry tomography: (a) 64-by-64 Phantom. (b) Sinogram. (c) Reconstruction by filtered backprojection. (d) Reconstruction by TV-model. (e) Reconstruction by TV^q-model.

$\mu = 10^{-6}$. In our implementation of the TV-model, α is chosen in order to obtain an image with optimal PSNR. In the right part of Table 1, we again compare the three candidate methods with respect to PSNR and CPU time. Similar to the compression sensing example, the TV^q method costs more CPU time than the other two methods (but still less than double the CPU time of the TV method) but yields an almost perfect reconstruction.

4.5 Reconstruction of Multi-coil MRI Raw Data

We now extend the methodology to magnetic resonance imaging (MRI), by considering the following model:

$$\min_{u\in\mathbb{R}^{|\Omega|}} \frac{1}{2}\sum_{l=1}^{L}\|K_l u - z_l\|^2 + \sum_{(i,j)\in\Omega}\frac{\alpha_g}{q}|(\nabla u)_{ij}|^q + \sum_{(i',j')\in\Xi}\frac{\alpha_w}{q}|(Wu)_{i'j'}|^q, \quad (9)$$

with $K_l := PF(\sigma_l u)$. Here α_g and α_w are two positive parameters, L is the number of coils of the MRI machine, (z_l) denote the raw data collected by each coil, (σ_l) are the given (or precomputed) coil sensitivities, F is the two-dimensional

Table 1. Comparison with respect to PSNR and CPU time (in seconds).

Compressed sensing				Integral-geometry tomography			
	IFT	TV	TV^q		FBP	TV	TV^q
PSNR	13.68	25.74	44.48	PSNR	15.14	23.18	51.26
CPU	$\ll 1$	6.68	12.72	CPU	$\ll 1$	3.82	7.27

discrete Fourier transform, and P represents some given radial projection operator in the k-space. Moreover, $W : \mathbb{R}^{|\Omega|} \to \mathbb{R}^{|\Xi|}$ is a user-chosen transform, typically a 2D discrete wavelet transform, and Ξ denotes the wavelet transformed domain.

Note that in addition to the TV^q regularization, we include the ℓ^q-norm of wavelet coefficients in the regularization in order to allow the reconstruction to be richer than patchwise constant images. Nevertheless, our algorithm presented in this paper can be extended without agonizing pain to the problem (9).

Indeed, as a straightforward extension of Theorem 1, the solution of (9) exists provided that $\mathrm{Ker}\nabla \cap \mathrm{Ker}W \cap (\cap_{l=1}^{L} \mathrm{Ker}K_l) = \{0\}$. The Euler-Lagrange equation for (9) appears as

$$\begin{cases} \alpha_g \nabla^\top p_g + \alpha_w W^\top p_w + \sum_{l=1}^{L} K_l^\top (K_l u - z_l) = 0, \\ (p_g)_{ij} = |(\nabla u)_{ij}|^{q-2} (\nabla u)_{ij}, & \text{if } (i,j) \in \Omega \wedge (\nabla u)_{ij} \neq 0, \\ (p_w)_{i'j'} = |(Wu)_{(i',j')}|^{q-2} (Wu)_{i'j'}, & \text{if } (i',j') \in \Xi \wedge (Wu)_{i'j'} \neq 0. \end{cases}$$

The associated Huberized problem can be analogously formulated as

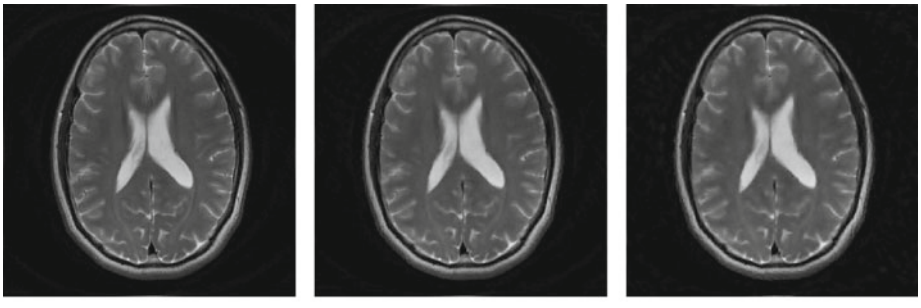
$$\min_{u \in \mathbb{R}^{|\Omega|}} \frac{1}{2} \sum_{l=1}^{L} \|K_l u - z_l\|^2 + \alpha_g \sum_{(i,j) \in \Omega} \varphi_\gamma(|(\nabla u)_{ij}|) + \alpha_w \sum_{(i',j') \in \Xi} \varphi_\gamma(|(Wu)_{i'j'}|),$$

and the corresponding Huberized Euler-Lagrange equation is given by

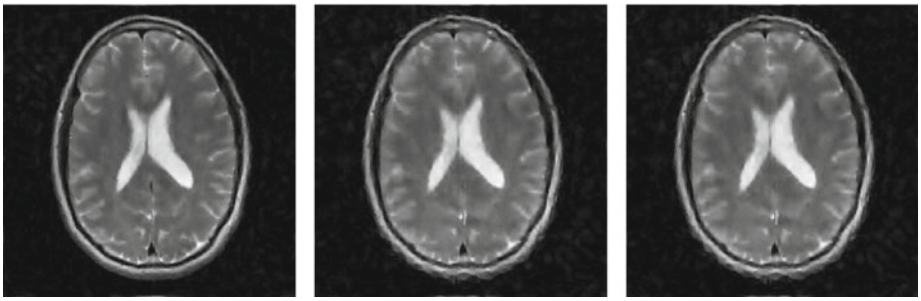
$$\alpha_g \nabla^\top (\max(|\nabla u|, \gamma)^{q-2} \nabla u) + \alpha_w W^\top (\max(|Wu|, \gamma)^{q-2} Wu) + \sum_{l=1}^{L} K_l^\top (K_l u - z_l) = 0.$$

We shall not go into further details but remark that the R-regularized method in the appendix can be used to solve the above smooth problem by treating the gradient term and the wavelet term independently. Thus, Algorithm 3 can be implemented.

In this experiment, the MRI data are collected from four coils, i.e. $L = 4$. We choose $q = 0.75$, $\alpha_g = 10^{-5}$, $\alpha_w = 2 \times 10^{-5}$, and W to be the 4th-order Daubechies wavelet transform [8]. The reconstructed images using various numbers of radial projections are shown in Fig. 8. Depending on the resolution (or detail) desired by practitioners, our method helps to reduce the necessary number of k-space samples and therefore to speed up the overall MRI data acquisition.



(a) 96 radial projections. (b) 48 radial projections. (c) 32 radial projections.



(d) 24 radial projections. (e) 16 radial projections. (f) 12 radial projections.

Fig. 8. Reconstruction of four-coil MRI raw data.

Appendix: *R*-Regularized Newton Method

Here we provide a brief and self-contained description of the R-regularized Newton method. The interested readers can find more details in the recent work [12].

A regularized-Newton-type structure generically arises in the classical lagged-diffusivity fixed-point iteration [20]. Let u^k be our current iterate in solving the Huberized problem (4). By introducing a lagged-diffusivity weight

$$w^k := \max(|\nabla u^k|, \gamma)^{q+r-2},$$

and a dual variable

$$p := w^k \max(|\nabla u|, \gamma)^{-r} \nabla u,$$

with $0 \leq r \leq 2 - q$, the reweighted Euler-Lagrange Eq. (5) appears as

$$\begin{cases} -\mu \Delta u + K^\top (Ku - z) + \alpha \nabla^\top p = 0, \\ (w^k)^{-1} \max(|\nabla u|, \gamma)^r p = \nabla u. \end{cases} \tag{10}$$

Given a current approximation (u^k, p^k), we apply a generalized linearization to (10) and obtain the generalized Newton system

$$\begin{bmatrix} -\mu \Delta + K^\top K & \alpha \nabla^\top, \\ -\widetilde{C}^k(r) \nabla & D((m^k)^{2-q} e) \end{bmatrix} \begin{bmatrix} \delta u^k \\ \delta p^k \end{bmatrix} = \begin{bmatrix} \mu \Delta u^k - K^\top (K u^k - z) - \alpha \nabla^\top p^k \\ \nabla u^k - (m^k)^{2-q} p^k \end{bmatrix},$$

where

$$m^k := \max(|\nabla u^k|, \gamma),$$
$$(\chi_{\mathcal{A}^k})_{ij} := \begin{cases} 1, \text{ if } |(\nabla u^k)_{ij}| > \gamma, \\ 0, \text{ otherwise,} \end{cases}$$
$$\widetilde{C}^k(r) := I - rD(\chi_{\mathcal{A}^k}(m^k)^{-q}p^k) \begin{bmatrix} D(\nabla_x u^k)\ D(\nabla_y u^k) \\ D(\nabla_x u^k)\ D(\nabla_y u^k) \end{bmatrix}.$$

Here $D(v)$ denotes a diagonal matrix with the vector v as diagonal entries. For $v \in \mathbb{R}^{|\Omega|}$ and $p = (p^1, p^2) \in \mathbb{R}^{2|\Omega|}$, the notation vp is understood as a vector in $\mathbb{R}^{2|\Omega|}$ such that $(vp)_{ij} = (v_{ij}p^1_{ij}, v_{ij}p^2_{ij})$ for all $(i,j) \in \Omega$.

After eliminating δp^k, we are left with the linear system

$$\widetilde{H}^k(r)\delta u^k = -\nabla f_\gamma(u^k), \tag{11}$$

where

$$\widetilde{H}^k(r) := -\mu\Delta + K^\top K + \alpha\nabla^\top D((m^k)^{q-2}e)\widetilde{C}^k(r)\nabla.$$

The Newton system (11) can be rewritten as

$$(H^k + \beta R^k)\delta u^k = -\nabla f_\gamma(u^k), \tag{12}$$

with $\beta = 2-q-r$, where $H^k := \widetilde{H}^k(2-q)$ is the Hessian from the non-reweighted primal-dual Newton method [11,20], and

$$R^k := \alpha\nabla^\top D(\chi_{\mathcal{A}^k}(m^k)^{-2}p^k) \begin{bmatrix} D(\nabla_x u^k)\ D(\nabla_y u^k) \\ D(\nabla_x u^k)\ D(\nabla_y u^k) \end{bmatrix} \nabla.$$

serves as a regularizer on the Hessian H^k. This coins the name *R-regularization* in connection with Newton's method.

Next we aim to establish a sufficient condition on the R-regularization weight β in order to guarantee that $\widetilde{H}^k(r)$ is positive definite and, therefore, δu^k is a descent direction for f_γ at u^k. For this purpose we invoke an *infeasible Newton technique* [11,12].

The infeasible Newton technique involves two modifications in constructing the system matrix $\widetilde{H}^k(r)$. First, we replace p^k by p^k_+, where

$$p^k_+ := \frac{\chi_{\mathcal{A}^k}(m^k)^{q-1}p^k}{\max((m^k)^{q-1}, |p^k|)} + (1 - \chi_{\mathcal{A}^k})p^k.$$

Note that the modified p^k_+ satisfies its feasibility condition, i.e.

$$|(p^k_+)_{ij}| \le |(\nabla u^k)_{ij}|^{q-1}, \quad \text{whenever } |(\nabla u^k)_{ij}| > \gamma. \tag{13}$$

Secondly, we replace $\widetilde{C}^k(r)$ by its symmetrization denoted by $\widetilde{C}^k_+(r)$, i.e.

$$\widetilde{C}^k_+(r) := \frac{\widetilde{C}^k(r) + \widetilde{C}^k(r)^\top}{2} = I - rD(\chi_{\mathcal{A}^k}(m^k)^{-q}) \cdot$$
$$\cdot \begin{bmatrix} D((p^k_+)_x\nabla_x u^k) & D(\frac{1}{2}((p^k_+)_x\nabla_y u^k + (p^k_+)_y\nabla_x u^k)) \\ D(\frac{1}{2}((p^k_+)_x\nabla_y u^k + (p^k_+)_y\nabla_x u^k)) & D((p^k_+)_y\nabla_y u^k) \end{bmatrix}.$$

Accordingly, the system matrix H^k in (12) is replaced by H^k_+ with

$$H^k_+ := -\mu\Delta + K^\top K + \alpha\nabla^\top D((m^k)^{q-2}e)\widetilde{C}^k_+(2-q)\nabla,$$

and the regularizer R^k is replaced by R^k_+ with

$$R^k_+ := \alpha\nabla^\top D(\chi_{\mathcal{A}^k}(m^k)^{-2})\cdot \cdot \begin{bmatrix} D((p^k_+)_x\nabla_x u^k) & D(\frac{1}{2}((p^k_+)_x\nabla_y u^k + (p^k_+)_y\nabla_x u^k)) \\ D(\frac{1}{2}((p^k_+)_x\nabla_y u^k + (p^k_+)_y\nabla_x u^k)) & D((p^k_+)_y\nabla_y u^k) \end{bmatrix}\nabla.$$

Lemma 4. *Let $0 \le r \le 1$ (or equivalently $1-q \le \beta \le 2-q$) and the feasibility condition (13) hold true. Then the matrix $\widetilde{C}^k_+(r)$ is positive semidefinite.*

The proof of Lemma 4 is given in [12]. Thus, the positive definiteness of the R-regularized Hessian $H^k_+ + \beta R^k_+$ follows immediately from its structure and Lemma 4.

Theorem 5. *Suppose the assumptions of Lemma 4 are satisfied. Then the R-regularized Hessian $H^k_+ + \beta R^k_+$ is positive definite.*

We remark that our choice of μ is related to Theorem 6. Under the condition (2), for any positive μ the matrix $-\mu\Delta + K^\top K$ is positive definite, and therefore Theorem 6 follows. However, whenever K is injective the same conclusion holds with $\mu = 0$. This is why we are allowed to choose $\mu = 0$ in the denoising example in Sect. 4.1.

Given the result in Theorem 5, the descent direction δu^k in the R-regularized Newton method implemented in step 1 of Algorithm 3 can be now obtained by solving the linear system

$$(H^k_+ + \beta R^k_+)\delta u^k = -\nabla f_\gamma(u^k),$$

with $1-q \le \beta \le 2-q$. Given δu^k, one can compute δp^k as

$$\delta p^k = (m^k)^{q-2}(\nabla u^k + \widetilde{C}^k(r)\nabla\delta u^k) - p^k,$$

and then update $u^{k+1} := u^k + a^k\delta u^k$ and $p^{k+1} := p^k + a^k\delta p^k$ with some step size a^k determined by the Armijo line search as in step 2 of Algorithm 3.

In all experiments in Sect. 4, we consistently choose $\beta = 1-q$. This choice has the interesting interpretation that we actually relax the TV^q-model to a weighted TV-model (with weight updating in the outer iterations); see [12].

Acknowledgement. The authors would like to thank Dr. Florian Knoll for contributing his data and codes to our experiments on MRI.

References

1. Burke, J.V., Lewis, A.S., Overton, M.L.: A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. SIAM J. Optim. **15**, 751–779 (2005)

2. Chan, T.F., Mulet, P.: On the convergence of the lagged diffusivity fixed point method in total variation image restoration. SIAM J. Numer. Anal. **36**, 354–367 (1999)
3. Chartrand, R.: Exact reconstruction of sparse signals via nonconvex minimization. IEEE Signal Process. Lett. **14**, 707–710 (2007)
4. Chartrand, R.: Fast algorithms for nonconvex compressive sensing: MRI reconstruction from very few data. In: Proceedings of the IEEE International Symposium on Biomedical Imaging, pp. 262–265 (2009)
5. Chartrand, R., Yin, W.: Iteratively reweighted algorithms for compressive sensing. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3869–3872 (2008)
6. Chen, X., Zhou, W.: Smoothing nonlinear conjugate gradient method for image restoration using nonsmooth nonconvex minimization. SIAM J. Imaging Sci. **3**, 765–790 (2010)
7. Clarke, F.H.: Optimization and Nonsmooth Analysis. John Wiley & Sons, New York (1983)
8. Daubechies, I.: Ten Lectures on Wavelets. SIAM, Philadelphia (1992)
9. Daubechies, I., DeVore, R., Fornasier, M., Güntürk, C.: Iteratively reweighted least squares minimization for sparse recovery. Comm. Pure Appl. Math. **63**, 1–38 (2010)
10. Grippo, L., Lampariello, F., Lucidi, S.: A nonmonotone line search technique for Newton's method. SIAM J. Numer. Anal. **23**, 707–716 (1986)
11. Hintermüller, M., Stadler, G.: An infeasible primal-dual algorithm for total bounded variation-based inf-convolution-type image restoration. SIAM J. Sci. Comput. **28**, 1–23 (2006)
12. Hintermüller, M., Wu, T.: Nonconvex TV^q-models in image restoration: analysis and a trust-region regularization based superlinearly convergent solver. SIAM J. Imaging Sciences (to appear)
13. Kiwiel, K.C.: Convergence of the gradient sampling algorithm for nonsmooth nonconvex optimization. SIAM J. Optim. **18**, 379–388 (2007)
14. Mäkelä, M., Neittaanmäki, P.: Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control. World Scientific, River Edge (1992)
15. Nikolova, M., Ng, M.K., Tam, C.-P.: Fast nonconvex nonsmooth minimization methods for image restoration and reconstruction. IEEE Trans. Image Process. **19**, 3073–3088 (2010)
16. Nikolova, M., Ng, M.K., Zhang, S., Ching, W.-K.: Efficient reconstruction of piecewise constant images using nonsmooth nonconvex minimization. SIAM J. Imaging Sci. **1**, 2–25 (2008)
17. Nocedal, J., Wright, S.: Numerical Optimization, 2nd edn. Springer, New York (2006)
18. Rudin, L., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Physica D **60**, 259–268 (1992)
19. Schramm, H., Zowe, J.: A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results. SIAM J. Optim. **2**, 121–152 (1992)
20. Vogel, C.R., Oman, M.E.: Iterative methods for total variation denoising. SIAM J. Sci. Comput. **17**, 227–238 (1996)

A Fast Continuous Max-Flow Approach to Non-convex Multi-labeling Problems

Egil Bae[1(✉)], Jing Yuan[2], Xue-Cheng Tai[3], and Yuri Boykov[2]

[1] Department of Mathematics, University of California, Los Angeles, USA
ebae@math.ucla.edu
[2] Computer Science Department, Middlesex College,
University of Western Ontario, London, ON, Canada
cn.yuanjing@gmail.com, yuri@csd.uwo.ca
[3] Department of Mathematics, University of Bergen, Bergen, Norway
tai@math.uib.no

Abstract. This paper studies continuous image labeling problems with an arbitrary data term and a total variation regularizer, where the labels are constrained to a finite set of real numbers. Inspired by Ishikawa's multi-layered graph construction for the same labeling problem over a discrete image domain, we propose a novel continuous max-flow model and build up its duality to a convex relaxed formulation of image labeling under a new variational perspective. Via such continuous max-flow formulations, we show that exact and global optimizers can be obtained to the original non-convex labeling problem. We also extend the studies to problems with continuous-valued labels and introduce a new theory to this problem. Finally, we show the proposed continuous max-flow models directly lead to new fast flow-maximization algorithmic schemes which outperform previous approaches in terms of efficiency. Such continuous max-flow based algorithms can be validated by convex optimization theories and accelerated by modern parallel computational hardware.

1 Introduction

Many practical problems in image processing and computer vision can be modeled as multilabel problems, where the task is to optimally assign the unknown variable l, chosen from some finite set $\{l_1, \ldots, l_n\}$, at each point of the image domain Ω. It has become an important paradigm to formulate such labeling problems as the optimization of an energy function/functional which mathematically encodes all the information needed for the specified imaging and vision task. Such optimization problems can be formulated by either regarding the image domain as discrete or continuous.

In the spatially discrete setting, graph cut has become one of the most important and efficient techniques to tackle such problems, by computing max-flow or min-cut on appropriately constructed graphs. Applications of max-flow/min-cut in computer vision range from image segmentation or labeling [1,2], stereo [3,4], 3D reconstruction [5] etc. Unfortunately, most minimization problems

A. Bruhn et al. (Eds.): Global Optimization Methods, LNCS 8293, pp. 134–154, 2014.
DOI: 10.1007/978-3-642-54774-4_7,

involving more than two labels are NP-hard, therefore only approximate algorithms are available [1,4]. However, for a particular set of multilabeling problems with convex interaction penalty, Ishikawa [6] showed that exact solutions can be computed by max-flow and min-cut. Such energy functions are important in e.g. stereo reconstruction. Despite the efficiencies of graph-based methods, their computation results are often comparatively rough and biased by the discrete graph setting, i.e. metrication errors occur. Reducing such visual artifacts requires either considering more neighbour nodes, which increases memory burden largely, or applying more complex schemes such as high-order potentials [7].

Recently, the variational approach has become more and more popular for obtaining optimal labelings in the spatially continuous setting, where the problem is formulated as the minimization of a continuous energy functional. In contrast to the graph-based model, there are many advantages of the variational approach: the variational model perfectly avoids metrication errors due to its crucial rotation invariance; moreover, its reduced continuous numerical scheme is reliable, tractable, and can be easily implemented and accelerated in many different ways, e.g. parallel, multigrid or GPU hardwares; last but not least, the continuous models require far less memory in computation.

The application of variational methods to optimal labelings is often to relax the combinatorial constraints to a proper convex set. It leads to a constrained convex minimization problem such that global and exact optimums, in some special cases, are available. For example, Chan et al. [8] showed that global and exact binary optimizers can be obtained by thresholding the computation result of the convex relaxed model; therefore, a sequence of so-called continuous min-cuts can be obtained. [9,10] generalized Ishikawa's work [6] to the spatially continuous setting, where both the image domain and label values are continuous, by representing the optimal labeling function as the discontinuity set of a binary function in a one-dimensional higher space, i.e. a spatially continuous min-cut. Such a lifting approach is related to earlier mathematical theories of calibrations and Cartesian currents [11,12]. Optimal labeling functions could be obtained by applying the result of Chan et al. in the higher dimensional space, i.e. first solve the relaxed binary problem and then threshold the result.

Motivations and Contributions

For discrete graphs, it is well known that the minimum cut problem is dual to the maximum flow problem by the *max-flow and min-cut theorem* [13]. Actually, the fastest graph cut algorithms are based on maximizing flow instead of computing the min-cut directly, e.g. the Ford-Fulkerson algorithm [14] and the push-relabel algorithm [15]. The minimal 'cut' is finally recovered along edges with 'saturated' flows, i.e. cuts appear at the flow-bottlenecked edges [4,16]. In contrast, max-flow models and algorithms in the spatially continuous setting have been much less studied. Some work has appeared that deal with partitioning problems involving two regions: Strang [17] was the first to formulate max-flow and min-cut problems over a continuous domain; In [18], edge based max-flow and min-cut was formulated in which certain interior and exterior points must be

specified in advance; Yuan et al. [19,20] proposed a direct continuous analogue of the typical discrete max-flow and min-cut models that are used for solving binary labeling problems in image processing and computer vision. Max-flow and min-cut interpretations of recent convex relaxations for Potts model have been made in [21]. However in these cases there is generally a duality gap and the original problems can only be solved approximately.

To our knowledge, this is the first work to address continuous max-flow and min-cut duality for problems where the labeling function can take several discrete values. Motivated by Ishikawa [6] and Yuan et al. [19], we interpret the problem as a continuous min-cut problem over a mixed continuous/discrete domain and build up a novel continuous max-flow model in analogy with Ishikawa's discrete graph construction. The max-flow model can be used to produce global solutions of the original non-convex problem with discrete label values. In particular, it is shown that the max-flow model is dual to an exact convex relaxation of the original problem. Strict duality is also established between the max-flow model and the original problem, by extending the thresholding scheme of [8] from two to several regions.

A new continuous max-flow based algorithm is proposed. Its efficiency and convergence can be verified by standard convex optimization theories. The labeling function is updated as an unconstrained lagrange multiplier each iteration, and does not need to be projected back onto any feasible set. Numerical experiments show a significantly faster convergence rate than the primal-dual algorithm in Pock et al. [9,10] and later [22], especially at high precisions.

A significantly extended version of this paper is available at [23], which contains extensions to other regularizers and more experiments. This conference paper contains some novelties which are not in [23], such as the discussion on saturated/unsaturated edges in Sect. 3.5.

2 Preliminaries: Ishikawa's Work

Ishikawa [6] studied image labeling problems over an image graph which can be generally formulated as:

$$\min_{u\in U} \sum_{v\in\mathcal{P}} \rho(u_v, v) + \alpha \sum_{(v,w)\in\mathcal{N}} g(u_v - u_w)\,, \tag{1}$$

where $\mathcal{P}$ denotes a discrete image grid in 2-D or N-D; $\mathcal{N} \subset \mathcal{P}\times\mathcal{P}$ is a neighborhood system on $\mathcal{P}$; $U = \{u : \mathcal{P} \mapsto L$ is the set of all feasible labeling functions, where $L = \{\ell_1, ..., \ell_n\}\}$. The potential prior $g(x)$ in (1) is assumed to be convex and ρ is any bounded function, but not necessarily convex. It was shown by [6] that problems of the form (1) can be exactly optimized by finding the minimal cut over a specially constructed multi-layered graph $G = (\mathcal{V}, \mathcal{E})$, where each layer corresponds to one label.

We adopt Ishikawa's notations [6] in this work and study the simplified graph which uses $n-1$ layers instead of n and avoids infinite capacities on the source edges [24] (see Fig. 1 for a 1-D example). The vertex set $\mathcal{V}$ and the edge set $\mathcal{E}$ are defined as follows:

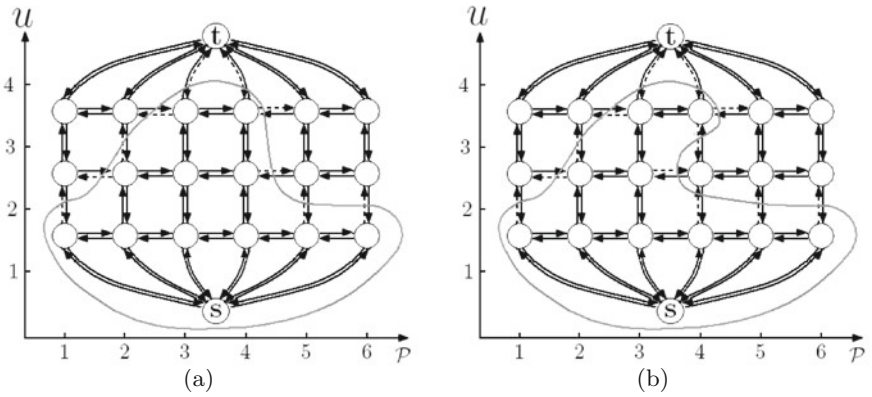


Fig. 1. 1D illustration: (a) Legal cut, (b) Illegal cut. Severed edges are depicted as dotted arrows. The gray curve visualizes the cut. Vertices interior to the curve belongs to V_s while vertices exterior to the curve belongs to V_t.

$$\mathcal{V} \;=\; \mathcal{P} \times L \cup \{s,t\} \;=\; \{u_{v,i}\,|v \in \mathcal{P}\,;\; i = 1,...,n-1\} \cup \{s,t\} \tag{2a}$$

$$\mathcal{E} \;=\; \mathcal{E}_D \cup \mathcal{E}_C \cup \mathcal{E}_P \tag{2b}$$

where the edge set $\mathcal{E}$ is composed of three types of edges

– Data edges $\mathcal{E}_D = \bigcup_{v\in\mathcal{P}} \mathcal{E}_D^v$, where

$$\mathcal{E}_D^v \;=\; (s, u_{v,1}) \cup \{(u_{v,i}, u_{v,i+1})\,|\,i = 1,\ldots,n-2\} \cup (u_{v,n-1}, t)\,. \tag{3}$$

– Penalty edges $\mathcal{E}_P = \bigcup_{v\in\mathcal{P}} \mathcal{E}_C^v$, where

$$\mathcal{E}_C^v \;=\; (u_{v,1}, s) \cup \{(u_{v,i+1}, u_{v,i}) \cup (t, u_{v,n-1})\,|\,i = 1,\ldots,n-2\}\,. \tag{4}$$

– Regularization edges $\mathcal{E}_R$:

$$\mathcal{E}_R \;=\; \{(u_{v,i}, u_{w,j})\,|\,(v,w) \in \mathcal{N}\,,\; i,j = 1,...,n\}\,. \tag{5}$$

2.1 Anisotropic Total-Variation Regularization

When a pairwise prior $g(u_v - u_w) = C(v,w)\,|u_v - u_w|$ is given, (1) corresponds to an anisotropic total-variation regularized image labeling problem, i.e.

$$\min_{u\in U} \sum_{v\in\mathcal{P}} \rho(u_v, v) + \alpha \sum_{(v,w)\in\mathcal{N}} C(v,w)\,|u_v - u_w| \tag{6}$$

which is a discrete counterpart of the total-variation regularizer.

Now we define flow configurations over the graph (2a) and (2b) such that its max-flow corresponds to the minimizer of (6):

– *Capacity of source flows:* the directed flow $p_1(v)$ along each edge from the source s to the node $u_{v,1}$ of the first layer, i.e. the edge $(s, u_{v,1})$, is constrained by

$$p_1(v) \leq \rho(\ell_1, v)\,, \quad \forall v \in \mathcal{P}\,; \tag{7}$$

– *Capacity of flows between layers:* the directed flow $p_i(v)$ along each edge $(u_{v,i}, u_{v,i+1})$ from the node $u_{v,i}$ of the i-th layer to the node $u_{v,i+1}$ of the $i+1$-th layer is constrained by

$$p_i(v) \leq \rho(\ell_i, v)\,, \quad \forall v \in \mathcal{P} \quad i = 1, ..., n-2 \tag{8}$$

– *Capacity of sink flows:* the directed flow $p_n(v)$ along each edge from the node $u_{v,n-1}$ of the last layer to the sink t is constrained by

$$p_n(v) \leq \rho(\ell_n, v)\,, \quad \forall v \in \mathcal{P}\,; \tag{9}$$

– *Capacity of spatial flows at each layer:* the undirected flow $q_i(v, w)$ of each edge $(v, w) \in \mathcal{N}$ at the layer i, $i = 1, \ldots, n-1$, is constrained by

$$|q_i(v, w)| \leq C(v, w)\,; \tag{10}$$

this corresponds to the well-known anisotropic total-variation regularizer in case of a 4 nearest neighborhood system $\mathcal{N}$;

– *Conservation of flows:* flow conservation means that in-coming flows should be balanced by out-going flows at any node $v \in \mathcal{P}$ of each layer $i = 1, ..., n-1$, i.e.

$$\Big(\sum_{w:(w,v)\in\mathcal{N}} q_i(v, w) - \sum_{w:(v,w)\in\mathcal{N}} q_i(v, w) \Big) - p_i(v) + p_{i+1}(v) \;=\; 0\,. \tag{11}$$

Since there is no lower bound on the flows (7)–(9), the flow on the penalty edges (4) can become arbitrarily large. This implies that each edge in the set $\mathcal{E}_D^v$ which links the source and sink can only be cut once, i.e. illegal cuts as shown in Fig. 1(b) have infinite cost and are not allowed.

Therefore, the max-flow problem over the graph is to find the largest amount of flow allowed to pass from the source s to sink t through the $n-1$ graph layers, i.e.

$$\max_{p,q} \sum_{v\in\mathcal{P}} p_1(v) \tag{12}$$

subject to the flow constraints (7), (8), (9), (10) and (11).

Due to duality between the max-flow and min-cut problem [13], one can solve the max-flow problem and then extract a solution to the min-cut problem (6).

3 Multilabeling by Continuous Max-Flow and Min-Cut

Define the feasible set of functions as $U = \{u : \Omega \mapsto \{\ell_1, ..., \ell_n\}$ s.t. $\int_\Omega |\nabla u| \leq \infty\}$, where $\ell_1 < ... < \ell_n$ are real numbers. The continuous counterpart of (1) can be formulated as

$$\min_{u\in U} \int_\Omega \rho(u(x), x)\, dx + \int_\Omega C(x)|\nabla u(x)|\, dx\,, \tag{13}$$

where $\rho : \mathbb{R} \times \Omega \mapsto \mathbb{R}$ is any bounded function, not necessarily convex. The set U is a non-convex set of discrete valued labeling functions. This is in contrast to Pock et al. who considered a convex feasible set of continuous valued labeling functions. We show this problem can be regarded as a continuous min-cut problem by following the ideas of Ishikawa.

We start by rewriting (13) in terms of the upper level sets of $u \in U$

$$\lambda_i(x) = \begin{cases} 1\,, & \text{if } u(x) > \ell_i \\ 0\,, & \text{if } u(x) \leq \ell_i \end{cases}, \forall x \in \Omega \quad i = 1, \ldots, n-1\,.$$

Let $\lambda_0(x) = 1$ and $\lambda_n(x) = 0$, a.e. $x \in \Omega$. Clearly, we have

$$1 = \lambda_0(x) \geq \lambda_1(x) \geq \lambda_2(x) \geq \ldots \geq \lambda_{n-1}(x) \geq \lambda_n(x) = 0 \text{ a.e. } x \in \Omega. \tag{14}$$

By the coarea formula, we have for any function $u \in U$ that

$$\int_\Omega C(x)|\nabla u|\,dx \;=\; \sum_{i=1}^{n-1} \int_\Omega C_i(x)|\nabla \lambda_i|\,dx\,,$$

where $C_i(x) = (\ell_{i+1} - \ell_i)C(x),\ i = 1, \ldots, n-1$. In this work, we will mostly focus on the case where $C(x) = \alpha$ is constant for simplicity.

Therefore, (13) can be equivalently rewritten as

$$\min_{\{\lambda_i\}_{i=1}^{n-1} \in \mathcal{B}} \sum_{i=1}^{n} \int_\Omega (\lambda_{i-1} - \lambda_i)\,\rho(\ell_i, x)\,dx \;+\; \alpha \sum_{i=1}^{n-1} (\ell_{i+1} - \ell_i) \int_\Omega |\nabla \lambda_i|\,dx \tag{15}$$

subject to the constraint (14), where the binary constraint $\mathcal{B}$ is defined as

$$\mathcal{B} = \{\phi \,:\, \Omega \mapsto \{0, 1\},\ \text{s.t.}\ \int_\Omega |\nabla \phi| < \infty\} \tag{16}$$

The problem (15) is obviously a nonconvex optimization problem due to the binary constraints (16).

After solving (15), the labeling function u can be recovered from λ_i by $u = \sum_{i=1}^{n} (\lambda_{i-1} - \lambda_i)\ell_i$.

3.1 Primal Model: Continuous Max-Flow

In this section, we build up a max-flow model in continuous settings, which simulates Ishikawa's graph configuration. It will be shown that solutions of (15) and (13) can be obtained by exploring the dual of this maximization problem.

To this end, we place $n-1$ image domains Ω_i, $i = 1, \ldots, n-1$ with $\Omega_i = \Omega$, layered in a sequential order between two terminals: the source s and the sink t. The source s is linked to each image pixel x of the first layer Ω_1 by an edge $e_1(x)$; the same image pixel x between two sequential image layers Ω_{i-1} and Ω_i, $i = 2, \ldots, n-1$, is linked by the edge $e_i(x)$; and the pixel x at the last layer Ω_{n-1} is also linked to the sink t by the edge $e_n(x)$. Define flow functions $p_i : \Omega \mapsto \mathbb{R}$

corresponding to each edge function e_i , $i = 1, \ldots, n$. Within each image layer Ω_i, $i = 1, \ldots, n-1$, the spatial flow functions are given by $q_i \in C^\infty(\Omega)^N$, where N is the dimension of the image domain.

As a generalization of the discrete constraints (7)–(11), we now give constraints on flow functions $p_i \in L^1(\Omega)$, $i = 1, \ldots, n$, and $q_i \in C^\infty(\Omega)^N$, $i = 1, \ldots, n-1$

$$|q_i(x)| \le C_i(x) \qquad \text{for } x \in \Omega\,,\ i = 1, \ldots, n-1 \tag{17}$$

$$p_i(x) \le \rho(\ell_i, x) \qquad \text{for } x \in \Omega\,,\ i = 1, \ldots, n \tag{18}$$

$$\big(\operatorname{div} q_i - p_i + p_{i+1}\big)(x) = 0 \qquad \text{for } x \in \Omega\,,\ i = 1, \ldots, n-1 \tag{19}$$

$$q_i \cdot n = 0 \qquad \text{on } \partial\Omega\,,\ i = 1, \ldots, n-1\,. \tag{20}$$

Therefore, the continuous max-flow model, in analogue with Ishikawa's discrete one (12), can be formulated by

$$\sup_{p,q} E^P(p) = \int_\Omega p_1(x)\, dx \tag{21}$$

subject to the constraints (17)–(20). In this work, we call (21) the *primal model*. Observe the maximization problem (21) is bounded above by $\int_\Omega \rho(\ell_1(x), x)\, dx$ due to the constraint (18).

3.2 Primal-Dual Model

By introducing multiplier functions $\lambda_i(x)$, $i = 1, \ldots, n-1$, to the linear equality constraints of flow conservation (19), we have the equivalent *primal-dual model* of (21):

$$\inf_{\lambda} \sup_{p,q} \quad E(p, q; \lambda) = \int_\Omega \Big\{ p_1 + \sum_{i=1}^{n-1} \lambda_i \big(\operatorname{div} q_i - p_i + p_{i+1} \big) \Big\}\, dx \tag{22}$$

subject to (17), (18) and (20).

After rearrangement, the above primal-dual formulation (22) can be equivalently written as

$$\inf_{\lambda} \sup_{p,q} \quad E(p, q; \lambda) = \sum_{i=1}^{n} \int_\Omega (\lambda_{i-1} - \lambda_i) p_i\, dx + \sum_{i=1}^{n-1} \int_\Omega \lambda_i \operatorname{div} q_i\, dx \tag{23}$$

subject to (17), (18) and (20).

3.3 Dual Model: Continuous Min-Cut

Now we show that optimizing the primal-dual model (23) over all the flow functions p and q leads to the equivalent *dual model*, i.e. the continuous min-cut model:

$$\inf_{\lambda} E^D(\lambda) = \sum_{i=1}^{n} \int_\Omega (\lambda_{i-1} - \lambda_i) \rho(\ell_i, x)\, dx + \sum_{i=1}^{n-1} \int_\Omega C_i(x)\, |\nabla \lambda_i|\, dx \tag{24}$$

$$\text{s.t.} \quad 1 = \lambda_0(x) \ge \lambda_1(x) \ge \ldots \ge \lambda_{n-1}(x) \ge \lambda_n(x) = 0\,, \quad \forall x \in \Omega\,.$$

Optimization of Flow Functions. In this regard, we consider the optimization problem

$$f(v) \;=\; \sup_{w \le C} v \cdot w \,, \tag{25}$$

where v, w and C are scalars. When $v < 0$, w can be arbitrarily large in order to maximize the value $v \cdot w$, i.e. $f(v) = +\infty$. Therefore, we must have $v \ge 0$ so as to make the function $f(v)$ meaningful and

$$\begin{cases} \text{if } v = 0\,, \text{ then } w < C \text{ and } f(v) \text{ reaches its maximum } 0 \\ \text{if } v > 0\,, \text{ then } w = C \text{ and } f(v) \text{ reaches its maximum } v \cdot C \end{cases} .$$

Therefore, we have

$$f(v) \;=\; \begin{cases} v \cdot C\,, & v \ge 0\,, \\ \infty & v < 0 \end{cases} . \tag{26}$$

The function $f(v)$ given in (25) provides us with a prototype to maximize the flow functions $p_i(x)$, $i = 1, \ldots, n$, in the primal-dual model (23).

For each $x \in \Omega$, consider

$$f_i(x) \;=\; \sup_{p_i(x) \le \rho(\ell_i, x)} \left(\lambda_{i-1}(x) - \lambda_i(x)\right) p_i(x)\,, \quad i = 1, \ldots, n\,.$$

In view of (26), we have

$$f_i(x) \;=\; \begin{cases} \left(\lambda_{i-1}(x) - \lambda_i(x)\right) \rho(\ell_i, x)\,, & \lambda_{i-1}(x) \ge \lambda_i(x) \\ \infty & \lambda_{i-1}(x) < \lambda_i(x) \end{cases} , \; i = 1, \ldots, n\,. \tag{27}$$

On the other hand, it is well known that for any $\lambda_i \in BV(\Omega)$

$$\sup_{q_i} \int_\Omega \lambda_i(x) \operatorname{div} q_i(x)\, dx \;=\; \int_\Omega C_i(x) |\nabla \lambda_i(x)|\, dx\,, \tag{28}$$

when q_i is optimized over the set (17) and (20). In view of (27) and (28), maximizing (23) over all the flow functions p and q leads directly to the equivalent dual model (24). The constraints (14) must be satisfied for an optimal ϕ, otherwise the energy would be infinite, contradicting boundedness of the max-flow problem from above.

Note that a solution to the problem (24) exists since (24) is convex, lower semi-continuous and bounded from below and the constraints (24) are convex. Regarding existence of a solution to the max-flow problem (21), due to boundedness from above a maximizing sequence $\{p^i, q^i\}_{i=1}^\infty$ exists to the problem (21). However, it may not admit a maximizing subsequence w.r.t. q^i which converges to a $q^* \in C_\nu$ because the supremum may be attained for a discontinuous q^* which lies in the closure of the set of smooth vector fields $C^\infty(\Omega)^N$ and not in the set itself. In this paper we still speak of (p^*, q^*) as a primal-dual solution even though q^* may be discontinuous to ease readability. A more formal presentation can be given if arguments involving (p^*, q^*) are replaced with $\lim_{i \to \infty} \{p^i, q^i\}_{i=1}^\infty$ for the maximizing sequence $\{p^i, q^i\}_{i=1}^\infty$.

3.4 Exact and Global Optimums

The functions λ_i, $i = 1 \dots n-1$, of the convex model (24) are relaxed to take values in the convex set $[0, 1]$, which is in contrast to the binary constraints of the original nonconvex formulation (15). The following proposition establishes a primal-dual relationship between the max-flow problem (21) and the original non-convex problem (15). By solving the max-flow problem (21) a set of optimizers to the original binary constrained problem (15) can be obtained by thresholding each layer function λ_i^*.

Proposition 1. *Assume ϕ^* is a minimizer of (24) and let $\{t_i\}_{i=1}^{n-1}$ be a sequence such that $0 < t_1 = t_2 = ... = t_{n-1} \leq 1$. Define the level sets*

$$S_i^{t_i} = \{x \,:\, \lambda_i^*(x) \geq t_i\}\,, \quad i = 1 \dots n-1 \tag{29}$$

and let $\lambda_i^{t_i}(x)$ be the characteristic function of $S_i^{t_i}$, i.e.

$$\lambda_i^{t_i}(x) := \begin{cases} 1\,, \lambda_i^*(x) \geq t_i \\ 0\,, \lambda_i^*(x) < t_i \end{cases}.$$

then the set of binary functions $\lambda_i^{t_i}(x)$, $i = 1, \dots, n-1$, is a global optimum of the original nonconvex multi-labeling problem (15). Furthermore, if $(p^*, q^*; \lambda^*)$ is any optimal primal-dual solution of (22), the cut given by $\lambda_i^{t_i}(x)$, $i = 1, \dots, n-1$, has an energy equal to the max flow energy in (21), i.e.

$$E^D(\lambda^t) = \int_\Omega p_1^*(x)\, dx = E^P(p^*).$$

Proof. Since p_i^*, $i = 1, ..., n$ and q_i^*, λ_i^*, $i = 1, ..., n-1$ is a global optimum of the primal-dual problem (22), then p_i^*, q_i^* optimize the dual problem (21) and $\lambda_i^*(x)$ optimizes (24).

For simplification reasons, define $t_0 = 0$ such that $S_0^{t_0} = \Omega$. Since l_i is increasing with i we must have

$$S_0^{t_0} \supseteq S_1^{t_1} \supseteq S_2^{t_2} \supseteq ... \supseteq S_{n-1}^{t_{n-1}}$$

Since the variables are optimal, the flow conservation condition (19) must hold, i.e

$$\operatorname{div} q_i^*(x) - p_i^*(x) + p_{i+1}^*(x) \;=\; 0\,, \quad \text{a.e. } x \in \Omega,\; i = 1, ..., n-1.$$

The proof is given by induction in $S_i^{t_i}$. For any $k \in \{1, ..., n-1\}$ define the function

$$E^k \;= \sum_{i=1}^{k} \int_{S_{i-1}^{\ell_{i-1}} \setminus S_i^{t_i}} \rho(\ell_i, x)\, dx + \int_{S_k^{\ell_k}} p_{k+1}^*(x)\, dx + \alpha \sum_{i=1}^{k} L_{S_i^{t_i}}$$

where $L_{S_i^{t_i}}$ is the length $|\partial S_i^{t_i} \setminus (\partial S_i^{t_i} \cap \partial\Omega)|$. We will prove $E^k = E^P(p^*)$ for any $k \in \{1, ..., n-1\}$ and start by considering $k = 1$. By the formula (27), it follows that

$$p_1^*(x) = \rho(\ell_1, x), \quad \text{for any point } \; x \in \Omega \backslash S_1^{t_1} = S_0^{t_0} \backslash S_1^{t_1}$$

This, together with the fact that

$$p_1^*(x) = p_2^*(x) + \operatorname{div} q_1^*(x), \quad \text{a.e. } x \in S_1^{t_1}$$

implies that the total max-flow energy defined in (21) can be written

$$\begin{aligned} E^P(p^*) &= \int_{\Omega \backslash S_1^{t_1}} \rho(\ell_1, x)\, dx + \int_{S_1^{t_1}} \left(p_2^*(x) + \operatorname{div} q_1^*(x)\right) dx \\ &= \int_{\Omega \backslash S_1^{t_1}} \rho(\ell_1, x)\, dx + \int_{S_1^{t_1}} p_2^*(x)\, dx + \int_{S_1^{t_1}} \operatorname{div} q_1^*(x)\, dx \\ &= \int_{S_0^{t_0} \backslash S_1^{t_1}} \rho(\ell_1, x)\, dx + \int_{S_1^{t_1}} p_2^*(x)\, dx + \alpha L_{S_1^{t_1}} = E^1 \end{aligned}$$

The last term follows because

$$\int_{S_i^{\ell_i}} \operatorname{div} q_i^*(x)\, dx \;=\; \int_{\Omega} \lambda_i^* \operatorname{div} q_i^*\, dx \;=\; \alpha \int_{\Omega} |\nabla \lambda_i^{\ell_i}|\, dx = \alpha \left| \partial S_i^{\ell_i} \backslash (\partial S_i^{t_i} \cap \partial \Omega) \right| . \tag{30}$$

where the second equality is due to Prop. 4 of [25]. Note that the boundary length $L_{S_1^{t_1}}$ is necessarily finite, otherwise the energy would be infinite, contradicting boundedness from above.

Assume now that $E^k = E^P(p^*)$ for some $k \in \{1, ..., n-2\}$, we will show this implies $E^{k+1} = E^P(p^*)$

$$E^P(p^*) = E^k \;= \sum_{i=1}^{k-1} \int_{S_{i-1}^{\ell_{i-1}} \backslash S_i^{t_i}} \rho(t_i, x)\, dx + \int_{S_{k-1}^{\ell_{k-1}}} p_k^*(x)\, dx + \alpha \sum_{i=1}^{k-1} L_{S_i^{t_i}} .$$

By the definition (29) it follows that $\lambda_{k-1}(x) - \lambda_k(x) > t_{k-1} - t_k = 0$ for all $x \in S_{k-1}^{t_{k-1}} \backslash S_k^{t_k}$. Therefore, by formula (27), for any point $x \in S_{k-1}^{t_{k-1}} \backslash S_k^{t_k}$ we must have $p_k^*(x) = \rho(\ell_k, x)$. Combining this with the fact that

$$p_k^*(x) = p_{k+1}^*(x) + \operatorname{div} q_k^*(x), \text{ a.e. } x \in \Omega$$

the above expression can be written

$$\begin{aligned} E^P(p^*) = E^k \;= &\sum_{i=1}^{k-1} \int_{S_{i-1}^{\ell_{i-1}} \backslash S_i^{t_i}} \rho(t_i, x)\, dx + \int_{S_{k-1}^{\ell_{k-1}} \backslash S_k^{\ell_k}} \rho(\ell_k, x)\, dx \\ &+ \int_{S_k^{\ell_k}} p_{k+1}^*(x)\, dx + L_{S_k^{t_k}} + \alpha \sum_{i=1}^{k-1} L_{S_i^{t_i}} \;= E^{k+1}. \end{aligned} \tag{31}$$

Hence, we can conclude that also $E^{n-1} = E^P(p^*)$. By noting from (27) that for all $x \in S_{n-1}^{t_{n-1}}$ we must have $p_n^*(x) = \rho(\ell_n, x)$, the total max flow energy defined

in (21) can be written

$$E^P(p^*) = E^{n-1} = \int_{\Omega \setminus S_1^{t_1}} \rho(\ell_1, x)\, dx + \sum_{i=2}^{n-1} \int_{S_{i-1}^{\ell_{i-1}} \setminus S_i^{t_i}} \rho(t_i, x)\, dx + \int_{S_{n-1}^{t_{n-1}}} \rho(\ell_n, x)\, dx + \alpha \sum_{i=1}^{n-1} L_{S_i^{t_i}} \tag{32}$$

By writing this expression in terms of the characteristic functions $\lambda_i^{t_i}$ of each region $S_i^{t_i}$, we get

$$E^P(p^*) = \sum_{i=1}^{n} \int_\Omega (\lambda_{i-1}^{\ell_{i-1}}(x) - \lambda_i^{t_i}(x))\, \rho(t_i, x)\, dx \; + \; \alpha \sum_{i=1}^{n-1} \int_\Omega |\nabla \lambda_i^{t_i}|\, dx = E^D(\lambda^\ell)$$

which is exactly the primal model energy (24) of the set of binary functions $\lambda_i^{t_i}$. Therefore, by duality between the max-flow problem (21) and the convex relaxed problem (24), $\lambda_i^{t_i}$ must be a global minimum of the min-cut problem (24) and therefore also a global minimum of the original problem (15).

3.5 'Saturated'/'Unsaturated' Edges

In the discrete setting, it is well known that the minimum cut severs edges that are saturated in the max-flow problem. This section attempts to give a variational explanation to the phenomena for the continuous max-flow and min-cut problems studied in this work. Let $\lambda_1^*, ..., \lambda_{n-1}^*$ be optimal to the dual problem (24). Assume that for some $x \in \Omega$ and $i \in 1, ..., n$ $\lambda_i(x) > t > \lambda_{i+1}(x)$, where $t \in (0, 1)$. Thresholding at t will generate the binary solution $\lambda_0(x) = ... = \lambda_i(x) = 1$ and $\lambda_{i+1}(x), ..., \lambda_n(x) = 0$. Therefore the cut generated by the binary function 'severs' the edge $e_{i+1}(x)$ between layer i and $i+1$. Since $\lambda_i(x) > \lambda_{i+1}(x)$ it follows by (27) that the optimal flow function must satisfy $p_i^*(x) = \rho(\ell_i, x)$, i.e. the edge $e_i(x)$ is saturated. Assume on the contrary that for some $x \in \Omega$ and $i \in 1, ..., n$ $p_i^*(x) < \rho(\ell_i, x)$. In this case $\lambda_i^*(x) = \lambda_{i+1}^*(x)$, otherwise $p_i^*(x)$ would not be optimal since increasing $p_i^*(x)$ would also increase the energy. Consequently, for any threshold level $t \in (0, 1]$, $\lambda_i^t(x) = \lambda_{i+1}^t(x)$, i.e. the edge $e_i(x)$ is not severed by the cut.

Similar interpretations of the spatial flow can be made by using the identity

$$\int_\Omega \lambda \text{ div } q\, dx = \int_\Omega q \cdot \nabla \lambda\, dx \tag{33}$$

If for some $x \in \Omega$ and a neighborhood $\mathcal{N}_\epsilon(x) = \{y \in \Omega : ||y - x|| < \epsilon\}$, $|q_i^*(y)| < \alpha$ for all $y \in \mathcal{N}_\epsilon(x)$, we say the spatial flow is unsaturated in $\mathcal{N}_\epsilon(x)$. Then λ_i is constant in $\mathcal{N}_\epsilon(x)$. Consequently, for any threshold $t \in (0, 1]$, $\lambda_i^t(y)$ is either identically 0 or 1 in $\mathcal{N}_\epsilon(x)$ and the cut will not sever the spatial domain $\mathcal{N}_\epsilon(x)$ at the i-th layer. Assume $\nabla \lambda_i \neq 0$ in some domain $S \subset \Omega$, then by (33) $|q_i^*(x)| = |\alpha \nabla \lambda_i^* / |\nabla \lambda_i^*|| = \alpha$ a.e. $x \in S$. Consequently, for any threshold $t \in (0, 1]$, $|q_i^*| = \alpha$ whenever $\nabla \lambda_i^t \neq 0$ in the distributional sense.

3.6 Extension to Continuous Labelings

Assume now that the feasible label values are constrained to the continuous interval $[\ell_{\min}, \ell_{\max}]$. As the number of labels goes to the limit of infinity, the max-flow problem (21) with the flow constraints (17)–(19) turns into

$$\sup_{p,q} \quad \int_{\Omega} p(\ell_{\min}, x)\, dx \tag{34}$$

$$\text{s.t.} \quad p(\ell, x) \leq \rho(\ell, x)\,, \quad |q(\ell, x)| \leq \alpha, \qquad \forall x \in \Omega, \quad \forall \ell \in [\ell_{\min}, \ell_{\max}] \tag{35}$$

$$\operatorname{div}_x q(\ell, x) + \partial_\ell\, p(\ell, x) = 0\,, \qquad \text{a.e. } x \in \Omega, \quad \ell \in [\ell_{\min}, \ell_{\max}]. \tag{36}$$

where $\ell \in [\ell_{\min}, \ell_{\max}]$ is the set of all feasible continuous-valued labels. The flow functions $p(x)$ and $q(x)$ are defined in the one dimensional higher space $[\ell_{\min}, \ell_{\max}] \times \Omega$. By carrying out similar steps as in the last section, the following dual problem can be derived

Proposition 2. *The max-flow model (34) with continuous label-values is dual / equivalent to the following min-cut model over* $[\ell_{\min}, \ell_{\max}] \times \Omega$*:*

$$\min_{\lambda(\ell,x)\in[0,1]} \int_{\ell_{\min}}^{\ell_{\max}} \int_{\Omega} \left\{ \alpha\, |\nabla_x \lambda| \, - \, \rho(\ell, x) \partial_\ell\, \lambda(\ell, x) \right\} dx d\ell$$
$$+ \int_{\Omega} (1 - \lambda(\ell_{\min}, x)) \rho(\ell_{\min}, x) + \lambda(\ell_{\max}, x) \rho(\ell_{\max}, x)\, dx \tag{37}$$

subject to

$$\partial_\ell\, \lambda(\ell, x) \leq 0\,, \quad \lambda(\ell_{\min}, x) \leq 1\,, \quad \lambda(\ell_{\max}, x) \geq 0\,, \quad \forall x \in \Omega, \quad \forall \ell \in [\ell_{\min}, \ell_{\max}]. \tag{38}$$

The proof can be found in [23].

The labeling function $u(x)$ can finally be reconstructed from the binary function $\lambda(\ell, x)$ by $u(x) = \ell_{\min} + \int_{\ell_{\min}}^{\ell_{\max}} \lambda(\ell, x)\, d\ell$.

In [9], Pock et al. gave a similar formulation of continuous labeling problems, as the search for a binary function defined over $[\ell_{\min}, \ell_{\max}] \times \Omega$, which minimizes

$$\min_{\lambda(\ell,x)\in\{0,1\}} \int_{\ell_{\min}}^{\ell_{\max}} \int_{\Omega} \left\{ \alpha\, |\nabla_x \lambda| + \rho(\ell, x)\, |\partial_\ell \lambda(\ell, x)| \, \right\} dx d\ell\,. \tag{39}$$

subject to

$$\lambda(\ell_{\min}, x) = 1\,, \quad \lambda(\ell_{\max}, x) = 0\,, \quad x \in \Omega \tag{40}$$

In order to solve this non-convex binary problem, the convex relaxation of [8] was adopted by minimizing over $\lambda(x, \ell) \in [0, 1]$. By applying the thresholding result of [8], binary optimizers could be obtained by thresholding the computed result.

Some differences can be observed between our formulation (37), (38) and the formulation (39), (40): The constraint $\partial_\ell \lambda(\ell, x) \leq 0$ is not forced explicitly in [9]. However, it turns out the presence of the absolute value of the term

$\rho(\ell, x)\,|\partial_\ell \lambda(\ell, x)|$ forces this constraint to hold. Observe that if $\rho(\ell, x) < 0$ is negative, the formulation of (39) is non-convex, and can therefore not be solved globally. This is in contrast to our formulation (37), which is convex also in this case. The functional (39) could be made convex by adding a sufficiently large number to the data term at every $x \in \Omega$. In the more recent work of Pock et al. [10], a more strict derivation resulted in a little different formulation. In this formulation, the integrand of the energy functional is infinite if $\partial_\ell \lambda(\ell, x) \leq 0$, hence this constraint is forced to hold. Their derivations rely heavily on results from the theory of calibrations [12] and cartesian currents [26,27]. Label values ranged over the whole real line $\mathbb{R}$ was assumed, which required to impose limits at infinity: $\lim_{\ell \mapsto +\infty} \lambda(\ell, x) = 0$ and $\lim_{\ell \mapsto -\infty} \lambda(\ell, x) = 1$.

We eventually stick to a finite label value set in practice. After discretization, the label space also becomes discrete in [10]. However, it has not been proven if all properties, such at the thresholding scheme and monotonicity constraint hold exactly after discretization. In contrast, these properties were proved to hold exactly for our model with discrete label values developed in Sect. 3.

Last but not the least, a primal-dual algorithm was proposed in [10], which consists of iteratively taking ascent steps over the dual variables p and q and descent step over the primal variable λ, followed by projections of all the variables onto the nearest points of the feasible sets iteratively until convergence.

4 Algorithms

4.1 Multiplier-Based Max-Flow Algorithm

In this section, it is assumed that the image domain Ω is discrete and the differential operators are discretized, such that the optimization problems become finite dimensional. We stick to the continuous notation, using $\int, \nabla$ and $\div$ to ease readability. As stated in the previous section, the energy formulation of (22) is just the Lagrangian function of (21) and λ_i, $i = 1, \ldots, n-1$, are the multiplier functions. To this end, we define its respective augmented Lagrangian function as

$$L_c(p, q, \lambda) := \int_\Omega p_1 + \sum_{i=1}^{n-1} \lambda_i (\operatorname{div} p_i + p_{i+1} - p_i) - \frac{c}{2} |\operatorname{div} p_i + p_{i+1} - p_i|^2 \, dx, \quad (44)$$

where $c > 0$.

We propose an algorithm for the continuous maximal flow problem (21) based on the augmented Lagrangian method [29] , see Algorithm 3.6. Algorithm 3.6 is an example of an alternating direction method of multipliers, where (44) is maximized alternatively with respect to each variable p_i, q, followed by an updating of the Lagrange multipliers λ_i, $i = 1, \ldots, n-1$ at each iteration. For the two-label case, a similar flow-maximization scheme for the continuous min-cut problem was proposed in [19,20].

Algorithm 1 Multiplier-Based Maximal-Flow Algorithm

Choose some starting values for p^1, q^1 and λ^1, let $k, i = 1$ and start $k-$th iteration, which contains the following steps, until convergence:

- For each layer $i = 1 \dots n$ solve
  - Optimize p_i by fixing other variables

$$\begin{aligned}\tilde{p}_i^{k+1} &:= \arg \max_{p_i(x) \le \rho(\ell_i, x)\ \forall x \in \Omega} L_c((p_{j<i}^{k+1}, p_i, p_{j>i}^{k}), (q_{j<i}^{k+1}, q_{j\ge i}^{k}), \lambda^k) \\ &:= \arg \max_{p_i(x) \le \rho(\ell_i, x)} -\frac{c}{2} \left\| p_i + \operatorname{div} q_{i-1}^{k+1} - p_{i-1}^{k+1} - \lambda_{i-1}^k / c \right\|^2 \\ &\quad - \frac{c}{2} \left\| p_i - (p_{i+1}^k + \operatorname{div} q_i^k) + \lambda_i^k / c \right\|^2\end{aligned}$$

    which can be explicitly computed at each point $x \in \Omega$; At the first and last layer, $i = 1$ and $i = n - 1$ the update formulas are a little different and are given in (42) and (43) below.
  - Optimize q_i, by introducing the new value of p_i^{k+1} and fixing other variables

$$\begin{aligned}q_i^{k+1} &:= \arg \max_{\|q\|_\infty \le \alpha} L_c((\tilde{p}_{i\le j}^{k+1}, p_{i>j}^{k}), (q_{j<i}^{k+1}, q_i, q_{j>i}^{k}), \lambda^k) \\ &:= \arg \max_{\|q\|_\infty \le \alpha} -\frac{c}{2} \left\| \operatorname{div} q_i + p_{i+1}^k - \tilde{p}_i^{k+1} - \lambda_i^k / c \right\|^2 , \end{aligned} \tag{41}$$

    which can either be solved iteratively by the projected-gradient algorithm [28], or approximately by one linearized step (45);
  - Optimize p_i again, by introducing the new values of q_i^{k+1} and fixing others

$$p_i^{k+1} := \arg \max_{p_i(x) \le \rho(\ell_i, x)\ \forall x \in \Omega} L_c((p_{j<i}^{k+1}, p_i, p_{j>i}^{k}), (q_{j\le i}^{k+1}, q_{j>i}^{k}), \lambda^k) ,$$

    which can be explicitly computed at each point $x \in \Omega$;
- Update multipliers λ_i, $i = 1, \dots, n-1$, by

$$\lambda_i^{k+1} = \lambda_i^k - c\,(\operatorname{div} q_i^{k+1} - p_i^{k+1} + p_{i+1}^{k+1}) ;$$

- Set $k \leftarrow k + 1$ and repeat until convergence.

At the first and last layer $i = 1$ and $i = n$ the update formulas for p_1 and p_n are:

$$\begin{aligned}p_1^{k+1} &:= \arg \max_{p_1(x) \le \rho(\ell_1, x)} L_c(p_1, p_2^k, ..., p_n^k, q^{k+1}, \lambda^k) \\ &:= \arg \max_{p_1(x) \le \rho(\ell_1, x)} \int_\Omega p_1\, dx - \frac{c}{2} \left\| p_1 - (p_2^k + \operatorname{div} q_1^{k+1}) + \lambda_1^k / c \right\|^2 , \end{aligned} \tag{42}$$

and

$$\begin{aligned}p_n^{k+1} &:= \arg \max_{p_n(x) \le \rho(\ell_n, x)} L_c(p_1^{k+1}, ..., p_{n-1}^{k+1}, p_n, q^{k+1}, \lambda^k) \\ &:= \arg \max_{p_n(x) \le \rho(\ell_1, x)} -\frac{c}{2} \left\| p_n + \operatorname{div} q_{n-1}^{k+1} - p_{n-1}^{k+1} - \lambda_{n-1}^k / c \right\|^2 . \end{aligned} \tag{43}$$

Both can be computed explicitly;

Instead of solving the sub-problem (41) iteratively by the projected-gradient algorithm [28], an inexact solution can be obtained by the linearization:

$$q_i^{k+1} = \Pi_\alpha \left(q_i^k + c\nabla(\operatorname{div} q_i^k + p_{i+1}^k - p_i^{k+1} - \lambda_i^k / c). \right) \tag{45}$$

where Π_α is the projection onto the convex set $C_\alpha = \{q \mid \|q\|_\infty \leq \alpha\}$. There are extended convergence results for such a linearization for closely related problems [30].

5 Numerical Experiments

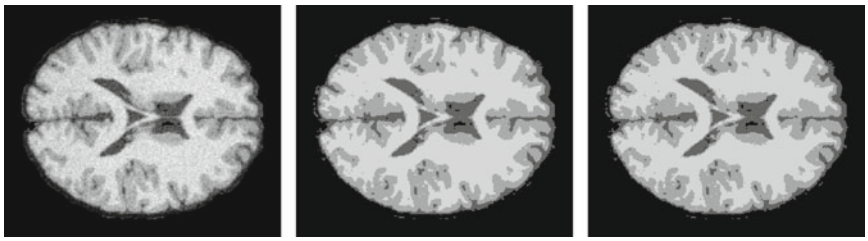
In this work, we focus on applications to image segmentation and stereo reconstruction. Comparisons are made to the discrete approach [6] and the primal-dual algorithm of [9].

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

Fig. 2. (a) Ground truth, (b) input, (c) Rescaled labeling function before threshold, (d) Rescaled labeling function after thresholding each λ_i at 0.5.

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

Fig. 3. (a) Input image damaged by impulse noise; (b) reconstructed labeling function with non-convex data term (47) before threshold, (c) labeling function after thresholding each λ_i at 0.5, (d) reconstructed labeling function with convex data term (46) and $\beta = 1$.



(a) (b) (c)

Fig. 4. (a) Input, (b) Labeling function before threshold (c) Labeling function after thresholding each λ_i at 0.5.

In case of image segmentation we assume $\ell_i = i$, $i = 1, ..., n$ and n is the number of regions. $\rho(i, x)$ is the data cost of assigning pixel x to region i. One possibility is

$$\rho(i, x) = |I(x) - c_i|^\beta, \quad i = 1, ..., n \tag{46}$$

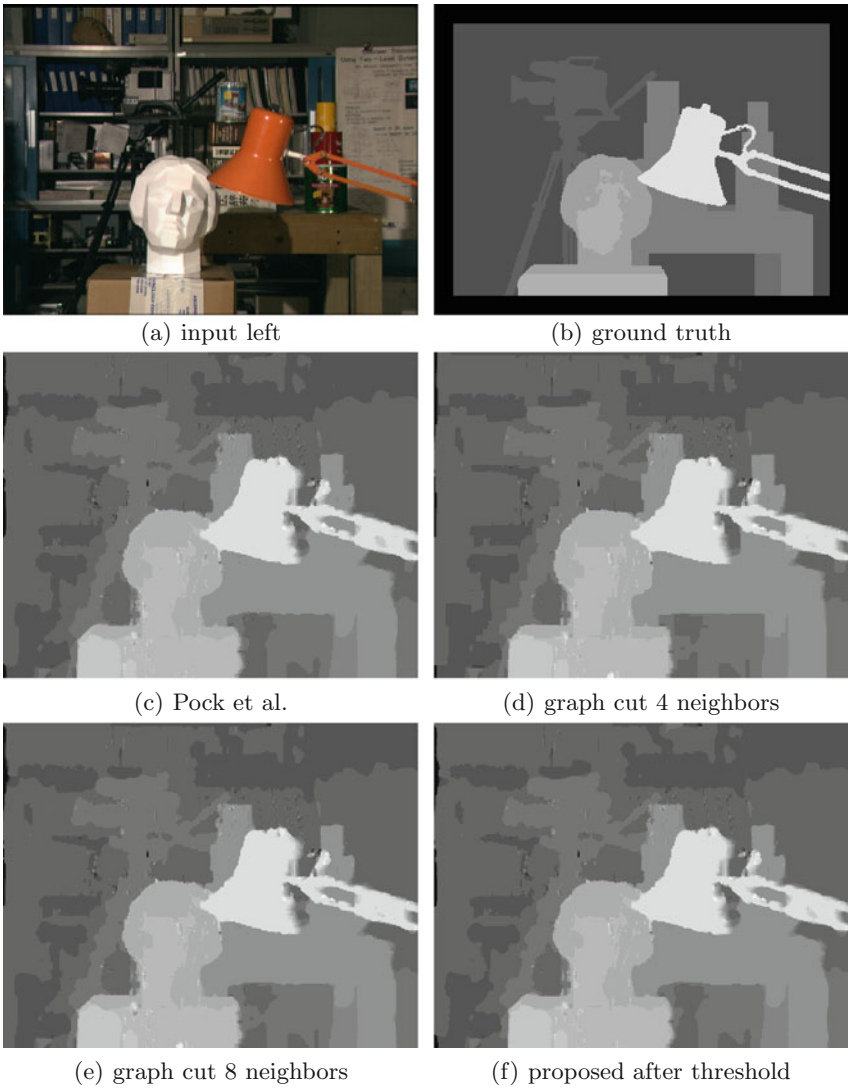
where I is the input image and c_i is the average intensity value of region i. They are assumed to be fixed in this work. Such a data term is convex for $\beta \geq 1$ and non-convex for $\beta < 1$. Results with $\beta = 2$ are shown in Figs. 2, 4. We also demonstrate image segmentation with a non-convex data term in Fig. 3. The ground truth image from Fig. 2(a) has been damaged by impulse noise in Fig. 3(a). More specifically, 70% of the pixels have been randomly selected and given a random number between 0 and 255 (max gray value). For this type of noise, the convex data terms does not perform well, as shown in Fig. 3(d) where we have selected (46) with $\beta = 1$. Instead the following non-convex data term can be used

$$\rho(i, x) := \begin{cases} 0\,, \text{ if } i = \text{argmin}_k\, |I(x) - c_k| \\ 1\,, \text{ else} \end{cases}. \tag{47}$$

In the stereo application we are given two color images I_L and I_R of a scene taken from horizontally slightly different viewpoints and would like to reconstruct the depth map u. The quality of the matching between I_L and I_R for a depth value u is measured by using the following ρ in the data term of (13)

$$\rho(u, x) = \sum_{j=1}^{3} |I_L^j(x) - I_R^j(x + (u, 0)^T)|. \tag{48}$$

Here $I^j(x)$ denotes the jth component of the color vector $I(x)$. The above data term (48) is obviously highly non-convex. The results on a standard example are shown in Fig. 5, where comparison are also given [10] and graph cut with a neighborhood system of 4 and 8. Graph cut produces a single non-unique solution which is shown in Fig. 5(f) and (g) with 4 and 8 neighbors respectively. As we see, such solutions suffer from metrication artifacts because of the discrete grid bias.



(a) input left (b) ground truth

(c) Pock et al. (d) graph cut 4 neighbors

(e) graph cut 8 neighbors (f) proposed after threshold

Fig. 5. Stereo depth estimation.

Iteration counts for all experiments are presented in Table 1 and CPU times are shown in Table 2. The two variants of Algorithm 1 are evaluated against the primal-dual method of Pock et al. [10]. The relative energy precision at iteration i is given by

$$\varepsilon = \frac{E^i - E^*}{E^*}, \tag{49}$$

where E^i is the energy at iteration i and E^* is the final energy. A good estimate of E^* is obtained by using a huge amount of iterations of each method and

Table 1. Iteration counts for each experiment. Number of iterations to reach an energy precision of 10^{-3} and 10^{-4} are shown. PD = Primal-dual. Proposed 1 stands for Algorithm 1 where the subproblem is solved by 5 iterations of Chambolle's algorithm each outer iteration (indicated by the number in the parenthesis). Proposed 2 stands for Algorithm 1 with the subproblems solved inexactly in one step through the linearization (45).

	Energy precision $\varepsilon < 10^{-3}$			Energy precision $\varepsilon < 10^{-4}$		
	PD [10]	Proposed 1	Proposed 2	PD [10]	Proposed 1	Proposed 2
Brain	280	50 (× 5)	110	430	65 (× 5)	280
Figure 2	295	35 (× 5)	115	640	65 (× 5)	290
Stereo	4055	550 (× 5)	1070	14305	920 (× 5)	3905

Table 2. CPU time in seconds for each experiment for reaching an energy precision of 10^{-3} and 10^{-4}. PD = Primal-dual. Proposed 1 stands for Algorithm 1 where the subproblem is solved by 5 iterations of Chambolle's algorithm each outer iteration (indicated by the number in the parenthesis). Proposed 2 stands for Algorithm 1 with the subproblems solved inexactly in one step through the linearization (45).

	Energy precision $\varepsilon < 10^{-3}$			Energy precision $\varepsilon < 10^{-4}$		
	PD [10]	Proposed 1	Proposed 2	PD [10]	Proposed 1	Proposed 2
Brain	86	68	38	132	89	96
Figure 2	1.34	0.64	0.47	2.61	1.18	1.32
Stereo	2027	1214	598	7153	2029	2182

each experiment. The table shows how many iterations are required to reach an energy precision of 10^{-3} and 10^{-4}. Our algorithms are implemented with a mimetic finite difference spatial discretization [31,32]. In order to make the comparison as accurate as possible, the primal-dual algorithm [10] is also implemented with such a mimetic finite difference discretization, although a slightly different forward scheme for the gradient and backward scheme for the divergence was used in [10].

The first variant of Algorithm 3.6 solves the subproblem (41) iteratively by Chambolle's algorithm [28]. Since the previous solution is available as a good initialization, not many iterations of this algorithm is required. In our experiments, 5 inner iterations was used for each step. Increasing the number of inner iterations beyond 5 did not seem to have any impact on the convergence rate in our experience.

The primal-dual method of [10] avoids the inner problem, but as we see requires significantly more iterations to reach the same energy precisions. Our algorithm also requires less total number of iterations (inner times outer iterations). The difference becomes progressively clearer with higher energy precision. For the stereo example, which is by far most difficult computationally, our approach reached an energy precision of $\epsilon < 10^{-5}$ after 1310 iterations, $\epsilon < 10^{-6}$ after 1635 iterations and $\epsilon < 10^{-7}$ after 2340 iteration. The primal-dual

Table 3. Iteration counts for stereo experiment. Number of iterations to reach an energy precision of 10^{-4}, 10^{-5} and 10^{-6} are shown. PD = Primal-dual.

	Energy prec. $\varepsilon < 10^{-4}$		Energy prec. $\varepsilon < 10^{-5}$		Energy prec. $\varepsilon < 10^{-6}$	
	PD [10]	Proposed 1	PD [10]	Proposed 1	PD [10]	Proposed 1
Stereo	14305	920 (× 5)	> 30000	1310 (× 5)	> 30000	1635 (× 5)

algorithm [10] failed to ever reach an energy precision of 10^{-5} or lower within our predetermined number of maximum iterations (30000). We believe this difference is due to the fact that our approach avoids the iterative projections of the labeling function and hence progresses in the exact steepest descent direction every iteration.

The second variant of the Algorithm 1 instead computes an inexact solution to (41) through the linearization (45) and hence avoids the inner iterations. However, the penalty parameter c must be set lower to maintain convergence, hence more outer iterations are required. Overall it converges a little faster than the first variant and outperforms the primal-dual algorithm [10] for all the experiments.

Comparison to discrete graph cut [33] is more complicated. Our algorithms are implemented in matlab, in contrast to the optimized c++ discrete max-flow implementation of [33]. Our algorithm consists mainly of floating point matrix and vector arithmetic and is therefore highly suited for massive parallel implementation on GPU. Traditional max-flow algorithms have a much more serial nature, which makes them more dependent on an efficient serial CPU. In the near future, hardware improvements are also expected to be largely of the parallel aspect. Hence, we see our work as more suited for the current and future generation of hardware.

6 Conclusions

In this paper we proposed and investigated a novel max-flow formulation of multilabelings in the continuous setting. It is a direct mapping of Ishikawa's graph-based configuration to the continuous setting. We proved the maximization problem is dual to an equivalent min-cut formulation by variational analysis. In addition, we proposed a new and reliable multiplier-based max-flow algorithm with convergence that can verified by optimization theories, which was demonstrated to significantly outperform eariler approaches. Due to its continuous formulation, the algorithm can easily be speeded up by a multigrid or parallel implementation, in contrast to graph-based methods. The memory requirement is also not as strict.

Acknowledgements. This research has been supported by the Norwegian Research Council eVita project 214889 and eVita project 166075 and Natural Sciences and Engineering Research Council of Canada (NSERC) Accelerator Grant R3584A04.

References

1. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **23**, 1222–1239 (2001)
2. Appleton, B., Talbot, H.: Globally optimal surfaces by continuous maximal flows. In: DICTA, pp. 987–996 (2003)
3. Kolmogorov, V., Zabih, R.: Multi-camera scene reconstruction via graph cuts. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002, Part III. LNCS, vol. 2352, pp. 82–96. Springer, Heidelberg (2002)
4. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **26**, 65–81 (2004)
5. Lempitsky, V., Boykov, Y.: Global optimization for shape fitting. In: CVPR (2007)
6. Ishikawa, H.: Exact optimization for markov random fields with convex priors. IEEE Trans. Pattern Anal. Mach. Intell. **25**, 1333–1336 (2003)
7. Kohli, P., Kumar, M.P., Torr, P.H.: p^3 and beyond: move making algorithms for solving higher order functions. IEEE Trans. Pattern Anal. Mach. Intell. **31**, 1645–1656 (2009)
8. Chan, T.F., Esedoḡlu, S., Nikolova, M.: Algorithms for finding global minimizers of image segmentation and denoising models. SIAM J. Appl. Math. **66**, 1632–1648 (2006). (electronic)
9. Pock, T., Schoenemann, T., Graber, G., Bischof, H., Cremers, D.: A convex formulation of continuous multi-label problems. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part III. LNCS, vol. 5304, pp. 792–805. Springer, Heidelberg (2008)
10. Pock, T., Cremers, D., Bischof, H., Chambolle, A.: Global solutions of variational models with convex regularization. Technical report, Institute for Computer Graphics and Vision, Graz University of Technology (2010)
11. Bouchitt'e, G.: Recent convexity arguments in the calculus of variations. In: Lecture Notes from the 3rd International Summer School on the Calculus of Variations, Pisa (1998)
12. Alberti, G., Bouchitté, G., Maso, G.D.: The calibration method for the mumford-shah functional and free-discontinuity problems. Calc. Var. Partial Differ. Eqn. **16**, 299–333 (2003)
13. Ford, L.R., Fulkerson, D.R.: Flows in Networks. Princeton University Press, Princeton (1962)
14. Ford, L.R., Fulkerson, D.R.: Maximal flow through a network. Can. J. Math. **8**, 399–404 (1956)
15. Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum-flow problem. J. ACM **35**, 921–940 (1988)
16. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd edn. MIT Press, Cambridge (2001)
17. Strang, G.: Maximal flow through a domain. Math. Program. **26**, 123–143 (1983)
18. Appleton, B., Talbot, H.: Globally minimal surfaces by continuous maximal flows. IEEE Trans. Pattern Anal. Mach. Intell. **28**, 106–118 (2006)
19. Yuan, J., Bae, E., Tai, X.: A study on continuous max-flow and min-cut approaches. In: IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, USA, pp. 2217–2224 (2010)
20. Yuan, J., Bae, E., Tai, X., Boykov, Y.: A study on continuous max-flow and min-cut approaches. Technical report CAM10-61, UCLA, CAM (2010)

21. Yuan, J., Bae, E., Tai, X.-C., Boykov, Y.: A continuous max-flow approach to potts model. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part VI. LNCS, vol. 6316, pp. 379–392. Springer, Heidelberg (2010)
22. Chambolle, A., Pock, T.: A first-order primal-dual algorithm for convex problems with applications to imaging. J. Math. Imaging Vis. **40**, 120–145 (2011)
23. Bae, E., Yuan, J., Tai, X., Boykov, Y.: A fast continuous max-flow approach to non-convex multilabeling problems. Technical report, UCLA, CAM-report 10-62 (2010)
24. Bae, E., Tai, X.-C.: Graph cut optimization for the piecewise constant level set method applied to multiphase image segmentation. In: Tai, X.-C., Mørken, K., Lysaker, M., Lie, K.-A. (eds.) SSVM 2009. LNCS, vol. 5567, pp. 1–13. Springer, Heidelberg (2009)
25. Bae, E., Yuan, J., Tai, X.C.: Global minimization for continuous multiphase partitioning problems using a dual approach. Int. J. Comput. Vision **92**, 112–129 (2011)
26. Giaquinta, M., Modica, G., Soucek, J.: Cartesian Currents in the Calculus of Variations I. Ergebnisse der Mathematik und ihrer Grenzgebiete. 3. Folge A Series of Modern Surveys in Mathematics, vol. 37. Springer, Heidelberg (1998)
27. Giaquinta, M., Modica, G., Soucek, J.: Cartesian Currents in the Calculus of Variations II. Ergebnisse der Mathematik und ihrer Grenzgebiete. 3. Folge A Series of Modern Surveys in Mathematics, vol. 38. Springer, Heidelberg (1998)
28. Chambolle, A.: An algorithm for total variation minimization and applications. J. Math. Imaging Vis. **20**, 89–97 (2004)
29. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific, Belmont (1999)
30. Esser, J.E.: Primal dual algorithms for convex models and applications to image restoration, registration and nonlocal inpainting (2010)
31. Hyman, J.M., Shashkov, M.J.: Natural discretizations for the divergence, gradient, and curl on logically rectangular grids. Comput. Math. Appl. **33**, 81–104 (1997)
32. Hyman, J.M., Shashkov, M.J.: Adjoint operators for the natural discretizations of the divergence, gradient and curl on logically rectangular grids. Appl. Numer. Math. **25**, 413–442 (1997)
33. Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Trans. Pattern Anal. Mach. Intell. **26**, 359–374 (2001)

A Geometric Multigrid Solver on Tsubame 2.0

Harald Köstler[1(✉)], Christian Feichtinger[1], Ulrich Rüde[1], and Takayuki Aoki[2]

[1] Chair for System Simulation, University of Erlangen-Nuremberg, Erlangen, Germany
Harald.Koestler@informatik.uni-erlangen.de
[2] Global Scientific Information and Computing Center, Tokyo Institute of Technology, Yokohama, Japan

Abstract. Tsubame 2.0 is currently one of the largest installed GPU clusters and number 5 in the Top 500 list ranking the fastest supercomputers in the world. In order to make use of Tsubame, there is a need to adapt existing software design concepts to multi-GPU environments. We have developed a modular and easily extensible software framework called waLBerla that covers a wide range of applications ranging from particulate flows over free surface flows to nano fluids coupled with temperature simulations and medical imaging. In this article we report on our experiences to extend waLBerla in order to support geometric multigrid algorithms for the numerical solution of partial differential equations (PDEs) on multi-GPU clusters. We discuss the software and performance engineering concepts necessary to integrate efficient compute kernels into our waLBerla framework and show first weak and strong scaling results on Tsubame for up to 1029 GPUs for our multigrid solver.

Keywords: GPGPU · CUDA · Parallel multigrid solver · waLBerla · Tsubame 2.0

1 Introduction

Many imaging applications exhibit high memory and compute power requirements, either due to the large amount of data being processed or runtime restrictions e.g. for real-time imaging. Graphics processing units (GPUs) typically offer hundreds of specialized compute units operating on dedicated memory and reach outstanding compute and memory performance in this way. Therefore, they are more and more used for compute-intensive applications also in imaging. GPUs are best suitable for massively-data parallel algorithms, inadequate problems, that e. g. require a high degree of synchronization or provide only limited parallelism, are left to the host CPU. For high performance computing (HPC) heterogeneous multi-GPU clusters are built up consisting of thousands of GPUs. In the Top 500 list[1] from November 2011 of the fastest machines world-wide there were three of these multi-GPU clusters in the Top 5.

[1] http://www.top500.org, Nov. 2011.

A. Bruhn et al. (Eds.): Global Optimization Methods, LNCS 8293, pp. 155–173, 2014.
DOI: 10.1007/978-3-642-54774-4_8,

However, in order to achieve good performance on these clusters, software development has to adapt to the new needs of the massively parallel hardware. As a starting point, GPU vendors offer proprietary environments for general purpose GPU computing. NVIDIA, e. g., provides the possibility to write single-source programs that execute kernels written in a subset of C and C++ on their Compute Unified Device Architecture (CUDA) [1]. An alternative would have been to use the Open Compute Language (OpenCL)[2]. Within OpenCL one can write code that runs in principle on many different hardware platforms, e. g. Intel CPUs, ATI/AMD or NVIDIA GPUs, and even the ICM Cell processor, but to achieve optimal performance the implementation has to be adapted to the specific features of the hardware. Since we are exclusively working on NVIDIA GPUs in this article and we found no considerable difference in the kernel performance if we tune OpenCL towards NVIDIA GPUs, we have done our implementations in CUDA. Both CUDA and OpenCL are low-level languages. To make code development more efficient, one either has to provide wrappers for high-level languages like e.g. OpenMP [2] and PyCUDA [3] or easy to use frameworks, where we follow the latter approach.

Our contributions in this article are specifically that

- we discuss the concepts necessary to integrate efficient GPU compute kernels for a geometric multigrid solver into our software framework waLBerla that is discussed in more detail in Sect. 3,
- and then show first weak and strong scaling results of our solver on Tsubame 2.0 located in Japan.

While waLBerla was at first developed for simulating fluid flow using the Lattice Boltzmann method on 3D structured domains, it is now also capable of solving elliptic PDEs like Poisson's equation numerically via multigrid.

One possible imaging application for our multigrid solver is high dynamic range (HDR) compression. HDR tries to allow a wide dynamic range of luminance between the lightest and darkest areas within an image. Often, HDR compression is only one step within the image acquisition pipeline and there are hard time constraints that have to be met in practical applications. In [4] one finds a state-of-the-art HDR compression algorithm in the gradient space that can be accelerated by our multigrid solver. In general, for gradient space imaging one has to transform an input image $I : \Omega \mapsto \mathbb{R}$ defined in the domain $\Omega \subset \mathbb{R}^3$ to gradient space and back. While the forward transformation to gradient space is fast by using simple finite differences to obtain the image gradient ∇I, the backward transformation requires the solution of Poisson's equation

$$\Delta u = f \quad \text{in } \Omega \tag{1a}$$

$$u = 0 \quad \text{on } \partial\Omega \tag{1b}$$

typically assuming homogeneous Dirichlet boundary conditions. Here,

$$f = \operatorname{div}(\Phi \nabla I) \ , \tag{2}$$

[2] http://www.khronos.org/opencl/, Mai 2012.

where $\Phi\nabla I$ are compressed dynamic range image derivatives and $\Phi : \mathbb{R}^3 \mapsto \mathbb{R}$ is a position-dependent attenuating function (see [4] for more details). The solution $u : \Omega \mapsto \mathbb{R}$ is the HDR compressed image.

Most of the overall runtime for HDR compression is spent in the numerical solution of (1a, 1b), where we can apply a parallel, geometric multigrid solver.

Besides HDR compression there are a variety of applications in imaging and computer vision, where multigrid methods are used. Especially for variational models the arising Euler-Lagrange equations are often treated via efficient multigrid solvers. In this way, applications ranging from image denoising, image inpainting, and image segmentation to optical flow and image registration are found (see [5] for more details about different multigrid methods and for further references).

There exists already also a variety of other implementations of different multigrid algorithms on GPU like in [6,7], conjugate gradients (CG) and multigrid on NVIDIA GeForce FX [8], mixed-precision multigrid solvers [9], finite element multigrid solvers on GPU clusters [10,11], or algebraic multigrid [12]. Parallel multigrid methods on GPUs are incorporated in software packages like OpenCurrent [13] or PETSc [14], and GPU multigrid is also used in imaging, e.g. for nonlinear denoising or variational optical flow (see e.g. [15–17]).

In previous work, we have run a multi-GPU Lattice Boltzmann simulation on Tsubame [18] and highly scalable multigrid solvers on CPU clusters [19–21]. Furthermore, we optimized a 2D multigrid solver on GPU to do real-time HDR compression [22] for a series of X-ray images. In addition to that, we show weak and strong scaling results on an IBM Bluegene/P up to nearly 300.000 cores and an Intel CPU cluster in [23], where we used a 3D multigrid solver on a block-structured tetrahedral finite element mesh. Now we integrate a multi-GPU geometric multigrid solver in waLBerla. An alternative is to implement a finite element based multigrid solver on GPU for gradient space imaging [24], however, it is computationally more expensive than our finite difference based solver on a regular grid. Note that our multigrid solver scales also on CPU-clusters [25] and works also for more general elliptic PDEs with variable coefficients [26].

The paper is organized as follows: In Sect. 2 we briefly describe the multigrid algorithm and its parallelization on GPUs. Section 3 summarizes the MPI-parallel waLBerla framework that easily enables us to extend our code to multi-GPUs. The hardware details of the Tsubame 2.0 cluster and a simple performance model for our multigrid solver to estimate the runtime of our software are introduced in Sect. 4. In Sect. 5 we present weak and strong scaling results on Tsubame 2.0 before concluding the paper in Sect. 6.

2 Parallel Multigrid

2.1 Multigrid Algorithm

Multigrid is not a single algorithm, but a general approach to solve problems by using several levels or resolutions [27,28]. We restrict ourselves to *geometric multigrid (MG)* in this article that identifies each level with a (structured) grid.

Typically, multigrid is used as an iterative solver for large linear systems of equations that have a certain structure, e.g. that arise from the discretization of PDEs and lead to sparse and symmetric positive definite system matrices. The main advantage of multigrid solvers compared to other solvers like CG is that multigrid can reach an asymptotically optimal complexity of $\mathcal{O}(N)$, where N is the number of unknowns or grid points in the system. For good introductions and a comprehensive overview on multigrid methods, we, e.g., refer to [29,30], for details on efficient multigrid implementations see [31–33].

We assume that we want to solve the PDE (1a, 1b) with solution $\boldsymbol{u} : \mathbb{R}^3 \to \mathbb{R}$, right hand side (RHS) $f : \mathbb{R}^3 \to \mathbb{R}$, and Dirichlet boundary conditions on a rectangular domain $\Omega \subset \mathbb{R}^3$. Equation (1a, 1b) is discretized by finite differences on a structured grid. This results in a linear system

$$A^h u^h = f^h \,, \quad \sum_{j \in \Omega^h} a_{ij}^h u_j^h = f_i^h \,, i \in \Omega^h \tag{3}$$

with system matrix $A^h \in \mathbb{R}^{N \times N}$, unknown vector $u^h \in \mathbb{R}^N$ and right hand side (RHS) vector $f^h \in \mathbb{R}^N$ on a discrete grid Ω^h with mesh size h.

In order to solve the above linear system, we note that during the iteration the algebraic error $e^h = u_*^h - u^h$ is defined to be the difference between the exact solution u_*^h of Eq. (3) and the approximate solution u^h. With the residual equation $r^h = f^h - A^h u^h$ we obtain there so-called error equation

$$A^h e^h = r^h. \tag{4}$$

The multigrid idea is now based on two principles:

Smoothing Property: Classical iterative solvers like red-black Gauß-Seidel (RBGS) are able to smooth the error after very few steps. That means the high frequency components of the error are removed well by these methods. But they have little effect on the low frequency components. Therefore, the convergence rate of classical iterative methods is good in the first few steps and decreases considerably afterward.

Coarse Grid Principle: A smooth function on a fine grid can be approximated satisfactorily on a grid with less discretization points, whereas oscillating functions would disappear. Furthermore, a procedure on a coarse grid is less expensive than on a fine grid. The idea is now to approximate the low frequency error components on a coarse grid.

Multigrid combines these two principles into one iterative solver. The smoother reduces the high frequency error components first, and then the low frequency error components are approximated on coarser grids, interpolated back to the finer grids and eliminated there. In other words, on the finest grid Eq. (1a, 1b) first is solved approximately by a few smoothing steps and then an approximation to the error equation is computed on the coarser grids. This leads to recursive algorithms which traverse between fine and coarse grids in a grid hierarchy. Two successive grid levels Ω^h and Ω^H typically have fine mesh size h and coarse mesh size $H = 2h$.

One multigrid iteration, here the so-called *V-cycle*, is summarized in Algorithm 1. Note that in general the operator A^h has to be computed on each grid level. This is either done by rediscretization of the PDE or by Galerkin coarsening, where $A^H = RA^hP$.

Algorithm 1 Recursive V-cycle: $u_h^{(k+1)} = V_h(u_h^{(k)}, A^h, f^h, \nu_1, \nu_2)$

1: **if** coarsest level **then**
2: solve $A^h u^h = f^h$ exactly or by several CG iterations
3: **else**
4: $\bar{u}_h^{(k)} = \mathcal{S}_h^{\nu_1}(u_h^{(k)}, A^h, f^h)$ {presmoothing}
5: $r^h = f^h - A^h \bar{u}_h^{(k)}$ {compute residual}
6: $r^H = Rr^h$ {restrict residual}
7: $e^H = V_H(0, A^H, r^H, \nu_1, \nu_2)$ {recursion}
8: $e^h = Pe^H$ {interpolate error}
9: $\tilde{u}_h^{(k)} = \bar{u}_h^{(k)} + e^h$ {coarse grid correction}
10: $u_h^{(k+1)} = \mathcal{S}_h^{\nu_2}(\tilde{u}_h^{(k)}, A^h, f^h)$ {postsmoothing}
11: **end if**

In our node-based multigrid solver we use the following components:

- A ω-RBGS (or red-black SOR) smoother $\mathcal{S}_h^{\nu_1}, \mathcal{S}_h^{\nu_2}$ with ν_1 pre- and ν_2 postsmoothing steps.
- The restriction operator R from fine to coarse grid is full weighting.
- We apply a trilinear interpolation operator P for the error.
- The coarse grid problem is solved by a sufficient number of CG iterations.
- The discretization of the Laplacian was done via the standard 7-point stencil (cf. Eq. (1a, 1b)), on coarser grids we rediscretize the Laplacian.

Note that the required number of CG iterations on the coarsest grid is proportional to the diameter of the computational domain (see e.g. [23,34]) and thus increases linearly with growing diameter.

2.2 GPU Implementation

To implement the multigrid algorithm on GPU we have to parallelize it and write kernels for smoothing, computation of the residual, restriction, and interpolation together with coarse grid correction. In the following, we choose the ω-RBGS kernel as an example and discuss it in more detail. Algorithm 2 shows the source code of a straightforward, unoptimized CUDA RBGS kernel. Here, the `solution` and `rhs` fields are stored in global GPU memory. Due to the splitting in red and black points within the RBGS to enable parallelization, only every second solution value is written back, whereas the whole solution vector is processed.

Algorithm 2 ω-red-black Gauß-Seidel smoother kernel in CUDA.

```
__global__ void kernel_RBGS(Real* solution, Real* rhs,
                            const Uint xSize, const Uint ySize,
                            const Uint zSize, const Uint red_black)
{

  unsigned int x = threadIdx.x;
  unsigned int y = blockIdx.x;
  unsigned int z = blockIdx.y;
  Real w = 1.15;

  if ((x > 0) && (y > 0) && (z > 0) &&
      (x < xSize-1) && (y < ySize-1) && (z < zSize-1) )
  {

    Real new_val = (1-w)*GET3D(solution,x,y,z)
    + w*((GET3D(rhs,x,y,z) + GET3D(solution,x-1,y,z)
    + GET3D(solution,x+1,y,z) + GET3D(solution,x,y+1,z)
    + GET3D(solution,x,y-1,z) + GET3D(solution,x,y,z+1)
    + GET3D(solution,x,y,z-1)) / Real(6.));

    if (((x+y+z)%2) == red_black)
      GET3D(solution,x,y,z) = new_val;
  }
}
```

Possible optimizations are e.g. to split the red and black points into separate arrays in memory, or blocking techniques (see [22] for a detailed performance analysis in 2D). Additionally, the thread block size depends on the number of grid points in x-direction. Best performance can be achieved for larger thread block sizes, e.g. 256 or 512, therefore the kernel becomes inefficient for a smaller number of grid points in x-direction and 2D thread blocks become necessary.

For multi-GPU, the distributed memory parallelization is simply done by decomposing each grid into several smaller sub-grids and introducing a layer of ghost cells between them. Now the sub-grids can be distributed to different MPI processes and only the ghost cells have to be communicated to neighboring sub-grids. The function calling the kernel handles the ghost cell exchange. Buffers are sent to neighboring processes via communication routines provided by the waLBerla framework introduced in the next section. Within Algorithm 1 one has to exchange the ghost layer of the solution resp. the error after smoothing and interpolation (steps 4, 8, and 10), the ghost layer of the residual after step 5. On the coarsest level we have only a few grid points left per sub-grid and thus we transfer the whole RHS from GPU to CPU and do the parallel CG iterations on CPU. After that, the solution is transfered back from CPU to GPU.

3 Walberla

WaLBerla is a massively parallel software framework developed for HPC applications on block-structured domains [35]. It has been successfully used in many multi-physics simulation tasks ranging from free surface flows [36] to particulate flows [37] and fluctuating lattice Boltzmann [38] for nano fluids.

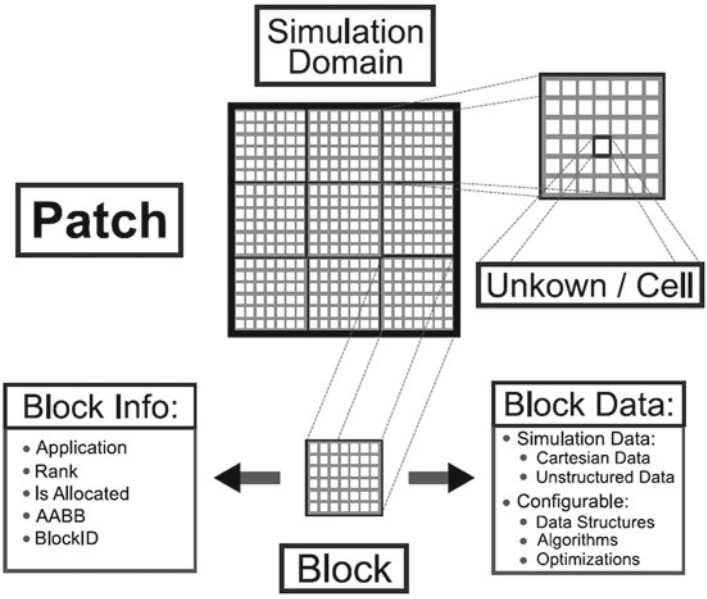


Fig. 1. Patches and Blocks in waLBerla [39].

The main design goals of the waLBerla framework are to provide excellent application performance across a wide range of computing platforms and the easy integration of new algorithms. The current version waLBerla 2.0 is capable of running heterogeneous simulations on CPUs and GPUs with static load balancing [39].

3.1 Patch, Block, and Sweep Concept

A fundamental design concept of waLBerla is to rely on block-structured grids, what we call our Patch and Block data structure. We restrict ourselves to block-structured grids in order to support efficient massively parallel simulations.

In our case a Patch denotes a cuboid describing a region in the simulation that is discretized with the same resolution (see Fig. 1) . This Patch is further subdivided into a Cartesian grid of Blocks consisting of cells. The actual simulation data is located on these cells. In parallel one or more Blocks can be assigned to each process in order to support load balancing strategies. Furthermore, we may specify for each Block, on which hardware it is executed. Of course, this requires also to be able to choose different implementations that run on a certain Block, what is realized by our functionality management.

The functionality management in waLBerla 2.0 controls the program flow. It allows to select different functionality (e.g. kernels, communication functions) for different granularities, e.g. for the whole simulation, for individual processes, and for individual Blocks.

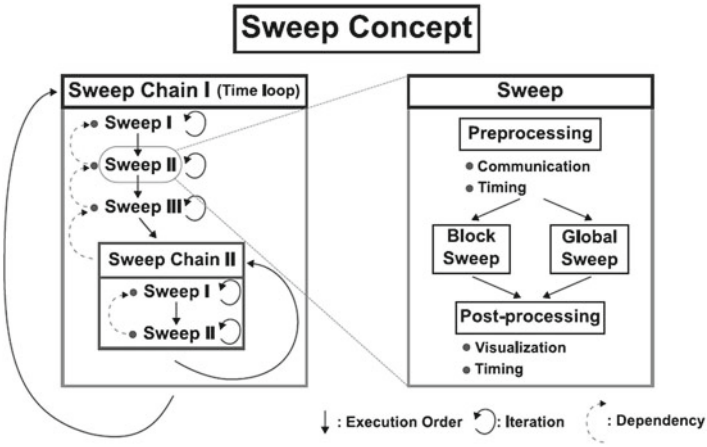


Fig. 2. Sweep concept in waLBerla [39].

When the simulation runs, all tasks are broken down into several basic steps, so-called Sweeps. A Sweep consists of two parts as shown in Fig. 2: a communication step fulfilling the boundary conditions for parallel simulations by nearest neighbor communication and a communication independent work step traversing the process-local Blocks and performing operations on all cells. The work step usually consists of a kernel call, which is realized for instance by a function object or a function pointer. As for each work step there may exist a list of possible (hardware dependent) kernels, the executed kernel is selected by our functionality management.

3.2 MPI Parallelization

The parallelization of waLBerla can be broken down into three steps:

1. a data extraction step,
2. a MPI communication step, and
3. a data insertion step.

During the data extraction step, the data that has to be communicated is copied from the simulation data structures of the corresponding Blocks. Therefore, we distinguish between process-local communication for Blocks lying on the same and MPI communication for those on different processes.

Local communication directly copies from the sending Block to the receiving Block, whereas for the MPI communication the data has first to be copied into buffers. For each process to which data has to be sent, one buffer is allocated. Thus, all messages from Blocks on the same process to another process are serialized. To extract the data to be communicated from the simulation

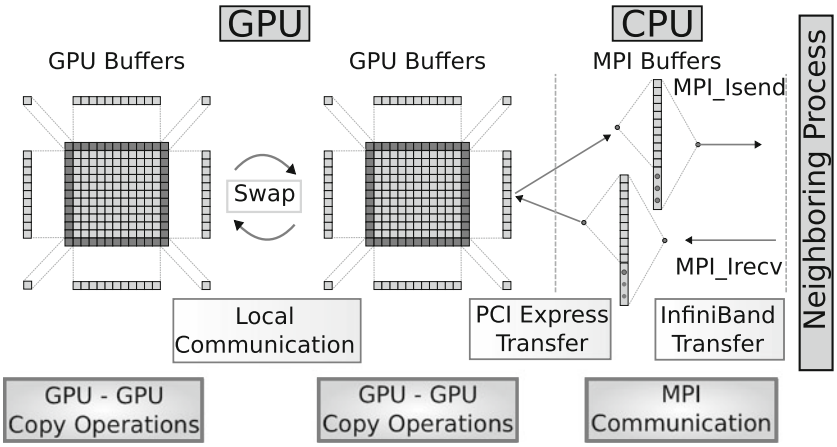


Fig. 3. Communication concept within WaLBerla [39]. Depicted is a process having two Blocks. Communication between the process-local Blocks is realized by swapping of the corresponding buffers, whereas MPI communication involves PCIe transfers of the GPU buffers. GPU-GPU copy operations are required to extract and insert data from the data fields to and from the buffers.

data, extraction function objects are used that are again selected via the functionality management. The data insertion step is similar to the data extraction, besides that we traverse the block messages in the communication buffers instead of the Blocks.

3.3 Multi-GPU Implementation

For parallel simulations on GPUs, the boundary data of the GPU has first to be copied by a PCIe transfer to the CPU and then be communicated via MPI routines. Therefore, we need buffers on GPU and CPU in order to achieve fast PCIe transfers. In addition, on-GPU copy kernels are added to fill these buffers. The whole communication concept is depicted in Fig. 3.

The only difference between parallel CPU and GPU implementation is that we need to adapt the extraction and insertion functions. For the local communication they simply swap the GPU buffers, whereas for the MPI communication we copy the data directly from the GPU buffers into the MPI buffers and vice versa. To support heterogeneous simulations on GPUs and CPUs, we execute different kernels on CPU and GPU and also define a common interface for the communication buffers, so that an abstraction from the hardware is possible. Additionally, the work load of the CPU and the GPU processes can be balanced e.g. by allocating several Blocks on each GPU and only one on each CPU-only process. In addition to that it is also possible to divide a Block into several Sub-Blocks of different sizes to enable load balancing on heterogeneous compute nodes containing e.g. GPUs and CPUs.

Table 1. Specifications of the Tsubame 2.0 cluster.

Compute Nodes	1408
Processor	Intel Xeon X5670
GPU	NVIDIA Tesla M2050
GPUs per Compute Node	3
LINPACK[a] Performance	1192 TFLOPS
Power Consumption	1398.61 KW
Flops per Watt	852.27 FLOPS/W
Network Type	Fat Tree
Interconnect	QDR Infiniband

[a] http://www.netlib.org/linpack

4 Hardware and Performance Model

4.1 Tsubame 2.0

We perform all numerical tests in this article on Tsubame 2.0[3] that is currently (Nov. 2011) number 5 in the TOP 500 list. The detailed hardware specifications of this multi-GPU cluster are listed in Table 1.

All 1408 compute nodes are equipped with three NVIDIA Tesla M2050 GPU accelerators each having 3 GB of GPU memory. NVIDIA Tesla M2050 has a floating-point performance (single precision) of 1030 GFLOP/s and 515 GFLOP/s (double precision) coming from 448 CUDA streaming processors capable of doing 2 floating point operations per cycle and a processor frequency of 575 MHz. Thus, most of Tsubame's 2.4 PFlops peak performance comes from its 4224 GPUs. The GPU memory frequency is 1550 MHz with DDR5 (factor 2) RAM and a 384 Bit memory bus what results in 148 GB/s peak memory bandwidth.

4.2 Performance Model

Next we derive a very simple performance model for our multigrid solver in order to identify performance bottlenecks and to estimate the overall runtime for a given problem size on Tsubame 2.0. In general, we can split the runtime t into the time for the compute kernels, e.g. for the smoother, restriction or interpolation, and the time for communicating data between neighboring processes, mainly exchanging ghost layers after smoothing, residual, and interpolation. An important decision is, if one overlaps computation and communication. If we do not overlap them, the runtime is just the sum

$$t = t_{\text{kernel}} + t_{\text{buffer}} + t_{\text{PCI}} + t_{\text{MPI}} \tag{5}$$

of runtime of all kernels t_{kernel}, the time for copying data from ghost layers to send and receive buffers t_{buffer}, the time for PCIe transfers t_{PCI}, and the time for MPI communication t_{MPI}.

[3] http://www.gsic.titech.ac.jp/en/tsubame2, Nov. 2011.

In order to enable overlapping, we have to split the kernels into inner kernels and outer kernels, where the latter are just processing the points lying near boundary layers. After the outer kernels are finished and the GPU buffers are filled, we can communicate the ghost layers via several CUDA streams and asynchronous PCIe transfers. In parallel run the inner kernels, i.e.

$$t = t_{\text{o,kernel}} + t_{\text{buffer}} + \max(t_{\text{i,kernel}}, t_{\text{PCI}} + t_{\text{MPI}}). \tag{6}$$

Kernel Performance. First we take a closer look at the kernel performance on a single GPU. From previous work we already know that our multigrid algorithm is bounded by memory bandwidth (this also can be easily checked e.g. by a profiler). For the most time consuming part, the smoother, where we basically do a sparse matrix (stencil) vector product, we have to load per grid point one value of the right hand side, seven values of the solution, and we store one value of the solution. The loads of the solution can be partly cached (at least three rows in one grid plane), such that we can assume to require only one load per plane in the solution array, i.e. instead of seven we have three loads. Since we do not split the red and black points of the solution into separate arrays in memory, we assume that we must load and store the full array twice, once within the update iteration of the red and once of the black points. Table 2 summarizes the estimated load and store instructions for the different multigrid components. We denote the number of grid points on the finest grid $l = 0$ by $N = N_0$. On the coarser grids we have $N_l = \frac{N_{l-1}}{8}$ grid points in 3D. Thus, the overall number of grid points on $L-1$ grid levels is roughly $N_{\text{mg}} = N_0 \cdot (1 + \frac{1}{8} + \ldots + \frac{1}{8^L}) \approx N_0 \cdot 1.14$.

Table 2. Number of load and store instructions for different multigrid components per (fine) grid point. Additionally we list the required number of ghost layer exchanges in the multi-GPU case.

Component	Loads	Stores	Ghost layer exchanges
Smoothing	$2 \cdot (3 + 1) = 8$	2	2
Residual	$3 + 1 = 4$	1	1
Restriction	3	$\frac{1}{8}$	0
ProlongationAdd	$1 + \frac{3}{8}$	1	1

Table 3. Memory bandwidths on Tsubame 2.0 for data transfers within one GPU (DDR5), between two GPUs on one compute node (PCIe), and between two compute nodes (Infiniband).

	Theoretical Memory Bandwidth [GB/s]	Attainable Memory Bandwidth [GB/s]
DDR5	148	95
PCIe	8	6 (shared by 3 GPUs)
Infiniband (bidirectional)	5	5

Since we know that our multigrid solver is bandwidth limited we can estimate the kernel time and communication time from the maximum attainable memory and network bandwidth that we measured in Table 3 via standard streaming benchmarks. Note that we neglect the fact that the PCIe bandwidth for GPU to CPU resp. GPU to CPU transfers differs and that the bandwidth depends on the message size, i.e. for smaller message sizes the bandwidth is much lower and the latency dominates.

As an example one ω-RBGS iteration for problem size $N = 512^2 \times 256$ takes 60.5 ms what corresponds to approximately 89 GB/s memory bandwidth on one GPU. All our numerical tests run with double floating-point precision. Our performance model with data from Tables 2 and 3 estimates

$$t_{\text{RBGS}} = \frac{8 \cdot N \cdot 10}{95\,\text{GB/s}} \approx 56.5\,\text{ms}\,. \tag{7}$$

Thus, our model is quite accurate for the RBGS smoother. However, this holds only for larger problem sizes. In order to show the dependency of our smoother on the problem size we depict the runtimes and bandwidth of one ω-RBGS iteration with varying sizes in Fig. 4.

For smaller problems, the GPU overhead e.g. for CUDA kernel calls becomes visible and there is not enough work to be done in parallel and thus most of the compute cores idle.

For one multigrid V(2,2)-cycle with 6 grid levels we measure 364 ms on one GPU (corresponding to approximately 85 GB/s memory bandwidth), our performance model predicts 325 ms.

In order to give more insight in the runtime behavior of the different parts of the multigrid solver, Fig. 5 shows the portions of predicted and measured runtime spent in different components of a V(2,2)-cycle on one GPU. The problem size shrinks by a factor of 8 for each grid level, thus one expects the coarse grid (this includes all the previous components on all coarser grids plus solving the

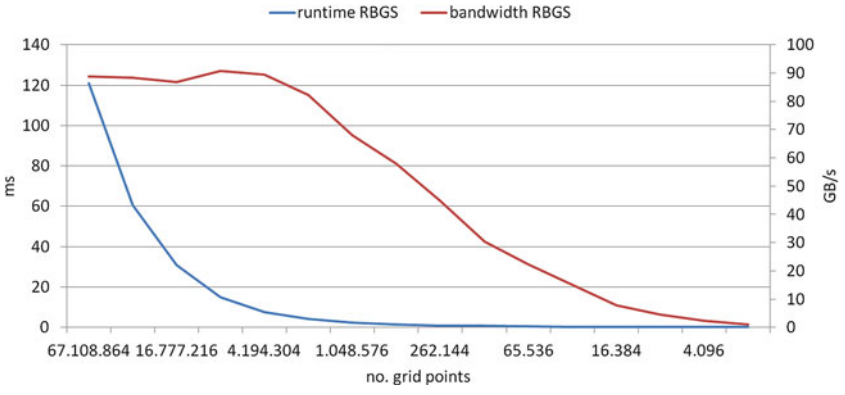
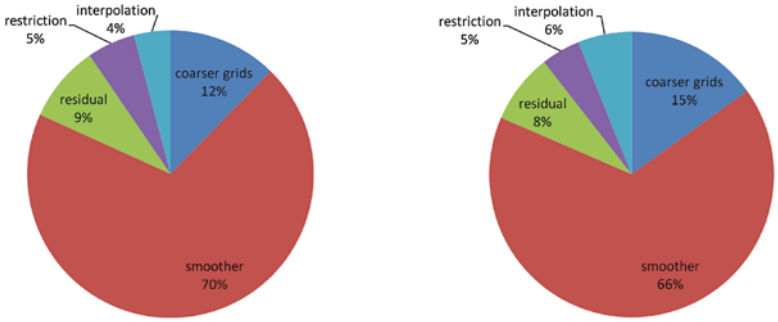


Fig. 4. Single GPU ω-RBGS runtime and bandwidth for different problem sizes.



(a) (b)

Fig. 5. Runtime percentage for different components predicted by our performance model (a) and measured (b) on one GPU (problem size $512^2 \times 256$).

problem on the coarsest grid with CG iterations) to require about 1/8 of the runtime. The measurement lies a little bit higher especially for GPUs, because the smaller sizes are not very efficient on GPU as seen before.

Summarizing, our predictions on one GPU are quite accurate and the model error is typically below 10 %.

For overlapping computation and communication we split the smoother kernel into an inner and outer kernel. This increases the runtime e.g. for one ω-RBGS iteration for problem size $512^2 \times 256$ by approximately 6 % on one GPU. Therefore, we assume $t_{\text{i,kernel}} = t_{\text{kernel}}$ and $t_{\text{o,kernel}} = 0.06 \cdot t_{\text{kernel}}$ in the following. A simple bandwidth based runtime estimate for $t_{\text{o,kernel}}$ is not feasible because of the relatively small number of boundary points and the non-contiguous memory accesses for four of the six boundary layers.

Communication Time. The same problems as for $t_{\text{o,kernel}}$ we also have when trying to estimate t_{buffer} for the multi-GPU case. Thus, we also fall back to measured times here that depend on the number of neighboring processes. In worst case, six ghost layers have to be copied into and from buffers. From this we measure $t_{\text{buffer}} \approx 0.05 \cdot t_{\text{kernel}}$. t_{PCI} and t_{MPI} we are able to predict using information about the number of ghost layer exchanges from Table 2 and the bandwidths from Table 3. Note that within one smoothing iteration we have two ghost layer exchanges, one after updating the red points and one after updating the black points. The PCIe transfer time is the sum of the transfer from GPU to CPU and back (if more than one of the three GPUs on a compute node is used the attainable PCIe bandwidth is shared and thus reduces to 3 resp. 2 GB/s). We neglect that the MPI communication time differs from within one node and between two nodes.

For problem size $N = 512^2 \times 256$ the number of ghost points on the six boundary planes is $512^2 + 4 \cdot 256 \cdot 512$, i.e. the surface to volume ratio is $1 : 64$. On the

coarser grids the ratio goes down to 1 : 2 on grid level 6. In this setting we measure 89 ms on 48 GPUs for one ω-RBGS iteration on the finest grid level, if we do not overlap computation and communication. Our communication model predicts $t_{\text{PCI}} = 16.8\,\text{ms}$ and $t_{\text{MPI}} = 3.4\,\text{ms}$, i.e. $t_{\text{RBGS}} = 56.5 + 16.8 + 3.4 + 5 = 81.7\,\text{ms}$.

To estimate the overall time for overlapping computation and communication, we observe that the sum $t_{\text{PCI}} + t_{\text{MPI}}$ is much lower than the time for an inner smoothing kernel, therefore the communication time should not be visible for the parallel smoother, i.e. $t = t_{\text{o,kernel}} + t_{\text{buffer}} + t_{\text{i,kernel}}$.

5 Scaling Experiments

Next we check the achievable parallel efficiency and speedup of our multigrid multi-GPU implementation on Tsubame 2.0. Baseline is the performance on one GPU.

We distinguish two types of experiments: *Weak scaling* relates to experiments were the problem size is increased linearly with the number of involved GPUs, whereas the term *strong scaling* implies that we have a constant global problem size and vary only the number of processes. Assuming a perfect parallelization, we expect the runtime to be constant in weak scaling experiments, while we expect the runtime to be reciprocally proportional to the number of parallel processes in strong scaling experiments. To estimate the quality of our parallelization we compute speedup $S_p = \frac{t_1}{t_p}$ and parallel efficiency $E_p = \frac{S_p}{p}$ given the runtimes t_1 and t_p on one and on p GPUs.

We measure the runtime of one V(2,2)-cycle (i.e. V-cycles with two ω-RBGS iterations for pre- and postsmoothing each) on six grid levels with parameters from Sect. 2, if not stated otherwise. On the coarsest grid between 15 and 20 CG iterations are performed. All our experiments are done with double floating point accuracy.

5.1 Weak Scaling

Figure 6 shows the weak scaling behavior of the code for problem size $512^2 \times 256$ for non-overlapping communication and computation and when overlapping communication and computation within the smoother. Here, we measure the time spent to do pre- and postsmoothing (step 4 and 10 in Algorithm 1) on the finest grid level (Smoother fine), the time spent to solve the problem on all coarser grid levels (Coarser Grids), and the overall time for one V(2,2)-cycle. In addition to that the efficiency for one V-cycle is shown. In contrast to nearly perfect weak scaling also on large CPU clusters (cf. [23]) the overall parallel efficiency drops to 35 % in the first case and to 42 % in the second case on 1029 GPUs, what was the maximum number of GPUs available for us on Tsubame 2.0. This has mainly two reasons: first the effect of additional intra-node memory transfers of ghost layers between CPU and GPU via the PCIe bus when using GPUs, and second the CG iterations on the coarsest grid that are done on CPU and require a global all-to-all communication. Overlapping of computation and

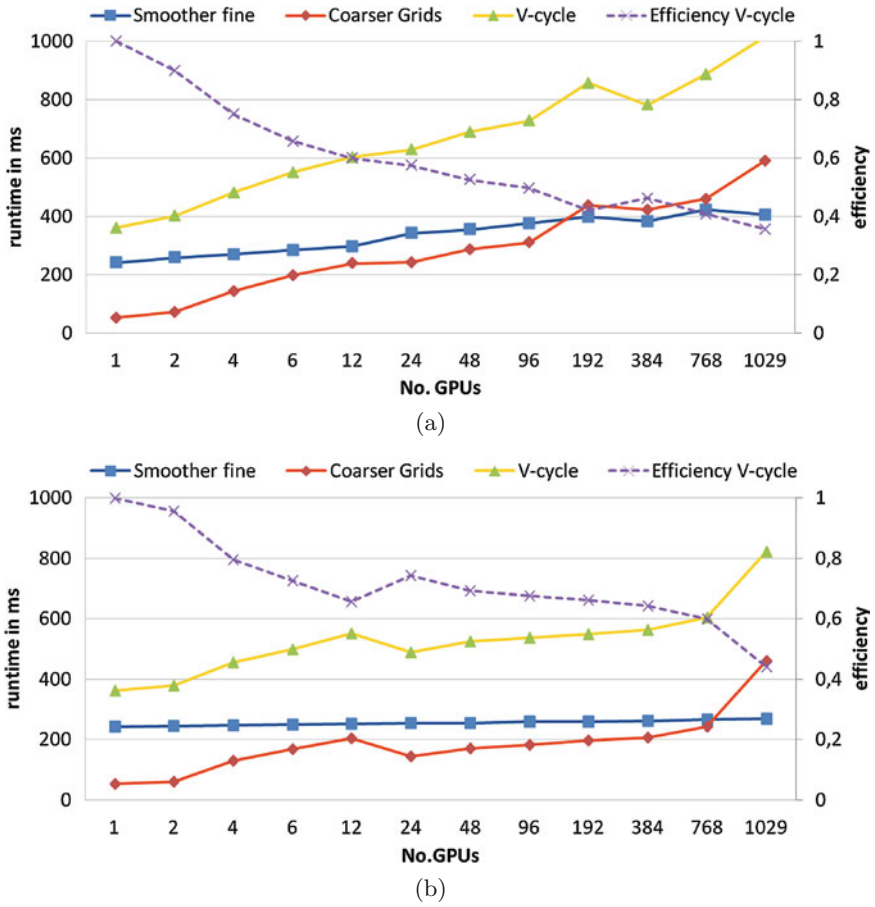


Fig. 6. Weak scaling behavior and parallel efficiency for non-overlapping communication (a) and overlapping communication in the smoother (b) from one to 1029 GPUs on Tsubame performing one multigrid V(2,2)-cycle.

communication within the smoother improves the parallel efficiency and the overall runtime on 1029 GPUs is about 870 ms in this case, where 40 % of the time are spent within the smoother and about 30 % on the coarsest grid doing CG iterations.

5.2 Strong Scaling

Next, we scale the number of involved processing units, but leave the total problem size, i.e. the number of grid points, constant. In this subsection we do not overlap communication and computation. Figure 7 shows the runtimes for $512^2 \times 256$ solved on up to 16 GPUs. The maximal speedup is just 2.3 achieved on 8 GPUs, which is a result of different factors: on the one hand the problems

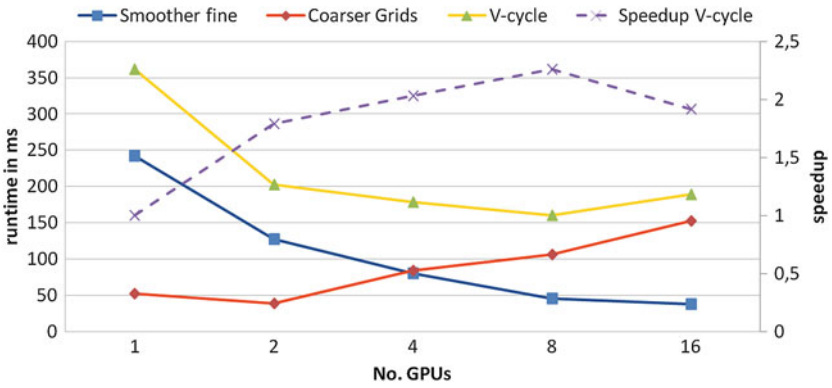


Fig. 7. Strong scaling and speedups for one V(2,2)-cycle with 512×256^2 grid points per GPU.

Table 4. Runtimes of one V(2,2)-cycle for varying problem sizes and 5 or 6 grid levels on 1029 GPUs

Unknowns (in million)	No. of levels	Runtime in ms
69055	6	1025
34528	6	583
17264	6	**322**
8632	5	461
4316	5	261
2158	5	171
1079	5	127
539	5	**125**
270	5	130

for small size mentioned when discussing the single-node performance and on the other hand the communication overhead addressed within the weak scaling experiments.

Table 4 shows runtime results for different problem sizes on 1029 GPUs in order to determine the optimal problem size on this number of GPUs. For the largest run taking 1025 ms our performance model predicts only 445 ms with a ratio computation to communication of 2.7 : 1, the model error is mainly caused by the coarse grid solver. The minimal runtime on 1029 GPUs we find for 539 million grid points, here one V(2,2)-cycle takes 125 ms and communications dominates computation roughly 4 : 1 due to our performance model.

6 Conclusions and Future Work

We have implemented a geometric multigrid solver on GPU and integrated it into the waLBerla framework. First results show acceptable scalability on Tsubame 2.0 up to 1029 GPUs.

One of the next steps is a performance optimization of our code. On one GPU, one obvious improvement would be to use an optimized data layout by splitting the red and black grid points into two separate arrays in memory. In the multi-GPU case we next implement overlapping communication also for the remaining multigrid components besides the smoother. We will also further investigate the CG coarse grid solver and possible alternative parallel (direct) solvers. It is possible to refine the performance model, e.g. to take into account different bandwidths for each grid level like in [22] or to model the performance of the CG solver as done in [23]. The next major change in waLBerla will be to support local grid refinement within the computational domain. Besides adaptive multigrid methods this allows us to reduce the number of processes on coarser grids to achieve a better scalability.

Acknowledgment. We are grateful to have the opportunity to test our multigrid solver on Tsubame 2.0.

References

1. NVIDIA Cuda Programming Guide 4.2. http://developer.nvidia.com/nvidia-gpu-computing-documentation (2012)
2. Ohshima, S., Hirasawa, S., Honda, H.: OMPCUDA : OpenMP execution framework for CUDA based on omni OpenMP compiler. In: Sato, M., Hanawa, T., Müller, M.S., Chapman, B.M., de Supinski, B.R. (eds.) IWOMP 2010. LNCS, vol. 6132, pp. 161–173. Springer, Heidelberg (2010)
3. Klöckner, A., Pinto, N., Lee, Y., Catanzaro, B., Ivanov, P., Fasih, A.: PyCUDA and PyOpenCL: a scripting-based approach to GPU run-time code generation. Parallel Comput. **38**, 157–174 (2012)
4. Fattal, R., Lischinski, D., Werman, M.: Gradient domain high dynamic range compression. ACM Trans. Graph. **21**(3), 249–256 (2002)
5. Köstler, H.: A Multigrid Framework for Variational Approaches in Medical Image Processing and Computer Vision. Verlag Dr. Hut, München (2008)
6. Goodnight, N., Woolley, C., Lewin, G., Luebke, D., Humphreys, G.: A multigrid solver for boundary value problems using programmable graphics hardware. In: ACM SIGGRAPH 2005 Courses, p. 193. ACM Press, New York (2005)
7. Feng, Z., Li, P.: Multigrid on GPU: tackling power grid analysis on parallel simt platforms. In: IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2008, pp. 647–654. IEEE Computer Society, Washington, DC (2008)
8. Bolz, J., Farmer, I., Grinspun, E., Schröder, P.: Sparse matrix solvers on the GPU: conjugate gradients and multigrid. In: ACM SIGGRAPH 2003 Papers, pp. 917–924. ACM (2003)
9. Göddeke, D., Strzodka, R.: Cyclic reduction tridiagonal solvers on gpus applied to mixed-precision multigrid. IEEE Trans. Parallel Distrib. Syst. **22**(1), 22–32 (2011)
10. Göddeke, D., Strzodka, R., Mohd-Yusof, J., McCormick, P., Wobker, H., Becker, C., Turek, S.: Using GPUs to improve multigrid solver performance on a cluster. Int. J. Comput. Sci. Eng. **4**(1), 36–55 (2008)
11. Göddeke, D.: Fast and Accurate Finite-Element Multigrid Solvers for PDE Simulations on GPU Clusters. Logos Verlag, Berlin (2011)

12. Haase, G., Liebmann, M., Douglas, C.C., Plank, G.: A parallel algebraic multigrid solver on graphics processing units. In: Zhang, W., Chen, Z., Douglas, C.C., Tong, W. (eds.) HPCA 2009. LNCS, vol. 5938, pp. 38–47. Springer, Heidelberg (2010)
13. Cohen, J.: OpenCurrent, Nvidia research. http://code.google.com/p/opencurrent/ (2011)
14. Balay, S., Buschelman, K., Gropp, W.D., Kaushik, D., Knepley, M.G., McInnes, L.C., Smith, B.F., Zhang, H.: PETSc Web page. http://www.mcs.anl.gov/petsc (2009)
15. Grossauer, H., Thoman, P.: GPU-based multigrid: real-time performance in high resolution nonlinear image processing. In: Gasteratos, A., Vincze, M., Tsotsos, J.K. (eds.) ICVS 2008. LNCS, vol. 5008, pp. 141–150. Springer, Heidelberg (2008)
16. Gwosdek, P., Zimmer, H., Grewenig, S., Bruhn, A., Weickert, J.: A highly efficient GPU implementation for variational optic flow based on the Euler-Lagrange framework. In: Kutulakos, K.N. (ed.) ECCV 2010 Workshops, Part II. LNCS, vol. 6554, pp. 372–383. Springer, Heidelberg (2012)
17. Zimmer, H., Bruhn, A., Weickert, J.: Optic flow in harmony. Int. J. Comput. vis. **93**(3), 368–388 (2011)
18. Wang, X., Aoki, T.: Multi-GPU performance of incompressible flow computation by lattice boltzmann method on GPU cluster. Parallel Comput. **37**, 512–535 (2011)
19. Gradl, T., Rüde, U.: High performance multigrid in current large scale parallel computers. In: 9th Workshop on Parallel Systems and Algorithms (PASA), vol. 124, pp. 37–45 (2008)
20. Gradl, T., Freundl, C., Köstler, H., Rüde, U.: Scalable multigrid. In: High Performance Computing in Science and Engineering, Garching/Munich 2007, pp. 475–483 (2009)
21. Bergen, B., Gradl, T., Hülsemann, F., Rüde, U.: A massively parallel multigrid method for finite elements. Comput. Sci. Eng. **8**(6), 56–62 (2006)
22. Köstler, H., Stürmer, M., Pohl, T.: Performance engineering to achieve real-time high dynamic range imaging. J. Real-Time Image Proc., pp. 1–13 (2013)
23. Gmeiner, B., Köstler, H., Stürmer, M., Rüde, U.: Parallel multigrid on hierarchical hybrid grids: a performance study on current high performance computing clusters. Practice and Experience, Concurrency and Computation (2012)
24. Kazhdan, M., Hoppe, H.: Streaming multigrid for gradient-domain operations on large images. ACM Trans. Graph. (TOG) **27**, 21 (2008). (ACM Press, New York)
25. Bartuschat, D., Ritter, D., Rüde, U.: Parallel multigrid for electrokinetic simulation in particle-fluid flows. In: 2012 International Conference on High Performance Computing and Simulation (HPCS), pp. 374–380. IEEE (2012)
26. Köstler, H., Ritter, D., Feichtinger, C.: A geometric multigrid solver on GPU clusters. In: Yuen, D.A., Wang, L., Chi, X., Johnsson, L., Ge, W., Shi, Y. (eds.) GPU Solutions to Multi-scale Problems in Science and Engineering. Lecture Notes in Earth System Sciences, pp. 407–422. Springer, Heidelberg (2013)
27. Brandt, A.: Multi-level adaptive solutions to boundary-value problems. Math. Comput. **31**(138), 333–390 (1977)
28. Hackbusch, W.: Multi-Grid Methods and Applications. Springer, Heidelberg (1985)
29. Briggs, W., Henson, V., McCormick, S.: A Multigrid Tutorial, 2nd edn. Society for Industrial and Applied Mathematics (SIAM), Philadelphia (2000)
30. Trottenberg, U., Oosterlee, C., Schüller, A.: Multigrid. Academic Press, San Diego (2001)
31. Douglas, C., Hu, J., Kowarschik, M., Rüde, U., Weiß, C.: Cache optimization for structured and unstructured grid multigrid. Electron. Trans. Numer. Anal. (ETNA) **10**, 21–40 (2000)

32. Hülsemann, F., Kowarschik, M., Mohr, M., Rüde, U.: Parallel geometric multigrid. In: Bruaset, A., Tveito, A. (eds.) Numerical Solution of Partial Differential Equations on Parallel Computers. Lecture Notes in Computational Science and Engineering, vol. 51, pp. 165–208. Springer, Heidelberg (2005)
33. Stürmer, M., Wellein, G., Hager, G., Köstler, H., Rüde, U.: Challenges and potentials of emerging multicore architectures. In: Wagner, S., Steinmetz, M., Bode, A., Brehm, M., eds.: High Performance Computing in Science and Engineering, Garching/Munich 2007, LRZ, KONWIHR, pp. 551–566. Springer, Heidelberg (2008)
34. Shewchuk, J.: An introduction to the conjugate gradient method without the agonizing pain (1994)
35. Feichtinger, C., Donath, S., Köstler, H., Götz, J., Rüde, U.: WaLBerla: HPC software design for computational engineering simulations. J. Comput. Sci. **2**(2), 105–112 (2011)
36. Donath, S., Feichtinger, Ch., Pohl, T., Götz, J., Rüde, U.: Localized parallel algorithm for bubble coalescence in free surface lattice-boltzmann method. In: Sips, H., Epema, D., Lin, H.-X. (eds.) Euro-Par 2009. LNCS, vol. 5704, pp. 735–746. Springer, Heidelberg (2009)
37. Götz, J., Iglberger, K., Feichtinger, C., Donath, S., Rüde, U.: Coupling multibody dynamics and computational fluid dynamics on 8192 processor cores. Parallel Comput. **36**(2–3), 142–151 (2010)
38. Dünweg, B., Schiller, U., Ladd, A.J.C.: Statistical mechanics of the fluctuating lattice boltzmann equation. Phys. Rev. E **76**(3), 036704 (2007)
39. Feichtinger, C., Habich, J., Köstler, H., Hager, G., Rüde, U., Wellein, G.: A flexible patch-based lattice boltzmann parallelization approach for heterogeneous GPU-CPU clusters. J. Parallel Comput. **37**(9), 536–549 (2011)

Author Index