Pattern Recognition 137 (2023) 109318

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Arbitrary Order Total Variation for Deformable Image Registration

Jinming Duan[a,b,*], Xi Jia[a], Joseph Bartlett[a,c], Wenqi Lu[d], Zhaowen Qiu[e,*]

[a] School of Computer Science, University of Birmingham, Birmingham, UK
[b] Alan Turing Institute, London, UK
[c] Department of Biomedical Engineering & Melbourne Brain Centre Imaging Unit, University of Melbourne, Melbourne, Australia
[d] Tissue Image Analytics Centre, Department of Computer Science, University of Warwick, Coventry, UK
[e] Institute of Information Computer Engineering, Northeast Forestry University, Harbin, China

ARTICLE INFO

Article history:
Received 6 September 2022
Revised 6 December 2022
Accepted 8 January 2023
Available online 13 January 2023

Keywords:
Image Registration
Nonlinear Optimisation
ADMM
Total Variation
Arbitrary Order Derivatives

ABSTRACT

In this work, we investigate image registration in a variational framework and focus on regularization generality and solver efficiency. We first propose a variational model combining the state-of-the-art sum of absolute differences (SAD) and a new arbitrary order total variation regularization term. The main advantage is that this variational model preserves discontinuities in the resultant deformation while being robust to outlier noise. It is however non-trivial to optimize the model due to its non-convexity, non-differentiabilities, and generality in the derivative order. To tackle these, we propose to first apply linearization to the model to formulate a convex objective function and then break down the resultant convex optimization into several point-wise, closed-form subproblems using a fast, over-relaxed alternating direction method of multipliers (ADMM). With this proposed algorithm, we show that solving higher-order variational formulations is similar to solving their lower-order counterparts. Extensive experiments show that our ADMM is significantly more efficient than both the subgradient and primal-dual algorithms particularly when higher-order derivatives are used, and that our new models outperform state-of-the-art methods based on deep learning and free-form deformation. Our code implemented in both Matlab and Pytorch is publicly available at https://github.com/j-duan/AOTV.

1. Introduction

Image registration is a process of matching two or more images of the same or similar object from different times, viewpoints, or imaging modalities. It has been widely used in many areas, such as art, astronomy, biology, chemistry, criminology, physics, remote sensing, etc. In medical imaging, registration enables direct comparison of images taken at different stages of progression of a disease, which is essential for disease diagnosis, treatment, and monitoring.

In the case of mono-modal registration, this task can be approached using a variational framework. A common method of estimating geometric transformations (deformations) is to minimise the sum of squared differences (SSD) between the warped source image (aka. moving or template image) and the target image (aka. fixed or reference image). SSD assumes that the underlying noise follows a Gaussian distribution, and is not robust to outlier noise. To overcome this, the L_1 similarity measure may be used. The registration problem is often ill-posed as there is no guarantee of an unique solution, making regularization vital. Many works tackle the question of how to select optimal regularization terms which allow plausible and reliable deformations to be computed. In the literature, regularizations range from using first-order derivatives, such as the diffusion regularization [1,2], the total variation regularization [3–6], the fluid regularization [7,8], and the bounded deformation regularization [9], to using higher-order counterparts, such as the bending energy [10], the biharmonic (aka. linear curvature) regularization [11,12], the linear/non-linear elasticity regularization [12–14], the mean curvature regularization [15,16], and the Gaussian curvature regularization [17]. There are also regularizations that combine hybrid derivatives [18,19], explore effectiveness of non-local (graph) [20–23] or fractional-order derivatives [24,25], or are learned directly from data [71].

However, there remain limitations in current regularizations: (1) Most regularizations were developed in a quadratic form [7,8,10–12,15–18]. They may be able to generate a globally smooth

* Corresponding authors.
E-mail addresses: j.duan@bham.ac.uk (J. Duan), qiuzw@nefu.edu.cn (Z. Qiu).

https://doi.org/10.1016/j.patcog.2023.109318

and satisfactory local deformation field, but cannot preserve discontinuities in the deformation. Such discontinuities often appear in medical imaging at organ boundaries primarily due to respiratory motion but also intensity inhomogeneity, pathological tissues or heavy noise [9]. Although some regularizations [3–6,9] allow discontinuities, they are based on first-order derivatives, and the smoothness induced may not be strong enough to regularize the resulting deformation. Furthermore, according to [11,12] first-order methods are more dependant on initial alignments between images and they are more suitable when an affine linear pre-registration is available, which is not the case for higher-order models because they do not penalize linear transformations[1]. As such, it is challenging to design a variational model that is able to induce both smoothness and discontinuity whilst being less sensitive to initializations. (2) Some higher-order regularizations are non-convex [13–17]. Together with an almost unavoidably non-convex data term, it becomes drastically difficult to devise effective and efficient numerical algorithms to optimize such an energy functional involving dual non-convexities. (3) Almost all higher-order models are second order [10–12,15–18]. As such, it is unclear how the numerical behaviour changes when a derivative order higher than two is used. (4) Non-local and fractional methods [20,21,24] define derivatives using many more pixel points, making their implementation non-trivial and expensive.

The minimization process of a variational registration model entails the *calculus of variations*, by which one obtains an *Euler-Lagrange* equation with respect to the deformation, which is normally a coupled, higher-order, and non-linear *partial differential equation* (PDE). Various numerical algorithms have been proposed to solve such equations. Among them, most use time-dependent gradient descent by introducing an artificial time variable and then determining the steady state solution of such PDEs [1,11–14,26]. These algorithms are explicit methods based on the time marching schemes and unfortunately are quite slow. Others handled the PDEs directly, with common choices being the semi-implicit fixed-point iteration [27,28], Newton-type methods [29,30], multigrid methods [5,18], etc. These algorithms however require advanced knowledge about discretization and can be difficult to implement for higher-order PDEs (e.g. over fourth order). Furthermore, the fast primal-dual algorithm was proposed in [3,4,31] to handle the total variation regularization of displacements. However, due to the iterative nature of these methods, they require a sufficiently large number of iterations to achieve high accuracy (e.g. machine precision), thus leading to slow convergence speed. To overcome the limitations of existing image registration methods, in this paper we make four novel contributions by proposing new models and algorithms:

- We propose a new variational model for image registration that uses an arbitrary order total variation regularization. Such a regularization is convex and integrates over the magnitude of an arbitrary order spatial derivative derived from the binomial theorem (see Table 1). Moreover, the data term is a sum of absolute differences (SAD). By combining the SAD data consistency term with the arbitrary order total variation regularization, the proposed model is capable of capturing discontinuities in the resultant deformation, while being robust to outlier noise. However, it is non-trivial to minimize such a model due to the non-convexity of data term, the generality of regularization order, and the non-differentiabilities of both data and regularization terms.
- We propose an effective and efficient algorithm to optimize the proposed variational model. We first tackle its non-convexity by linearizing the data term with the first-order Taylor theorem. As a result, the original problem is converted into a convex optimization, for which we propose a fast ADMM solver based on variable splitting. We then break down this convex problem into three linear subproblems, among which two can be solved by simple thresholding equations (which overcome the non-differentiabilities) and another one solved by the fast discrete cosine transform (DCT) (which handles the generality). All resultant solutions are point-wise and closed-form without any iterations, and are therefore accurate and efficient. Finally, an extra over-relaxation step is introduced to further accelerate the convergence of ADMM without increasing its overall computational complexity.
- We provide rigorous derivations and proofs in Appendix A regarding how DCT can be used to analytically solve a linear PDE of arbitrary order. To the best of our knowledge, this is the first time that DCT is used in image registration to solve arbitrary order PDEs. We also implement a subgradient method in Appendix D as well as a primal-dual method in Appendix B which serve as two baselines to compare with our proposed ADMM. Our finding is that ADMM becomes increasingly more efficient than these competing algorithms for the cases when higher-order derivatives are used. Furthermore, in Appendix C we propose a novel method, based on primal dual, to derive the closed-form solution (8) associated with the SAD data term in our model. The method also applies to deriving soft thresholding (12) associated with the regularization term.
- We perform extensive experiments on three challenging MRI datasets to explain and understand different models and algorithms. Specifically, we design proper numerical experiments to: show the impact of using different higher-order regularizations, compare the robustness of SAD (L_1) and SSD (L_2) data terms, demonstrate the convergence rate and computational efficiency of different algorithms, select reasonable built-in model parameters for speed and accuracy, and compare our method with state-of-the-art (SOTA) methods including a second-order free-form deformation (FFD) [10] and four unsupervised deep learning methods [2,32,57–59].

2. Arbitrary order total variation for image registration

We first propose a non-linear variational registration model using a general, isotropic *nth*-order regularization, given by

$$\min_{u_1,u_2} \int_\Omega |I_1(\mathbf{x}+\mathbf{u}(\mathbf{x})) - I_0(\mathbf{x})| d\mathbf{x} + \lambda \int_\Omega \sqrt{\sum\nolimits_p |\nabla^n u_p(\mathbf{x})|^2} d\mathbf{x}, \quad (1)$$

where $\mathbf{u}(\mathbf{x}) = (u_1(\mathbf{x}), u_2(\mathbf{x}))^{\mathrm{T}} : (\Omega \subseteq \mathbb{R}^2) \to \mathbb{R}^2$ is the two-dimensional displacement field, and $I_0(\mathbf{x})$ and $I_1(\mathbf{x}+\mathbf{u}) : \Omega \to \mathbb{R}$ are a pair of input images. Here, we use the *intensity consistency constraint*, which assumes the intensity at point $\mathbf{x}$ in the target image is the same as that at point $\mathbf{x}+\mathbf{u}$ in the source image, i.e. $I_0(\mathbf{x}) = I_1(\mathbf{x}+\mathbf{u})$. In addition, $\nabla^n u_p(\mathbf{x}) : \Omega \to \mathbb{R}^{2^n}$ is a nth-order distributional derivative of $u_p(\mathbf{x})$ and its definition can be found in Table 1. Theoretically, n can be chosen as an arbitrary integer. However, for larger n the subsequent computations become increasingly challenging. We will investigate and analyze the impact of n in Section 4.1.

The objective here is to find $\mathbf{u}^*$ that minimizes (1). In the data term of (1), we notice that the non-linearity in the function $I_1(\mathbf{x}+\mathbf{u})$ with respect to $\mathbf{u}$ poses a challenge for optimization. To benefit from closed-form solutions, we borrow the idea from the Gauss-Newton algorithm to handle (1). Specifically, if we expand $I_1(\mathbf{x}+\mathbf{u})$ at $\mathbf{u}^\omega$ (here ω is an integer denoting iteration number) using the first-order Taylor theorem (i.e. 2a), then we can approxi-

[1] Consider an affine as a degree one polynomial and it will disappear if we differentiate it more than once.

Table 1
Detailed forms of $\nabla^n u$ and $|\nabla^n u|$ in the case when n varies from 1 to s. When $n = s$, the representation of $\binom{s}{p}$ denotes 's choose p' which is a binomial coefficient. Note that if one computes the variational derivative of a respective $|\nabla^n u|$, it will result in a PDE of order $2n$. $|\cdot|$ represents the Euclidean norm. $t\oplus$ means we repeat the respective differential operator t times

Order	$\nabla^n u(x,y)$	Detailed form of $\nabla^n u$	Magnitude $\lvert\nabla^n u\rvert$	PDE order
n=1	∇u	$\left(\frac{\partial u}{\partial x}\ \frac{\partial u}{\partial y}\right)^\top \in \mathbb{R}^2$	$\sqrt{(\frac{\partial u}{\partial x})^2 + (\frac{\partial u}{\partial y})^2}$	2
n=2	$\nabla^2 u$	$\left(\frac{\partial^2 u}{\partial x^2}\ 2\oplus\frac{\partial^2 u}{\partial x\partial y}\ \frac{\partial^2 u}{\partial y^2}\right)^\top \in \mathbb{R}^4$	$\sqrt{(\frac{\partial^2 u}{\partial x^2})^2 + 2(\frac{\partial^2 u}{\partial x\partial y})^2 + (\frac{\partial^2 u}{\partial y^2})^2}$	4
n=3	$\nabla^3 u$	$\left(\frac{\partial^3 u}{\partial x^3}\ 3\oplus\frac{\partial^3 u}{\partial x^2\partial y}\ 3\oplus\frac{\partial^3 u}{\partial x\partial y^2}\ \frac{\partial^3 u}{\partial y^3}\right)^\top \in \mathbb{R}^8$	$\sqrt{(\frac{\partial^3 u}{\partial x^3})^2 + 3(\frac{\partial^3 u}{\partial x^2\partial y})^2 + 3(\frac{\partial^3 u}{\partial x\partial y^2})^2 + (\frac{\partial^3 u}{\partial y^3})^2}$	6
⋮	⋮	⋮	⋮	⋮
$n = s$	$\nabla^s u$	$\left(\binom{s}{p}\oplus\frac{\partial^s u}{\partial x^{s-p}\partial y^p}\right)^\top \in \mathbb{R}^{2^n},\ p = 1,\ldots,s$	$\sqrt{\sum_{p=0}^{s}(\binom{s}{p}\frac{\partial^s u}{\partial x^{s-p}\partial y^p})^2}$	$2s$

mate (1) around the displacement field u^ω using (2b)

$$I_1(\mathbf{x}+\mathbf{u}) = I_1(\mathbf{x}+\mathbf{u}^\omega) + \langle \nabla I_1(\mathbf{x}+\mathbf{u}^\omega), \mathbf{u}-\mathbf{u}^\omega\rangle, \tag{2a}$$

$$\mathbf{u}^{\omega+1} = \arg\min_{\mathbf{u}} \int_\Omega |\rho_{\mathbf{u}^\omega}(\mathbf{u})| d\mathbf{x} + \lambda \int_\Omega \sqrt{\textstyle\sum_p |\nabla^n u_p|^2}\, d\mathbf{x}, \tag{2b}$$

where in (2b), $\rho_{\mathbf{u}^\omega}(\mathbf{u})$ is the data term linearized with respect to $\mathbf{u}$ around $\mathbf{u}^\omega$:

$$\rho_{\mathbf{u}^\omega}(\mathbf{u}) = I_1(\mathbf{x}+\mathbf{u}^\omega) + \langle \nabla I_1(\mathbf{x}+\mathbf{u}^\omega), \mathbf{u}-\mathbf{u}^\omega\rangle - I_0(\mathbf{x}). \tag{3}$$

In (2a) and (3), $I_1(\mathbf{x}+\mathbf{u}^\omega)$ (or I_1^ω for short) is the geometrically transformed/warped image from $I_1(\mathbf{x})$ using the current iteration of the deformation field $\mathbf{u}^\omega$, ∇ is the gradient operator and $\nabla I_1(\mathbf{x}+\mathbf{u}^\omega)$ represents the spatial derivatives of I_1^ω along the vertical and horizontal directions in the image. $\langle\cdot,\cdot\rangle$ denotes the inner product. To solve (1) we need to iterate between (2a) and (2b). Furthermore (2b) alone, which is the linearized version of (1) also needs to be solved iteratively, therefore, the resulting numerical implementation consists of 2 nested loops.[2].

There are three significant properties of the variational model (1): First, the data term uses SAD, which is based on robust estimation [33]. With this term, the resultant displacement field will be less sensitive to outlier noise in the images to be aligned, see our experiments in Section 4.2 for evidence of this claim. Second, the regularization term uses an arbitrary order total variation (i.e. integration over the magnitude of a nth-order distributional derivative), which is capable of recovering discontinuities (e.g. edges) in the final displacement field. Third, the regularization allows reconstruction of a piecewise polynomial function of arbitrary degree for the estimated displacement field. For example, setting $n = 1$, $n = 2$, and $n = 3$ will cause the displacement field exhibit piecewise constant, piecewise linear, and piecewise quadratic behaviour, respectively. However, it is non-trivial to find an optimal solution to (2b) due to its non-differentiabilities in both data and regularization terms, as well as the generality in the derivative order. To benefit from the advances of the model, in the next section we develop an effective and efficient algorithm to minimize (1). Using the proposed algorithm, we show that solving higher-order variational formulations can be as easy as solving their lower-order counterparts.

The proposed arbitrary order total variation regularization in (1) and (2b) has a close relationship to 1D L_1 trend filtering for signal processing. More technically, when we discretize our regularization using the forward finite difference (which is the case in this paper) and discard the second dimension (e.g. y axis), our regularization is exactly 1D L_1 trend filtering (for an evenly-spaced grid). Trend filtering was independently proposed by Steidl et al. [34], Poschl et al. [35], and Kim et al. [36] and has been recently studied systematically by Tibshirani [37]. Trend filtering has a rich history dating back to 100 years ago and was studied much earlier than total variation denoising [38]. However, apart from these two tasks (signal processing versus image processing) being fundamentally different, there are other distinctive technical differences. For example, in 1D gradient is a scalar but in 2D it is a vector whose dimension increases exponentially with the order of the derivative. We also find a general 2D gradient follows a binomial distribution (see our Table 1) which is not the case in 1D. Further, in 1D discretization is straightforward but in 2D we need to consider spatial relations between pixels and pay special attention to boundary conditions. For these, we need to use the Kronecker products detailed in our Appendix A, which is not the case in 1D. To the best of our knowledge, the proposed general regularization has not been studied in image registration. For these reasons, we consider our regularization a novel contribution.

Our regularization contains only single term and hence is simpler than other higher-order regularizations that hybridize first- and second-order derivatives, such as the total generalized variation (TGV) [39,40] and those proposed in [41–43]. This simplicity allows us to study numerical behaviours of very high-order regularizations (e.g. fourth order). On the other hand, with these hybrid regularizations we will end up with additional regularization parameters as well as more penalty weights (if one decides to use our ADMMs), which may be challenging to find an optimal combination. The overall implementation will be also more complicated than our Algorithm 1 especially for very high-order derivatives. Moreover, first-order methods are often dependant on initial alignments and are more suitable when an affine pre-registration is available (see [11,12] or our Fig. 4). As such, in image registration where large deformations persist it may not be beneficial to mix first- with higher-order derivatives.

3. Faster ADMM with Over-relaxation

We first focus on the minimization problem (2b) with $\rho_{\mathbf{u}^\omega}(\mathbf{u})$ defined in (3). To solve this convex problem, we need to discretize (2b) and (3) to be a function of the pixels in the image domain as

[2] For now we do not consider multi-scale implementation, otherwise we have three loops in the implementation.

opposed to the continuous domain Ω:[3]

$$\min_{\mathbf{u}} \|\rho_{\mathbf{u}^\omega}(\mathbf{u})\|_1 + \lambda \sum_{i,j} \sqrt{|(\nabla^n u_1)_{i,j}|^2 + |(\nabla^n u_2)_{i,j}|^2}, \tag{4a}$$

$$(\rho_{\mathbf{u}^\omega}(\mathbf{u}))_{i,j} = (I_1^\omega - I_0)_{i,j} + (\nabla I_1^\omega)_{i,j}^{\mathrm{T}}(\mathbf{u}_{i,j} - \mathbf{u}_{i,j}^\omega), \tag{4b}$$

where $\mathbf{u} = (u_1, u_2) \in (\mathbb{R}^{M\times N})^2$, $\nabla^n u_1 \in (\mathbb{R}^{M\times N})^{2^n}$, $\nabla^n u_2 \in (\mathbb{R}^{M\times N})^{2^n}$, $I_1^\omega \in \mathbb{R}^{M\times N}$, $I_0 \in \mathbb{R}^{M\times N}$, and $\nabla I_1^\omega \in (\mathbb{R}^{M\times N})^2$. Here n denotes the derivative order, $M \times N$ represents the image size, and 2^n denotes the number of components in $\nabla^n u_1$ or $\nabla^n u_2$ (see the 3rd column of Table 1). Depending on the value of n, the definition of $|(\nabla^n u_p)_{i,j}|, \forall p \in \{1, 2\}$ can be found in the 4th column of Table 1.

Following the philosophy of ADMM for convex problems [44], we introduce the auxiliary splitting variables $\mathbf{w}_1 \in (\mathbb{R}^{M\times N})^{2^n}$, $\mathbf{w}_2 \in (\mathbb{R}^{M\times N})^{2^n}$, $v_1 \in \mathbb{R}^{M\times N}$ and $v_2 \in \mathbb{R}^{M\times N}$, converting (4a) into an equivalent constrained minimization problem

$$\begin{aligned} &\min_{\mathbf{u},\mathbf{v},\mathbf{w}} \|\rho_{\mathbf{u}^\omega}(\mathbf{u})\|_1 + \lambda \sum_{i,j} \sqrt{|(\mathbf{w}_1)_{i,j}|^2 + |(\mathbf{w}_2)_{i,j}|^2} \\ &s.t.\ u_1 = v_1, u_2 = v_2, \mathbf{w}_1 = \nabla^n v_1, \mathbf{w}_2 = \nabla^n v_2. \end{aligned} \tag{5}$$

The introduction of the first two constraints decouples $\mathbf{u}$ in the regularization from that in the data term so that a multi-channel nth-order total variation denoising problem can be explicitly formulated. The last two constraints effectively handle the non-differentiability, non-linearity and generality in the regularization. To guarantee an optimal solution, the above constrained problem (5) can be converted to a saddle problem solved by ADMM. Let $\mathcal{L}_{\mathcal{A}}(\mathbf{u}, \mathbf{v}, \mathbf{w}; \mathbf{b}, \mathbf{d})$ be the augmented Lagrange functional of (5), defined as

$$\begin{aligned} &\mathcal{L}_{\mathcal{A}}(\mathbf{u}, \mathbf{v}, \mathbf{w}; \mathbf{b}, \mathbf{d}) = \\ &\quad \|\rho_{\mathbf{u}^\omega}(\mathbf{u})\|_1 + \lambda \sum_{i,j} \sqrt{|(\mathbf{w}_1)_{i,j}|^2 + |(\mathbf{w}_2)_{i,j}|^2} \\ &\quad + \frac{\theta_1}{2} \sum_{i,j} |(\mathbf{w}_1 - \nabla^n v_1 - \mathbf{b}_1)_{i,j}|^2 \\ &\quad + \frac{\theta_1}{2} \sum_{i,j} |(\mathbf{w}_2 - \nabla^n v_2 - \mathbf{b}_2)_{i,j}|^2 \\ &\quad + \frac{\theta_2}{2} \|v_1 - u_1 - d_1\|_2^2 + \frac{\theta_2}{2} \|v_2 - u_2 - d_2\|_2^2, \end{aligned} \tag{6}$$

where $\mathbf{b}_1 \in (\mathbb{R}^{M\times N})^{2^n}$, $\mathbf{b}_2 \in (\mathbb{R}^{M\times N})^{2^n}$, $d_1 \in \mathbb{R}^{M\times N}$ and $d_2 \in \mathbb{R}^{M\times N}$ are Lagrangian multipliers or dual variables, and θ_1 and θ_2 are penalty weights which have an impact on how fast the minimization process converges and will be studied numerically in Section 4.4. In (6), we have three sets of primal variables $(\mathbf{u}, \mathbf{v}, \mathbf{w})$ and two sets of dual variables $(\mathbf{b}, \mathbf{d})$. Saddle points can be computed when (6) is minimized with respect to primal variables while maximized with respect to dual variables. As per convex optimization [45], one of saddle points for the augmented Lagrange functional (6) will give a minimizer for the constrained minimization problem (5).

3.1. Solving subproblems w.r.t. model variables

To optimize (6), we need to decompose it into three subproblems with respect to $\mathbf{u} = (u_1, u_2)$, $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2)$ and $\mathbf{v} = (v_1, v_2)$, and then update the Lagrangian multipliers $\mathbf{b} = (\mathbf{b}_1, \mathbf{b}_2)$ and $\mathbf{d} = (d_1, d_2)$ until the process converges.

1) $\mathbf{u}$−***subproblem***: This subproblem $\mathbf{u}^{k+1} \leftarrow \min_{\mathbf{u}} \mathcal{L}_{\mathcal{A}}(\mathbf{u}, \mathbf{v}^k, \mathbf{w}^k; \mathbf{b}^k, \mathbf{d}^k)$ is an affine linear L_1 problem which can be solved by considering the following minimization problem

$$\mathbf{u}^{k+1} = \operatorname*{argmin}_{\mathbf{u}} \|\rho_{\mathbf{u}^\omega}(\mathbf{u})\|_1 + \frac{\theta_2}{2} \sum_{p}^{2} \|v_p^k - u_p - d_p^k\|_2^2, \tag{7}$$

which has a closed-form, point-wise solution, given by the following thresholding equation

$$\mathbf{u}^{k+1} = \mathbf{v}^k - \mathbf{d}^k - \frac{\hat{z}}{\max\left(\mathrm{abs}(\hat{z}), 1\right)} \frac{\nabla I_1^\omega}{\theta_2}, \tag{8}$$

where $\hat{z} = \theta_2 \rho_{\mathbf{u}^\omega}(\mathbf{v}^k - \mathbf{d}^k)/(|\nabla I_1^\omega|^2 + \epsilon)$ and $\mathrm{abs}(\cdot)$ denotes absolute value of a scalar input. Note that for simplicity we omit subscripts i, j on each variable in the point-wise solution (8) and that ϵ is a small positive value to avoid division by zeros. In Appendix C, we propose a novel primal-dual method to derive (8) in detail.

2) ***Over-relaxation***: One approach to accelerate the convergence of ADMM is to account for past values when computing subsequent iterates. This technique is called *relaxation* and amounts to replacing the closed-form solution of $\mathbf{u}$ with $\widehat{\mathbf{u}} = (\widehat{u}_1, \widehat{u}_2)$ which is a convex combination of $\mathbf{u}^{k+1}$ and $\mathbf{v}^k$

$$\widehat{\mathbf{u}}^{k+1} = \alpha \mathbf{u}^{k+1} + (1 - \alpha)\mathbf{v}^k. \tag{9}$$

The parameter $\alpha \in (0, 2)$ is called the *relaxation parameter*. Note that letting $\alpha = 1$ recovers the plain ADMM iterations without acceleration. Empirical studies show that over-relaxation, i.e., letting $\alpha > 1$, is often advantageous and the guideline $\alpha \in [1.5, 1.8]$ has been proposed [46]. In [47], the authors have also shown it is beneficial to use *over-relaxation*. In experiments, we will examine the impact of using different $\alpha's$ and select the one which accelerates convergence speed most.

3) $\mathbf{v}$−***subproblem***: This subproblem has the form of $\mathbf{v}^{k+1} \leftarrow \min_{\mathbf{v}} \mathcal{L}_{\mathcal{A}}(\widehat{\mathbf{u}}^{k+1}, \mathbf{v}, \mathbf{w}^k; \mathbf{b}^k, \mathbf{d}^k)$, which is a linear L_2 problem that can be solved by optimizing the following problem

$$\begin{aligned} v_p^{k+1} = \operatorname*{argmin}_{v_p} &\frac{\theta_1}{2} \sum_{i,j} |(\mathbf{w}_1^k - \nabla^n v_1 - \mathbf{b}_1^k)_{i,j}|^2 \\ &+ \frac{\theta_1}{2} \sum_{i,j} |(\mathbf{w}_2^k - \nabla^n v_2 - \mathbf{b}_2^k)_{i,j}|^2 \\ &+ \frac{\theta_2}{2} \|v_1 - \widehat{u}_1^{k+1} - d_1^k\|_2^2 + \frac{\theta_2}{2} \|v_2 - \widehat{u}_2^{k+1} - d_2^k\|_2^2, \end{aligned}$$

where $p \in \{1, 2\}$. The minimization process of this subproblem entails the following equation with respect to v_p

$$v_p + \tfrac{\theta_1}{\theta_2}(-1)^n \mathrm{div}^n(\nabla^n v_p) = \tfrac{\theta_1}{\theta_2}(-1)^n \mathrm{div}^n(\mathbf{w}_p^k - \mathbf{b}_p^k) + \widehat{u}_p^{k+1} + d_p^k,$$

which is a $2nth$-order PDE and has a closed-form solution that can be obtained using the *discrete cosine transform* (DCT) under the assumption of the *Neuman boundary conditions* (normal derivative equal to zero on the boundaries). We note that in many recent image restoration works [45,48–51] the *discrete Fourier transform* (DFT) was frequently used to solve similar PDEs under the *periodic boundary conditions*. Imposing periodicity however introduces discontinuities (therefore artefacts) to the boundaries[4]. Therefore, we propose to use DCT instead of DFT. We note that DCT has been used in various imaging problems. For example, NG et al. [52] used DCT to diagonalize block Toeplitz-plus-Hankel matrices derived from a quadratic image deblurring problem. Strang et al. [53] studied up to eight variants of DCTs (ranging from DCT-1 to DCT-8) and applied DCT-2 to an image compression problem. Setzer et al. [42] used DCT to tackle a mixed first- and second-order

[3] We use $|\cdot|$ and $|\cdot|^2$ to denote Euclidean norm and squared Euclidean norm for vectors defined at each pixel, and $\|\cdot\|_1$ and $\|\cdot\|_2^2$ to denote L_1 norm and squared L_2 norm for vectors defined in the image space.

[4] When DFT is used, the higher order the PDE is, the more severe boundary artefacts one will see. In fact, by *calculus of variations*, the minimization of an energy functional often induces a *Euler-Lagrange* with the Neumann boundary conditions.

Algorithm 1: Closed-Form, Over-Relaxed ADMM for Arbitrary Order Total Variation Registration (1)

Input paired images: f_0 and f_1
Input parameters: $(n, \alpha, \theta_1, \theta_2, \lambda, \epsilon_1, \epsilon_2, N_{warp}, N_{iter}, scale)$ ▷ choose parameters
for all $s \in scale$ **do** ▷ $scale = \{..., 8, 4, 2, 1\}$, from coarse to fine
  $I_0 \leftarrow resize^-(f_0, s)$ ▷ downsample with factor s
  $I_1 \leftarrow resize^-(f_1, s)$ **if** $s = scale\{0\}$ **then** ▷ at first (coarsest) scale
    $\mathbf{u}^s = \mathbf{0}$ ▷ initialize displacement field
  else
    $\mathbf{u}^s \leftarrow 2 \times resize^+(\mathbf{u}^s, 2)$ ▷ upsampling with factor 2
  end
  while $\omega < N_{warp}$ *or (13) does not hold* **do** ▷ # Taylor expansions
    if $\omega = 0$ **then**
      $\mathbf{u}^w \leftarrow \mathbf{u}^s$ ▷ initialize $\mathbf{u}^w$ with displacement field at scale s
    end
    $I_1^\omega \leftarrow warping(I_1, \mathbf{u}^\omega)$ ▷ I_1^ω denotes $I_1(\mathbf{x} + \mathbf{u}^\omega)$
    set $\mathbf{w}^0 = \mathbf{b}^0 = \mathbf{0}$, $\mathbf{v}^0 = \mathbf{d}^0 = \mathbf{0}$ ▷ initialize variables and Lagrangian multipliers
    while $k < N_{iter}$ *or (14) does not hold* **do** ▷ # iterations
      update $\mathbf{u}^{k+1}$ using (8) with I_1^ω and $\rho_{\mathbf{u}^\omega}$ ▷ closed-form, point-wise solution
      update $\widehat{\mathbf{u}}^{k+1}$ using (9) ▷ accelerated over-relaxation step
      update $\mathbf{v}^{k+1}$ using (10) ▷ closed-form, point-wise solution
      update $\mathbf{w}^{k+1}$ using (12) ▷ closed-form, point-wise solution
      update $\mathbf{b}^{k+1} = \mathbf{b}^k + \nabla^n \mathbf{v}^{k+1} - \mathbf{w}^{k+1}$ ▷ Lagrangian multipliers
      update $\mathbf{d}^{k+1} = \mathbf{d}^k + \widehat{\mathbf{u}}^{k+1} - \mathbf{v}^{k+1}$ ▷ Lagrangian multipliers
    end
    $\mathbf{u}^w \leftarrow \mathbf{u}^{k+1}$ ▷ return solution of (2b)
  end
  $\mathbf{u}^s \leftarrow \mathbf{u}^w$ ▷ update displacement field at scale s
end
return $\mathbf{u}^* \leftarrow \mathbf{u}^s$ ▷ return solution of (1)

image denoising problem, which is similar to TGV [39] and infimal-convolution [41]. However, it has not been used to solve our arbitrary order PDEs.

With DCT, we have the closed-form solution to the linear PDE above

$$v_p^{k+1} = \mathcal{D}^{-1}\left(\frac{\mathcal{D}\left(\widehat{u}_p^{k+1} + d_p^k + \frac{\theta_1}{\theta_2}(-1)^n \mathrm{div}^n(\mathbf{w}_p^k - \mathbf{b}_p^k)\right)}{1 + \frac{\theta_1}{\theta_2}(-1)^n \mathcal{D}(\mathrm{div}^n \nabla^n)}\right), \tag{10}$$

where $\mathcal{D}$ and $\mathcal{D}^{-1}$ respectively denote the DCT and inverse DCT; ∇^n is the nth-order differential operators which are discretized via the forward finite difference; $(-1)^n \mathrm{div}^n$ is the nth-order adjoint operators which are discretized via the backward finite difference; $\mathrm{div}^n(\nabla^n)$ is the $2nth$-order differential operator; "$-$" stands for the point-wise division of matrices. The entries of the general coefficient matrix are

$$\mathcal{D}(\mathrm{div}^n \nabla^n) = \left(2\cos\left(\frac{\pi q}{W}\right) + 2\cos\left(\frac{\pi r}{H}\right) - 4\right)^n, \tag{11}$$

where H and W stand for image height and width, and $r \in [0, H)$ and $q \in [0, W)$ are frequency indices. Such a coefficient matrix essentially consists of the eigenvalues of $\mathrm{div}^n(\nabla^n)$ whose eigenvectors are DCT basis functions. In Appendix 1, we prove how to derive a solution like (10) and (11) from a general, linear PDE.

4) $\mathbf{w}$–***subproblem***: This subproblem has the form of $\mathbf{w}^{k+1} \leftarrow \min_{\mathbf{w}} \mathcal{L}_{\mathcal{A}}(\mathbf{u}^{k+1}, \mathbf{v}^{k+1}, \mathbf{w}; \mathbf{b}^k, \mathbf{d}^k)$, which is a linear L_1 problem that can be solved by optimizing the following problem

$$\begin{aligned}\mathbf{w}_p^{k+1} = \underset{\mathbf{w}_p}{\mathrm{argmin}}\, & \lambda \sum_{i,j} \sqrt{|(\mathbf{w}_1)_{i,j}|^2 + |(\mathbf{w}_2)_{i,j}|^2} \\ & + \frac{\theta_1}{2} \sum_{i,j} |(\mathbf{w}_1 - \nabla^n v_1^{k+1} - \mathbf{b}_1^k)_{i,j}|^2 \\ & + \frac{\theta_1}{2} \sum_{i,j} |(\mathbf{w}_2 - \nabla^n v_2^{k+1} - \mathbf{b}_2^k)_{i,j}|^2,\end{aligned}$$

where $p \in \{1, 2\}$. The solution of the problem is the following analytical, point-wise generalized soft thresholding equation

$$\mathbf{w}_p^{k+1} = \max\left(\sqrt{|\mathbf{y}_1|^2 + |\mathbf{y}_2|^2} - \frac{\lambda}{\theta_1}, 0\right) \frac{\mathbf{y}_p}{\sqrt{|\mathbf{y}_1|^2 + |\mathbf{y}_2|^2}}, \tag{12}$$

with the convention that $0 = 0/0$ and $\mathbf{y}_p = \nabla^n v_p^{k+1} + \mathbf{b}_p^k$. Here, for simplicity we omit subscripts i, j on each variable in the point-wise solution (12). Note that (12) can be also derived using the primal-dual method developed in Appendix C.

5) **b** ***and*** **d** ***updates***: At each iteration, one also needs to update the augmented Lagrangian multipliers $\mathbf{b}^{k+1}$ and $\mathbf{d}^{k+1}$, which are shown in Algorithm 1.

3.2. Stopping criteria

In Algorithm 1, there are two stop criteria that are defined to automatically break the loops. The first one is given as

$$\frac{||I_1(\mathbf{x} + \mathbf{u}^{k+1}) - I_0(\mathbf{x})||_1 - ||I_1\left(\mathbf{x} + \mathbf{u}^k\right) - I_0(\mathbf{x})||_1}{||I_1\left(\mathbf{x} + \mathbf{u}^k\right) - I_0(\mathbf{x})||_1} < \epsilon_1, \tag{13}$$

which says that the relative change between the data term (without linearization) at two consecutive iterations should be smaller than a positive threshold ϵ_1.

The second stopping criterion, which is the relative residual associated with the displacement field $\mathbf{u} = (u_1, u_2)$, is defined as

$$\max\left(\frac{||u_1^{k+1} - u_1^k||_1}{||u_1^{k+1}||_1}, \frac{||u_2^{k+1} - u_2^k||_1}{||u_2^{k+1}||_1}\right) < \epsilon_2. \tag{14}$$

In summary, to handle the original non-linear, non-differentiable and non-convex minimization problem (1), we break it down into a warping problem (2a) and a linearized, convex minimization problem (2b). We then develop an accelerated, ADMM-based iterative algorithm to decompose (2b) into an over-relaxation step and three simple subproblems. Each subproblem has a closed-form, point-wise solution and can be efficiently handled using existing numerical methods (i.e. DCT and soft thresholding). As such, the algorithm is very accurate and efficient, which can be confirmed in Section 4.3.

Since the Taylor expansion is used to linearize the intensity difference, we are only able to recover small displacements through minimizing (2b). As such, Algorithm 1 needs an extra *warping* operation (via $\mathbf{u}^{\omega}$). With *warping*, we can convert a relatively large displacement into N_{warp} small ones, each of which can be solved iteratively and optimally. In the algorithm, $resize^{-}$ indicates we downsample the image at integer locations, while $resize^{+}$ denotes the image is upsampled via the bilinear interpolation, which is also used in the *warping* operation. Since minimizing a large displacement field is a non-convex problem, in our implementation we also embed Algorithm 1 into a multi-scale (pyramid) scheme to avoid convergence to local minima. When a N_{scale} pyramid is used, the total number of iterations in the algorithm becomes $N_{scale} \times N_{warp} \times N_{iter}$.

4. Experimental results

To evaluate the performance of different models and algorithms, we introduce five subsections: impact of derivative orders, robustness against outliers, algorithm efficiency, parameter selection, and comparison with the SOTA. First, we investigate and analyze how increased derivative order n will affect numerical results and model performance. Second, we compare their robustness and performance between L_1 (SAD) and L_2 (SSD) data terms. Third, we show the impact of the over-relaxation parameter α and compare our accelerated ADMM with a subgradient method and a popular primal-dual algorithm for non-smooth optimization. Next, we explain what the role each built-in parameter in the algorithm plays and then show how to tune reasonable values for these parameters in order to maximize both accuracy and speed. Finally, we introduce the datasets and quantitative matrices used for experimental comparison. Implementation details of other competing methods are then given. We quantitatively and qualitatively study all methods in the end.

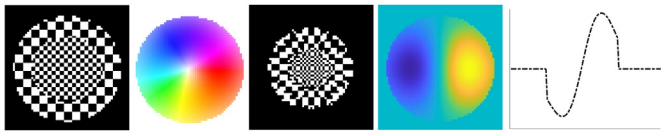

Fig. 2. A piecewise quadratic displacement field (2nd image) is simulated to warp source (1st image) to target (3rd image). 4th image is the displacement field in the x direction, and 5th image the middle cross section profile of the 4th image. Source and target are used to compute results in Fig. 3.

4.1. Impact of derivative orders

To understand the numerical behaviour of using varying derivative orders in our proposed model, we analyze a L_2 smoothing problem: $\min_u \int_\Omega |u(\mathbf{x}) - f(\mathbf{x})|^2 d\mathbf{x} + \lambda \int_\Omega |\nabla^n u(\mathbf{x})|^2 d\mathbf{x}$, which is a simplified version of $\mathbf{v}$−***subproblem*** in Section 3.1. As such, our numerical analysis established below may help understand the L_1 formulation (2b) as $\mathbf{v}$−***subproblem*** is one of its subproblems. Here, $f(\mathbf{x})$ and $u(\mathbf{x}) : \Omega \to \mathbb{R}^2$ and λ is a smooth parameter. Note that this problem can be explicitly tackled by DCT, and the optimal solution is $u^* = \mathcal{D}^{-1}(\mathcal{M} \cdot \mathcal{D}(f))$, where $\mathcal{M} = [1 + \lambda(-1)^n \mathcal{D}(\mathrm{div}^n \nabla^n)]^{-1}$ and $\cdot$ denotes the point-wise multiplication. In u^*, $\mathcal{M}$ is a soft mask whose values are determined by λ and n.

In Fig. 1 left, we plot different $\mathcal{M}'s$ by varying λ and n. In Fig. 1 right, we obtain the same number of smoothed results by using these $\mathcal{M}'s$. From Fig. 1, we can see when n goes higher, λ must increase exponentially to reduce the soft mask size such that the corresponding deformation can be smoothed. In other words, large n makes the problem less sensitive to λ. Numerically speaking, the use of big λ results in systems for inversion that are ill-conditioned, thus making the overall computation difficult. To prove this claim, in Table 2 we show the condition numbers of $\mathcal{M}'s$ computed using different $\lambda's$. One can see larger $\lambda's$ and $n's$ produce bigger condition numbers. For numerical easability, we study only $n = \{1, 2, 3, 4\}$, resulting in the total variations of first order (1stO-TV), second order (2ndO-TV), third order (3rdO-TV), and fourth order (4thO-TV) for comparison.

Next we perform the experiments on two toy examples to illustrate the effectiveness of using higher-order derivatives. As can

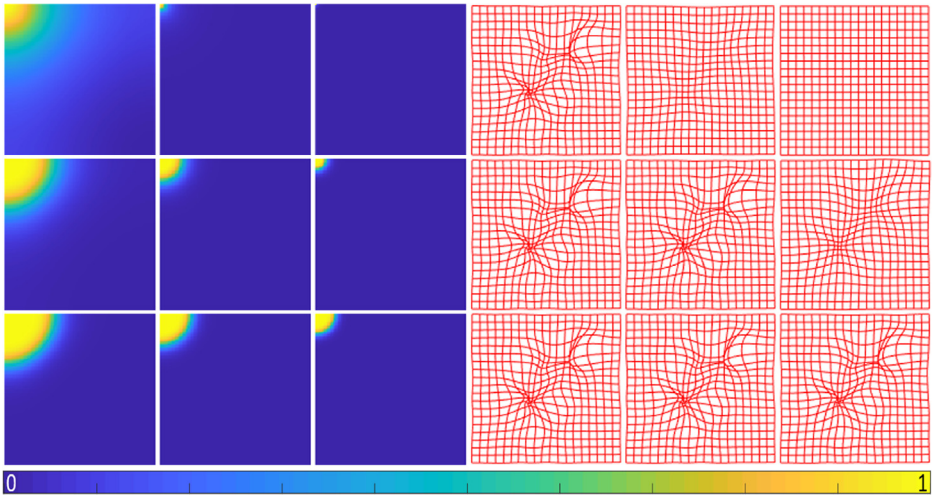

Fig. 1. Impact of the derivative order n and of the smooth parameter λ. Left panel: soft masks produced by $\mathcal{M}'s$ using different combinations of λ and n. In this panel, from top to bottom n is respectively 1, 3, and 5 and from left to right λ respectively 1, 10^2, and 10^4. Right panel: corresponding smoothed deformations produced by u^* with $\mathcal{M}'s$ on the left. It is clear that these deformations become increasingly smooth as the soft masks get smaller. This is because after an deformation is transformed into the DCT space, the corner locations contain mostly low frequency information. Applying a soft mask is equivalent to performing a low frequency filter on the deformation.

Table 2
Condition numbers of $\mathcal{M}$ (after diagonalization) with different combinations of n and λ. Numerically, inverting a matrix with a large condition number gives a less accurate and stable result.

	$\lambda = 1$	$\lambda = 10^2$	$\lambda = 10^4$	$\lambda = 10^5$
$n = 1$	8.997	8.007×10^2	7.998×10^4	7.997×10^6
$n = 3$	5.126×10^2	5.116×10^4	5.115×10^6	5.115×10^8
$n = 5$	3.273×10^4	3.272×10^6	3.272×10^8	3.272×10^{10}
$n = 7$	2.093×10^6	2.093×10^8	2.093×10^{10}	2.093×10^{12}

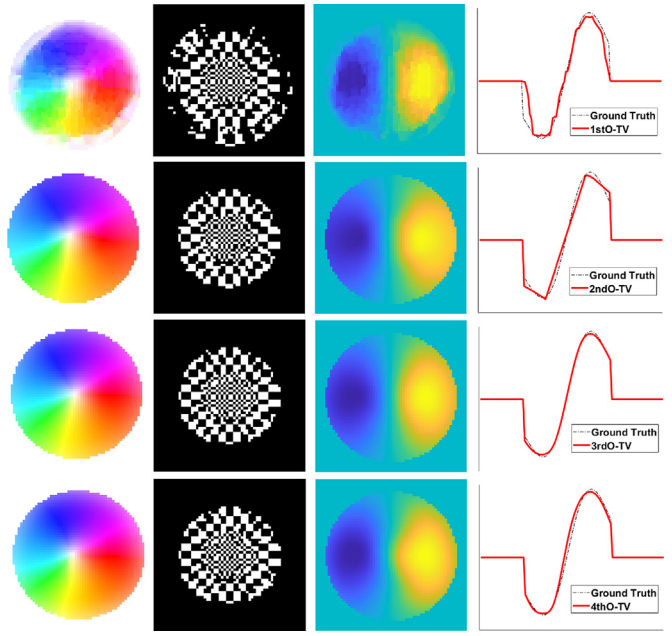

Fig. 3. 1st-4th rows: 1st-order TV, 2nd-order TV, 3rd-order TV, and 4th-order TV, respectively. Clearly the best results are acquired by using 3rd and 4th-order TVs, since they are capable of modeling piecewise quadratic signals.

be seen from Fig. 3, setting $n = 1$, $n = 2$, and $n = 3$ make the displacement field respectively exhibit piecewise constant, piecewise linear, and piecewise quadratic behaviour. It is also obvious from Fig. 3 and 4 that higher-order models can induce stronger smoothness and are more robust to larger deformation. Two simple examples here (Fig. 3 and 4) suggest higher-order TVs can be more effective than their lower order counterparts when the underlying displacement field possesses the behavior of high-order derivatives.

4.2. Robustness against outliers

We now examine whether the application of L_1 data term (SAD) in (1) leads to more robust performance against outlier noise in images. For this, we compare L_1 data term with its L_2 counterpart (SSD), both of which are implemented with the 1st-order TV regularization for noisy images. The results are given in Fig. 5, from which it is evident that L_1 is more robust than L_2 against noise.

Technically, the use of L_2 norm between prediction and observation is known as least squares regression, whilst the use of L_1 norm as least absolute deviations regression. The latter is a robust fit method, meaning that the least absolute deviations regression is insensitive to outlier data points [33]. More specifically, L_2, which squares the error/residual, increases the importance of larger errors. This results in a fit that is especially focused on trying to minimize these large errors - often due to outliers in data. In other words, L_2 tends to overfit to outliers in a dataset. In contrast, by using the absolute error (L_1) we treat negative and positive errors equally, but do not exaggerate the importance of large

Fig. 4. Top row: 1st image (target) is first locally deformed to 2nd image (source 1), which is then further affine-tranformed to 3rd image (source 2). Bottom row shows results using 1st-order TV between source 1 and target, results using 1st-order TV between source 2 and target, and results using 3rd-order TV between source 2 and target, respectively. 1st-order methods are more dependant on initial alignments between images so are more suitable when an affine linear pre-registration is available, which is not the case for higher-order methods.

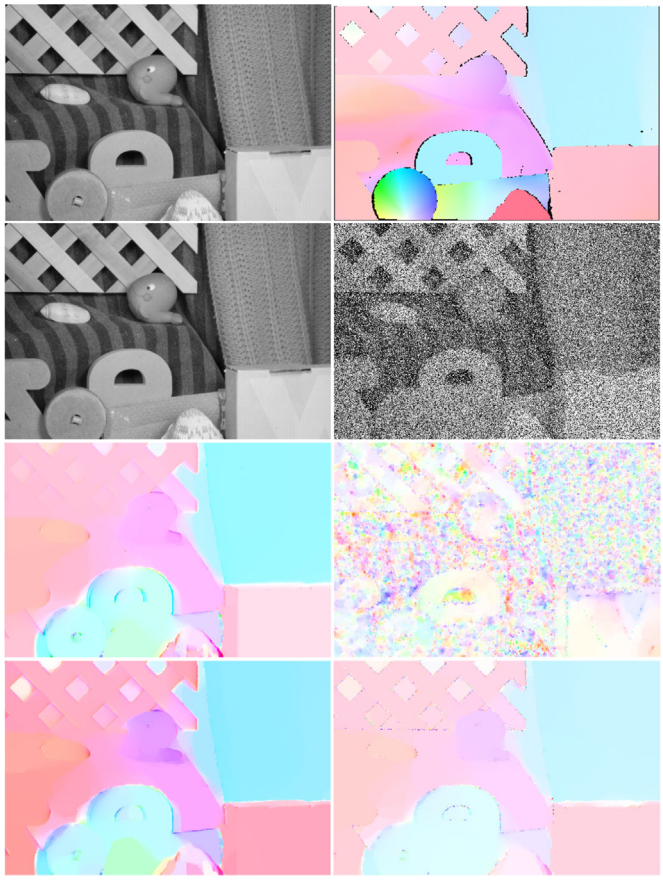

Fig. 5. Performance comparison between L_2 and L_1 data terms. 1st row: source image and ground truth displacement field between source and target images. 2nd row: target image and its corrupted version by 50% salt and pepper noise. 3rd row: L_2 results (left: displacement field between source and target images; right: displacement field between source and corrupted target images). 4th row: L_1 results (left: displacement field between source and target; right: displacement field between source and corrupted target).

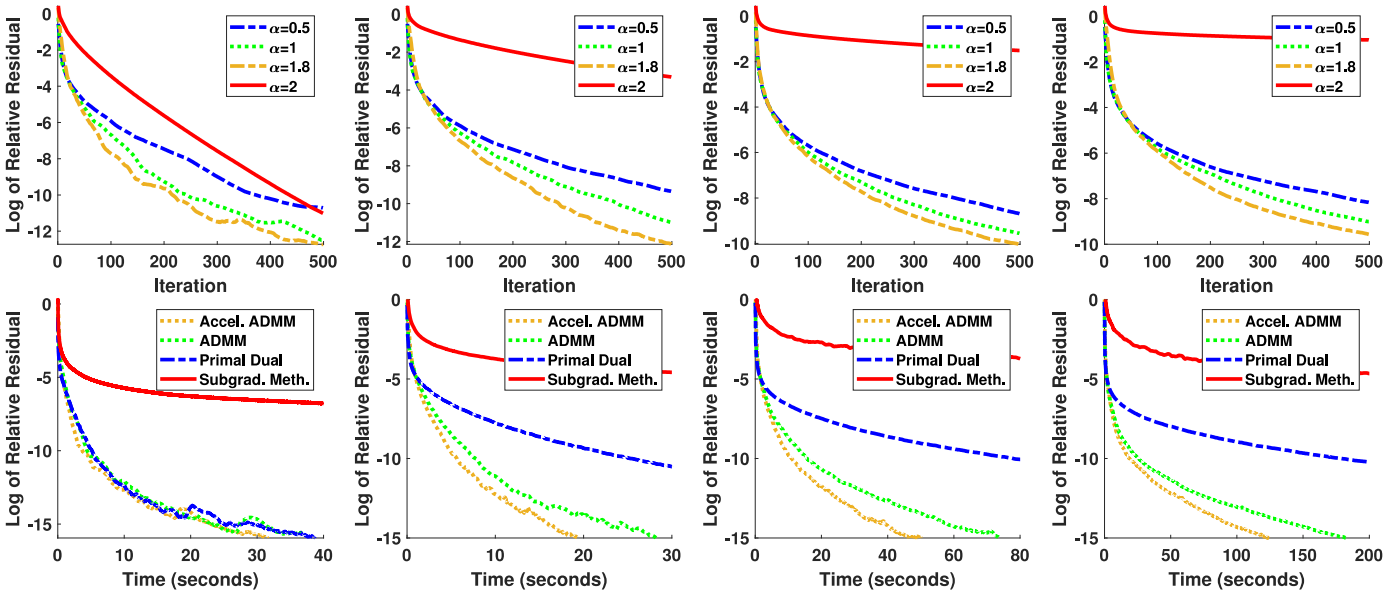

Fig. 6. Comparison of convergence rate (top) and computational time (bottom). Top row: impact of using different over-relaxation parameter α in (9). Bottom row: comparing computational efficiency between our ADMMs ($\alpha = 1.8$ and $\alpha = 1$), a state-of-the-art primal dual implemented in Appendix B, and the subgradient method in Appendix D. From left to right show 1stO-TV, 2ndO-TV, 3rdO-TV, and 4thO-TV results, respectively.

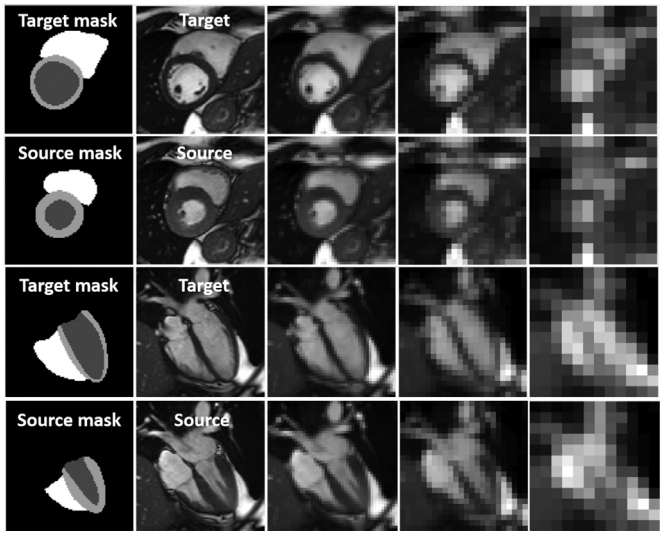

Fig. 7. Segmentation masks versus multi-scale images. 1st column: segmentation masks of the images in the 2nd column. 2nd-5th columns: images at different scales. From left to right the image is downsampled 1, 2, 4, and 8 times, respectively. In the masks, the while region represents the right ventricle cavity (RV); the light gray region denotes the left ventricle cavity (LV); and the dark gray region indicates the LV myocardium (LVM). 1st-2nd rows: short-axis images; 3rd-4th rows: image-axis images.

errors/residuals, which leads to a robust fit insensitive to outlier noise, as proven in Fig. 5.

4.3. Algorithm efficiency

In Fig. 6, we compare the computational speed between the accelerated ADMM ($\alpha = 1.8$), the standard ADMM ($\alpha = 1$), the primal dual, and the subgradient method on an image pair called 'Yosemite' from the Middlebury dataset. For all algorithms, we set their common parameters ($\lambda = 0.1, \epsilon_1 = \times, \epsilon_2 = 2^{-23}, N_{warp} = 1, N_{iter} = \times, scale = \{1\}$), where '$= \times$' means we do not use that stopping criterion to break the loop. For both ADMMs, we set ($\theta_1 = 1, \theta_2 = 0.1$). We also use the optimal parameters for both primal dual and subgradient methods (see Appendix B and Appendix D). Among all algorithms, the subgradient method is the slowest, which is not surprising because subgradient methods have a convergence rate of $O(1/\sqrt{k})$ [54]. In contrast, the primal dual has a rate of $O(1/k)$ [31]. It is also clear that the accelerated ADMM is faster than the standard ADMM. Our accelerated ADMM is the fastest algorithm especially for higher-order cases.

In Table 3, we further compare the runtime of accelerated ADMM and primal dual on three other image pairs from the dataset using different $\lambda's$. For this comparison, we used $\theta_1 = 1$ and $\theta_2 = 0.1$ for ADMM, $\sigma = \tau = \frac{1}{\lambda\sqrt{8^n}}$ for primal dual, and ($\epsilon_1 = \times, \epsilon_2 = 10^{-3}, N_{warp} = 5, N_{iter} = \times, scale = \{4, 2, 1\}$) for both algorithms. From the table, we observe that for $n = 1$ ADMM has a comparable speed as primal dual but for $n = 2$ ADMM gets much faster. In essence, primal dual is a proximal gradient method and its convergence speed is determined by the step sizes σ and τ. For higher-order models, the step sizes should be decreased exponentially to guarantee convergence, drastically slowing down its speed. This comparative experiment is done in Matlab 2020 using a Dell laptop with a 2.60GHz Intel i7-9850H CPU (16GB RAM). In addition, we have also implemented our ADMM using a V100 GPU in Pytorch with the same model parameters, which boosts the speed by a factor of 10.57 on average as compared to that in Table 3.

4.4. Parameter selection

Now we explain how to select the built-in parameters ($n, \alpha, \theta_1, \theta_2, \lambda, \epsilon_1, \epsilon_2, N_{warp}, N_{iter}, scale$) in Algorithm 1 for our following final comparative experiments:

- $n = \{1, 2, 3, 4\}$ will be studied separately and α as per last section is set to 1.8 to give a faster speed.
- θ_1 and θ_2 are convergence parameters, different combinations of which affect the convergence rate but may not change the minimiser of (2b) due to its convexity. In Table 4, we prove this claim using the short-axis image pair in Fig. 7 from the UK Biobank (UKBB) [55]. We set ($\alpha = 1.8, \epsilon_1 = \times, \epsilon_2 = 2^{-23}, N_{warp} = 5, N_{iter} = \times, scale = \{4, 2, 1\}$) with an optimal λ for each model. As is evident, within each model the mean Dice and the mean square error are almost identical for differ-

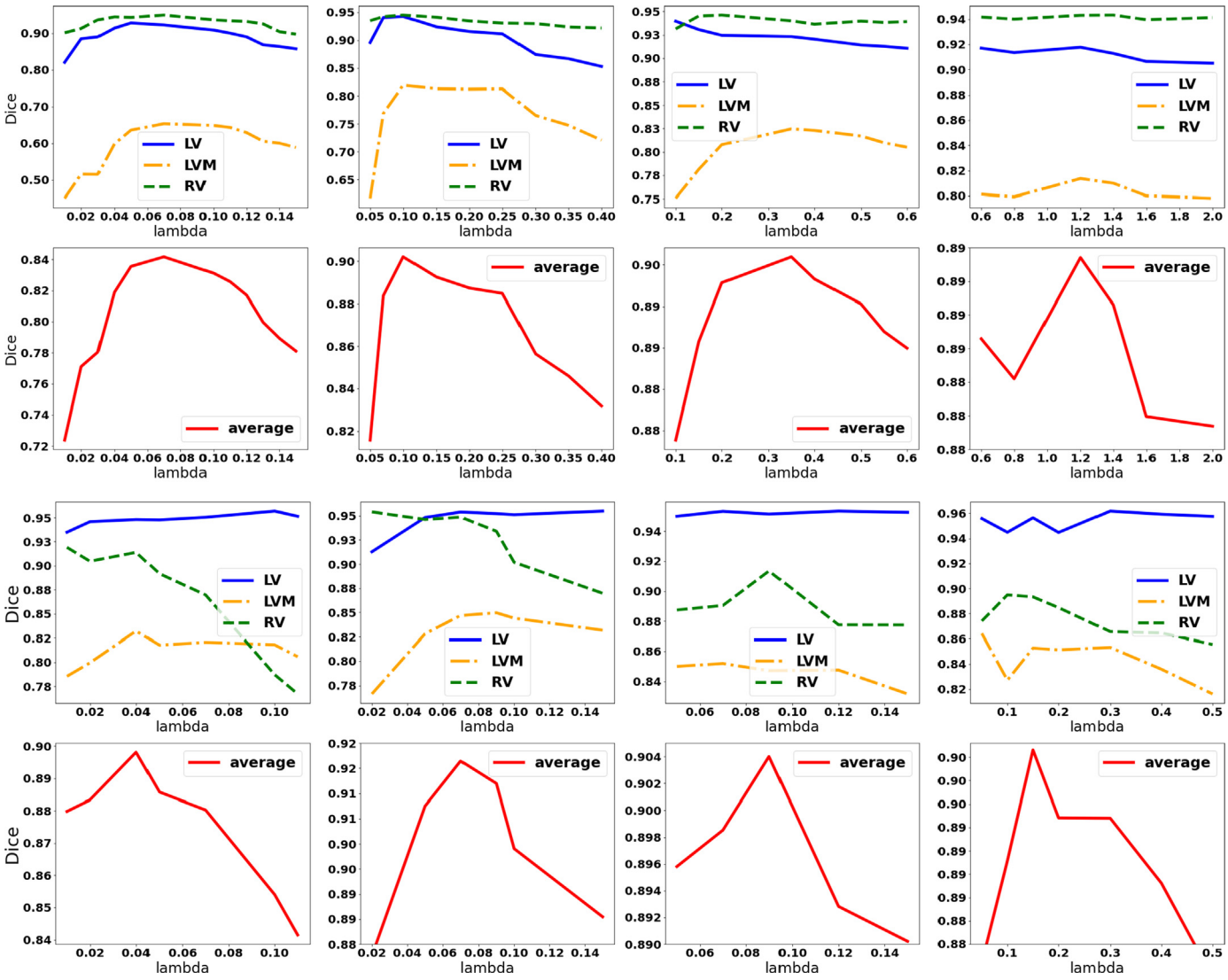

Fig. 8. Selection of the optimal λ for different registration models. 1st and 3rd rows: Dice values versus λ values for LV, LVM and RVM. 2nd and 4th rows: average Dice values of LV, LVM and RVM versus λ values. From left to right show the results of 1stO-TV, 2ndO-TV, 3rdO-TV and 4thO-TV, respectively. λ is selected when the average Dice value is maximized.

Table 3
Time trial results. Iteration counts are reported for fast ADMM and primal dual with total runtime (secs) in parentheses.

			1stO-TV		2ndO-TV	
Image	Size	λ	Accel. ADMM	Primal Dual	Accel. ADMM	Primal Dual
Army	584×388	0.05	84 (32s)	385 (45s)	104 (55s)	1010 (196s)
Army	584×388	0.1	94 (36s)	513 (58s)	172 (82s)	1371 (246s)
Army	584×388	0.15	118 (45s)	613 (70s)	246 (103s)	1611 (268s)
Teddy	420×360	0.1	117 (50s)	411 (51s)	138 (76s)	1055 (166s)
Teddy	420×360	0.15	136 (47s)	547 (59s)	164 (82s)	1260 (209s)
Teddy	420×360	0.2	120 (47s)	617 (68s)	188 (89s)	1402 (226s)
Minicooper	640×480	0.1	81 (54s)	287 (47s)	128 (101s)	651 (194s)
Minicooper	640×480	0.15	89 (53s)	306 (54s)	163 (120s)	715 (212s)
Minicooper	640×480	0.2	98 (58s)	322 (57s)	181 (130s)	755 (208s)

ent combinations of θ_1 and θ_2. However, the number of iterations varies drastically. By selecting suitable θ_1 and θ_2, we may achieve fast convergence speed while still benefiting from same numerical accuracy.

- λ is a smooth parameter, controlling the smoothness and accuracy of final deformations. To find the optimal λ for each model, we perform *parameter sweeps* on a set of λ′*s*. Here we use both image pairs in Fig. 7 as an example. Specifically, first: we define a set of λ′*s* and for each λ together with $(\alpha = 1.8, \epsilon_1 = \times, \epsilon_2 = 2^{-23}, N_{warp} = 5, N_{iter} = \times, scale = \{4, 2, 1\})$ and the optimal θ'*s* identified in Table 4, we use Algorithm 1 to compute the deformation between the source and target im-

Table 4
Numerical impact of using different combinations of θ_1 and θ_2. The three numbers (i.e. $\cdot/\cdot/\cdot$) represent the average Dice value of LV, LVM and RV, the mean square error (MSE) between $I_1(\mathbf{x}+\mathbf{u}^*)$ and $I_0(\mathbf{x})$, as well as the total number of iterations using the parameters ($\alpha = 1.8, \epsilon_1 = \times, \epsilon_2 = 2^{-23}, N_{warp} = 5, N_{iter} = \times, scale = \{4, 2, 1\}$).

	1stO-TV with $\lambda^* = 0.07$				2ndO-TV with $\lambda^* = 0.1$			
	$\theta_1 = 0.1$	$\theta_1 = 1$	$\theta_1 = 10$	$\theta_1 = 100$	$\theta_1 = 0.1$	$\theta_1 = 1$	$\theta_1 = 10$	$\theta_1 = 100$
$\theta_2 = 10^{-2}$	0.841/2.12e-3/63k	0.842/2.12e-3/56k	0.842/2.11e-3/110k	0.841/2.17e-3/579k	0.902/2.75e-3/107k	0.902/2.75e-3/101k	0.902/2.75e-3/100k	0.903/2.74e-3/167k
$\theta_2 = 10^{-1}$	0.841/2.09e-3/36k	0.842/2.11e-3/**20k**	0.841/2.12e-3/100k	0.842/2.10e-3/562k	0.902/2.75e-3/44k	0.902/2.75e-3/**24k**	0.902/2.75e-3/29k	0.903/2.74e-3/158k
$\theta_2 = 10^{0}$	0.841/2.10e-3/37k	0.841/2.08e-3/23k	0.841/2.13e-3/103k	0.842/2.12e-3/560k	0.903/2.75e-3/48k	0.902/2.75e-3/30k	0.902/2.75e-3/41k	0.903/2.74e-3/160k
$\theta_2 = 10^{1}$	0.842/2.08e-3/118k	0.842/2.06e-3/118k	0.841/2.10e-3/157k	0.843/2.11e-3/579k	0.903/2.75e-3/205k	0.903/2.75e-3/206k	0.903/2.75e-3/212k	0.904/2.75e-3/284k
$\theta_2 = 10^{2}$	0.842/2.15e-3/770k	0.842/2.13e-3/773k	0.839/2.18e-3/792k	0.842/2.10e-3/1.0m	0.902/2.73e-3/1.2m	0.902/2.74e-3/1.2m	0.902/2.74e-3/1.2m	0.902/2.74e-3/1.2m
	3rdO-TV with $\lambda^* = 0.35$				4thO-TV with $\lambda^* = 1.2$			
	$\theta_1 = 0.1$	$\theta_1 = 1$	$\theta_1 = 10$	$\theta_1 = 100$	$\theta_1 = 0.1$	$\theta_1 = 1$	$\theta_1 = 10$	$\theta_1 = 100$
$\theta_2 = 10^{-2}$	0.897/3.82e-3/353k	0.896/3.81e-3/203k	0.896/3.81e-3/210k	0.896/3.81e-3/210k	0.894/4.04e-3/1.7m	0.892/4.06e-3/404k	0.893/4.12e-3/303k	0.893/4.11e-3/291k
$\theta_2 = 10^{-1}$	0.896/3.81e-3/229k	0.896/3.81e-3/72k	0.896/3.81e-3/55k	0.896/3.81e-3/54k	0.894/4.03e-3/1.6m	0.892/4.04e-3/232k	0.892/4.04e-3/94k	0.890/4.04e-3/75k
$\theta_2 = 10^{0}$	0.896/3.82e-3/226k	0.896/3.81e-3/45k	0.897/3.81e-3/**33k**	0.896/3.81e-3/34k	0.891/4.03e-3/1.5m	0.890/4.05e-3/228k	0.891/4.05e-3/52k	0.891/4.05e-3/**42k**
$\theta_2 = 10^{1}$	0.895/3.82e-3/269k	0.897/3.82e-3/196k	0.897/3.82e-3/200k	0.897/3.82e-3/202k	0.892/4.02e-3/1.4m	0.894/4.04e-3/310k	0.892/4.05e-3/227k	0.892/4.05e-3/233k
$\theta_2 = 10^{2}$	0.896/3.83e-3/1.2m	0.896/3.83e-3/1.2m	0.896/3.83e-3/1.2m	0.896/3.83e-3/1.2m	0.893/4.07e-3/1.7m	0.895/4.05e-3/1.3m	0.894/4.05e-3/1.3m	0.894/4.05e-3/1.3m

ages in Fig. 7 for each model. Second: we warp the source segmentation mask using the deformation. Third: for each region (i.e. LV, LVM and RV), we compute a Dice value between the warped source mask and the target mask. The optimal λ for each model on both image pairs is identified by the peak point, as shown in the 2nd and 4th rows of Fig. 8. It is clear that as the regularization order goes higher, we need a larger λ to reach comparable accuracy, which is largely in line with the observation in Fig. 1.

- ϵ_1 and ϵ_2 are two relative residuals, given in (13) and (14), which determine the actual number of warpings and iterations in Algorithm 1, respectively. In the final comparative experiments next, we assign 2% to ϵ_1 and 10^{-5} to ϵ_2. We also set $N_{warp} = 5$ and $N_{iter} = 500$ to guarantee the algorithm will stop at some point.
- *scale* is a multi-scale parameter and introduced to ease the minimization difficulty of original non-convex problem (1), and it relates to the size of object of interest in the image. For our datasets, downsampling the images to 1/4 of original size is favourable as otherwise the heart regions vanish, as shown in the last column of Fig. 7. For this reason, we select $scale = \{4, 2, 1\}$ for the comparative experiments next.

4.5. Comparison with the state-of-the-art

Knowing the behavior of built-in parameters, we then compare their performance with SOTA methods on three MR image datasets, both quantitatively and qualitatively. The first dataset used is UKBB-1 [55], where we randomly select 220 healthy subjects and for each subject we have a corresponding 4D (3D+t) volume with $t = 50$ representing a complete cardiac cycle. For each volume, the in-plane resolution of each image slice is $1.8 \times 1.8mm^2$ per pixel and the through-plane resolution is $10mm$. Due to large gaps between slices, it is physiologically implausible to do 3D registration. The second dataset comes from ACDC [56] which is composed of 100 patients with four types of pathology. In this dataset, the image slice thickness changes from $5mm$ to $10mm$ and the spatial resolution varies from $1.34mm$ to $1.68mm$ per pixel. We resample in-plane resolutions of all images to $1.8 \times 1.8mm^2$ before experiments. In addition to the first two datasets which contain only short-axis cardiac images, we use a third dataset which is from UKBB-2. This dataset contains long-axis images from 220 subjects and the resolution of each image is $1.8 \times 1.8mm^2$ per pixel. For all three datasets, we perform experiments on images at the end-diastolic (ED) and end-systolic (ES) frames and crop the image to 128×128 pixels. For short-axis datasets, we only use middle ventricular slices. As an example, we show a pair of short-axis images (top) and long-axis images (bottom) in Fig. 7.

To evaluate the accuracy of resulting deformations, we use Dice and Hausdorff distance (HD) as the quantitative matrices. For Dice, the segmentation masks of LV, LVM and RV at ED (target) and ES (source) frames are used, which are available in all datasets. Dice varies from 0-1, with high values corresponding to a better match. HD is computed on an open-ended scale, with smaller values implying a better result. To compute these matrices for paired images, we estimate the deformation by registering the ES image to its respective ED image. With the deformation, we deform the segmentation mask at ES to ED and measure its overlap and contour distance to the ED mask using Dice and HD, respectively.

We compare our models with the following 5 SOTA approaches. Note that for learning-based methods, we split UKBB-1 into 100/20/100, ACDC into 40/20/40, and UKBB-2 into 100/20/100 for training. validation, and test. The regularization weights in the three methods are selected to maximize the performance on validation sets. All methods are compared on test sets only.

Table 5
Comparison of image registration performance using different methods. 'All' means that Dice or HD is computed by averaging that of LV, LVM and RV of all subjects in the test set. Here mean values are reported for UKBB-1, ACDC, and UKBB-2, separately. Note that apart from FFD which is tested on a CPU, we use a A100 GPU (40G RAM) to run the other methods.

Methods	UKBB-1									
	Dice				HD				Runtime	
	LV	LVM	RV	ALL	LV	LVM	RV	ALL	Training (hour)	Testing (ms)
FFD [10]	0.947	0.755	0.875	0.859	4.104	5.078	8.625	5.936	–	16630
VoxelMorph [2]	0.949	0.816	0.886	0.884	3.474	3.507	8.397	5.126	3.42	6.24
Siamese [32,57]	0.955	0.685	0.852	0.831	3.635	8.450	10.41	7.498	3.42	7.03
SYM-Net [58]	**0.958**	**0.840**	0.894	**0.897**	3.143	**3.139**	8.304	4.862	3.10	7.84
Bspline-Net [59]	0.953	0.825	0.893	0.890	3.333	3.374	8.192	4.966	3.23	8.39
1stO-TV	0.955	0.811	0.893	0.886	3.307	3.856	8.248	5.137	–	570.13
2ndO-TV	**0.958**	0.833	**0.895**	0.895	**3.040**	3.174	**7.680**	**4.631**	–	746.07
3rdO-TV	0.956	0.832	0.889	0.892	3.089	3.338	7.878	4.769	–	797.88
4thO-TV	0.954	0.829	0.891	0.891	3.238	3.863	7.819	4.973	–	1029.96
Methods	ACDC									
	Dice				HD				Runtime	
	LV	LVM	RV	ALL	LV	LVM	RV	ALL	Training (hour)	Testing (ms)
FFD [10]	0.939	0.799	0.850	0.863	**4.612**	5.619	**7.814**	6.015	–	31230
VoxelMorph [2]	0.933	0.801	0.854	0.863	4.878	5.269	8.992	6.380	1.32	5.65
Siamese [32,57]	0.907	0.692	0.846	0.815	7.248	8.592	9.126	8.322	1.34	5.97
SYM-Net [58]	0.916	0.797	0.848	0.854	5.105	5.110	8.155	6.124	1.32	6.67
Bspline-Net [59]	0.906	0.786	0.827	0.839	5.432	4.899	8.508	6.279	1.28	6.36
1stO-TV	0.922	0.796	**0.868**	0.862	5.495	6.247	8.263	6.668	–	497.62
2ndO-TV	**0.944**	0.817	0.865	**0.875**	4.750	5.055	8.328	6.044	–	435.56
3rdO-TV	0.939	0.817	0.857	0.871	5.236	5.384	8.031	6.217	–	828.02
4thO-TV	0.933	**0.821**	0.855	0.870	5.060	**4.718**	8.165	**5.981**	–	937.83
Methods	UKBB-2									
	Dice				HD				Runtime	
	LV	LVM	RV	ALL	LV	LVM	RV	ALL	Training (hour)	Testing (ms)
FFD [10]	0.875	0.757	0.724	0.785	7.609	3.602	9.216	6.809	–	5150
VoxelMorph [2]	0.884	0.717	0.760	0.787	7.999	9.106	12.322	9.809	1.30	3.83
Siamese [32,57]	0.897	0.744	0.726	0.789	7.487	7.664	12.851	9.334	1.33	4.09
SYM-Net [58]	**0.900**	0.769	0.745	0.808	7.119	8.120	12.908	9.307	1.22	3.67
Bspline-Net [59]	0.898	0.738	0.762	0.803	**6.323**	7.372	12.644	8.687	1.22	3.94
1stO-TV	0.883	0.763	0.743	0.796	7.018	3.872	9.128	6.672	–	435.20
2ndO-TV	0.877	**0.800**	**0.766**	**0.814**	6.845	**3.188**	**8.563**	**6.199**	–	655.78
3rdO-TV	0.869	0.782	0.760	0.804	7.709	4.223	8.865	6.932	–	823.81
4thO-TV	0.866	0.774	0.764	0.801	8.471	4.935	8.612	7.340	–	1070.49

- The first one is an optimization-based method using B-spline free form deformation (FFD) [10] which ranks top three among 14 non-rigid registration methods [60] and therefore is a strong baseline. For this method, we use the official MIRTK implementation[5]. In terms of parameters, we use SSD as similarity and Bending energy as regularization. Moreover, a three-level multi-scale approach (similar to ours) is used where the spacing of B-spline control points on the highest scale is set to $8mm$.
- The second method is a learning-based method called VoxelMorph[6] from [2]. For this method, we first pass a pair of ED and ES images through a 2D deterministic U-Net [61], the output prediction of which is a displacement field. The ES image is then warped using the displacement field in a bilinear spatial transformation layer [62]. The loss function is a combination of similarity (MSE between the warped ES image and the ED image) and smoothness (first-order diffusion regularization). We note that a recent work [63] has proved that using U-Net as backbone is still very competitive and can be more effective than transformer-based methods [64].
- The third method[7] we compare is from [32,57]. Its idea is different from VoxelMorph in two aspects: (1) the U-Net is replaced with a Siamese style multi-scale network; (2) the loss controlling smoothness is replaced with the Huber loss [65].
- The fourth method we compare is SYM-Net[8] from [58]. The backbone of this network is U-Net, which predicts a stationary velocity field. Scaling and Squaring is then applied to this velocity field to produce the final deformation. We use the MSE and diffusion regularization loss for SYM-Net.
- The fifth method we compare is Bspline-Net[9] from [59]. The backbone of this network is U-Net, which predicts stationary velocities of the control points. The deformation is obtained by first evaluating B-spline functions at the control points and then integrating via Scaling and Squaring. We use the MSE and diffusion regularization loss for Bspline-Net.

In Table 5, we show the quantitative results of using different methods on UKBB-1, ACDC, and UKBB-2. We observe that the best results on UKBB-1, highlighted in bold, come from either SYM-Net

[5] https://github.com/BioMedIA/MIRTK.
[6] https://github.com/voxelmorph/voxelmorph.
[7] https://github.com/qiuhuaqi/cardiac-motion.
[8] https://github.com/cwmok/Fast-Symmetric-Diffeomorphic-Image-Registration-with-Convolutional-Neural-Networks.
[9] https://github.com/qiuhuaqi/midir.

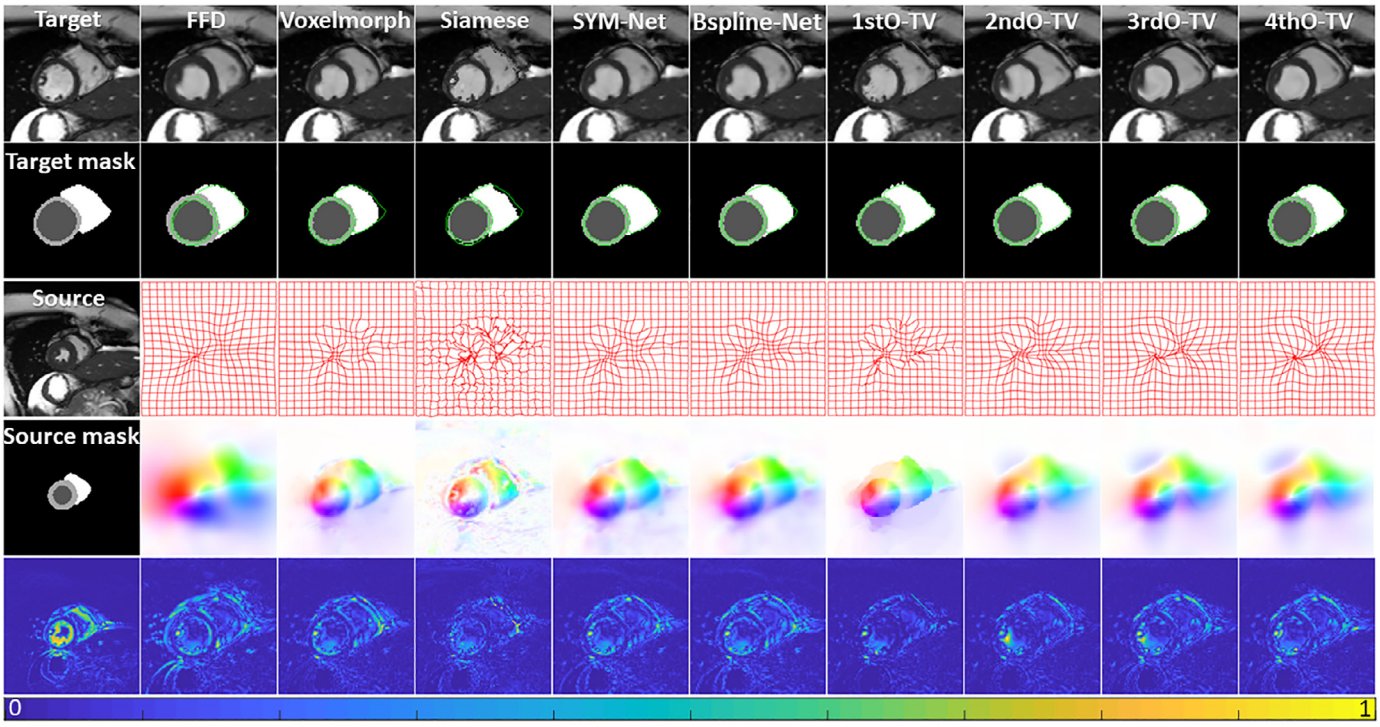

Fig. 9. Comparing visual results obtained by different registration methods on one representative example from UKBB-1. 1st column shows original images (on 1st and 3rd slots), segmentation masks (on 2nd and 4th slots), and absolute differences (on 5th slot) between corresponding source and target; 2nd-10th columns show FFD, VoxelMorph, Siamese, SYM-Net, Bspline-Net, 1stO-TV, 2ndO-TV, 3rdO-TV, and 4thO-TV results, respectively; 1st-5th rows (excluding 1st column) show warped sources, warped source masks (with contours from target mask superimposed), deformation grids, displacement fields, and absolute differences between corresponding warped sources and target, respectively.

or our proposed methods, among which 2ndO-TV performs the best and both 3rdO-TV and 4thO-TV are competitive. On ACDC, FFD and 4thO-TV are better in term of HD, while 1stO-TV, 2ndO-TV and 4thO-TV are better in terms of Dice. On UKBB-2, our 2ndO-TV consistently outperforms other methods on most anatomies in terms of both Dice and HD. Overall, our methods are superior to other compared methods in most cases, indicating the effectiveness of using model-based approaches for this unsupervised task. When considering only inference time in Table 5, deep-learning based methods outperform our proposed methods, but this does not account for the significant amount of training time required by deep learning-based methods. One advantage of our method is their overall efficiency (up to a second runtime), owing to the fact they are model-driven and free of training.

In Fig. 9 and 10, we compare visual results of different methods. FFD, built on a L_2 regularization, over-smooths the displacement field, leading to a under-estimated mask warping. Our methods (last 4 columns of each figure), which use a L_1 regularization, are able to preserve discontinuities. As such, the resultant displacement fields are more dense and clustered (indicated by red arrows in Fig. 10), and their warped masks are therefore closer to their respective ground truths. For displacement fields, 1stO-TV produces staircase artifacts and the smoothness induced does not appear strong enough to regularize the deformation grid. In contrast, higher-order models (last 3 columns of each figure) are able to reconstruct piecewise smooth deformations. We also observe that learning-based methods perform less accurately than model-based methods. The improved accuracy of our methods is prominent in the first row in Fig. 10 where the mitral valve (indicated in red circles) in the warped source images can be seen to be significantly closer to its position in the target image when our methods are applied. Additionally an improved correspondence to the target mask can be observed in the second row of Fig. 10. Again, this is particularly obvious in the region close to the mitral valve. Collectively, Fig. 9 and 10, together with Table 5, provide ample evidence that our methods have the capability of registering images robustly and accurately.

5. Conclusion

In this paper, we have proposed a new variational model, where the regularization is a term that involves a derivative of arbitrary order. We have then proposed a point-wise, closed-form and over-relaxed ADMM solver, providing an easy way to accurately and efficiently solve this general model. We have also presented comprehensive derivations, theorems, proofs and experiments for relevant formulations and claims. Extensive experiments have shown that the proposed approaches outperform state-of-the-art methods, including the subgradient method, primal dual, FFD, and learning-based methods. The proposed regularization method and accelerated ADMM solver can be generalized to many other image processing problems.

These experimental results in Fig. 3 and Table 5 deliver two important messages: (1) The best performance is not always achieved by only one specific model (e.g. 2nd-order TV); (2) Depending on applications, one should select suitable models carefully. For example, when the underlying distribution of a displacement field is piecewise constant, the 1st-order TV should be used, which is however not suitbale for large displacements as compared to its higher-order counterparts. As the displacement field in Fig. 3 is piecewise quadratic, we are expecting the 3rd-order and 4th-order TVs to outperform the 1st-order and 2nd-order TVs. This

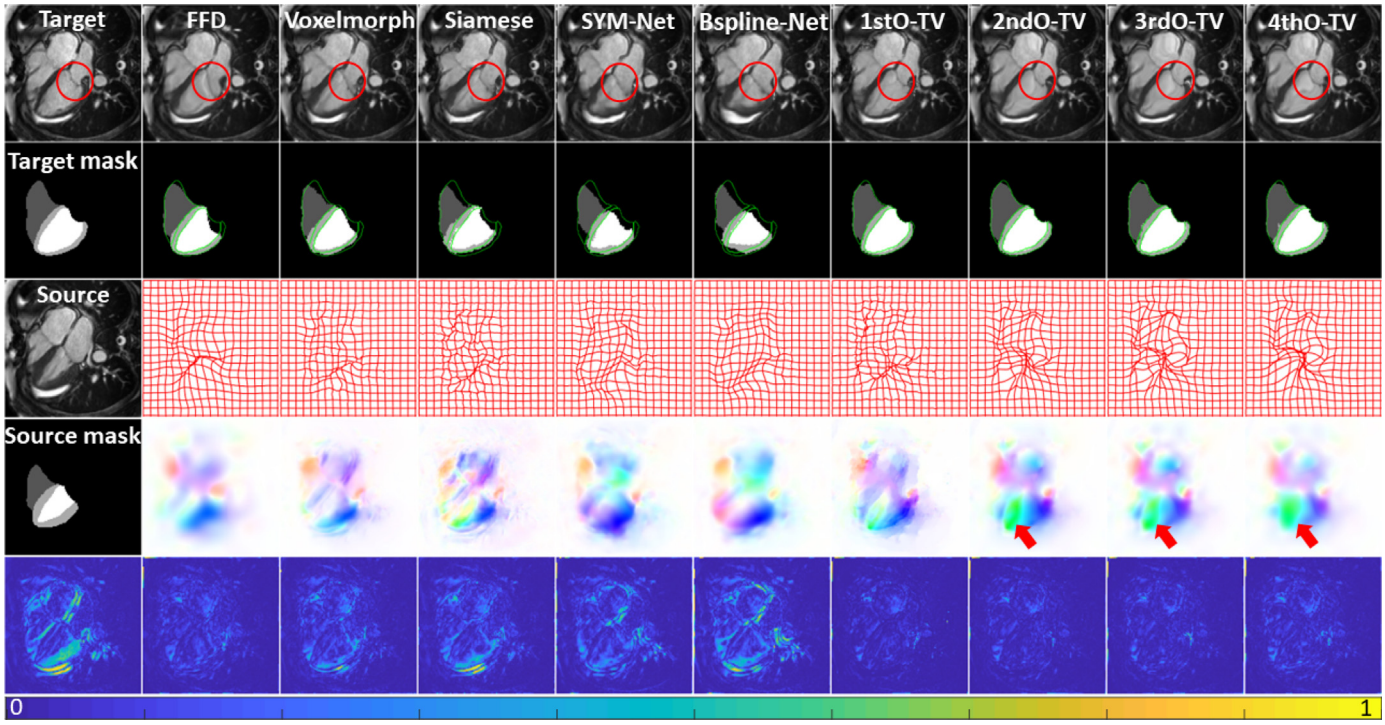

Fig. 10. Same visual results as in Fig. 9, but from a different example in UKBB-2.

is because both the 3rd-order and 4th-order derivatives possess a quadratic behaviour. The reason why in Table 5 most of the best quantitative results come from the 2nd-order TV is that the underlying displacement field mostly likely follows a piecewise linear distribution.

One drawback of our proposed methods is that they produce folding (singular, non-invertible points) in resultant deformation grids (see Fig. 9 and 10), which may lead to numerical instability or warping artifacts. This problem can be eased by using a large regularization parameter λ and a good example can be seen in Fig. 1, where the deformation grid gradually gets disentangled as we increase the value of λ. However, setting λ too big reduces the deformation magnitude and therefore loses the ability to model large deformations. To tackle this dilemma, we will combine diffeomorphisms [7,66,67] with our discontinuous regularizations in our future research, which are capable to compute smooth, invertible spatial transformations between images.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgement

Zhaowen Qiu is supported by the Key R&D Project of Heilongjiang Province (2022ZX01A30), by the Science and Technology Program of Suzhou (ZXL2021431 and RC2021130), and by the Fundamental Research Funds for the Central Universities (2572020DR10). This research used the UK Biobank Resource under the application number 40119.

The GPU computations described in this research were performed using the Baskerville Tier 2 HPC service. Baskerville is funded by the EPSRC and UKRI (EP/T022221/1 and EP/W032244/1) and is operated by Advanced Research Computing at the University of Birmingham. Jinming Duan is partially funded by the BHF Accelerator Award (AA/18/2/34218) and by the Korea Cardiovascular Bioresearch Foundation (CHORUS Seoul 2022). Xi Jia is partially supported by the Chinese Scholarship Council.

Appendix A

In this section, we show in detail how to derive the solution of **v**−***subproblem*** in Section 3.1. At its core, this requires to solve a Poisson-like equation. In the following, we start from a 1D problem, followed by deriving the respective 2D problem. The general cases are given at the end of this section. Note that our derivations on the 1D problem below are inspired by Section 4.5.1 in [68]. However, for the Neumann boundary conditions their derivations do not up directly with the DCT coefficients (eigenvalues) and basises (eigenvectors).

A1. One-dimensional problem

The 1D Poisson equation is given as

$$U(x) - \frac{d^2U(x)}{dx^2} = F(x),\ a \le x \le b. \tag{A.1}$$

Under the *Neumann boundary conditions*, we have $U'(a) = 0$ and $U'(b) = 0$. On a regular grid, the resulting discretization (using a finite difference method) of above differential equation leads to a linear system:

$$(\mathcal{I}_n - \mathcal{T}_n)u = f. \tag{A.2}$$

Here, u and f are respectively the discrete vectors of $U(x)$ and $F(x)$ and each vector has a length of n; $\mathcal{I}_n$ is an identity matrix of size $n \times n$; and $\mathcal{T}_n$ is a tridiagonal coefficient matrix of size $n \times n$, which

has the form of

$$\mathcal{T}_n = \begin{pmatrix} -1 & 1 & & & \\ 1 & -2 & 1 & & \\ & \cdot & \cdot & \cdot & \\ & & 1 & -2 & 1 \\ & & & 1 & -1 \end{pmatrix}. \tag{A.3}$$

Of note, $\mathcal{T}_n$ has a highly structured eigensystem that is closely related to some fast trigonometric transform. In other words, $\mathcal{T}_n$ can be efficiently diagonalized by some eigenvectors. The goal here therefore is to establish these eigensystem/transform connections to show how they can be used to design a very fast and effective Possion solver. Next, we examine the eigenstructure of such a matrix.

Let θ be any real number and for any integer k, let $c_k = \cos\left(\left(k+\frac{1}{2}\right)\theta\right)$ and $s_k = \sin\left(\left(k+\frac{1}{2}\right)\theta\right)$. Considering the following two trigonometric identities:

$$c_{k-1} = \cos(\theta)c_k + \sin(\theta)s_k \quad \text{and} \quad c_{k+1} = \cos(\theta)c_k - \sin(\theta)s_k.$$

By adding the first identity to the second we have

$$c_{k-1} - 2\cos(\theta)c_k + c_{k+1} = 0, \tag{A.4}$$

which will be used to compute the eigenvectors and eigenvalues of $\mathcal{T}_n$.

Lemma 1. *Suppose $\theta \in \mathbb{R}$, if $c_k = \cos\left(\left(k+\frac{1}{2}\right)\theta\right)$ for $k = 0, \ldots, n-1$, we have*

$$\mathcal{T}_n \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{bmatrix} = 2(\cos(\theta)-1) \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{bmatrix} + \begin{bmatrix} c_0 - c_{-1} \\ 0 \\ \vdots \\ 0 \\ c_{n-1} - c_n \end{bmatrix}. \tag{A.5}$$

Proof. From the definition of $\mathcal{T}_n$ in (A.3), we obtain

$$\mathcal{T}_n \begin{bmatrix} c_0 \\ \vdots \\ c_k \\ \vdots \\ c_{n-1} \end{bmatrix} = \begin{bmatrix} c_{-1} - 2c_0 + c_1 \\ \vdots \\ c_{k-1} - 2c_k + c_{k+1} \\ \vdots \\ c_{n-2} - 2c_{n-1} + c_n \end{bmatrix} + \begin{bmatrix} c_0 - c_{-1} \\ \vdots \\ 0 \\ \vdots \\ c_{n-1} - c_n \end{bmatrix}.$$

Focusing on the first component in the right-hand side of the equation above, together with the identity (A.4), we have the following simple derivation

$$c_{k-1} - 2c_k + c_{k+1} = \underbrace{c_{k-1} - 2\cos(\theta)c_k + c_{k+1}}_{0} + 2(\cos(\theta)-1)c_k,$$

which verifies that (A.5) holds. □

We will use this lemma to identify important properties (i.e. eigenvectors and eigenvalues) of the matrix $\mathcal{T}_n$ that arise in solving (A.2). We start with the following theorem.

Theorem 2. *Let $C_{k,j} = \cos\left(\left(k+\frac{1}{2}\right)\frac{j\pi}{n}\right)$ for $k = 0, \ldots, n-1$ and $j = 0, \ldots, n-1$. If $\mathcal{V} = C_{k,j}$ and*

$$\lambda_j = 2\left(\cos\left(\frac{j\pi}{n}\right) - 1\right), \tag{A.6}$$

we have

$$\mathcal{V}^{-1}\mathcal{T}_n\mathcal{V} = \operatorname{diag}(\lambda_0, \ldots, \lambda_{n-1}), \tag{A.7}$$

where $\mathcal{V}$ consists of the eigenvectors of $\mathcal{T}_n$ and the diagonal entries λ_j are the corresponding eigenvalues of $\mathcal{T}_n$.

Proof. Let's pull our attention back to the equation (A.5) in Lemma 1, and recall that $c_k = \cos\left(\left(k+\frac{1}{2}\right)\theta\right)$ and that the cosine function is an even function. As such, the following equation automatically holds

$$c_0 - c_{-1} = \cos\left(\frac{1}{2}\theta\right) - \cos\left(-\frac{1}{2}\theta\right) = 0.$$

In addition, if we define $n\theta = j\pi$ for $j = 0, \ldots, n-1$, we end up with

$$\theta = \frac{j\pi}{n},$$

and the following equation also holds

$$\begin{aligned} c_{n-1} - c_n &= \cos\left(\left(n-\tfrac{1}{2}\right)\theta\right) - \cos\left(\left(n+\tfrac{1}{2}\right)\theta\right) \\ &= 2\sin(n\theta)\sin\left(\tfrac{\theta}{2}\right) = 0. \end{aligned}$$

Now if we plug $c_0 - c_{-1} = 0$, $c_{n-1} - c_n = 0$ and $\theta = j\pi/n$, $j = 0, \ldots, n-1$ into (A.5), we obtain

$$\mathcal{T}_n \begin{bmatrix} \cos\left(\left(0+\frac{1}{2}\right)\frac{j\pi}{n}\right) \\ \vdots \\ \cos\left(\left(n-1+\frac{1}{2}\right)\frac{j\pi}{n}\right) \end{bmatrix} = 2\left(\cos\left(\tfrac{j\pi}{n}\right) - 1\right) \begin{bmatrix} \cos\left(\left(0+\frac{1}{2}\right)\frac{j\pi}{n}\right) \\ \vdots \\ \cos\left(\left(n-1+\frac{1}{2}\right)\frac{j\pi}{n}\right) \end{bmatrix}.$$

The theorem has been proved because above the eigenvalues are equal to that defined in (A.6) and the eigenvectors are equal to $\mathcal{V}$ or $C_{k,j}$ in (A.7). □

Now, we are almost ready to seek a solution to the discrete differential equation (A.2). First, with the eigensystem (A.7) we can derive

$$(\mathcal{I}_n - \mathcal{T}_n)^{-1} = \mathcal{V}(\mathcal{I}_n - \mathcal{D}_n)^{-1}\mathcal{V}^{-1}, \tag{A.8}$$

where $\mathcal{D}_n = \operatorname{diag}(\lambda_0, \ldots, \lambda_{n-1})$. By using the fast eigensystem (A.8) of $\mathcal{T}_n$, we are finally able to solve the linear system of $(\mathcal{I}_n - \mathcal{T}_n)u = f$ as follows:

$$\begin{aligned} &u \leftarrow \mathcal{V}^{-1}f, \\ &u \leftarrow (\mathcal{I}_n - \mathcal{D}_n)^{-1}u, \\ &u \leftarrow \mathcal{V}u. \end{aligned}$$

By definition, one knows that the multiplication of a vector by $\mathcal{V}$ is equivalent to performing an inverse *discrete cosine transform* (DCT) with some scaling weights[10] on a 1D grid.

A2. Two-dimensional problem

We now extend the Poisson problem from 1D to 2D. For the latter case, we are given a function $F(x, y)$ defined on

$$\mathbb{R} = \{(x, y), a \le x \le b,\ c \le y \le d\},$$

and want to determine $U(x, y)$ such that

$$U(x, y) - \left(\frac{\partial^2 U(x, y)}{\partial x^2} + \frac{\partial^2 U(x, y)}{\partial y^2}\right) = F(x, y), \tag{A.9}$$

subject to the *Neumann boundary conditions*. On a 2D Cartesian grid of size $m \times n$, the resulting discretization (using a 2D finite difference method) of the above second-order PDE leads to a linear system:

$$\mathcal{M}u = f \text{ with } \mathcal{M} = \mathcal{I}_m \otimes \mathcal{I}_n - (\mathcal{T}_m \otimes \mathcal{I}_n + \mathcal{I}_m \otimes \mathcal{T}_n). \tag{A.10}$$

[10] In DCT, the scaling weights are applied to make sure the norm of each eigenvector is unit

In the linear system, $\otimes$ denotes *Kronecker product*. u and f are two vectors resulting from the discretization of $U(x,y)$ and $F(x,y)$ respectively and the length of each vector is of mn. In practical, one needs to concatenate all the columns (or rows) in a discrete rectangle matrix (i.e. an image) to form a long vector, like u or f. $\mathcal{M} \in \mathbb{R}^{mn\times mn}$ above is a sparse matrix. The following theorem tells us that $\mathcal{M}$ can be diagonalized efficiently.

Theorem 3. *Suppose* $\mathcal{M} = \mathcal{I}_m \otimes \mathcal{I}_n - (\mathcal{T}_m \otimes \mathcal{I}_n + \mathcal{I}_m \otimes \mathcal{T}_n)$ *is non-singular with* $\mathcal{T}_m \in \mathbb{R}^{m\times m}$ *and* $\mathcal{T}_n \in \mathbb{R}^{n\times n}$. *If* $\mathcal{V}_m^{-1}\mathcal{T}_m\mathcal{V}_m = \mathcal{D}_m$, $\mathcal{V}_n^{-1}\mathcal{T}_n\mathcal{V}_n = \mathcal{D}_n$, *and* $f \in \mathbb{R}^{mn}$, *then the solution to* $\mathcal{M}u = f$ *is given by*

$$u = (\mathcal{V}_m \otimes \mathcal{V}_n)\mathcal{A}\left(\mathcal{V}_m^{-1} \otimes \mathcal{V}_n^{-1}\right)f,$$

where $\mathcal{A} = [\mathcal{I}_m \otimes \mathcal{I}_n - (\mathcal{D}_m \otimes \mathcal{I}_n + \mathcal{I}_m \otimes \mathcal{D}_n)]^{-1}$.

Proof. According to the Kronecker mixed-product property, the following identities hold

$$\begin{aligned}
&(\mathcal{V}_m^{-1} \otimes \mathcal{V}_n^{-1})\ (\mathcal{I}_m \otimes \mathcal{I}_n)(\mathcal{V}_m \otimes \mathcal{V}_n)\\
&\quad = (\mathcal{V}_m^{-1}\mathcal{I}_m\mathcal{V}_m) \otimes (\mathcal{V}_n^{-1}\mathcal{I}_n\mathcal{V}_n) = \mathcal{I}_m \otimes \mathcal{I}_n,\\
&(\mathcal{V}_m^{-1} \otimes \mathcal{V}_n^{-1})\ (\mathcal{T}_m \otimes \mathcal{I}_n)(\mathcal{V}_m \otimes \mathcal{V}_n)\\
&\quad = (\mathcal{V}_m^{-1}\mathcal{T}_m\mathcal{V}_m) \otimes (\mathcal{V}_n^{-1}\mathcal{I}_n\mathcal{V}_n) = \mathcal{D}_m \otimes \mathcal{I}_n,\\
&(\mathcal{V}_m^{-1} \otimes \mathcal{V}_n^{-1})\ (\mathcal{I}_m \otimes \mathcal{T}_n)(\mathcal{V}_m \otimes \mathcal{V}_n)\\
&\quad = (\mathcal{V}_m^{-1}\mathcal{I}_m\mathcal{V}_m) \otimes (\mathcal{V}_n^{-1}\mathcal{T}_n\mathcal{V}_n) = \mathcal{I}_m \otimes \mathcal{D}_n.
\end{aligned}$$

Combing these three identities, we have

$$(\mathcal{V}_m^{-1} \otimes \mathcal{V}_n^{-1})\mathcal{M}(\mathcal{V}_m \otimes \mathcal{V}_n) = \mathcal{I}_m \otimes \mathcal{I}_n - (\mathcal{D}_m \otimes \mathcal{I}_n + \mathcal{I}_m \otimes \mathcal{D}_n),$$

which implies that the theorem follows. □

Putting this in an array format, we have the following framework that solves the linear system (A.10) formally:

$$\begin{aligned}
&u \leftarrow (\mathcal{V}_m^{-1} \otimes \mathcal{V}_n^{-1})f,\\
&u \leftarrow [\mathcal{I}_m \otimes \mathcal{I}_n - (\mathcal{D}_m \otimes \mathcal{I}_n + \mathcal{I}_m \otimes \mathcal{D}_n)]^{-1}u,\\
&u \leftarrow (\mathcal{V}_m \otimes \mathcal{V}_n)u.
\end{aligned} \tag{A.11}$$

Note that here the multiplication of a vector by $\mathcal{V}_m^{-1} \otimes \mathcal{V}_n^{-1}$ is equivalent to applying an 2D DCT to the 2D matrix representation of that vector (normally through reshaping) on a $m \times n$ sized grid. As both the matrices $\mathcal{I}_m \otimes \mathcal{I}_n$ and $\mathcal{D}_m \otimes \mathcal{I}_n + \mathcal{I}_m \otimes \mathcal{D}_n$ are diagonal, we can convert the matrix-vector multiplication into a point-wise division performing on the grid, which reads

$$[\mathcal{I}_m \otimes \mathcal{I}_n - (\mathcal{D}_m \otimes \mathcal{I}_n + \mathcal{I}_m \otimes \mathcal{D}_n)]^{-1}u = \frac{\mathcal{U}}{1-\left(2\cos\left(\frac{q\pi}{m}\right)+2\cos\left(\frac{r\pi}{n}\right)-4\right)},$$

where $r \in [0,n)$ and $q \in [0,m)$ are the integer indices and $\mathcal{U} \in \mathbb{R}^{m\times n}$ is equivalent to u after reshaping back to the $m \times n$ sized grid. $1-\left(2\cos\left(\frac{q\pi}{m}\right)+2\cos\left(\frac{r\pi}{n}\right)-4\right)$ is a coefficient matrix resulting from solving the discrete second-order PDE of (A.9).

A3. General higher-order problems

With the knowledge from Section A.2, we further study similar, but more general higher-order[11] PDE problems in 2D. However, the theorems established here have no problem to extend to 3D cases. Given an arbitrary order PDE with the *Neumann boundary conditions*, we should be aware that there exists a respective linear system $\mathcal{M}u = f$, where the sparse matrix $\mathcal{M}$ is different depending on the order. For a general higher-order PDE defined on a $m \times n$ 2D grid, $\mathcal{M}$ has the following general form:

$$\mathcal{M} = \mathcal{I}_m \otimes \mathcal{I}_n + (-1)^s \sum_{p=0}^{s} \binom{s}{p} \left(\mathcal{T}_m^{s-p} \otimes \mathcal{T}_n^{p}\right), \tag{A.12}$$

[11] By higher order, we mean that the order s in (A.12) should be greater than 1. However, our derivations in this section still follow when $s = 1$, because (A.12) is equivalent to $\mathcal{M}$ in (A.10) when $s = 1$.

and

$$\binom{s}{p} = \prod_{l=0}^{p-1} \frac{s-l}{p-l}$$

is a positive integer known as the binomial coefficient. Note that the subscript $s-p$ on $\mathcal{T}_m$ and the superscript p on $\mathcal{T}_n$ denote the power of the matrix. $(-1)^s \sum_{p=0}^{s} \binom{s}{p} \left(\mathcal{T}_m^{s-p} \otimes \mathcal{T}_n^{p}\right)$ is equivalent to $(-1)^s\mathcal{D}(\mathrm{div}^s\nabla^s)$ in (10).

Theorem 4. *Suppose that* $\mathcal{M} = \mathcal{I}_m \otimes \mathcal{I}_n + (-1)^s \sum_{p=0}^{s} \binom{s}{p} \left(\mathcal{T}_m^{s-p} \otimes \mathcal{T}_n^{p}\right)$ *is non-singular with* $\mathcal{T}_m \in \mathbb{R}^{m\times m}$ *and* $\mathcal{T}_n \in \mathbb{R}^{n\times n}$. *If* $\mathcal{V}_m^{-1}\mathcal{T}_m\mathcal{V}_m = \mathcal{D}_m$, $\mathcal{V}_n^{-1}\mathcal{T}_n\mathcal{V}_n = \mathcal{D}_n$, *and* $f \in \mathbb{R}^{mn}$, *then the solution to* $\mathcal{M}u = f$ *is given by*

$$u = (\mathcal{V}_m \otimes \mathcal{V}_n)\mathcal{B}\left(\mathcal{V}_m^{-1} \otimes \mathcal{V}_n^{-1}\right)f,$$

where $\mathcal{B} = [\mathcal{I}_m \otimes \mathcal{I}_n + (-1)^s \sum_{p=0}^{s} \binom{s}{p} \left(\mathcal{D}_m^{s-p} \otimes \mathcal{D}_n^{p}\right)]^{-1}$.

Proof. First, we show that the following derivation follows

$$\begin{aligned}
(\mathcal{V}_n^{-1}\mathcal{T}_n\mathcal{V}_n)^p &= \underbrace{(\mathcal{V}_n^{-1}\mathcal{T}_n\mathcal{V}_n)(\mathcal{V}_n^{-1}\mathcal{T}_n\mathcal{V}_n)\cdots(\mathcal{V}_n^{-1}\mathcal{T}_n\mathcal{V}_n)}_{\text{repeat } p \text{ times}}\\
&= \mathcal{V}_n^{-1}\mathcal{T}_n^{p}\mathcal{V}_n = \mathcal{D}_n^{p}.
\end{aligned}$$

As such, we obtain

$$\begin{aligned}
&(\mathcal{V}_m^{-1} \otimes \mathcal{V}_n^{-1})\left((-1)^s \sum_{p=0}^{s} \binom{s}{p} \left(\mathcal{T}_m^{s-p} \otimes \mathcal{T}_n^{p}\right)\right)(\mathcal{V}_m \otimes \mathcal{V}_n)\\
&\quad = (-1)^s \sum_{p=0}^{s} \binom{s}{p} \left(\mathcal{V}_m^{-1}\mathcal{T}_m^{s-p}\mathcal{V}_m \otimes \mathcal{V}_n^{-1}\mathcal{T}_n^{p}\mathcal{V}_n\right)\\
&\quad = (-1)^s \sum_{p=0}^{s} \binom{s}{p} \left(\mathcal{D}_m^{s-p} \otimes \mathcal{D}_n^{p}\right),
\end{aligned}$$

which implies that the theorem follows. □

Putting this in an array format, we have the following framework that formally solves the linear system $\mathcal{M}u = f$ with $\mathcal{M}$ defined in (A.12):

$$\begin{aligned}
&u \leftarrow (\mathcal{V}_m^{-1} \otimes \mathcal{V}_n^{-1})f,\\
&u \leftarrow [\mathcal{I}_m \otimes \mathcal{I}_n + (-1)^s \textstyle\sum_{p=0}^{s} \binom{s}{p} \left(\mathcal{D}_m^{s-p} \otimes \mathcal{D}_n^{p}\right)]^{-1}u,\\
&u \leftarrow (\mathcal{V}_m \otimes \mathcal{V}_n)u.
\end{aligned} \tag{A.13}$$

Finally, as both the matrices $\mathcal{I}_m \otimes \mathcal{I}_n$ and $(-1)^s \sum_{p=0}^{s} \binom{s}{p} \left(\mathcal{D}_m^{s-p} \otimes \mathcal{D}_n^{p}\right)$ are diagonal, we can convert the matrix-vector multiplication into a point-wise division performing on the 2D grid, which reads

$$\begin{aligned}
&[\mathcal{I}_m \otimes \mathcal{I}_n + (-1)^s \sum_{p=0}^{s} \binom{s}{p} \left(\mathcal{D}_m^{s-p} \otimes \mathcal{D}_n^{p}\right)]^{-1}u\\
&= \frac{\mathcal{U}}{1+(-1)^s\left(2\cos\left(\frac{q\pi}{m}\right)+2\cos\left(\frac{r\pi}{n}\right)-4\right)^s},
\end{aligned}$$

where $(-1)^s\left(2\cos\left(\frac{q\pi}{m}\right)+2\cos\left(\frac{r\pi}{n}\right)-4\right)^s$ is the coefficient matrix resulting from applying the fast trigonometric transform (i.e. DCT) to the higher-order linear PDE. This coefficient is the same to (11).

Appendix B

In this section, we develop and implement a primal-dual algorithm to minimize the proposed model (2b). We first introduce the general primal dual and then apply it to (2b).

B1. General primal dual

Similar to ADMM, primal dual tackles saddle-point problems and usually considers the form

$$\min_{x\in X}\max_{y\in Y} F(x)+\lambda\langle Kx,y\rangle - G(y), \tag{B.1}$$

where X and Y are two finite-dimensional real vector spaces, $K: X\to Y$ is a continuous linear operator, and $F: X\to[0,+\infty)$ and $G: Y\to[0,+\infty)$ are convex functions. λ usually is a smooth parameter. Given the adjoint operator K^*, the primal-dual algorithm to optimise such an objective function is listed in Algorithm 2.

Algorithm 2: General Primal-Dual Algorithm

Input variables: $x^0\in X$, $y^0\in Y$ and $\widehat{x}^0=x^0$
Input parameters: $\lambda>0$, $\sigma>0$, $\tau>0$
while *Not Converged* **do**
$y^{k+1}=(I+\sigma\partial G)^{-1}\left(y^k+\lambda\sigma K\widehat{x}^k\right)$
$x^{k+1}=(I+\tau\partial F)^{-1}\left(x^k-\lambda\tau K^*y^{k+1}\right)$
$\widehat{x}^{k+1}=x^{k+1}+\theta\left(x^{k+1}-x^k\right)$

Note that $x-update$ and $y-update$ in the algorithm respectively use the proximal operators of G and F, which correspond to the following two minimization problems

$$\begin{aligned}(I+\sigma\partial G)^{-1}(\widehat{y}) &= \arg\min_{y\in Y} G(y)+\tfrac{1}{2\sigma}\|y-\widehat{y}\|_2^2,\\ (I+\tau\partial F)^{-1}(\widehat{x}) &= \arg\min_{x\in X} F(x)+\tfrac{1}{2\tau}\|x-\widehat{x}\|_2^2.\end{aligned}$$

Note that λ and the step sizes σ and τ need to satisfy $\lambda\sigma\tau<\frac{1}{r(K^*K)}$ for the algorithm to converge, where $r(K^*K)$ denotes the spectral radius of the matrix K^*K. Also note that $\widehat{x}-update$ in Algorithm 2 is an extrapolation prediction step of the form $\widehat{x}^{k+1}=x^{k+1}+\theta\left(x^{k+1}-x^k\right)$, where $\theta=1$ has been used in convex problems such as motion estimation [31]. It have been analyzed for $[-1,1]$ in [69]. Some authors have also suggested adaptive step sizes [70]. In this study, we set $\theta=1$ to gain the acceleration of speed.

B2. Applying primal dual to (2b)

We now apply Algorithm 2 to our problem (2b), which is

$$\min_{\mathbf{u}}\|\rho_{\mathbf{u}^\omega}(\mathbf{u})\|_1+\lambda\sum_{i,j}\sqrt{|(\nabla^n u_1)_{i,j}|^2+|(\nabla^n u_2)_{i,j}|^2}. \tag{B.2}$$

The minimization problem (B.2) can be converted equivalently to a saddle-point problem by writing the regularization term as a maximization, i.e.

$$\max_{\|\mathbf{q}\|_\infty\le 1}\langle\mathbf{q},\nabla^n\mathbf{u}\rangle, \tag{B.3}$$

over the dual variables $\mathbf{q}=(\mathbf{q}_1,\mathbf{q}_2)$ for $\mathbf{q}_1\in(\mathbb{R}^{M\times N})^{2^n}$ and $\mathbf{q}_2\in(\mathbb{R}^{M\times N})^{2^n}$. The inner product in (B.3) is defined as

$$\langle\mathbf{q},\nabla^n\mathbf{u}\rangle=\sum_{i,j}\sum_{l=1}^{2^n}\left[(\mathbf{q}_1)_{i,j}^l(\nabla^n u_1)_{i,j}^l+(\mathbf{q}_2)_{i,j}^l(\nabla^n u_2)_{i,j}^l\right],$$

and $\|\mathbf{q}\|_\infty$ in (B.3) is the maximum norm which has the form of

$$\|\mathbf{q}\|_\infty=\max_{i,j}\sqrt{|(\mathbf{q}_1)_{i,j}|^2+|(\mathbf{q}_2)_{i,j}|^2}.$$

With these, we can define a convex set for which the dual maximization (B.3) is taken over

$$Q=\left\{\mathbf{q}=(\mathbf{q}_1\in(\mathbb{R}^{M\times N})^{2^n},\mathbf{q}_2\in(\mathbb{R}^{M\times N})^{2^n}):\|\mathbf{q}\|_\infty\le 1\right\}.$$

We then introduce the indicator function of the set Q

$$\delta(\mathbf{q})=\begin{cases}0 & \text{if } \mathbf{q}\in Q\\ +\infty & \text{if } \mathbf{q}\notin Q\end{cases}. \tag{B.4}$$

With (B.3) and (B.4), the non-smooth, regularized problem (B.2) can be written in its primal-dual form

$$\min_{\mathbf{u}}\max_{\mathbf{q}}\|\rho_{\mathbf{u}^\omega}(\mathbf{u})\|_1+\lambda\langle\mathbf{q},\nabla^n\mathbf{u}\rangle-\delta(\mathbf{q}), \tag{B.5}$$

which is a saddle-point problem in the form of (B.1) with $x=\mathbf{u}$, $y=\mathbf{q}$, $F=\|\rho_{\mathbf{u}^\omega}(\mathbf{u})\|_1$, $G=\delta(\mathbf{q})$, $X=(\mathbb{R}^{M\times N})^2$, $Y=Q$, $K=\nabla^n$ and $K^*=(-1)^n\mathrm{div}^n$.

To apply Algorithm 2, we need to specify the point-wise proximal operators in $y-update$ and $x-update$, which respectively correspond to $\mathbf{q}-update$ and $\mathbf{u}-update$ below, given by

$$\begin{aligned}(I+\sigma\partial G)^{-1}(\widehat{\mathbf{q}}) &= \arg\min_{\mathbf{q}}\delta(\mathbf{q})+\tfrac{1}{2\sigma}\|\mathbf{q}-\widehat{\mathbf{q}}\|_2^2\\ &= \frac{\mathbf{q}_{i,j}}{\max\left(\sqrt{|(\mathbf{q}_1)_{i,j}|^2+|(\mathbf{q}_2)_{i,j}|^2},1\right)},\end{aligned}$$

and

$$(I+\tau\partial F)^{-1}(\widehat{\mathbf{u}})=\arg\min_{\mathbf{u}}\|\rho_{\mathbf{u}^\omega}(\mathbf{u})\|_1+\frac{1}{2\tau}\|\mathbf{u}-\widehat{\mathbf{u}}\|_2^2,$$

whose closed-form solution can be found in (8). Finally, the primal dual to optimize (2b) is given in Algorithm 3.

Algorithm 3: Primal-Dual Algorithm for (2b)

Input variables: $\mathbf{u}^0=\mathbf{0}$, $\mathbf{q}^0=\mathbf{0}$, and $\widehat{\mathbf{u}}^0=\mathbf{u}^0$
Input parameters: $\lambda>0$, $\sigma>0$, $\tau>0$, $\lambda\sigma\tau<\frac{1}{8^n}$, and $\theta=1$
while *Not Converged* **do**
$\mathbf{q}^{k+1}=(I+\sigma\partial G)^{-1}\left(\mathbf{q}^k+\lambda\sigma\nabla^n\widehat{\mathbf{u}}^k\right)$
$\mathbf{u}^{k+1}=(I+\tau\partial F)^{-1}\left(\mathbf{u}^k-\lambda\tau(-1)^n\mathrm{div}^n\left(\mathbf{q}^{k+1}\right)\right)$
$\widehat{\mathbf{u}}^{k+1}=\mathbf{u}^{k+1}+\theta\left(\mathbf{u}^{k+1}-\mathbf{u}^k\right)$

We remark that the spectral radius (largest absolute eigenvalue) of the matrix K^*K can be very efficiently computed by applying DCT to the matrix, i.e., see (11). Depending on the derivative order n, we have $\lambda\sigma\tau<\frac{1}{8^n}$, which can used to select λ, σ and τ in Algorithm 3. The proof for this inequality is straightforward as the upper bound of (11) is 8^n. Also note that Algorithm 3 needs to be combined with Algorithm 1 to minimize (1). That is to say, in order to compute $\mathbf{u}^{k+1}$ one just replaces the ADMM updates in the third loop in Algorithm 1 with the primal dual updates in Algorithm 3.

Appendix C

In this section, we propose to derive the solution (8) for $\mathbf{u}-$***subproblem*** in Section 3.1 using a new primal-dual method different to Appendix Appendix B. Note that our derivations below are also different to those proposed in [71]. First, we recall the problem (7) as

$$\min_{\mathbf{u}}\|\rho_{\mathbf{u}^\omega}(\mathbf{u})\|_1+\frac{\theta_2}{2}\sum_{p}^{2}\|v_p-u_p-d_p\|_2^2,$$

where the discrete $\rho_{\mathbf{u}^\omega}(\mathbf{u})$ has been given in (4b). This minimization problem can be converted equivalently to a saddle-point problem by writing the first term as a maximization, i.e.

$$\|\rho_{\mathbf{u}^\omega}(\mathbf{u})\|_1=\max_{\|z\|_\infty\le 1}\langle\rho_{\mathbf{u}^\omega}(\mathbf{u}),z\rangle,$$

over the dual variable $z\in\mathbb{R}^{M\times N}$, where $\|z\|_\infty=\max_{i,j}(\mathrm{abs}(z_{i,j}))$ and $\langle\rho_{\mathbf{u}^\omega}(\mathbf{u}),z\rangle=\sum_{i,j}(\rho_{\mathbf{u}^\omega}(\mathbf{u}))_{i,j}z_{i,j}$. As such, the above minimization problem is equivalent to the following primal-dual (min-max)

problem, i.e.

$$\min_{\mathbf{u}} \max_{z:\|z\|_\infty \le 1} \langle \rho_{\mathbf{u}^\omega}(\mathbf{u}), z\rangle + \frac{\theta_2}{2}\sum_p^2 \|v_p^k - u_p - d_p^k\|_2^2, \tag{C.1}$$

over the primal variable $\mathbf{u}$ and the dual variable z, respectively.

If we differentiate (C.1) with respect to $\mathbf{u}$ and then set the derivative zero, we can obtain the following system of equations, from which we have the closed-form solution of $\mathbf{u}$:

$$\theta_2(\mathbf{u} - \mathbf{v}^k + \mathbf{d}^k) + z\nabla I_1^\omega = 0 \rightarrow \mathbf{u} = \mathbf{v}^k - \mathbf{d}^k - z\frac{\nabla I_1^\omega}{\theta_2}. \tag{C.2}$$

Now we can plug the solution of $\mathbf{u}$ in (C.2) back to (C.1). In this case, (C.1) is converted into a constrained maximization problem only with respect to z:

$$\max_{z:\|z\|_\infty \le 1} \left\langle \rho_{\mathbf{u}^\omega}\left(\mathbf{v}^k - \mathbf{d}^k - z\frac{\nabla I_1^\omega}{\theta_2}\right), z\right\rangle + \frac{\theta_2}{2}\sum_{i,j}\left|z_{i,j}\frac{(\nabla I_1^\omega)_{i,j}}{\theta_2}\right|_2^2, \tag{C.3}$$

which can be solved by projected gradient ascent. An important note here is there is no coupling in (C.3), so we can optimize it point-wisely. As (C.3) is differentiable, it is trivial for us to derive its gradient with respect to z, which is $\rho_{\mathbf{u}^\omega}\left(\mathbf{v}^k - \mathbf{d}^k - z\frac{\nabla I_1^\omega}{\theta_2}\right)$. With the gradient, the projected gradient ascent iterates as:

$$z_{i,j}^+ = P_Z\left(z_{i,j} + t\rho_{\mathbf{u}^\omega}\left(\mathbf{v}_{i,j}^k - \mathbf{d}_{i,j}^k - z_{i,j}\frac{(\nabla I_1^\omega)_{i,j}}{\theta_2}\right)\right), \tag{C.4}$$

where $P_Z(x)$ is a projection operator and defined as follows. This step guarantees the solution after projection to satisfy the constraint $\|z\|_\infty \le 1$

$$P_Z(x) = \frac{x}{\max(\mathrm{abs}(x), 1)}.$$

On the other hand, t in (C.4) is a step size, which can be found by using a line search process by plugging the quantity of P_Z in (C.4) to (C.3) and then maximizing the objective with respect to t. The optimal t is therefore:

$$t = \frac{\theta_2}{|(\nabla I_1^\omega)_{i,j}|^2}.$$

If we now plug this step size back to (C.4) we have a closed-form solution for z without iterations, which is

$$z_{i,j} = P_Z\left(\frac{\theta_2}{|(\nabla I_1^\omega)_{i,j}|^2}\rho_{\mathbf{u}^\omega}\left(\mathbf{v}_{i,j}^k - \mathbf{d}_{i,j}^k\right)\right). \tag{C.5}$$

If we plug (C.5) back into (C.2) we have the closed-form solution for $\mathbf{u}$ without involving the dual variable z

$$\mathbf{u}_{i,j} = \mathbf{v}_{i,j}^k - \mathbf{d}_{i,j}^k - P_Z\left(\frac{\theta_2}{|(\nabla I_1^\omega)_{i,j}|^2}\rho_{\mathbf{u}^\omega}\left(\mathbf{v}_{i,j}^k - \mathbf{d}_{i,j}^k\right)\right)\frac{(\nabla I_1^\omega)_{i,j}}{\theta_2}, \tag{C.6}$$

which is the same to the solution (8) in Section 3.1. Finally, we note that the proposed primal dual here naturally applies to deriving the point-wise soft thresholding equation (12) from $\mathbf{w}$–***subproblem*** in Section 3.1.

For the non-smooth, convex problem (2b), we can implement its subgradient method directly without introducing any auxiliary variable:

$$\mathbf{u}_{i,j}^{k+1} = \mathbf{u}_{i,j}^k - t^k(\mathbf{g}_{i,j}^k + \mathbf{r}_{i,j}^k), \tag{D.1}$$

where the subgradients $\mathbf{g} \in (\mathbb{R}^{M\times N})^2$ and $\mathbf{r} \in (\mathbb{R}^{M\times N})^2$ are associated with data and regularization term, respectively. By the chain rule, these two subgradients can be derived as:

$$\mathbf{g}_{i,j}^k = (\nabla I_1^\omega)_{i,j}\begin{cases}\mathrm{sign}[(\rho_{\mathbf{u}^\omega}(\mathbf{u}^k))_{i,j}] & \text{if } (\rho_{\mathbf{u}^\omega}(\mathbf{u}^k))_{i,j} \neq 0\\ [-1,1] & \text{if } (\rho_{\mathbf{u}^\omega}(\mathbf{u}^k))_{i,j} = 0\end{cases},$$

and

$$\mathbf{r}_p^k = \lambda(-1)^n \mathrm{div}^n(\gamma),$$

with $\gamma \in \mathbb{R}^{M\times N}$ defined as

$$\gamma_{i,j} = \begin{cases}\frac{(\nabla^n u_p^k)_{i,j}}{\sqrt{\sum_p |(\nabla^n u_p^k|^2)_{i,j}}} & \text{if } \sum_p |(\nabla^n u_p^k)_{i,j}|^2 \neq 0\\ \{\boldsymbol{z} : |\boldsymbol{z}| \le 1\} & \text{if } \sum_p |(\nabla^n u_p^k)_{i,j}|^2 = 0\end{cases}.$$

For $\mathbf{g}$, the gradient of this function is $(\rho_{\mathbf{u}^\omega}(\mathbf{u}^k))_{i,j}/|(\rho_{\mathbf{u}^\omega}(\mathbf{u}^k))_{i,j}|$ when $(\rho_{\mathbf{u}^\omega}(\mathbf{u}^k))_{i,j} \neq 0$ but not well defined at $(\rho_{\mathbf{u}^\omega}(\mathbf{u}^k))_{i,j} = 0$ since the denominator will become 0. In $\mathbf{r}_p^k, \forall p = 1, 2, \boldsymbol{z} \in \mathbb{R}^{2n}$. For $\mathbf{r}_p$, the gradient is also not well defined when the denominator is 0. In this case, the gradient is any element of the L_2 ball defined as $\{\boldsymbol{z} : |\boldsymbol{z}| \le 1\}$.

For the subgradient method, we use $\lambda = 0.1$ and $t^k = 1/\sqrt{k}$. The definition follows the diminishing step size rule [54] to guarantee algorithm convergence. The formal subgradient algorithm to optimize (2b) is given in Algorithm 4. Note that Algorithm 4 needs to be combined with Algorithm 1 to minimize (1). That is to say, in order to compute $\mathbf{u}^{k+1}$ one just replaces the ADMM updates in the third loop in Algorithm 1 with the subgradient updates in Algorithm 4.

Algorithm 4: Subgradient Algorithm for (2b)

Input variables: $\mathbf{u}^0 = \mathbf{0}$
Input parameters: $\lambda > 0$, and $t = 1$
while *Not Converged* **do**
  $t^k = 1/\sqrt{k}$
  $\mathbf{u}_{i,j}^{k+1} = \mathbf{u}_{i,j}^k - t^k(\mathbf{g}_{i,j}^k + \mathbf{r}_{i,j}^k)$

References

[1] B. Fischer, J. Modersitzki, Fast diffusion registration, Contemporary Mathematics 313 (2002) 117–128.
[2] G. Balakrishnan, A. Zhao, M.R. Sabuncu, J. Guttag, A.V. Dalca, Voxelmorph: a learning framework for deformable medical image registration, IEEE Transactions on Medical Imaging 38 (8) (2019) 1788–1800.
[3] C. Zach, T. Pock, H. Bischof, A duality based approach for realtime TV-L1 optical flow, in: Joint Pattern Recognition Symposium, Springer, 2007, pp. 214–223.
[4] A. Wedel, T. Pock, C. Zach, H. Bischof, D. Cremers, An improved algorithm for TV-L1 optical flow, in: Statistical and Geometrical Approaches to Visual Motion Analysis, Springer, 2009, pp. 23–45.
[5] C. Frohn-Schauf, S. Henn, K. Witsch, Multigrid based total variation image registration, Computing and Visualization in Science 11 (2) (2008) 101–113.
[6] V. Vishnevskiy, T. Gass, G. Szekely, C. Tanner, O. Goksel, Isotropic total variation regularization of displacements in parametric image registration, IEEE Transactions on Medical Imaging 36 (2) (2016) 385–395.
[7] M.F. Beg, M.I. Miller, A. Trouvé, L. Younes, Computing large deformation metric mappings via geodesic flows of diffeomorphisms, International Journal of Computer Vision 61 (2) (2005) 139–157.
[8] C. Chen, B. Gris, O. Oktem, A new variational model for joint image reconstruction and motion estimation in spatiotemporal imaging, SIAM Journal on Imaging Sciences 12 (4) (2019) 1686–1719.
[9] Z. Nie, X. Yang, Deformable image registration using functions of bounded deformation, IEEE Transactions on Medical Imaging 38 (6) (2019) 1488–1500.
[10] D. Rueckert, L.I. Sonoda, C. Hayes, D.L. Hill, M.O. Leach, D.J. Hawkes, Non-rigid registration using free-form deformations: application to breast MRI, IEEE Transactions on Medical Imaging 18 (8) (1999) 712–721.
[11] B. Fischer, J. Modersitzki, Curvature based image registration, Journal of Mathematical Imaging and Vision 18 (1) (2003) 81–85.
[12] J. Modersitzki, Numerical methods for image registration, Oxford University Press on Demand, 2004.
[13] T. Lin, C. Le Guyader, I. Dinov, P. Thompson, A. Toga, L. Vese, Gene expression data to mouse atlas registration using a nonlinear elasticity smoother and landmark points constraints, Journal of Scientific Computing 50 (3) (2012) 586–609.
[14] L.A. Vese, C. Le Guyader, Variational methods in image processing, CRC Press, 2016.
[15] N. Chumchob, K. Chen, C. Brito-Loeza, A fourth-order variational image registration model and its fast multigrid algorithm, Multiscale Modeling & Simulation 9 (1) (2011) 89–128.

[16] N. Chumchob, K. Chen, Improved variational image registration model and a fast algorithm for its numerical approximation, Numerical Methods for Partial Differential Equations 28 (6) (2012) 1966–1995.
[17] M. Ibrahim, K. Chen, C. Brito-Loeza, A novel variational model for image registration using gaussian curvature, Geometry, Imaging and Computing 1 (4) (2014) 417–446.
[18] H. Köstler, K. Ruhnau, R. Wienands, Multigrid solution of the optical flow system using a combined diffusion-and curvature-based regularizer, Numerical Linear Algebra with Applications 15 (2-3) (2008) 201–218.
[19] C. Vogel, S. Roth, K. Schindler, An evaluation of data costs for optical flow, in: German Conference on Pattern Recognition, Springer, 2013, pp. 343–353.
[20] M. Werlberger, T. Pock, H. Bischof, Motion estimation with non-local total variation regularization, in: CVPR, 2010, pp. 2464–2471.
[21] R. Ranftl, K. Bredies, T. Pock, Non-local total generalized variation for optical flow estimation, in: European Conference on Computer Vision, Springer, 2014, pp. 439–454.
[22] B.W. Papież, A. Szmul, V. Grau, J.M. Brady, J.A. Schnabel, Non-local graph-based regularization for deformable image registration, in: Medical Computer Vision and Bayesian and Graphical Models for Biomedical Imaging, Springer, 2016, pp. 199–207.
[23] B. Huang, L. Ge, G. Chen, M. Radenkovic, X. Wang, J. Duan, Z. Pan, Nonlocal graph theory based transductive learning for hyperspectral image classification, Pattern Recognition 116 (2021) 107967.
[24] J. Zhang, K. Chen, Variational image registration by a total fractional-order variation model, Journal of Computational Physics 293 (2015) 442–461.
[25] C. Xu, Y. Wen, B. He, A novel fractional order derivate based log-demons with driving force for high accurate image registration, in: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2019, pp. 1997–2001.
[26] D.P.L. Ferreira, E. Ribeiro, C.A.Z. Barcelos, A variational approach to non-rigid image registration with bregman divergences and multiple features, Pattern Recognition 77 (2018) 237–247.
[27] T. Brox, A. Bruhn, N. Papenberg, J. Weickert, High accuracy optical flow estimation based on a theory for warping, in: European Conference on Computer Vision, Springer, 2004, pp. 25–36.
[28] W. Lu, M.-L. Chen, G.H. Olivera, K.J. Ruchala, T.R. Mackie, Fast free-form deformable registration via calculus of variations, Physics in Medicine & Biology 49 (14) (2004) 3067.
[29] E. Haber, J. Modersitzki, A multilevel method for image registration, SIAM Journal on Scientific Computing 27 (5) (2006) 1594–1607.
[30] K. Chen, G.N. Grapiglia, J. Yuan, D. Zhang, Improved optimization methods for image registration problems, Numerical Algorithms 80 (2) (2019) 305–336.
[31] A. Chambolle, T. Pock, A first-order primal-dual algorithm for convex problems with applications to imaging, Journal of Mathematical Imaging and Vision 40 (1) (2011) 120–145.
[32] C. Qin, W. Bai, J. Schlemper, S.E. Petersen, S.K. Piechnik, S. Neubauer, D. Rueckert, Joint learning of motion estimation and segmentation for cardiac MRI sequences, in: MICCAI, 2018, pp. 472–480.
[33] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Robust Estimation in Numerical recipes in Fortran 77, volume 1, Cambridge university press Cambridge, 1992.
[34] G. Steidl, S. Didas, J. Neumann, Splines in higher order TV regularization, International Journal of Computer Vision 70 (3) (2006) 241–255.
[35] C. Poschl, O. Scherzer, Characterization of minimizers of convex regularization functionals, Contemporary Mathematics 451 (2008) 219–248.
[36] S.-J. Kim, K. Koh, S. Boyd, D. Gorinevsky, l_1 trend filtering, SIAM Review 51 (2) (2009) 339–360.
[37] R.J. Tibshirani, Divided differences, falling factorials, and discrete splines: Another look at trend filtering and related problems, arXiv preprint arXiv:2003.03886 (2020).
[38] L.I. Rudin, S. Osher, E. Fatemi, Nonlinear total variation based noise removal algorithms, Physica D: Nonlinear Phenomena 60 (1-4) (1992) 259–268.
[39] K. Bredies, K. Kunisch, T. Pock, Total generalized variation, SIAM Journal on Imaging Sciences 3 (3) (2010) 492–526.
[40] J. Duan, W. Lu, C. Tench, I. Gottlob, F. Proudlock, N.N. Samani, L. Bai, Denoising optical coherence tomography using second order total generalized variation decomposition, Biomedical Signal Processing and Control 24 (2016) 120–127.
[41] A. Chambolle, P.-L. Lions, Image recovery via total variation minimization and related problems, Numerische Mathematik 76 (2) (1997) 167–188.
[42] S. Setzer, G. Steidl, Variational methods with higher order derivatives in image processing, Approximation 12 (2008) 360–386.
[43] K. Papafitsoros, C.-B. Schönlieb, A combined first and second order variational approach for image reconstruction, Journal of Mathematical Imaging and Vision 48 (2) (2014) 308–338.
[44] S. Boyd, N. Parikh, E. Chu, B. Peleato, et al., Distributed optimization and statistical learning via the alternating direction method of multipliers, Foundations and Trends in Machine learning 3 (1) (2011) 1–122.
[45] X.-C. Tai, J. Hahn, G.J. Chung, A fast algorithm for Euler's elastica model using augmented lagrangian method, SIAM Journal on Imaging Sciences 4 (1) (2011) 313–344.
[46] J. Eckstein, Parallel alternating direction multiplier decomposition of convex programs, Journal of Optimization Theory and Applications 80 (1) (1994) 39–62.
[47] E. Ghadimi, A. Teixeira, I. Shames, M. Johansson, Optimal parameter selection for the alternating direction method of multipliers (ADMM): quadratic problems, IEEE Transactions on Automatic Control 60 (3) (2014) 644–658.
[48] M.V. Afonso, J.M. Bioucas-Dias, M.A. Figueiredo, Fast image recovery using variable splitting and constrained optimization, IEEE Transactions on Image Processing 19 (9) (2010) 2345–2356.
[49] B. Wohlberg, Efficient algorithms for convolutional sparse representations, IEEE Transactions on Image Processing 25 (1) (2015) 301–315.
[50] W. Lu, J. Duan, Z. Qiu, Z. Pan, R.W. Liu, L. Bai, Implementation of high-order variational models made easy for image processing, Mathematical Methods in the Applied Sciences 39 (14) (2016) 4208–4233.
[51] J. Duan, W.O. Ward, L. Sibbett, Z. Pan, L. Bai, Introducing diffusion tensor to high order variational model for image reconstruction, Digital Signal Processing 69 (2017) 323–336.
[52] M.K. Ng, R.H. Chan, W.-C. Tang, A fast algorithm for deblurring models with Neumann boundary conditions, SIAM Journal on Scientific Computing 21 (3) (1999) 851–866.
[53] G. Strang, The discrete cosine transform, SIAM Review 41 (1) (1999) 135–147.
[54] S. Boyd, L. Xiao, A. Mutapcic, Subgradient methods, Lecture Notes of EE392o, Stanford University, Autumn Quarter 2004 (2003) 2004–2005.
[55] S.E. Petersen, P.M. Matthews, F. Bamberg, D.A. Bluemke, J.M. Francis, M.G. Friedrich, P. Leeson, E. Nagel, S. Plein, F.E. Rademakers, et al., Imaging in population science: cardiovascular magnetic resonance in 100,000 participants of UK biobank-rationale, challenges and approaches, Journal of Cardiovascular Magnetic Resonance 15 (1) (2013) 46.
[56] O. Bernard, A. Lalande, C. Zotti, F. Cervenansky, X. Yang, P.-A. Heng, I. Cetin, K. Lekadir, O. Camara, M.A.G. Ballester, et al., Deep learning techniques for automatic MRI cardiac multi-structures segmentation and diagnosis: is the problem solved? IEEE Transactions on Medical Imaging 37 (11) (2018) 2514–2525.
[57] H. Qiu, C. Qin, L. Le Folgoc, B. Hou, J. Schlemper, D. Rueckert, Deep learning for cardiac motion estimation: Supervised vs. unsupervised training, in: International Workshop on Statistical Atlases and Computational Models of the Heart, Springer, 2019, pp. 186–194.
[58] T.C. Mok, A. Chung, Fast symmetric diffeomorphic image registration with convolutional neural networks, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 4644–4653.
[59] H. Qiu, C. Qin, A. Schuh, K. Hammernik, D. Rueckert, Learning diffeomorphic and modality-invariant registration using b-splines, Medical Imaging with Deep Learning, 2021.
[60] A. Klein, J. Andersson, B.A. Ardekani, J. Ashburner, B. Avants, M.-C. Chiang, G.E. Christensen, D.L. Collins, J. Gee, P. Hellier, et al., Evaluation of 14 nonlinear deformation algorithms applied to human brain MRI registration, Neuroimage 46 (3) (2009) 786–802.
[61] O. Ronneberger, P. Fischer, T. Brox, U-Net: Convolutional networks for biomedical image segmentation, in: MICCAI, 2015, pp. 234–241.
[62] M. Jaderberg, K. Simonyan, A. Zisserman, et al., Spatial transformer networks, in: Advances in Neural Information Processing Systems, 2015, pp. 2017–2025.
[63] X. Jia, J. Bartlett, T. Zhang, W. Lu, Z. Qiu, J. Duan, U-net vs transformer: Is u-net outdated in medical image registration? arXiv preprint arXiv:2208.04939 (2022).
[64] J. Chen, E.C. Frey, Y. He, W.P. Segars, Y. Li, Y. Du, Transmorph: Transformer for unsupervised medical image registration, arXiv preprint arXiv:2111.10480 (2021).
[65] J. Caballero, C. Ledig, A. Aitken, A. Acosta, J. Totz, Z. Wang, W. Shi, Real-time video super-resolution with spatio-temporal networks and motion compensation, in: CVPR, 2017, pp. 4778–4787.
[66] A. Thorley, X. Jia, H.J. Chang, B. Liu, K. Bunting, V. Stoll, A. de Marvao, D.P. O'Regan, G. Gkoutos, D. Kotecha, et al., Nesterov accelerated admm for fast diffeomorphic image registration, in: MICCAI, 2021, pp. 150–160.
[67] J. Wu, X. Tang, Brain segmentation based on multi-atlas and diffeomorphism guided 3d fully convolutional network ensembles, Pattern Recognition 115 (2021) 107904.
[68] C. Van Loan, Computational frameworks for the fast Fourier transform, SIAM, 1992.
[69] B. He, X. Yuan, Convergence analysis of primal-dual algorithms for a saddle–point problem: from contraction perspective, SIAM Journal on Imaging Sciences 5 (1) (2012) 119–149.
[70] T. Goldstein, M. Li, X. Yuan, E. Esser, R. Baraniuk, Adaptive primal-dual hybrid gradient methods for saddle-point problems, arXiv preprint arXiv:1305.0546 (2015).
[71] X. Jia, A. Thorley, W. Chen, H. Qiu, L. Shen, I.B. Styles, H.J. Chang, A. Leonardis, A. De Marvao, D.P. O'Regan, et al., Learning a model-driven variational network for deformable image registration, IEEE Transactions on Medical Imaging 41 (1) (2021) 199–212.

Jinming Duan is now a Turing Fellow at Alan Turing Institute and an Assistant Professor within Computer Science at University of Birmingham. He is also a Fellow of the Higher Education Academy (FHEA) under the UK Professional Standards Framework for teaching and learning support in higher education. He was a Research Associate jointly within Department of Computing & Institute of Clinical Sciences at Imperial College London. There, he has been developed cutting-edge machine learning methods for cardiovascular imaging problems. Prior to that, he acquired his PhD degree in Computer Science at University of Nottingham. His PhD was funded by Engineering and Physical Sciences Research Council. Jinming's research includes deep neural nets, variational methods, partial/ordinary differential equations, numerical optimisation, and finite difference/element methods, with applications to image processing, computer vision and medical imaging analysis.