# Diffeomorphic Variational Models and Their Fast Algorithms for Image Registration Problems

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

**Daoping Zhang**
**Supervised by Prof. Ke Chen**

August 2019

# Contents

**Acknowledgements** vii

**Abstract** ix

**Publications and Presentations** xi

**1 Introduction** 1
- 1.1 Introduction to Image Registration . . . 1
- 1.2 Applications of Image Registration . . . 2
- 1.3 Outline of This Thesis . . . 3

**2 Mathematical Preliminaries** 5
- 2.1 Normed Linear Spaces . . . 5
- 2.2 The Calculus of Variations . . . 9
  - 2.2.1 First Variation and Euler-Lagrange Equation . . . 10
  - 2.2.2 The Direct Method . . . 12
- 2.3 Inverse Problems and Regularization . . . 13
- 2.4 Discretization and Notation . . . 14
  - 2.4.1 Discrete Schemes . . . 15
    - Cell-Centered Discretization . . . 15
    - Staggered Discretization . . . 15
    - Nodal Discretization . . . 16
  - 2.4.2 Difference Schemes . . . 16
  - 2.4.3 Matrix Notation . . . 17
- 2.5 Iterative Methods for Solving Linear Systems . . . 18
  - 2.5.1 Conjugate Gradient Method . . . 18
  - 2.5.2 Minimal Residual Method . . . 21
- 2.6 Optimization Methods . . . 22
  - 2.6.1 Unconstrained Optimization Methods . . . 22
    - First-Order Necessary Condition . . . 22
    - Armijo Condition . . . 23
    - The Steepest Descent Method . . . 24
    - Newton Method . . . 25
    - Quasi-Newton Method . . . 25
    - Gauss-Newton Method . . . 28
  - 2.6.2 Constrained Optimization Methods . . . 29
    - First-Order Necessary Condition . . . 29
    - Quadratic Penalty Method . . . 30

Augmented Lagrangian Method . . . 31
Alternating Direction Method of Multipliers . . . 32
Bregman Iteration . . . 33

**3 Mathematical Models for Image Registration 35**
3.1 Variational Framework for Image Registration . . . 35
3.2 Fidelity Terms . . . 36
3.2.1 Sum of Squared Differences (SSD) . . . 36
3.2.2 Normalized Cross Correlation (NCC) . . . 36
3.2.3 Normalized Gradient Fields (NGF) . . . 36
3.2.4 Mutual Information (MI) . . . 37
3.3 Regularizations . . . 37
3.3.1 Linear Elastic Regularizer . . . 38
3.3.2 Diffusion Regularizer . . . 38
3.3.3 Hyperelastic Regularizer . . . 38
3.3.4 Fluid Regularizer . . . 39
3.3.5 Total Variation Regularizer . . . 39
3.3.6 Linear Curvature Regularizer . . . 40
3.3.7 Henn and Witsch's Curvature Regularizer . . . 40
3.3.8 Mean Curvature Regularizer . . . 40
3.3.9 Gaussian Curvature Regularizer . . . 40
3.4 General Solution Framework . . . 41
3.4.1 First-Optimize-Then-Discretize . . . 41
3.4.2 First-Discretize-Then-Optimize . . . 42
3.5 Cubic Spline Interpolation . . . 43
3.6 Multilevel Strategy . . . 45
3.7 Conclusion . . . 46

**4 A Novel Diffeomorphic Model for Image Registration and Its Algorithm 49**
4.1 Introduction . . . 49
4.2 Diffeomorphic Transformation . . . 50
4.3 The Proposed Image Registration Model . . . 54
4.3.1 New Regularizer . . . 55
4.3.2 The Proposed Model . . . 55
4.4 The Numerical Algorithm . . . 56
4.4.1 Discretization . . . 57
Discretization of Term 1 in (4.19) . . . 57
Discretization of Term 2 in (4.19) . . . 58
Discretization of Term 3 in (4.19) . . . 58
4.4.2 Optimization Method for The Discretized Problem (4.30) . . . 60
Approximated Hessian $\hat{H}$ . . . 60
Search Direction . . . 62
Step Length . . . 62
Stopping Criteria . . . 63
Multi-Level Strategy . . . 66
4.5 Numerical Results . . . 66
4.5.1 Example 1 — Improvement over The Diffusion Model . . . 68
4.5.2 Example 2 — Test of Large Deformation and Comparison of Models 68
4.5.3 Example 3 — Comparison of Models for A Challenging Test . . . 72

4.5.4 Example 4 — Comparison of The New Model with Other Models . 73
4.6 Conclusion . . . 76
4.7 Appendix . . . 77
4.7.1 Computation of Matrices $A$ and $G$ in (4.26) . . . 77
4.7.2 Computation of The Vector $\vec{\boldsymbol{r}}(U)$ in (4.29) . . . 77
4.7.3 Computing The Gradient and Approximated Hessian of The Term (4.35) . . . 78

**5 Improved Optimization Methods for Image Registration Problems 81**
5.1 Introduction . . . 81
5.2 Optimization Methods . . . 82
5.2.1 Multilevel Gauss-Newton Method . . . 82
5.2.2 Two-Step Gauss-Newton Method . . . 84
5.2.3 Convergence Analysis . . . 86
5.2.4 Subspace Multilevel Technique . . . 91
5.3 Numerical Experiments . . . 92
5.4 Conclusion . . . 108

**6 An Efficient Iterative Algorithm for A Beltrami Based 3D Diffeomorphic Model 109**
6.1 Introduction . . . 109
6.2 Related Works and New Regularizers for 3D Image Registration . . . 110
6.2.1 Related Works . . . 110
6.2.2 Motivation and New Regularizers for 3D Image Registration . . . 111
Properties of 2D Regularizer $|\mu(f)|^2$ . . . 112
New Regularizers for 3D Image Registration . . . 112
6.3 New Models for 3D Image Registration and Their Numerical Algorithms . 116
6.3.1 New Models for 3D Image Registration . . . 116
6.3.2 Mathematical Analysis of The Proposed Models . . . 117
6.3.3 The Numerical Algorithm . . . 118
Discretization . . . 118
Optimization Method . . . 122
6.4 Numerical Experiments . . . 125
6.4.1 Example 1 . . . 126
6.4.2 Test of The Convergence and Parameters' Sensitivity . . . 128
6.4.3 Example 2 . . . 131
6.5 Conclusion . . . 132
6.6 Appendix . . . 134
6.6.1 Computation of $A$ in (6.19) . . . 134
6.6.2 Computation of $M_1$, $M_2$ and $M_3$ in (6.20) . . . 134
6.6.3 Computation of The Matrix-Vector Product $\hat{H}v$ . . . 135
6.6.4 The Diagonal of $\hat{H}$ . . . 137

**7 Conclusion and Future Research 139**
7.1 Conclusion . . . 139
7.2 Future Research . . . 140
3D 'Beltrami-like' Coefficient. . . . 140
Multimodality Image Registration . . . 140
Regularizers . . . 141
ADMM . . . 142
Multigrid Method. . . . 143

Deep Learning . . . . . 144

**Bibliography** **145**

# Acknowledgements

Firstly, I would like to thank my Ph.D. supervisor, Prof. Ke Chen. During my four years doctoral period at the University of Liverpool, Prof. Ke Chen provides his all supports and advice on my research and even daily life. It is fortunate for me to have this precious opportunity to work with Prof. Ke Chen.

I also would like to thank Prof. Bo Yu from Dalian University of Technology, China. Without his help and recommendation, I would never have the chance to go abroad to further my education. From Prof. Bo Yu's serious and rigorous character, I know that what quality an excellent researcher should possess.

Further, I would like to thank my colleagues, Prof. Jinyun Yuan, Dr. Geovani Nunes Grapiglia, Dr. Jack Spencer, Dr. Anis Theljani, Dr. Jianping Zhang, Dr. Xiaoliang Song, Mr. Michael Roberts, Mr. Abdul Jumaat and Mr. Anthony Thompson for their helpful and valuable discussions.

Moreover, I would like to thank my parents, Xuexiong Zhang and Weiqing Qu. No matter what difficulties I have encountered, they always support and encourage me silently and patiently. It is my great honor to be their son!

Finally, I would like to thank China Scholarship Council, the University of Liverpool, Chinese Service Center for Scholarly Exchange and Consulate-General of The People's Republic of China in Manchester for all supports throughout my doctoral studies.

# Abstract

Image registration aims to find an optimal geometric transformation to overlay two or more images of the same scene which are taken at different times, from different perspectives or various sensors. As one of the most fundamental tasks in image processing, image registration has a wide range of applications in computer vision, biological imaging, remote sensing, and medical imaging.

In general, image registration involves two images, where one called template is kept transformed, and the other one called reference is kept unchanged. To measure the similarity between the deformed template and the reference, a fidelity term should be chosen. However, only minimizing/maximizing the fidelity term is ill-posed in the sense of Hadamard since it is not sufficient to ensure the uniqueness and continuity of the solution. To overcome this difficulty, regularization is indispensable. Hence, in this thesis, we mainly consider the variational framework.

Nowadays, there exist many kinds of different regularizers. Unfortunately, when we solve the variational model for image registration, although we can obtain a visually satisfied deformed template, the corresponding transformation often has foldings and is not physically correct. Hence, over the last decade, more and more researchers have focused on diffeomorphic image registration, whose aim is to find an accurate diffeomorphic mapping, namely the transformation is continuously differentiable, and it has a continuously differentiable inverse. Here, we consider choosing the first-discretize-then-optimize method to solve the variational model for image registration, namely, first directly discretize the variational model and obtain a finite dimensional optimization problem then choose suitable optimization methods to solve the resulting optimization problem. However, in image registration, the number of variables is usually huge, and the dimension of the resulting optimization problem is huge as well. For example, when the size of the given images is $128 \times 64 \times 128$, the number of unknowns is over 3 million. Hence, designing an efficient and converging solver is also an important issue.

This thesis can be mainly classified into two parts: one is how to design the diffeomorphic registration model leading to a diffeomorphic transformation, and the other one is how to develop highly efficient and effective solvers.

In the first part, we first propose a new variational model with a special regularizer, based on the quasi-conformal theory, which can guarantee that the registration map is diffeomorphic. Also, since the Beltrami coefficient uses complex analysis and is only defined in 2D space, we further present two new formulations in 3D space motivated by the Beltrami concept and then set our new registration models. We propose a converging Gauss-Newton iterative method to solve the resulting nonlinear optimization problems and prove its convergence. Numerical experiments can demonstrate that the new 2D and 3D models can not only get diffeomorphic registrations even when the deformations are large, but also possess the accuracy in comparing with the state-of-the-art models.

In the second part, we consider using the subspace method to accelerate the algorithm. Here, we propose two simple techniques to improve the performance of the standard multilevel Gauss-Newton algorithm. The first technique consists of the possible use of a second step within each iteration of the Gauss-Newton method. This step is computed by minimizing a quadratic approximation of the objective function over a two-dimensional subspace. This subspace is spanned by the steepest descent direction and the L-BFGS direction concerning the current point given by the Gauss-Newton step. The second technique is a modification of the standard coarse-to-fine multilevel strategy. At each level, instead of using the interpolated solution of the previous level directly as the initial point, we try to find a better initial point by minimizing a quadratic approximation of the objective function over the subspace spanned by the interpolated solutions of all the previous levels. Numerical experiments illustrate that the subspace method can significantly reduce the running time compared with the standard multilevel Gauss-Newton method.

Overall, this thesis is concerned with the diffeomorphic variational models and their fast algorithms for image registration problems.

# Publications and Presentations

**Publications**

- **D Zhang** and K Chen. An Efficient Iterative Algorithm for A Generalized Beltrami Based 3D Diffeomorphic Model. In Preparation.
- **D Zhang**, A Theljani and K Chen. Springer Book Chapter: A Gauss-Newton Solver for New Diffeomorphic Multi-modality Registration Model in Image Processing and Inverse Problems Eds. H Liu, S Wei ,and X Tai (2020).
- **D Zhang**, K Chen. A Novel Diffeomorphic Model for Image Registration and Its Algorithm. Journal of Mathematical Imaging and Vision. 2018, 60(8), pp.1261-1283.
- K Chen, G Grapiglia, J Yuan, **D Zhang**. Improved Optimization Methods for Image Registration Problems. Numerical Algorithms. 2019, 80(2), pp. 305-336.

**Presentations**

- 2D and 3D Diffeomorphic Models for Image Registration. 2019.4 at Strathclyde SIAM-IMA Student Chapter Conference: Applications of Mathematics to Industrially Relevant Problems, Strathclyde, UK.
- Variational Diffeomorphic Models for Image Registration. 2018.6 at SIAM National Student Chapter Conference, Bath, UK.
- Diffeomorphic Image Registration Models by New Constraints. 2018.6 at SIAM Conference on Imaging Science, Bologna, Italy.
- An Overview of Image Analysis Work: Image Enhancement, Segmentation, and Registration. 2018.5 at Histology, Pathology & Digital Health Workshop, Liverpool, UK.
- Diffeomorphic Image Registration Models by New Constraints. 2018.5 at Developments in Healthcare Imaging-Connecting with Academia, Cambridge, UK.
- Variational Models for Diffeomorphic Image Registration. 2017.9 at Further Engagement with Healthcare Sector, Liverpool, UK.

- Two Novel Diffeomorphic Models for Image Registration. 2017.7 at Optimisation & Imaging Workshop, Liverpool, UK.
- On Diffeomorphic Image Registration Models. 2016.5 at SIAM Conference on Imaging Science, Albuquerque, USA.

**Posters**

- Converging Algorithms for Image Registration Problems. 2019.4 at Numerical Algorithms for High-Performance Computational Science, London, UK.
- A Novel Diffeomorphic Model for Image Registration. 2016.12 at Faculty of Science and Engineering Poster Day, Liverpool, UK

# Illustrations

## List of Figures


1.1 Illustration of image registration. . . . . 1

2.1 An example of the convex set (left) and the nonconvex set (right). . . . . 9
2.2 Cell-centered discretization of a square domain. . . . . 15
2.3 Staggered discretization of a square domain. . . . . 16
2.4 Nodal discretization of a square domain. . . . . 16
2.5 NP and ICFP represent no preconditioner and incomplete Cholesky factorization preconditioner respectively. . . . . 20
2.6 An example of the global minimizer (left) and the local minimizer(right). . . 23

3.1 An example of a cubic spline interpolation with natural boundary conditions. 44
3.2 B-spline of Order 4 (cubic B-spline). . . . . 45
3.3 An example of the multilevel representation of a pair of 2D images. . . . . 46

4.1 Partition of domain $\Omega = \cup_{ij}\Omega_{i,j}$. Note that solutions $u_1$ and $u_2$ are defined at nodes. . . . . 57
4.2 Partition of a cell, nodal point $\square$ and center point $\circ$. $\triangle V_1V_2V_5$ is $\Omega_{i,j,k}$. . . . 59
4.3 Example 1 results of Hand to Hand registration ($\alpha = 2$): in the top row, there are the template and reference. In the second row, there are the deformed templates obtained by model (4.19) and the diffusion model separately. Though the last column is visually fine, the transformation is not correct – see Table 4.1. . . . . 69
4.4 Zooming in the transformation (obtained by the diffusion model) where there is folding. . . . . 69
4.5 Example 2 results of Disc to C. The percentage value shows Re_SSD error. In the top row, there are the template and reference. In the second and third row, there are the deformed templates obtained by New 1-3 and 5 other models separately. The landmarks in the template and reference are only used for QCHR and the last result (j) by Diffusion is evidently not correct. . 71
4.6 Example 2 Relative Residual of New 3 and QCHR: The solid line indicates the relative residual of New 3. And the dotted line shows the relative residual of the second subproblem in QCHR. Here, we can find that in the same 50 iterations, the relative residual of New 3 is decreasing to below $10^{-2}$. However, the relative residual of QCHR is not decreasing and over 0.1. Hence, the convergence of the algorithm for QCHR cannot be guaranteed. . . . . 72

4.7 Example 3 results of a large Disc to a small letter C: in the top row, there are the template and reference. In the second and third row, there are the deformed templates obtained by model (4.19) and other models separately. The landmarks in the template and reference are only used for QCHR. . . . 74
4.8 Tests of QCHR with different landmarks: Example 2 (row 1) and Example 3 (row 2). On the left 3 columns of row 3, we show the registered templates for row 1. On the right 3 columns of row 3, we show the registered templates for row 2. Here, we can see that the accuracy of QCHR improves with the increase of landmarks. . . . . 75
4.9 Example 4 results of a pair of CT images: the template $T$ and the reference $R$ in the top row. The contours show the regions of interest. In the second and third rows, we show the deformed templates obtained by 8 models. The 5 landmarks in the template and the reference are only used by QCHR. . . 76

5.1 Problem hand. . . . . 92
5.2 Problem EPslice. . . . . 93
5.3 Problem brain. . . . . 93
5.4 Problem CT. . . . . 93
5.5 Problem MRI. . . . . 94
5.6 Problem lung. . . . . 94
5.7 Problem CT1. . . . . 94
5.8 Problem CT2. . . . . 95
5.9 Problem MRI2. . . . . 95
5.10 Problem breast. . . . . 95
5.11 Problem circle to c. . . . . 96
5.12 Problem c to circle. . . . . 96
5.13 Problem A to R. . . . . 96
5.14 Problem square to square. . . . . 97
5.15 Problem Lena. . . . . 97
5.16 Problem circle to square. . . . . 97
5.17 Problem molecule. . . . . 98
5.18 Problem F to F. . . . . 98
5.19 Problem circle to I. . . . . 98
5.20 Problem Rio. . . . . 99
5.21 Performance Profile based on CPU Time for the set of 20 problems with medical images and resolutions of $128 \times 128$ and $256 \times 256$. The red line and black line represents HYBRID and GN respectively. . . . . 103
5.22 Performance Profile based on Number of Iterations for the set of 20 problems with medical images and resolutions of $128 \times 128$, $256 \times 256$ and $512 \times 512$. The red line and black line represents HYBRID and GN respectively. . . . . 104

5.23 Performance Profile based on Number of Function Evaluations for the set of 20 problems with medical images and resolutions of $128 \times 128$, $256 \times 256$ and $512 \times 512$. The red line and black line represents HYBRID and GN respectively. . . . . 105
5.24 Registered images for problem MRI2 with resolution $512 \times 512$. . . . . 106
5.25 3D brain problem. . . . . 107
5.26 Template and Reference for the 3D brain problem. . . . . 107

6.1 Illustration of the distortion and dilatation in 3D space. Here, we can note that $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are orientation-preserving but $\boldsymbol{f}_3$ is not orientation-preserving (from the color). . . . . 115
6.2 Partition of the domain $\Omega$. Nodal grid $\square$ and cell-centered grid $\times$. . . . . 119
6.3 Partition of a voxel. $V_1, ..., V_8$ are vertices. . . . . 120
6.4 The structure of the preconditioner $L$. $L$ is composed of the diagonals of blocks of the approximated Hessian $\hat{H}$. The dimension of the problem is 14739 ($3 \times 17 \times 17 \times 17$). . . . . 124
6.5 Example 1: the left column and the right column show the template and the reference respectively. . . . . 127
6.6 The results of Example 1: the top row shows the deformed templates obtained by NM1 (left) and NM2 (right). The second row shows the deformed templates obtained by NM3 (left) and Hyper1 (right). The bottom row shows the deformed template obtained by Hyper2 (left) and LDDMM (right). The percentage represents the relative error. . . . . 128
6.7 The performance of different solvers with different preconditioners for NM1, NM2 and NM3 in Example 1. Here, MINRES, MINRESD and MINRESL represent that the solver is MINRES without preconditioner, with diagonal preconditioner and with $L$ preconditioner respectively. And CG, CGD and CGL represent that the solver is conjugate gradient method without preconditioner, with diagonal preconditioner and with $L$ preconditioner respectively. 129
6.8 The relative norm of the gradient and relative function values of NM1, NM2 and NM3 in Example1. . . . . 130
6.9 Parameters' sensitivity test of NM1, NM2 and NM3 (with regard to diffeomorphism, using Example 1). Here, for the three models, they all generate diffeomorphic transformations in the specific parameter domain. Hence, they are robust with respect to the diffeomorphism. . . . . 130
6.10 Test of sensitivity of relative residuals of NM1, NM2 and NM3 in their respective feasible regions from Figure 6.9. Here, we can observe that the relative residual generated by NM2 is not sensitive with respect to the parameters. However, for NM1 and NM3, when the parameters change, the relative residual generated by them will change dramatically. Hence, NM2 is the best model in terms of parameters' sensitivity. . . . . 131

6.11 Example 2: the left column and the right column show the template and the reference respectively. . . . . 132
6.12 The results of Example 2: the top row shows the deformed templates obtained by NM1 (left) and NM2 (right). The second row shows the deformed templates obtained by NM3 (left) and Hyper1 (right). The bottom row shows the deformed template obtained by Hyper2 (left) and LDDMM (right). The percentage represents the relative error. . . . . 133

7.1 A pair of MRI images (T1 and T2). The resulting transformation is diffeomorphic and the deformed template is also visually satisfied. . . . . 141
7.2 A pair of MRI images. . . . . 142

## List of Tables

4.1 Example 1 — Comparison of the new model (New 1-3) with the diffusion model based a fixed $\alpha$ and an optimized $\alpha$ for the latter. Clearly the latter model can produce an incorrect result if not tuned while New 1-3 are less sensitive to $\alpha$ with the help of $\beta$. . . . . 69
4.2 Example 2 — Comparison of the new model (New 1-3) with 5 other models. 71
4.3 Example 3 — Comparison of the new model (New 1-3) with 5 other models. 73
4.4 Example 4 — Comparison of the new model (New 1-3) with 5 other models . 75

5.1 Results for resolution $128 \times 128$. . . . . 99
5.2 Results for resolution $256 \times 256$. . . . . 100
5.3 Results for resolution $512 \times 512$. . . . . 100
5.4 Results for resolution $1024 \times 1024$. . . . . 101
5.5 Comparison of the total time (in seconds) to solve all 20 problems for each resolution between GN and SIG. . . . . 101
5.6 Comparison of the total time (in seconds) to solve all 20 problems for each resolution between GN and TS. . . . . 101
5.7 Comparison of the total time (in seconds) to solve all 20 problems for each resolution between GN and HYBRID. . . . . 102
5.8 Comparison of the total time (in seconds) to solve all 10 problems with medical images for each resolution between GN and SIG. . . . . 102
5.9 Comparison of the total time (in seconds) to solve all 10 problems with medical images for each resolution between GN and TS. . . . . 102
5.10 Comparison of the total time (in seconds) to solve all 10 problems with medical images for each resolution between GN and HYBRID. . . . . 102
5.11 Results for 3D problems. . . . . 106
5.12 Results for MGH problems. . . . . 108

6.1 Example 1 — Comparison of the new models with Hyper1, Hyper2 and LDDMM. . . . . 127

6.2 The number of iterations needed to reach the termination for different solver with different preconditioner in Example 1. . . . . 129
6.3 Example 2 — Comparison of the new models with Hyper1, Hyper2 and LDDMM. . . . . 132

7.1 Measurements of 2D Brain example. . . . . 142

# Chapter 1

# Introduction

## 1.1 Introduction to Image Registration

Image registration, also called image matching, image warping or image fusion, is one of the most fundamental tasks in image processing. It aims to find an optimal geometrical transformation to align two or more images of the same scene which are taken at different times, from different perspectives or from different sensors.

To use the mathematical language to express image registration, we first define image by the following continuous model.

**Definition 1.1** (**Image**)**.** Let $d \in \mathbb{N}$. We say that a continuous function $I(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$ is a $d-$dimensional image if $I(\boldsymbol{x})$ is compactly supported in $\Omega \subset \mathbb{R}^d$. Here, each spatial position $\boldsymbol{x}$ is equal to $(x_1, ..., x_d)$.

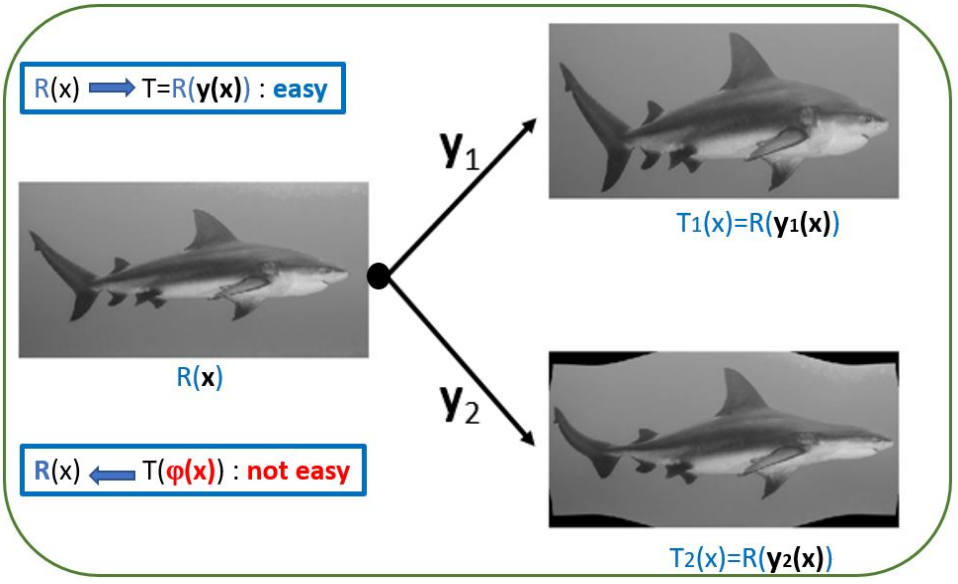



FIGURE 1.1: Illustration of image registration.

Image registration usually involves two images, the so-called template $T(\boldsymbol{x}) : \Omega \subset \mathbb{R}^d \rightarrow \mathbb{R}$ and the so-called reference $R(\boldsymbol{x}) : \Omega \subset \mathbb{R}^d \rightarrow \mathbb{R}$. From Figure 1.1, we can see that if the transformation $\boldsymbol{y} = (y_1(\boldsymbol{x}), ..., y_d(\boldsymbol{x}))$ is given, computing the deformed image is a forward problem but if the template and reference are given, computing the transformation to make the deformed template and reference similar is a backward problem. To generate a plausible transformation, the conventional method is to choose a fidelity term and measure the similarity between the deformed template and the reference. However, only minimizing/maximizing the fidelity term is ill-posed in the sense of Hadamard since it is not sufficient to ensure the uniqueness and continuity of the solution. For instance, if the template is a plain white disc on a plain black background and the reference is the same as the template, then any rotations around the center of the disc are all solutions [110]. To overcome this difficulty, regularization is indispensable. Hence, in this thesis, we mainly consider the variational framework.

All registration models are nonlinear but they can be classified into two main categories according to the way deformation mapping is represented: linear registration and nonlinear registration. In linear registration, the deformation model is linear and global, including rotation, translation, shearing and scaling [21, 91]. Although the computation speed of a linear model is fast since it contains few variables, it is commonly used as the preregistration for starting a more sophisticated model. This is mainly because linear models can not accommodate the local details (differences). In contrast, nonlinear registration models inspired by physical processes of transformations [114] such as the elastic model [9], fluid model [19], diffusion model [38], TV (total variation) model [43], MTV (modified TV) model [22], linear curvature model [39, 40], mean curvature model [24], Gaussian curvature model [71] and total fractional-order variation model [136] are proposed to account for localized variation in details, by allowing many degrees of freedom. The particular free-form deformation models based on B-splines lying between the above two types possess simplicity, smoothness, efficiency and ability to describe local deformation with few degrees of freedom [106, 108, 114]. For more details about image registration methodology, please refer to [10, 41, 67, 85, 91, 114, 138] and references therein.

## 1.2 Applications of Image Registration

As one of the most fundamental tasks in image processing, image registration has a wide range of applications in computer vision, biological imaging, remote sensing, industrial manufacturing, robotic navigation, and medical imaging. For example, in computer vision, we can recover shape from stereo or automatically change detection for security monitoring and in remote sensing, we can monitor the global land usage and landscape planning. Particularly, medical image registration has attracted more and more attention in the last decade since we can fuse different modality images (such as PET/CT) to help diagnose the diseases or monitor the growth of the tumor during the medical

treatment. For more details about the applications of image registration, please refer to [6, 10, 30, 45, 81, 85, 89, 90, 98, 112, 114, 138] and references therein.

## 1.3 Outline of This Thesis

The content of this thesis can be mainly classified into two parts: one is how to design the diffeomorphic registration model leading to a diffeomorphic transformation, and the other one is how to design highly efficient and effective solvers. Specifically, the work of this thesis is organized as follows:

- **Chapter 2** mainly reviews some basic mathematical tools which will be used throughout this thesis. It includes:
  - Basic definitions, theorems, and examples in normed linear spaces.
  - Basic notions about the calculus of variations.
  - An introduction to inverse problems and regularization.
  - A discussion of the discretization and notation on the regular domain with finite difference methods.
  - A brief review of two Krylov subspace methods, conjugate gradient method (CG) and minimal residual method (MINRES), for solving symmetric systems of linear equations.
  - A brief review of the fundamental theory of optimization methods, including unconstrained optimization methods and constrained optimization methods.
- **Chapter 3** reviews the related details in image registration. In includes:
  - The variational framework of image registration.
  - A brief review of the fidelity terms in image registration, including the sum of squared differences, normalized cross correlation, normalized gradient fields, and mutual information.
  - A brief review and discussion of the regularizers in image registration, including linear elastic regularizer, diffusion regularizer, hyperelastic regularizer, fluid regularizer, TV regularizer, linear curvature regularizer, Henn and Witsch's curvature regularizer, mean curvature regularizer, and Gaussian curvature regularizer.
  - A brief discussion of the general solution framework in image registration, including first-optimize-then-discretize and first-discretize-then-optimize.
  - A discussion of the cubic spline interpolation used in image registration.

  - A short introduction about the multilevel strategy in image registration.

- **Chapter 4** presents a novel diffeomorphic model for image registration and its algorithm. It includes:
  - A review of the previous works about diffeomorphic registrations.
  - Details of a new regularizer based on Beltrami coefficient and the proposed new registration model.
  - Details of the numerical implementation, including the discretization and the optimization method.
  - Numerical experiments showing the accuracy of our proposed new registration model.

- **Chapter 5** presents improved optimization methods for image registration problems. It includes:
  - A review and discussion of the standard multilevel Gauss-Newton method.
  - Details of the two-step Gauss-Newton method.
  - Details of the subspace multilevel technique.
  - Numerical experiments showing the effectiveness and efficiency of our proposed methods compared with the standard multilevel Gauss-Newton method.

- **Chapter 6** presents an efficient iterative algorithm for a generalized Beltrami based 3D diffeomorphic model. In includes:
  - A review of the related works about 3D diffeomorphic registration.
  - Details of our new 3D regularizers for diffeomorphic registration and the proposed new registration models.
  - Details of the numerical implementation, including the discretization and the optimization method.
  - Numerical experiments showing the accuracy of our proposed new registration models and the effectiveness of the proposed algorithm.

- **Chapter 7** concludes this thesis and lists some possible future works.

# Chapter 2

# Mathematical Preliminaries

This chapter reviews some necessary mathematical tools which will be used throughout this thesis, including normed linear spaces, the calculus of variations, inverse problems and regularization, discretization and notation, iterative methods for solving linear systems, and optimization methods.

## 2.1 Normed Linear Spaces

Here, we review some basic notions, theorems, and examples in normed linear spaces. For more details, please refer to [26, 68, 78].

**Definition 2.1** (**Field**)**.** Let $\mathbf{K}$ be a subset of the complex numbers $\mathbb{C}$. We say that $\mathbf{K}$ is a field if it satisfies the following conditions:

1. If $x, y \in \mathbf{K}$, $x + y$ and $xy$ are also elements of $\mathbf{K}$.
2. If $x \in \mathbf{K}$, $-x$ is also an element of $\mathbf{K}$. If furthermore $x \neq 0$, $x^{-1}$ is also an element of $\mathbf{K}$.
3. The elements 0 and 1 are elements of $\mathbf{K}$.

**Example 2.1.** *$\mathbb{R}$ and $\mathbb{C}$ are both fields.*

**Definition 2.2** (**Linear Vector Space**)**.** Let $\mathbf{K}$ be a field. We say that $\mathbf{V}$ is a linear vector space over the field $\mathbf{K}$ if it satisfies the following conditions:

1. If $u, v \in \mathbf{V}$, $u + v = v + u$.
2. If $u, v, w \in \mathbf{V}$, $(u + v) + w = u + (v + w)$.
3. There is an element of $\mathbf{V}$, denoted by 0, such that $0 + u = u + 0 = u$ for all elements $u \in \mathbf{V}$.
4. If $u \in \mathbf{V}$, there exists an element $-u \in \mathbf{V}$ such that $u + (-u) = 0$.

5. If $c \in \mathbf{K}$ and $u, v \in \mathbf{V}$, $c(u+v) = cu + cv$.
6. If $a, b \in \mathbf{K}$ and $u \in \mathbf{V}$, $(a+b)u = au + bu$.
7. If $a, b \in \mathbf{K}$ and $u \in \mathbf{V}$, $(ab)u = a(bu)$.
8. If $u \in \mathbf{V}$, we have $1 \cdot u = u$ (here 1 is the number one).

We can notice that a linear vector space $\mathbf{V}$ over the field $\mathbf{K}$ is a set of objects which can be added and multiplied by elements of $\mathbf{K}$. In other words, the sum of two elements of $\mathbf{V}$ is also an element of $\mathbf{V}$ and the product of an element of $\mathbf{V}$ by an element of $\mathbf{K}$ is also an element of $\mathbf{V}$.

**Example 2.2.** *Let $\boldsymbol{V} = \boldsymbol{K}^n$ be the set of $n-$tuples of elements of $\boldsymbol{K}$. Then $\boldsymbol{V}$ is a linear vector space over the field $\boldsymbol{K}$.*

**Definition 2.3** (**Linear Subspace**)**.** Let $\mathbf{V}$ be a vector space over the field $\mathbf{K}$ and $\mathbf{W}$ be a subset of $\mathbf{V}$. We say that $\mathbf{W}$ is a linear subspace of $\mathbf{V}$ if it satisfies the following conditions:

1. If $v, w \in \mathbf{W}$, $v + w$ is also an element of $\mathbf{W}$.
2. If $v \in \mathbf{W}$ and $c \in \mathbf{K}$, $cv$ is also an element of $\mathbf{W}$.
3. The element 0 of $\mathbf{V}$ is also an element of $\mathbf{W}$.

**Example 2.3.** *Let $\boldsymbol{V} = \boldsymbol{K}^n$ and $\boldsymbol{W}$ be the set of vectors in $\boldsymbol{V}$ whose last coordinate is equal to 0. Then $\boldsymbol{W}$ is a linear subspace of $\boldsymbol{V}$, which can be identified with $\boldsymbol{K}^{n-1}$.*

**Definition 2.4** (**Norm**)**.** Let $\mathbf{V}$ be a vector space over the field $\mathbf{K}$. We say that a nonnegative-valued scalar function $\|\cdot\|$ is a norm on $\mathbf{V}$ if it satisfies the following conditions:

1. $\|u\| = 0$ if and only if $u = 0 \in \mathbf{V}$.
2. $\|\lambda u\| = |\lambda| \|u\|$, for all $\lambda \in \mathbf{K}$ and all $u \in \mathbf{V}$ .
3. $\|u + v\| \leq \|u\| + \|v\|$, for all $u, v \in \mathbf{V}$.

**Definition 2.5** (**Seminorm**)**.** Let $\mathbf{V}$ be a vector space over the field $\mathbf{K}$. We say that a nonnegative-valued scalar function $\|\cdot\|$ is a seminorm on $\mathbf{V}$ if it satisfies the following conditions:

1. $\|\lambda u\| = |\lambda| \|u\|$, for all $\lambda \in \mathbf{K}$ and all $u \in \mathbf{V}$ .
2. $\|u + v\| \leq \|u\| + \|v\|$, for all $u, v \in \mathbf{V}$.

**Example 2.4.** *Let $x = (x_1, ..., x_n)$ be a vector. Then well-known examples of the vector norm are as follows:*

$$\|x\|_\infty = \max_{1\le i\le n} |x_i|, \quad (l^\infty - \text{norm}) \tag{2.1}$$

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|, \quad (l^1 - \text{norm}) \tag{2.2}$$

$$\|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2\right)^{1/2}, \quad (l^2 - \text{norm}). \tag{2.3}$$

*The above examples are special cases of $l^p-$norm which is defined by*

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}, \quad (l^p - \text{norm}). \tag{2.4}$$

**Example 2.5** ($L^p-$norm)**.** *Let $f$ be a function defined on a domain $\Omega$ and $1 \le p \le \infty$. The $L^p-$norm of $f$ on $\Omega$ is defined by*

$$\|f\|_{L^p} = \left(\int_\Omega |f(x)|^p \mathrm{d}x\right)^{1/p}. \tag{2.5}$$

*This is a generalization of the previous example. The special case when $p = \infty$ is defined by*

$$\|f\|_{L^\infty} = \sup_x |f(x)|. \tag{2.6}$$

**Example 2.6.** *Let $\boldsymbol{V}$ be a vector space over the field $\boldsymbol{K}$. $\|u\| = 0$ for all $u \in \boldsymbol{V}$ is a trivial seminorm.*

**Definition 2.6** (**Normed Linear Space**)**.** A normed linear space is a pair $(\mathbf{V}, \|\cdot\|)$ where $\mathbf{V}$ is a vector space over the field $\mathbf{K}$ and $\|\cdot\|$ is a norm on $\mathbf{V}$.

**Definition 2.7** (**Inner Product**)**.** Let $\mathbf{V}$ be a vector space over the field $\mathbf{K}$. We say that a function $\langle\cdot,\cdot\rangle : \mathbf{V} \times \mathbf{V} \to \mathbf{K}$ is an inner product on $\mathbf{V}$ if it satisfies the following conditions:

1. $\langle u, u\rangle \ge 0$, for all $u \in \mathbf{V}$.
2. $\langle u, u\rangle = 0$ if and only if $u = 0 \in \mathbf{V}$.
3. $\langle u + v, w\rangle = \langle u, w\rangle + \langle v, w\rangle$, for all $u, v, w \in \mathbf{V}$.
4. $\langle \lambda u, v\rangle = \lambda\langle u, v\rangle$, for all $\lambda \in \mathbf{K}$ and all $u, v \in \mathbf{V}$.
5. $\langle u, v\rangle = \overline{\langle v, u\rangle}$, for all $u, v \in \mathbf{V}$.

**Example 2.7.** *Two examples of the inner product are as follows:*

- *The Euclidean inner product on* $\boldsymbol{K}^n$ *is defined by*

$$\langle (w_1, ..., w_n), (z_1, ..., z_n) \rangle = w_1 \overline{z_1} + \cdots + w_n \overline{z_n}. \tag{2.7}$$

- *An inner product on the vector space of continuous real-valued functions on the interval* $[-1, 1]$ *can be defined by*

$$\langle f, g \rangle = \int_{-1}^{1} f(x) g(x) \mathrm{d}x. \tag{2.8}$$

**Definition 2.8** (**Cauchy Sequence and Completeness**)**.** We say that a sequence $\{x_n\}_{n=1}^{\infty}$ in a normed linear space $\mathbf{V}$ is a Cauchy sequence if given any $\epsilon > 0$, there exists $N_0 \in \mathbb{N}$ such that

$$|x_n - x_m| < \epsilon, \text{ for all } n, m > N_0. \tag{2.9}$$

In addition, we say that a normed linear space $\mathbf{V}$ is complete if every Cauchy sequence in $\mathbf{V}$ converges to an element in $\mathbf{V}$.

**Example 2.8.** $\boldsymbol{V} = [a, b]$ *is complete but* $\boldsymbol{V} = (a, b)$ *is not complete.*

**Definition 2.9** (**Banach Space and Hilbert Space**)**.** A complete normed linear space is called a Banach space. A complete inner product space concerning the norm induced by the inner product is called a Hilbert space.

**Example 2.9.** *The space* $L^2(\Omega)$ *with the inner product defined by*

$$\langle f, g \rangle_{L^2(\Omega)} = \int_{\Omega} f(x) g(x) \mathrm{d}x. \tag{2.10}$$

*is a Hilbert space. It is also a Banach space.*

**Definition 2.10** (**Linear Mapping**)**.** Let $\mathbf{V}$ and $\mathbf{W}$ be vector spaces over the same field $\mathbf{K}$. We say that a mapping $\mathcal{L} : \mathbf{V} \to \mathbf{W}$ is linear if it satisfies the following conditions:

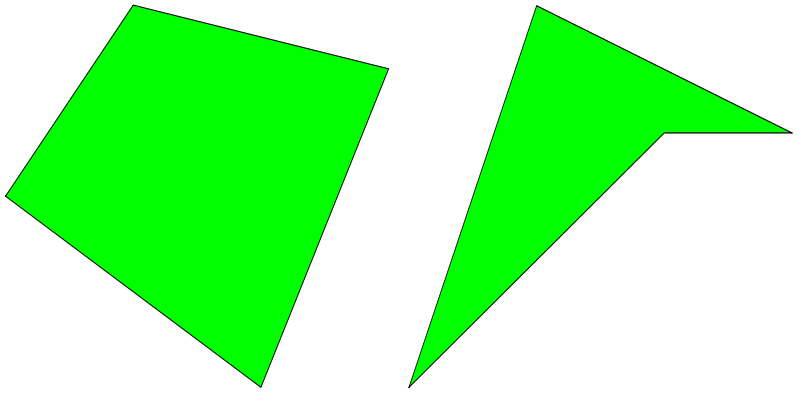
1. $\mathcal{L}(u + v) = \mathcal{L}(u) + \mathcal{L}(v)$, for all $u, v \in \mathbf{V}$.
2. $\mathcal{L}(\lambda u) = \lambda \mathcal{L}(u)$, for all $\lambda \in \mathbf{K}$ and all $u \in \mathbf{V}$.

**Example 2.10.** *Let* $\boldsymbol{V} = \mathbb{R}^3$ *and* $\boldsymbol{W} = \mathbb{R}^2$*. Define a projective mapping* $\mathcal{L} : \boldsymbol{V} \to \boldsymbol{W}$*, namely* $\mathcal{L}(x, y, z) = (x, y)$*. Then the mapping* $\mathcal{L}$ *is a linear mapping.*

**Definition 2.11** (**Convex Set**)**.** Let $S$ be a set. We say that $S$ is a convex set if it satisfies the following condition:

$$\lambda u + (1 - \lambda) v \in S, \text{ for all } \lambda \in [0, 1] \text{ and all } u, v \in S. \tag{2.11}$$

Figure 2.1 shows an example of the convex set and nonconvex set.



(a) A convex set (b) A nonconvex set

FIGURE 2.1: An example of the convex set (left) and the nonconvex set (right).

**Definition 2.12** (**Convex Function and Strict Convex Function**)**.** Let $S$ be a nonempty convex set. We say that $f : S \to \mathbb{R}$ is convex on $S$ if it satisfies the following condition:

$$f(\lambda u + (1-\lambda)v) \leq \lambda f(u) + (1-\lambda)f(v), \text{ for all } \lambda \in (0,1) \text{ and all } u, v \in S. \quad (2.12)$$

In addition, we say that $f : S \to \mathbb{R}$ is strictly convex on $S$ if it satisfies the following condition:

$$f(\lambda u + (1-\lambda)v) < \lambda f(u) + (1-\lambda)f(v), \text{ for all } \lambda \in (0,1) \text{ and all } u \neq v \in S. \quad (2.13)$$

**Example 2.11.** *Let $S$ be $[0, \pi]$. Then $f(x) = x^2$ is convex and strictly convex on $S$. However, $f(x) = sin(x)$ is not convex on $S$.*

**Example 2.12.** *Let $S \subset \mathbb{R}^n$ be a nonempty subset. The indicator function $I_S : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is defined by*

$$I_S(x) := \begin{cases} 0 & \text{if } x \in S, \\ +\infty & \text{otherwise.} \end{cases} \quad (2.14)$$

*Obviously, $I_S$ is convex if and only if $S$ is convex.*

## 2.2 The Calculus of Variations

A functional is a mapping from a vector space of functions into the real numbers. The calculus of variations is concerned with finding maxima and minima of functionals. In this section, we introduce the basic notions about the calculus of variations, such as first variation, Euler-Lagrange equation, and the direct method to prove the existence. For more details, please refer to [27, 28, 42, 123].

### 2.2.1 First Variation and Euler-Lagrange Equation

Firstly, we define the local extrema for the functional.

**Definition 2.13** (**Neighborhood**). The neighborhood of $u \in \mathcal{U}$ is defined by

$$\mathcal{N}_\epsilon(u) = \{\hat{u} \in \mathcal{U} | \|u - \hat{u}\| \leq \epsilon\}. \tag{2.15}$$

**Definition 2.14** (**Local Extrema**). Let $\mathcal{J} : \mathcal{U} \to \mathbb{R}$ be a functional defined on the function space $(\mathcal{U}, \|\cdot\|)$. We say that $\mathcal{J}$ has a local maximum at $u \in \mathcal{U}$ if the following condition is satisfied:

$$\mathcal{J}(\hat{u}) \leq \mathcal{J}(u) \text{ for all } \hat{u} \in \mathcal{N}_\epsilon(u). \tag{2.16}$$

We say that $\mathcal{J}$ has a local minimum at $u \in \mathcal{U}$ if $u \in \mathcal{U}$ is a local maximum for $-\mathcal{J}$. Often, $\mathcal{U}$ is a set of functions with certain boundary conditions.

Next, we illustrate the first variation and the Euler-Lagrange equation through a particular class of problems. Let $\mathcal{J} : C^2[x_0, x_1] \to \mathbb{R}$ be a functional defined by

$$\mathcal{J}(u) = \int_{x_0}^{x_1} f(x, u, u')\mathrm{d}x, \tag{2.17}$$

where $f$ is a function whose partial derivatives with respect to $x, u$ and $u'$ are second-order continuous. In addition, let $u_0, u_1$ be real numbers and $u(x_0) = u_0$, $u(x_1) = u_1$. Then

$$\mathcal{U} = \{u \in C^2[x_0, x_1] : u(x_0) = u_0 \text{ and } u(x_1) = u_1\}. \tag{2.18}$$

For simplifying the analysis, we set

$$\mathcal{H} = \{\eta \in C^2[x_0, x_1] : \eta(x_0) = \eta(x_1) = 0\}. \tag{2.19}$$

Assume that $\mathcal{J}$ has a local maximum at $u$. Then there exists $\epsilon > 0$ such that $\mathcal{J}(\hat{u}) \leq \mathcal{J}(u)$ for all $\hat{u} \in \mathcal{N}_\epsilon(u)$. For any $\hat{u} \in \mathcal{U}$ there is an $\eta \in \mathcal{H}$ such that $\hat{u} = u + \epsilon\eta$. For small $\epsilon$, Taylor's theorem implies that

$$\begin{aligned} f(x, \hat{u}, \hat{u}') &= f(x, u + \epsilon\eta, u' + \epsilon\eta') \\ &= f(x, u, u') + \epsilon\left(\eta\frac{\partial f}{\partial u} + \eta'\frac{\partial f}{\partial u'}\right) + O(\epsilon^2). \end{aligned} \tag{2.20}$$

Here, we regard $f$ as a function of the three independent variables $x, u$ and $u'$ and the partial derivatives in the above expression are all evaluated at the point $x, u$ and $u'$. Then we have

$$\begin{aligned} \mathcal{J}(\hat{u}) - \mathcal{J}(u) &= \int_{x_0}^{x_1} f(x, \hat{u}, \hat{u}')\mathrm{d}x - \int_{x_0}^{x_1} f(x, u, u')\mathrm{d}x \\ &= \epsilon\int_{x_0}^{x_1}\left(\eta\frac{\partial f}{\partial u} + \eta'\frac{\partial f}{\partial u'}\right)\mathrm{d}x + O(\epsilon^2). \end{aligned} \tag{2.21}$$

The quantity

$$\delta\mathcal{J}(\eta, u) = \int_{x_0}^{x_1} \left(\eta \frac{\partial f}{\partial u} + \eta' \frac{\partial f}{\partial u'}\right) \mathrm{d}x \tag{2.22}$$

is defined as the first variation of $\mathcal{J}$. For small $\epsilon$, the sign of $\mathcal{J}(\hat{u}) - \mathcal{J}(u)$ is determined by the sign of the first variation, unless $\delta\mathcal{J}(\eta, u) = 0$ for all $\eta \in \mathcal{H}$. Since $u \in \mathcal{U}$ is a local maximum of $\mathcal{J}$, $\mathcal{J}(\hat{u}) - \mathcal{J}(u)$ does not change sign for any $\hat{u} \in \mathcal{N}_\epsilon(u)$. Hence, if $\mathcal{J}(u)$ is a local maximum then

$$\delta\mathcal{J}(\eta, u) = \int_{x_0}^{x_1} \left\{\eta \frac{\partial f}{\partial u} + \eta' \frac{\partial f}{\partial u'}\right\} \mathrm{d}x = 0 \tag{2.23}$$

for all $\eta \in \mathcal{H}$. In addition, if $\mathcal{J}$ has a local minimum at $u \in \mathcal{U}$, (2.23) must also be satisfied. If $u$ satisfies (2.23) for all $\eta \in \mathcal{H}$, we say that $\mathcal{J}$ is stationary at $u$.

Furthermore, by employing the integration by parts and boundary conditions $\eta(x_0) = \eta(x_1) = 0$, (2.23) can be rewritten into the following formulation

$$\int_{x_0}^{x_1} \eta \left(\frac{\partial f}{\partial u} - \frac{d}{\mathrm{d}x}\left(\frac{\partial f}{\partial u'}\right)\right) \mathrm{d}x = 0. \tag{2.24}$$

Then by applying the fundamental lemma of the calculus of variations [72], we conclude the following theorem:

**Theorem 2.15** (Theorem 2.2.3 in [123])**.** *Let $\mathcal{J} : C^2[x_0, x_1] \to \mathbb{R}$ be a functional defined by*

$$\mathcal{J}(u) = \int_{x_0}^{x_1} f(x, u, u')\mathrm{d}x \tag{2.25}$$

*where $f$ has second-order continuous partial derivatives with respect to $x, u$ and $u'$ and $x_0 < x_1$. Let*

$$\mathcal{U} = \{u \in C^2[x_0, x_1] : u(x_0) = u_0 \text{ and } u(x_1) = u_1\}, \tag{2.26}$$

*where $u_0$ and $u_1$ are given real numbers. If $u \in \mathcal{U}$ is an extrema of $\mathcal{J}$, then*

$$\frac{\partial f}{\partial u} - \frac{d}{\mathrm{d}x}\left(\frac{\partial f}{\partial u'}\right) = 0 \tag{2.27}$$

*for all $x \in [x_0, x_1]$.*

(2.27) is a second-order ordinary differential equation that any extrema $u$ must satisfy. This differential equation is called the Euler-Lagrange equation.

**Example 2.13.** *Consider the following functional defined by*

$$\mathcal{J}(u) = \int_0^{\pi} (u'^2 - ku^2)\mathrm{d}x, \tag{2.28}$$

*with boundary conditions* $u(0) = 0$ *and* $u(\pi) = 0$. *If* $u$ *is an extrema for* $\mathcal{J}$, *then its Euler-Lagrange equation is*

$$u'' + ku = 0. \tag{2.29}$$

*The general solution for the Euler-Lagrange equation is*

$$u(x) = c_1 cos(\sqrt{k}x) + c_2 sin(\sqrt{k}x). \tag{2.30}$$

*Combining with the boundary conditions, we get an infinite number of extrema:* $u(x) = c_2 sin(\sqrt{k}x)$.

Although the Euler-Lagrange equation is the necessary condition for minimizing a functional, it may lead to false conclusions when the existence of a minimizer is not established beforehand. Next, we show the outline of the direct method in the calculus of variations, which is used to prove the existence of the minimizer for a functional.

### 2.2.2 The Direct Method

The direct method introduced by Zaremba and Hilbert around 1900 provides an outline to prove the existence of a minimizer for a given functional.

In order to ensure that the functional $\mathcal{J}$ has a minimizer, the necessary condition is that the functional $\mathcal{J}$ must be bounded, namely,

$$\inf\{\mathcal{J}(u)|u \in \mathcal{U}\} > -\infty. \tag{2.31}$$

Then there exists a sequence $\{u_k\}_{k=1}^{\infty}$ in $\mathcal{U}$ such that

$$\lim_{k\to\infty} \mathcal{J}(u_k) = \inf\{\mathcal{J}(u)|u \in \mathcal{U}\}. \tag{2.32}$$

Next, we list the main parts of the direct method in the calculus of variations:

1. Take a minimizing sequence $\{u_k\}_{k=1}^{\infty}$ for $\mathcal{J}$;
2. Prove that in $\{u_k\}_{k=1}^{\infty}$, there exists a subsequence $\{u_{k_l}\}_{l=1}^{\infty}$ which converges to a $u_0 \in \mathcal{U}$ with respect to a topology $\tau$ on $\mathcal{U}$;
3. Prove that $\mathcal{J}$ is sequentially lower semi-continuous with respect to the topology $\tau$.

Since the functional $\mathcal{J}$ is sequentially lower semi-continuous if

$$\lim_{k\to\infty} \inf \mathcal{J}(u_k) \geq \mathcal{J}(u_0) \tag{2.33}$$

for any convergence sequence $u_k \to u_0$ in $\mathcal{U}$, we have the following formulation:

$$\inf\{\mathcal{J}(u)|u \in V\} = \lim_{k\to\infty} \mathcal{J}(u_k) = \lim_{l\to\infty} \mathcal{J}(u_{k_l}) \geq \mathcal{J}(u_0) \geq \inf\{\mathcal{J}(u)|u \in \mathcal{U}\}. \tag{2.34}$$

This means

$$\mathcal{J}(u_0) = \inf\{\mathcal{J}(u)|u \in \mathcal{U}\} \tag{2.35}$$

and $u_0$ is a minimizer of $\mathcal{J}$ in $\mathcal{U}$.

## 2.3 Inverse Problems and Regularization

In the last fifty years, inverse problems have been widely applied in geophysics, oceanography, signal processing, machine learning, medical imaging and many other fields [126]. Forward problems start with the causes and calculate the results but inverse problems start with the results and calculate the causes. For example, in image denoising, we want to get the clean image from the observed image and in CT reconstruction, we want to get the CT image from the data source. However, inverse problems are usually ill-posed. The classical definition of an ill-posed problem is defined by Hadamard in 1902 [55]: if one of the following conditions can not be satisfied:

1. the solution exists;
2. the solution is unique;
3. the solution's behavior changes continuously with the initial conditions.

A problem is well-posed if it is not ill-posed.

**Example 2.14.** *Here, we use several simple examples to illustrate the ill-posed problems and well-posed problems. Let us consider the following system of linear equations:*

$$Ax = b, \tag{2.36}$$

*where $A \in \mathbb{R}^{2\times 2}$ is a matrix and $b \in \mathbb{R}^{2\times 1}$ is a vector.*

- *If $A = \begin{pmatrix} 3 & 4 \\ 3 & 4 \end{pmatrix}$ and $b = \begin{pmatrix} 2 \\ 7 \end{pmatrix}$, this problem does not exist a solution and it is ill-posed.*
- *If $A = \begin{pmatrix} 3 & 4 \\ 3 & 4 \end{pmatrix}$ and $b = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$, this problem has infinite solutions $x = \begin{pmatrix} \frac{2-4k}{3} \\ k \end{pmatrix}, k \in \mathbb{R}$ and it is ill-posed.*
- *If $A = \begin{pmatrix} 2.0002 & 1.9998 \\ 1.9998 & 2.0002 \end{pmatrix}$ and $b = \begin{pmatrix} 4 \\ 4 \end{pmatrix}$, this problem has only one solution $x = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. If we add a perturbation $\delta b = \begin{pmatrix} 2 \times 10^{-4} \\ -2 \times 10^{-4} \end{pmatrix}$ to $b$, the solution changes to $\bar{x} = \begin{pmatrix} 1.5 \\ 0.5 \end{pmatrix}$. Here, we can find that the relative error of the solution $\frac{\|\bar{x}-x\|_\infty}{\|x\|_\infty}$ is 10,000 times of the relative error of the right hand side $\frac{\|\delta b\|_\infty}{\|b\|_\infty}$. So it is ill-posed.*

- *If* $A = \begin{pmatrix} 3 & 4 \\ 5 & 1 \end{pmatrix}$ *and* $b = \begin{pmatrix} 7 \\ 6 \end{pmatrix}$, *this problem has a unique solution* $x = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. *The solution depends continuously on the right hand side* $b$ *and it is well-posed.*

Since it is very difficult to deal with the ill-posed problem, Andrei N. Tikhonov in [120] introduced the concept of the regularization which used a series of well-posed problems to approximate the ill-posed problem.

**Example 2.15.** *Let us consider the following least-square problem:*

$$\min_x \|Ax - b\|_2^2 \tag{2.37}$$

*where* $A \in \mathbb{R}^{m\times n}$ *is a matrix and* $b \in \mathbb{R}^{m\times 1}$ *is a vector. The normal equation of* (2.37) *is*

$$A^T Ax = A^T b. \tag{2.38}$$

*If* $n > m$, (2.38) *may have no solution or infinite solutions. Here, we try to find the solution with some properties, for example, it has the smallest* 2*-norm. Then we can convert the problem* (2.37) *into the following problem:*

$$\min_x \|Ax - b\|_2^2 + \alpha\|x\|_2^2. \tag{2.39}$$

*Here, the first term is the fitting term and the second term is the regularizer.* $\alpha$ *is a nonnegative parameter to balance the weight of these two terms. In order to minimize* (2.39)*, we only need to solve the following linear system:*

$$(A^T A + \alpha I)x = A^T b. \tag{2.40}$$

*Since* $\alpha > 0$ *and* $A^T A + \alpha I$ *is symmetric and positive definite,* (2.40) *has only one solution. Hence,* (2.39) *is well-posed.*

In the general case, Tikhonov regularization replaces

$$\min_u \mathcal{D}(u) \tag{2.41}$$

by

$$\min_u \mathcal{D}(u) + \alpha\mathcal{R}(u), \tag{2.42}$$

where $\mathcal{D}(u)$ is a fitting term and $\mathcal{R}(u)$ is a regularizer which can rule out the unwanted solutions according to the prior information.

## 2.4 Discretization and Notation

In the numerical implementation, we must discretize the continuous problem into the discrete problem since the computer only deals with the discrete data. How to discretize a continuous problem is very important because a proper discretization may affect the

rate of the convergence and an improper discretization may lead to a bad result or even divergence.

There exist several ways to discretize a continuous problem, including the finite element method, the finite volume method, and the finite difference method. For image registration problem, since the domain of the image is usually rectangular and the intensity values of the image are uniformly distributed in this domain, the natural way is to choose the finite difference method to discretize this domain. This domain is denoted by $\Omega \in \mathbb{R}^d$. For simplicity, in this section, we only consider the two-dimensional case, namely, $d = 2$ and $\Omega \in \mathbb{R}^2$, and it can be similarly extended to the three-dimensional case. Here, we set $\Omega = [a, b] \times [c, d]$. Employing a cartesian mesh, the lengths of the intervals in $x_1-$ and $x_2-$direction are $h_1 = (b-a)/n_1$ and $h_2 = (d-c)/n_2$ respectively.

### 2.4.1 Discrete Schemes

For the discretization of the image registration problem, there are three common discrete schemes: cell-centered discretization, staggered discretization, and nodal discretization.

#### Cell-Centered Discretization

In the so-called cell-centered discretization, the grids are located at the center of the cells (Figure 2.2). There are $n_1 \times n_2$ grid points and the position of the grid point $(i, j)$ is $(a + (i - \frac{1}{2})h_1, c + (j - \frac{1}{2})h_2)$ for $1 \leq i \leq n_1$ and $1 \leq j \leq n_2$.

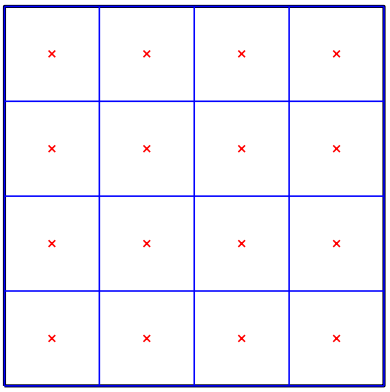
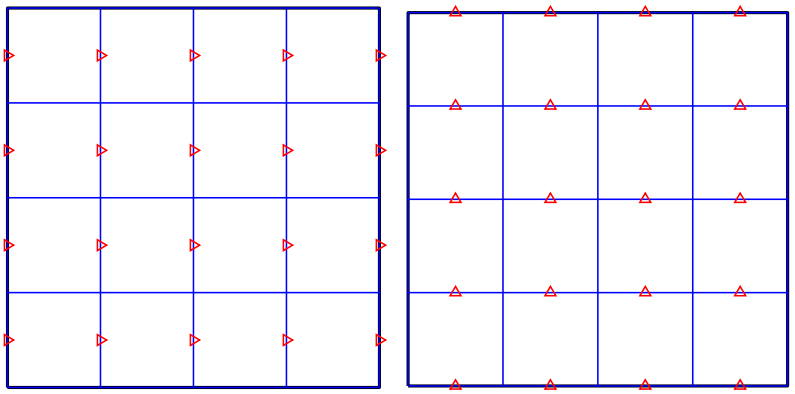


FIGURE 2.2: Cell-centered discretization of a square domain.

#### Staggered Discretization

In the so-called staggered discretization, the grids are located at the edge of the cells (Figure 2.3). There are $(n_1 + 1) \times n_2$ grid points and the position of the grid point $(i, j)$ is $(a + ih_1, c + (j - \frac{1}{2})h_2)$ for $0 \leq i \leq n_1$ and $1 \leq j \leq n_2$ (Figure 2.3 (a)) or there are $n_1 \times (n_2 + 1)$ grid points and the position of the grid point $(i, j)$ is $(a + (i - \frac{1}{2})h_1, c + jh_2)$ for $1 \leq i \leq n_1$ and $0 \leq j \leq n_2$ (Figure 2.3 (b)).



(a) (b)

FIGURE 2.3: Staggered discretization of a square domain.

### Nodal Discretization

In the so-called nodal discretization, the grids are located at the vertex of the cells (Figure 2.4). There are $(n_1+1)\times(n_2+1)$ grid points and the position of the grid point $(i,j)$ is $(a+ih_1, c+jh_2)$ for $0 \le i \le n_1$ and $0 \le j \le n_2$.

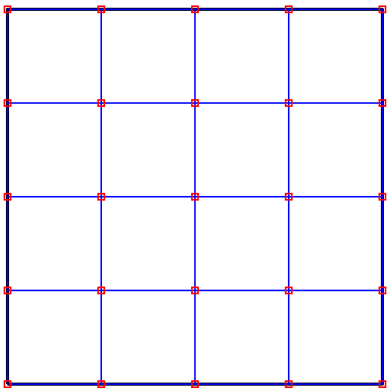


FIGURE 2.4: Nodal discretization of a square domain.

## 2.4.2 Difference Schemes

In the image registration problem, our aim is to find the suitable displacement $\boldsymbol{u}(\boldsymbol{x}) = (u_1(x_1,x_2), u_2(x_1,x_2))$(or transformation $\boldsymbol{y}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{u}$) which will be discussed in the next chapter. In the implementation, we usually need to discretize the derivative of the displacement $\boldsymbol{u}$. Here, for $u_1(x_1,x_2)$, we use difference schemes to approximate its partial derivatives:

- First order forward difference: $\partial_{x_1}^{i,j} u_1 \approx \frac{u_1^{i+1,j} - u_1^{i,j}}{h_1}$;
- First order backward difference: $\partial_{x_1}^{i,j} u_1 \approx \frac{u_1^{i,j} - u_1^{i-1,j}}{h_1}$;
- Second order central difference: $\partial_{x_1}^{i,j} u_1 \approx \frac{u_1^{i+1,j} - u_1^{i-1,j}}{2h_1}$;

- First order forward difference: $\partial_{x_2}^{i,j} u_1 \approx \frac{u_1^{i,j+1} - u_1^{i,j}}{h_2}$;
- First order backward difference: $\partial_{x_2}^{i,j} u_1 \approx \frac{u_1^{i,j} - u_1^{i,j-1}}{h_2}$;
- Second order central difference: $\partial_{x_2}^{i,j} u_1 \approx \frac{u_1^{i,j+1} - u_1^{i,j-1}}{2h_2}$;
- Second order central difference: $\partial_{x_1 x_1}^{i,j} u_1 \approx \frac{u_1^{i+1,j} - 2u_1^{i,j} + u_1^{i-1,j}}{h_1^2}$;
- Second order central difference: $\partial_{x_2 x_2}^{i,j} u_1 \approx \frac{u_1^{i,j+1} - 2u_1^{i,j} + u_1^{i,j-1}}{h_2^2}$;
- Second order mixed difference: $\partial_{x_1 x_2}^{i,j} u_1 \approx \frac{u_1^{i+1,j+1} - u_1^{i+1,j-1} - u_1^{i-1,j+1} + u_1^{i-1,j-1}}{4h_1 h_2}$;

where $u_1^{i,j} = u_1(x_1^i, x_2^j)$ is the value of $u_1$ at the grid point $(i, j)$. For the partial derivatives of $u_2(x_1, x_2)$, we have similar results.

### 2.4.3 Matrix Notation

In this thesis, we usually need to discretize a variational problem and get a summation. To simplify the formulation, by introducing the vector and matrix, we convert this summation into a matrix-vector product. We first define the Kronecker product and then employ a simple example to illustrate this idea.

**Definition 2.16** (**Kronecker Product**)**.** Let $A$ be an $m \times n$ matrix and $B$ be a $p \times q$ matrix. We say that $C$ is the Kronecker product of $A$ and $B$, written $C = A \otimes B$, provided by

$$C = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix}. \tag{2.43}$$

The dimension of $C$ is $mp \times nq$.

**Example 2.16.** *Let us discretize the following variational problem:*

$$\min_{\boldsymbol{u}} \int_\Omega \sum_{l=1}^{2} |\nabla u_l|^2 \mathrm{d}\boldsymbol{x}, \tag{2.44}$$

*where* $\Omega = [0, 1] \times [0, 1]$.

*We employ the nodal discretization and define a spatial partition* $\Omega_h = \{\boldsymbol{x}^{i,j} \in \Omega | \boldsymbol{x}^{i,j} = (x_1^j, x_2^j) = (ih, jh), 0 \le i \le n, 0 \le j \le n\}$*, where* $h = \frac{1}{n}$ *and the discrete domain consists of* $n^2$ *cells of size* $h \times h$*. We discretize* $\boldsymbol{u}$ *on the nodal grid, namely* $\boldsymbol{u}^{i,j} = (u_1^{i,j}, u_2^{i,j}) = (u_1(x_1^i, x_2^j), u_2(x_1^i, x_2^j))$*. Here, we use forward difference to approximate* $\partial_{x_1} u_l$ *and* $\partial_{x_2} u_l$ *for* $l = 1, 2$ *at the grid point* $(x_1^j, x_2^j)$*. Then we have the following approximation:*

$$\int_\Omega \sum_{l=1}^{2} |\nabla u_l|^2 \mathrm{d}\boldsymbol{x} \approx h^2 \sum_{l=1}^{2} \sum_{j=0}^{n-1} \sum_{i=0}^{n-1} \left( \left( \frac{u_l^{i+1,j} - u_l^{i,j}}{h} \right)^2 + \left( \frac{u_l^{i,j+1} - u_l^{i,j}}{h} \right)^2 \right). \tag{2.45}$$

*In order to convert* (2.45) *into a vector inner product, according to the lexicographical ordering, we reshape*

$$U = (u_1^{0,0}, ..., u_1^{n,0}, ..., u_1^{0,n}, ..., u_1^{n,n}, u_2^{0,0}, ..., u_2^{n,0}, ..., u_2^{0,n}, ..., u_2^{n,n})^T \in \mathbb{R}^{2(n+1)^2 \times 1}. \quad (2.46)$$

*Set* $A = \begin{pmatrix} B \\ C \end{pmatrix}$, $B = I_2 \otimes I_{n+1} \otimes \partial_n^{1,h}$, $C = I_2 \otimes \partial_n^{1,h} \otimes I_{n+1}$ *and*

$$\partial_n^{1,h} = \frac{1}{h} \begin{pmatrix} -1 & 1 & & \\ & -1 & 1 & \\ & \cdots & \cdots & \cdots \\ & & -1 & 1 \\ & & & -1 \quad 1 \end{pmatrix} \in \mathbb{R}^{n \times (n+1)}, \quad (2.47)$$

*where* $\otimes$ *denotes the Kronecker product. Then* (2.45) *can be reformulated as follows:*

$$\int_\Omega \sum_{l=1}^{2} |\nabla u_l|^2 \mathrm{d}\boldsymbol{x} \approx h^2 U A^T A U. \quad (2.48)$$

## 2.5 Iterative Methods for Solving Linear Systems

In this section, we mainly introduce two Krylov subspace methods: conjugate gradient method (CG) and minimal residual method (MINRES), for solving the symmetric systems of linear equations. For the classic iterative methods (such as Jacobi method, Gauss-Seidel method and SOR method) and other Krylov subspace methods (such as BICG and TFQMR), please refer to [3, 14, 66, 74, 97, 101, 121].

### 2.5.1 Conjugate Gradient Method

The conjugate gradient method (CG) is an iterative method for solving a system of linear equations

$$Ax = b, \quad (2.49)$$

where $A \in \mathbb{R}^{n\times n}$ is a symmetric and positive definite (SPD) matrix. The algorithm of the CG method is summarized in Algorithm 1.

**Algorithm 1:** CG Method

Given $x^0$;
Set $r^0 = Ax^0 - b, p^0 = -r^0, k = 0$;
**while** $r^k \neq 0$ **do**
  $\alpha^k = -\frac{(r^k)^T r^k}{(p^k)^T A p^k}$;
  $x^{k+1} = x^k + \alpha^k p^k$;
  $r^{k+1} = r^k + \alpha^k A p^k$;
  $\beta^{k+1} = \frac{(r^{k+1})^T r^{k+1}}{(r^k)^T r^k}$;
  $p^{k+1} = -r^{k+1} + \beta^{k+1} p^k$;
  $k = k + 1$;
**end**

The CG method has the following basic properties and convergent results.

**Theorem 2.17** (Theorem 5.3 in [97]). *Suppose that the kth iterate generated by the CG method is not the solution point $x^*$. Then we have the following four properties:*

$$(r^k)^T r^i = 0, \quad \text{for } i = 0, 1, ..., k-1, \tag{2.50}$$

$$span\{r^0, r^1, ..., r^k\} = span\{r^0, Ar^0, ..., A^k r^0\}, \tag{2.51}$$

$$span\{p^0, p^1, ..., p^k\} = span\{r^0, Ar^0, ..., A^k r_0\}, \tag{2.52}$$

$$(p^k)^T A p^i = 0, \quad \text{for } i = 0, 1, ..., k-1. \tag{2.53}$$

*In addition, the sequence $\{x^k\}$ converges to $x^*$ in at most n iterations.*

**Theorem 2.18** (Theorem 5.4 in [97]). *If A has only r distinct eigenvalues, then the CG method will terminate at the solution in at most r iterations.*

**Theorem 2.19** (Theorem 5.5 in [97]). *If A has eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, we have that*

$$\|x^{k+1} - x^*\|_A^2 \leq \left(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1}\right)^2 \|x^0 - x^*\|_A^2, \tag{2.54}$$

*where $\|x\|_A$ is denoted by $\sqrt{x^T A x}$.*

From Theorem 2.18 and 2.19, we know that the rate of the convergence of the CG method depends on the distribution of the eigenvalues of $A$. In order to accelerate the CG method, we can consider solving the following linear system

$$(C^{-T} A C^{-1})\hat{x} = C^{-T} b \tag{2.55}$$

where $\hat{x} = Cx$ and $C$ is a nonsingular matrix. We can choose $C$ such that the condition number of $C^{-T} A C^{-1}$ is much smaller than the condition number of $A$ or the eigenvalues

of $C^{-T}AC^{-1}$ are clustered. Then we have the preconditioned CG method and the algorithm is summarized in Algorithm 2.

**Algorithm 2:** Preconditioned CG Method

Given $x^0$, preconditioner $M$ $(M = C^TC)$;
Set $r^0 = Ax^0 - b, k = 0$;
Solve $My^0 = r^0$ for $y^0$;
Set $p^0 = -y^0$;
**while** $r^k \neq 0$ **do**
  $\alpha^k = -\frac{(r^k)^T y^k}{(p^k)^T A p^k}$;
  $x^{k+1} = x^k + \alpha^k p^k$;
  $r^{k+1} = r^k + \alpha^k A p^k$;
  Solve $My^{k+1} = r^{k+1}$;
  $\beta^{k+1} = \frac{(r^{k+1})^T y^{k+1}}{(r^k)^T y^k}$;
  $p^{k+1} = -y^{k+1} + \beta^{k+1} p^k$;
  $k = k + 1$;
**end**

**Example 2.17.** *Let us use the preconditioned CG method to solve the following linear system:*

$$Ax = b, \tag{2.56}$$

*where $A$ is the discrete Laplacian matrix by 5-point difference scheme with Dirichlet boundary conditions and $b$ is a vector whose elements are all* 1. *From Figure 2.5, we can*

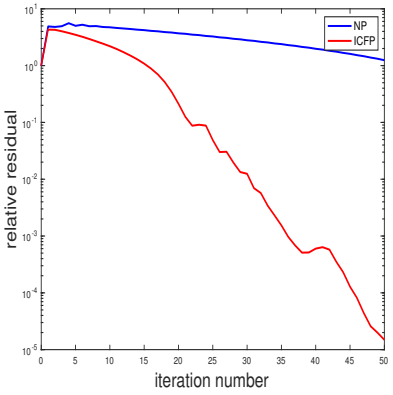



FIGURE 2.5: NP and ICFP represent no preconditioner and incomplete Cholesky factorization preconditioner respectively.

*notice that a good preconditioner can speed up the rate of the convergence significantly.*[1]

[1] https://www.mathworks.com/help/matlab/ref/pcg.html

### 2.5.2 Minimal Residual Method

Let us consider the following equations:

$$r + Ax = b, \qquad Ar = 0, \tag{2.57}$$

where $A \in \mathbb{R}^{n\times n}$ is a symmetric matrix which may be both indefinite and singular. We want to find an approximation of $x$ with the form $V_k y$, where $V_k \in \mathbb{R}^{n\times k}$ is a matrix and its columns $v_i, i \in \{1, ..., k\}$ are linearly independent mutually. Here, we want to find $y^k$ such that it is a stationary value to

$$f_k(y) = (AV_k y - b)^T (AV_k y - b). \tag{2.58}$$

Then we can minimize $\|Ax - b\|_2$ by solving

$$V_k^T A^2 V_k y^k = V_k^T Ab, \qquad x^k = V_k y^k. \tag{2.59}$$

If the vectors $v_i$ are computed by the Lanczos algorithm [100], we have

$$V_k^T A^2 V_k = T_k^2 + \beta_{k+1}^2 e_k e_k^T, \tag{2.60}$$

$$V_k^T Ab = \beta_1 V_k^T A v_1 = \beta_1 T_k e_1, \tag{2.61}$$

where $\beta_1 = \|b\|_2$, $e_k$ is a $k$ dimensional column vector whose $k$th component is 1 and remaining components are 0, and

$$T_k = \begin{pmatrix} \alpha_1 & \beta_2 & & \\ \beta_2 & \alpha_2 & \beta_3 & \\ & \cdot & \cdot & \\ & & \beta_k & \alpha_k \end{pmatrix}. \tag{2.62}$$

By using orthogonal factorization $T_k = \bar{L}_k Q_k$, we get

$$T_k^2 + \beta_{k+1}^2 e_k e_k^T = \bar{L}_k \bar{L}_k^T + \beta_{k+1}^2 e_k e_k^T = L_k L_k^T, \tag{2.63}$$

where $\bar{L}_k$ and $L_k$ are both lower triangular and $Q$ is unitary. Here, we use Givens rotation to decompose $T_k$:

$$T_k Q_{1,2} \cdots Q_{k-1,k} = T_k Q_k^T = \bar{L}_k = \begin{pmatrix} \gamma_1 & & & & \\ \delta_2 & \gamma_2 & & & \\ \epsilon_3 & \delta_3 & \gamma_3 & & \\ & \cdot & \cdot & \cdot & \\ & & \epsilon_k & \delta_k & \bar{\gamma_k} \end{pmatrix}, \tag{2.64}$$

where $Q_{i,i+1}$ is a Givens matrix whose non-zero elements are given by $q_{i,i} = -q_{i+1,i+1} = c_i, q_{i,i+1} = q_{i+1,i} = s_i, q_{k,k} = 1$ for $k \neq i, i+1$. In the next step, we compute

$$\gamma_k = (\bar{\gamma_k}^2 + \beta_{k+1}^2)^{1/2}, \quad c_k = \bar{\gamma_k}/\gamma_k, \quad s_k = \beta_{k+1}/\gamma_k. \tag{2.65}$$

Now, we only need to solve the following equation:

$$L_k L_k^T y^k = \beta_1 \bar{L}_k Q_k e_1. \tag{2.66}$$

We have $\bar{L}_k = L_k D_k$, $D_k = \text{diag}(1, \cdots, 1, c_k)$. Set $L_k^T y^k = \beta_1 D_k Q_k e_1 = t_k$ and $M_k = V_k L_k^{-T}$. Then we have

$$x^k = V_k y^k = M_k t_k. \tag{2.67}$$

## 2.6 Optimization Methods

In reality, many complex problems can be reformulated into optimization problems. In this section, the fundamental theory of optimization methods, including unconstrained optimization methods and constrained optimization methods, are presented. For more details about the theory of optimization methods, please refer to [8, 73, 97, 115].

### 2.6.1 Unconstrained Optimization Methods

In this subsection, we consider the following unconstrained optimization problem:

$$\min_x f(x), \tag{2.68}$$

where $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth function. The basic iterative scheme for the unconstrained optimization methods is

$$x^{k+1} = x^k + \alpha^k p^k, \tag{2.69}$$

where $x^k$ is the $k$th iterative point, $\alpha^k$ is the step length and $p^k$ is the search direction. Next, we first present the first-order necessary condition. Then the choices of the step length and the search directions are briefly discussed.

#### First-Order Necessary Condition

First, we give the definition of the global minimizer and the local minimizer of (2.68).

**Definition 2.20.** We say that a point $x^*$ is a global minimizer of (2.68) if the following condition is satisfied:

$$f(x^*) \leq f(x) \quad \text{for all } x. \tag{2.70}$$

In addition, we say that a point $x^*$ is a local minimizer of (2.68) if the following condition is satisfied:

$$f(x^*) \leq f(x) \quad \text{for all } x \in \mathcal{N}, \tag{2.71}$$

where $\mathcal{N}$ is a neighborhood of $x$.

Figure 2.6 shows an example of the global minimizer and the local minimizer.

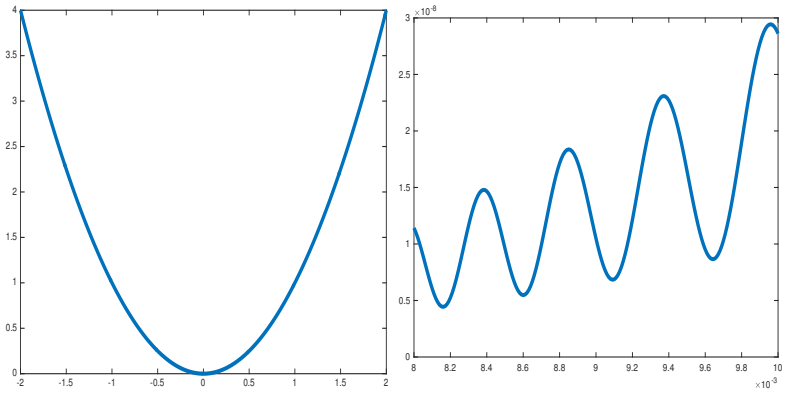


FIGURE 2.6: An example of the global minimizer (left) and the local minimizer(right).

Then we present the first-order necessary condition.

**Theorem 2.21.** *If $x^*$ is a local minimizer and $f$ is continuously differentiable in an open neighbourhood of $x^*$, then $\nabla f(x^*) = 0$.*

*Proof.* The proof can be found in [97]. □

Here, if $\nabla f(x^*) = 0$, we say that $x^*$ is a stationary point or a critical point. Hence, any local minimizer must be a stationary point.

**Theorem 2.22.** *When $f$ is convex, any local minimizer $x^*$ is a global minimizer of $f$. In addition, if $f$ is differentiable, any stationary point $x^*$ is a global minimizer of $f$.*

*Proof.* The proof can be found in [97]. □

### Armijo Condition

Assume that the search direction $p^k$ is fixed, the best choice of the step length $\alpha^k$ should be the global minimizer of the following function:

$$\phi(\alpha) = f(x^k + \alpha p^k), \qquad \alpha > 0. \tag{2.72}$$

However, it is usually too expensive to find this exact value. Hence, in practice, an inexact line search is often implemented rather than doing an exact line search. A popular inexact line search condition should satisfy the sufficient decrease:

$$f(x^k + \alpha p^k) \leq f(x^k) + \alpha\eta(\nabla f^k)^T p^k, \tag{2.73}$$

where $\nabla f^k$ is the gradient of $f(x)$ at $x^k$ and $\eta \in (0,1)$. Usually, $\eta$ is chosen as $10^{-4}$ [73, 92]. (2.73) is also called Armijo condition. We can use the backtracking strategy (Algorithm 3) to determine the step length $\alpha^k$.

**Algorithm 3:** Backtracking Strategy

```
Given ᾱ > 0, ρ, η ∈ (0, 1);
Set α = ᾱ;
while f(x^k + αp^k) > f(x^k) + αη(∇f^k)^T p^k do
    α = ρα;
end
α^k = α.
```

For the other inexact line search conditions, such as the Wolf condition, the strong Wolf condition and the Goldstein condition, please refer to [97, 115].

### The Steepest Descent Method

For the steepest descent method, the search direction is

$$p^k = -\nabla f^k. \tag{2.74}$$

Hence, the iterative scheme of the steepest descent method is

$$x^{k+1} = x^k - \alpha^k \nabla f^k, \tag{2.75}$$

where $\alpha^k$ is chosen by a line search condition.

Although the steepest descent method is one of the simplest and most fundamental methods in unconstrained optimization, the convergence rate may be very slow.

**Theorem 2.23** (Theorem 3.1.5 in [115])**.** *Consider the following unconstrained optimization problem*

$$\min_x \frac{1}{2} x^T G x, \tag{2.76}$$

*where $G \in \mathbb{R}^{n\times n}$ is a symmetric and positive definite matrix. Let $\lambda_n$ and $\lambda_1$ be the largest and smallest eigenvalues of $G$ respectively. Let $x^*$ be the minimizer of* (2.76)*. Then the sequence $\{x^k\}_{k=1}^{\infty}$ generated by the steepest descent method with the exact line search converges to $x^*$ and the following bound holds:*

$$\|x^{k+1} - x^*\|_G^2 \le \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}\right)^2 \|x^k - x^*\|_G^2, \tag{2.77}$$

*where $\|x^k - x^*\|_G^2 = (x^k - x^*)^T G (x^k - x^*)$.*

According to Theorem 2.23, we can know that the convergence rate of the steepest descent method depends on the condition number of $G$ ($\lambda_n/\lambda_1$). If the condition number of $G$ is very large, the convergence rate should be very slow.

### Newton Method

Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable and $\nabla^2 f^k$ (the Hessian of $f(x)$ at $x^k$) be positive definite. By using the Taylor expansion, we have

$$f(x^k + p) \approx q^k(p) = f(x^k) + (\nabla f^k)^T p + \frac{1}{2} p^T \nabla^2 f^k p, \tag{2.78}$$

where $p = x - x^k$. By minimizing $q^k(p)$, we have

$$x^{k+1} = x^k - [\nabla^2 f^k]^{-1} \nabla f^k, \tag{2.79}$$

which is the iterative scheme of the Newton method. The following theorem shows that the Newton method is local convergent and has the quadratic convergence rate.

**Theorem 2.24** (Theorem 3.2.2 in [115])**.** *Let $f \in C^2$ and $x^k$ be close enough to the minimizer $x^*$ with $\nabla f(x^*) = 0$. If the Hessian $\nabla^2 f(x^*)$ is positive definite and $\nabla^2 f(x)$ satisfies the Lipschitz condition*

$$|\nabla^2_{i,j} f(x) - \nabla^2_{i,j} f(y)| \leq \beta \|x - y\|, \text{ for some } \beta \text{ and all } i, j, \tag{2.80}$$

*where $\nabla^2_{i,j} f(x)$ is the $(i,j)$-element of $\nabla^2 f(x)$, then for all $k$, Newton's iteration* (2.79) *is well-defined and the generated sequence $\{x^k\}_{k=1}^{\infty}$ converges to $x^*$ with a quadratic rate.*

*Remark* 2.25. Although the Newton method can have the quadratic convergence rate, it severely depends on the location of the initial point. If the initial point is far from the minimizer, the Newton method may be not convergent.

### Quasi-Newton Method

Here, we introduce the most popular quasi-Newton method, the BFGS method, which is named after Broyden, Fletcher, Goldfarb, and Shanno. Let us consider the following approximation of the objective function $f(x)$ at the current iterate $x^k$:

$$f(x^k + p) \approx m^k(p) = f(x^k) + (\nabla f^k)^T p + \frac{1}{2} p^T B^k p, \tag{2.81}$$

where $B^k \in \mathbb{R}^{n \times n}$ is symmetric and positive definite. By minimizing $m^k(p)$ to obtain its minimizer $p^k$, we have

$$p^k = -[B^k]^{-1} \nabla f^k \tag{2.82}$$

and the iterative scheme of the quasi-Newton method is as follows:

$$x^{k+1} = x^k + \alpha^k p^k, \tag{2.83}$$

where $\alpha^k$ is chosen by a line search condition.

But how to choose $B^k$ at $k$th iteration? Let us consider

$$m^{k+1}(p) = f(x^{k+1}) + (\nabla f^{k+1})^T p + \frac{1}{2} p^T B^{k+1} p. \tag{2.84}$$

If we assume that the gradient of $m^{k+1}$ and the gradient of $f$ at the latest two iterates $x^k$ and $x^{k+1}$ are the same, then we have the following relationships:

$$\begin{aligned} \nabla m^{k+1}(0) &= \nabla f^{k+1}, \\ \nabla m^{k+1}(-\alpha^k p^k) &= \nabla f^k. \end{aligned} \tag{2.85}$$

The first equation in (2.85) is obviously satisfied and the second equation in (2.85) results in

$$B^{k+1}\alpha^k p^k = \nabla f^{k+1} - \nabla f^k. \tag{2.86}$$

By introducing $s^k = x^{k+1} - x^k$ and $y^k = \nabla f^{k+1} - \nabla f^k$, we have

$$B^{k+1} s^k = y^k. \tag{2.87}$$

(2.87) is called secant equation. In order to let $B^{k+1}$ be a symmetric and positive definite matrix, the curvature condition

$$(s^k)^T y^k > 0 \tag{2.88}$$

should also be satisfied. When condition (2.88) is satisfied, there always exist infinite solutions $B^{k+1}$ to (2.87). We aim to find a $B^{k+1}$ which is symmetric and positive definite, satisfies the secant equation and is closest to the current matrix $B^k$ in some sense. So we solve the following problem:

$$\min_B \|B - B^k\| \quad \text{subject to} \quad B = B^T, Bs^k = y^k, \tag{2.89}$$

where $s^k$ and $y^k$ satisfies the curvature condition (2.88) and $B^k$ is symmetric and positive definite. The matrix norm is chosen as $\|A\|_W = \|W^{1/2} A W^{1/2}\|_F$, where $\|\cdot\|_F$ is Frobenius norm and $W$ can be any matrix satisfying $Wy^k = s^k$. Then we can derive the DFP updating formula:

$$B^{k+1} = (\boldsymbol{I} - \rho^k y^k (s^k)^T) B^k (\boldsymbol{I} - \rho^k s^k (y^k)^T) + \rho^k y^k (y^k)^T, \tag{2.90}$$

where $\rho^k = \frac{1}{(y^k)^T s^k}$.

Furthermore, let $H^k$ be the inverse of $B^k$. If we solve the following problem:

$$\min_H \|H - H^k\| \quad \text{subject to} \quad H = H^T, Hy^k = s^k, \tag{2.91}$$

we can derive the BFGS updating formula:

$$H^{k+1} = (\boldsymbol{I} - \rho^k s^k (y^k)^T) H^k (\boldsymbol{I} - \rho^k y^k (s^k)^T) + \rho^k s^k (s^k)^T. \tag{2.92}$$

The algorithm of BFGS method is summarized in Algorithm 4.

**Algorithm 4:** BFGS Method

Given $x^0$, $\epsilon > 0$, initial inverse Hessian approximation $H^0$;
$k = 0$;
**while** $\|\nabla f^k\| \geq \epsilon$ **do**
    $p^k = -H^k \nabla f^k$;
    $x^{k+1} = x^k + \alpha^k p^k$ ($\alpha^k$ is computed from a line search condition.);
    $s^k = x^{k+1} - x^k$;
    $y^k = \nabla f^{k+1} - \nabla f^k$;
    compute $H^{k+1}$ from (2.92);
    $k = k + 1$;
**end**

*Remark* 2.26. At each iteration, the quasi-Newton method computes the gradient and the matrix-vector product and does not need to solve a linear system.

However, the inverse Hessian approximation $H^k$ will be in general not sparse and it is very difficult to store and compute a matrix-vector product $H^k \nabla f^k$. From (2.92), $H^k \nabla f^k$ can be computed by doing a sequence of inner products and vector summation involving $\nabla f^k$ and the pairs $\{s^i, y^i\}$. Hence, we just store a fixed number $m$ of the vector pairs $\{s^i, y^i\}$ to store a modified version of $H^k$ implicitly. The oldest vector pair in $\{s^i, y^i\}$ will be deleted and replaced by the new pair after the new iterate. Then the limited-BFGS approximation $H^k$ satisfies the following formula:

$$\begin{aligned} H^k =& ((V^{k-1})^T \cdots (V^{k-m})^T) H^0 (V^{k-m} \cdots V^{k-1}) \\ &+ \rho^{k-m} ((V^{k-1})^T \cdots (V^{k-m+1})^T) s^{k-m} (s^{k-m})^T (V^{k-m+1} \cdots V^{k-1}) \\ &+ \rho^{k-m+1} ((V^{k-1})^T \cdots (V^{k-m+2})^T) s^{k-m+1} (s^{k-m+1})^T (V^{k-m+2} \cdots V^{k-1}) \\ &+ \cdots \\ &+ \rho^{k-1} s^{k-1} (s^{k-1})^T. \end{aligned} \tag{2.93}$$

where $V^k = \boldsymbol{I} - \rho^k y^k (s^k)^T$ and $H^0$ is the initial inverse Hessian approximation. From (2.93), we can summarize Algorithm 5 to compute $H^k \nabla f^k$ efficiently.

**Algorithm 5:** L-BFGS recursion

$q = \nabla f^k$;
**for** $i = k-1, k-2, ..., k-m$ **do**
  $\alpha^i = \rho^i (s^i)^T q$ ;
  $q = q - \alpha^i y^i$;
**end**
$r = H^0 q$;
**for** $i = k-m, k-m+1, ..., k-1$ **do**
  $\beta = \rho^i (y^i)^T r$;
  $r = r + s^i(\alpha^i - \beta)$;
**end**
Stop with result $H^k \nabla f^k = r$.

Based on Algorithm 5, the L-BFGS method is summarized in Algorithm 6.

**Algorithm 6:** L-BFGS Method

Given $x^0$, $\epsilon > 0$, integer $m > 0$, initial inverse Hessian approximation $H^0$;
$k = 0$;
**while** $\|\nabla f^k\| \geq \epsilon$ **do**
  $p_k = -H^k \nabla f^k$ (Apply Algorithm 5);
  $x^{k+1} = x^k + \alpha^k p^k$ ($\alpha^k$ is computed from a line search condition.);
  **if** $k > m$ **then**
    Delete the vector pair $\{s^{k-m}, y^{k-m}\}$ from storage;
  **end**
  $s^k = x^{k+1} - x^k$;
  $y^k = \nabla f^{k+1} - \nabla f^k$;
  $k = k + 1$;
**end**

### Gauss-Newton Method

Let us consider the following unconstrained optimization problem:

$$\min_x f(x) = \frac{1}{2} \sum_{j=1}^{m} r_j^2(x), \tag{2.94}$$

where each $r_j : \mathbb{R}^n \to \mathbb{R}$ is a smooth function. Set $r(x) = (r_1(x), ..., r_m(x))^T$ and then $f(x)$ can be rewritten as $f(x) = \frac{1}{2}\|r(x)\|_2^2$. The gradient and Hessian of $f$ are as follows:

$$\nabla f(x) = J(x)^T r(x), \tag{2.95}$$

$$\nabla^2 f(x) = J(x)^T J(x) + \sum_{j=1}^{m} r_j(x)\nabla^2 r_j(x), \tag{2.96}$$

where

$$J(x) = \left[\frac{\partial r_j}{\partial x_i}\right]_{\substack{j = 1, ..., m \\ i = 1, ..., n}} \tag{2.97}$$

is the Jacobian of $r(x)$ and $\nabla^2 r_j(x)$ is the Hessian of $r_j(x)$. For (2.94), in order to find the search direction $p^k$, the Gauss-Newton method just solves the following equation

$$(J^k)^T J^k p = -\nabla f^k \tag{2.98}$$

instead of solving the Newton equation $\nabla^2 f^k p = -\nabla f^k$.

The Gauss-Newton method has several advantages: firstly, we only need to compute the Jacobian of $r(x)$ and avoid to compute $\nabla^2 r_j(x)$. In practice, it can save significant time. Secondly, $(J^k)^T J^k$ is usually much more important than the second order term when $\|r(x)\|$ or $\|\nabla^2 r_j(x)\|$ is small. Finally, when $J^k$ is full rank, $(J^k)^T J^k$ is positive definite and if $\nabla f^k$ is nonzero, (2.98) can lead to a descent direction.

### 2.6.2 Constrained Optimization Methods

In this subsection, we consider the following constrained optimization problem:

$$\min_x f(x) \qquad \text{subject to} \begin{cases} c_i(x) = 0, & i \in \mathcal{E} \\ c_i(x) \geq 0, & i \in \mathcal{I} \end{cases} \tag{2.99}$$

where $f, c_i : \mathbb{R}^n \to \mathbb{R}$ are all smooth functions and $\mathcal{E}$ and $\mathcal{I}$ are both finite sets of indices. Here, we first present the first-order necessary condition (KKT condition). Then we mainly focus on two methods: the quadratic penalty method and the augmented Lagrangian method.

#### First-Order Necessary Condition

Before giving the first-order necessary condition of (2.99), we first give the definition of the global minimizer and the local minimizer of (2.99).

**Definition 2.27.** We say that a point $x^* \in \Omega$ is a global minimizer of (2.99) if the following condition is satisfied:

$$f(x^*) \leq f(x) \quad \text{for all } x \in \Omega. \tag{2.100}$$

In addition, we say that a point $x^* \in \Omega$ is a local minimizer of (2.99) if the following condition is satisfied:

$$f(x^*) \leq f(x) \quad \text{for all } x \in \mathcal{N} \cap \Omega. \tag{2.101}$$

Here, $\Omega$ is the feasible set defined by

$$\Omega = \{x | c_i(x) = 0, i \in \mathcal{E};\ c_i(x) \geq 0, i \in \mathcal{I}\} \tag{2.102}$$

and $\mathcal{N}$ is a neighborhood of $x$.

Then the Lagrangian of the constrained optimization problem (2.99) is defined by the following formulation:

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(x). \tag{2.103}$$

Now, we present the first-order necessary condition.

**Theorem 2.28.** *Suppose that $x^*$ is a local solution of* (2.99) *and the linear independence constraint qualification holds at $x^*$. Then there exists a Lagrangian multiplier vector $\lambda^*$, with components $\lambda_i^*$, $i \in \mathcal{E} \cup \mathcal{I}$, such that the following conditions are satisfied at $(x^*, \lambda^*)$*

$$\nabla_x \mathcal{L}(x^*, \lambda^*) = 0, \tag{2.104}$$

$$c_i(x^*) = 0, \quad \text{for all } i \in \mathcal{E}, \tag{2.105}$$

$$c_i(x^*) \geq 0, \quad \text{for all } i \in \mathcal{I}, \tag{2.106}$$

$$\lambda^* \geq 0, \quad \text{for all } i \in \mathcal{I}, \tag{2.107}$$

$$\lambda_i^* c_i(x^*) = 0, \quad \text{for all } i \in \mathcal{E} \cup \mathcal{I}. \tag{2.108}$$

*Here, the linear independence constraint qualification (LICQ) is that the set of active constraint gradients $\{\nabla c_i(x^*), i \in \mathcal{A}(x^*)\}$ is linear independent, where the active set $\mathcal{A}(x^*)$ is $\mathcal{E} \cup \{i \in \mathcal{I} | c_i(x^*) = 0\}$.*

*Proof.* The proof can be found in [97]. □

This first-order necessary condition is also called Karush-Kuhn-Tucker (KKT) conditions.

### Quadratic Penalty Method

In this section, we focus on the following optimization problem with equality constraints:

$$\min_x f(x) \qquad \text{subject to} \quad c_i(x) = 0, i \in \mathcal{E}. \tag{2.109}$$

The quadratic penalty method is to minimize a sequence of quadratic penalty functions:

$$\mathcal{Q}(x; \sigma^k) = f(x) + \frac{1}{2\sigma^k} \sum_{i \in \mathcal{E}} c_i^2(x), \tag{2.110}$$

where $\sigma^k$ is the penalty parameter and $\lim_{k\to\infty}\sigma^k = 0$. Set $x^k$ as the minimizer of $\mathcal{Q}(x;\sigma^k)$. When $\sigma^k \to 0$, $\mathcal{Q}(x;\sigma^k)$ can penalize the constraints and $x^k$ can approximate the minimizer $x^*$ of (2.109). In addition, since $\mathcal{Q}(x;\sigma^k)$ is smooth, we can use the above mentioned unconstrained optimization methods to solve this quadratic penalty function. In practice, $x^k$ can be set as the initial guess for minimizing $\mathcal{Q}(x;\sigma^{k+1})$. Then the algorithm of the quadratic penalty method is summarized in Algorithm 7.

**Algorithm 7:** Quadratic Penalty Method

Given $\sigma^0$, $\epsilon^0 > 0$, initial guess $x^{0,s}$;
**for** $k = 0, 1, 2, \ldots$ **do**
  minimize $\mathcal{Q}(x;\sigma^k)$ with initial guess $x^{k,s}$ to find an approximated minimizer $x^k$ and the stopping criteria is $\|\nabla\mathcal{Q}(x;\sigma^k)\| \le \epsilon^k$;
  **if** *the final convergence test is satisfied* **then**
    Exit and find the approximated solution $x^k$;
  **end**
  Set the penalty parameter $\sigma^{k+1} \in (0, \sigma^k)$;
  Set the initial guess $x^{k+1,s} = x^k$;
  Set the tolerance $\epsilon^k > 0$;
**end**

For the quadratic penalty method, we have the following convergence theorem:

**Theorem 2.29** (Theorem 17.2 in [97])**.** *If the tolerance $\epsilon^k$ in Algorithm 7 satisfy $\lim_{k\to\infty}\epsilon^k = 0$ and the penalty parameters $\sigma^k$ satisfy $\lim_{k\to\infty}\sigma^k = 0$, then for all limit point $x^*$ of the sequence $\{x^k\}$ at which the constraint gradient $\nabla c_i(x^*)$ are linearly independent, we have that $x^*$ is a KKT point for the problem* (2.109)*. For such points, we have the infinite subsequence $\mathcal{K}$ such that $\lim_{k\in\mathcal{K}} x^k = x^*$ and*

$$\lim_{k\in\mathcal{K}} -c_i(x^k)/\sigma^k = \lambda_i^*, \quad \textit{for all } \ i \in \mathcal{E}, \tag{2.111}$$

*where $\lambda^*$ is multiplier vector that satisfies the KKT conditions.*

### Augmented Lagrangian Method

From (2.111), we know that the approximated minimizer $x^k$ of $\mathcal{Q}(x;\sigma^k)$ can not satisfy the equality constraints $c_i(x) = 0, i \in \mathcal{E}$ unless $\sigma^k$ is very small. In order to make the approximated minimizer satisfy the constraints easily and avoid decreasing $\sigma^k$ to 0, the augmented Lagrangian method was studied by Hestenes [65] and Powell [105]. The augmented Lagrangian is defined by

$$\mathcal{L}_A(x,\lambda;\sigma) = f(x) - \sum_{i\in\mathcal{E}}\lambda_i c_i(x) + \frac{1}{2\sigma}\sum_{i\in\mathcal{E}} c_i^2(x). \tag{2.112}$$

Differentiating (2.112) with respect to $x$, we get

$$\nabla_x \mathcal{L}_A(x, \lambda; \sigma) = \nabla f(x) - \sum_{i \in \mathcal{E}} [\lambda_i - c_i(x)/\sigma] \nabla c_i(x). \tag{2.113}$$

Here, we can deduce that

$$\lambda_i^* \approx \lambda_i^k - c_i(x^k)/\sigma^k, \quad \text{for all} \quad i \in \mathcal{E}. \tag{2.114}$$

Then we have $c_i(x^k) \approx -\sigma^k(\lambda_i^* - \lambda_i^k)$ for all $i \in \mathcal{E}$. If $\lambda_i^k$ is close to $\lambda_i^*$, $\|c_i(x^k)\|$ can be much smaller than $\sigma^k$ rather than the multiple of $\sigma^k$ shown in (2.111). Obviously, $\lambda_i^{k+1}$ can be updated by (2.114):

$$\lambda_i^{k+1} = \lambda_i^{k+1} - c_i(x^k)/\sigma^k, \quad \text{for all} \quad i \in \mathcal{E}. \tag{2.115}$$

Then the algorithm of the augmented Lagrangian method is summarized in Algorithm 8.

**Algorithm 8:** Augmented Lagrangian Method

Given $\sigma^0$, $\epsilon^0 > 0$, initial guess $x^{0,s}$ and $\lambda^0$;
**for** $k = 0, 1, 2, ...$ **do**
  minimize $\mathcal{L}_A(x, \lambda^k; \sigma^k)$ with initial guess $x^{k,s}$ to find an approximated minimizer $x^k$ and the stopping criteria is $\|\nabla_x \mathcal{L}_A(x, \lambda^k; \sigma^k)\| \leq \epsilon^k$;
  **if** *the final convergence test is satisfied* **then**
    Exit and find the approximated solution $x^k$;
  **end**
  Use (2.115) to update $\lambda^{k+1}$;
  Set the penalty parameter $\sigma^{k+1} \in (0, \sigma^k)$;
  Set the initial guess $x^{k+1,s} = x^k$;
  Set the tolerance $\epsilon^k > 0$;
**end**

### Alternating Direction Method of Multipliers

Here, we review the alternating direction method of multipliers (ADMM) which is a variant of the augmented Lagrangian method that partially updates the dual variables. Consider the following constrained problem:

$$\begin{aligned} &\min f(x) + g(z) \\ &\text{s.t. } Ax + Bz = c \end{aligned} \tag{2.116}$$

with $x \in \mathbb{R}^{n \times 1}$, $z \in \mathbb{R}^{m \times 1}$, $A \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{p \times m}$ and $c \in \mathbb{R}^{p \times 1}$. Its corresponding augmented Lagrangian function is defined by the following formulation:

$$\mathcal{L}_A(x, z, \lambda, \sigma) = f(x) + g(z) + \lambda^T(Ax + Bz - c) + \frac{1}{2\sigma}\|Ax + Bz - c\|^2, \tag{2.117}$$

where $\lambda$ is the Lagrangian multiplier and $\sigma > 0$ is the penalty parameter. Then ADMM consists of the following iterations

$$\begin{aligned} x^{k+1} &:= \text{argmin } \mathcal{L}_A(x, z^k, \lambda^k, \sigma), \\ z^{k+1} &:= \text{argmin } \mathcal{L}_A(x^{k+1}, z, \lambda^k, \sigma), \\ \lambda^{k+1} &:= \lambda^k + \frac{1}{\sigma}(Ax^{k+1} + Bz^{k+1} - c). \end{aligned} \tag{2.118}$$

ADMM can be viewed as a Gauss-Seidel pass over $x$ and $z$ instead of updating $x$ and $z$ simultaneously. By considering the structures of $f$ and $g$, we can design effective solves for subproblem $x$ and subproblem $z$ respectively and reduce the computational time significantly. Besides, ADMM can have excellent properties of the convergence under suitable assumptions, and for more details, please refer to [7, 13, 57, 70].

**Bregman Iteration**

Here, we introduce the Bregman iteration, which is one of the most popular methods in image processing and first used by Osher et al. in [99] to solve the ROF model for TV denoising. We consider the following constrained optimization problem:

$$\min_x f(x) \quad \text{subject to} \quad Ax = b, \tag{2.119}$$

where $A$ is a linear operator and $b$ is a vector. We use the quadratic penalty method to convert (2.119) into the following unconstrained optimization problem:

$$\min_x f(x) + \frac{1}{2\sigma}\|Ax - b\|_2^2. \tag{2.120}$$

Then the basic iterative scheme of the Bregman iteration is as follows:

$$x^{k+1} = \text{argmin } f(x) + \frac{1}{2\sigma}\|Ax - b^k\|_2^2, \tag{2.121}$$

$$b^{k+1} = b^k + b - Ax^{k+1}. \tag{2.122}$$

In [133], Yin et al. point out that the Bregman iteration is equivalent to the augmented Lagrangian method when the constriants are linear. More relevant details can be found in [36, 46, 102].

# Chapter 3

# Mathematical Models for Image Registration

In this chapter, we present the general variational framework for image registration. Firstly, the variational framework for image registration is established, and fidelity terms and regularizers are discussed. Then, the general solution scheme is considered, including first-optimize-then-discretize and first-discretize-then-optimize. Some other details, such as cubic spline interpolation and multilevel strategy, are also illustrated.

## 3.1 Variational Framework for Image Registration

The aim of image registration is to find a plausible transformation $\boldsymbol{y}(\boldsymbol{x}) : \Omega \subset \mathbb{R}^d \to \mathbb{R}^d$ such that the deformed template

$$(T \circ \boldsymbol{y})(\boldsymbol{x}) = T(\boldsymbol{y}(\boldsymbol{x})) \tag{3.1}$$

is similar with the reference $R(\boldsymbol{x})$. At the same time, we define $\boldsymbol{u}(\boldsymbol{x}) = (u_1(\boldsymbol{x}), ..., u_d(\boldsymbol{x}))$ is the displacement which shows how much $T$ moves. So $\boldsymbol{y}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x})$ and solving $\boldsymbol{y}(\boldsymbol{x})$ or $\boldsymbol{u}(\boldsymbol{x})$ is equivalent.

In order to measure the difference between the deformed template $T(\boldsymbol{x} + \boldsymbol{u})$ and the reference $R$, a suitable fidelity term $\mathcal{D}(T(\boldsymbol{x} + \boldsymbol{u}), R)$ should be introduced. Hence, the image registration problem can be formulated by minimizing the following functional:

$$\min_{\boldsymbol{u}} \mathcal{D}(T(\boldsymbol{x} + \boldsymbol{u}), R). \tag{3.2}$$

However, it is well known that only minimizing (3.2) is ill-posed in the sense of Hadamard since it is not sufficient to ensure the uniqueness and continuity of the solution. In order to overcome this difficulty, regularization is indispensable [11, 23, 32, 33, 59, 86, 87, 129]. Combining the fidelity term and the regularization, the image registration problem can

be well-posed as minimizing the following joint functional:

$$\min_{\boldsymbol{u}} \mathcal{D}(T(\boldsymbol{x}+\boldsymbol{u}), R) + \alpha \mathcal{R}(\boldsymbol{u}), \tag{3.3}$$

where $\alpha$ is a positive parameter to balance these two terms, $\mathcal{R}(\boldsymbol{u})$ is the regularizer to rule out the irregular and unwanted solutions based on prior information and $\boldsymbol{u}$ belongs to a specified feasible set.

## 3.2 Fidelity Terms

In image registration, how to measure the difference between the deformed template and the reference is one of the most basic tasks. For different applications, different fidelity terms should be considered. In this section, we present several commonly used choices for the fidelity terms, including the sum of squared differences, normalized cross correlation, normalized gradient fields, and mutual information.

### 3.2.1 Sum of Squared Differences (SSD)

In the mono-modal image registration, where the intensities of the given images are comparable, the most widely used fidelity term should be the so-called sum of squared differences (SSD) [91, 92] defined by:

$$\mathcal{D}^{\mathrm{SSD}}(T(\boldsymbol{x}+\boldsymbol{u}), R) = \frac{1}{2}\int_{\Omega} (T(\boldsymbol{x}+\boldsymbol{u}) - R)^2 \mathrm{d}\boldsymbol{x}. \tag{3.4}$$

When $T(\boldsymbol{x}+\boldsymbol{u}) = R$, (3.4) can reach its minimal value.

### 3.2.2 Normalized Cross Correlation (NCC)

When the intensities of the given images are not comparable, SSD can not be an appropriate fidelity term. In order to overcome this drawback, the normalized cross correlation (NCC) is proposed [92]:

$$\mathcal{D}^{\mathrm{NCC}}(T(\boldsymbol{x}+\boldsymbol{u}), R) = 1 - \frac{\langle T(\boldsymbol{x}+\boldsymbol{u}), R\rangle^2}{\|T(\boldsymbol{x}+\boldsymbol{u})\|^2 \|R\|^2}, \tag{3.5}$$

where $\|\cdot\| = \sqrt{\langle \cdot, \cdot \rangle}$. Here, we can find that when $T(\boldsymbol{x}+\boldsymbol{u})$ and $R$ are linearly dependent, (3.5) can reach its minimal value.

### 3.2.3 Normalized Gradient Fields (NGF)

In the multi-modal image registration, although the given images can be captured from different devices, such as CT and MRI, the reference image and the template image usually have similar shapes and features but different intensities. This phenomenon can be expressed by the fact that the changes of the intensities at corresponding positions are similar. Hence, introducing the information of the gradients of the images into

the fidelity term should be a good choice. Based on this idea, Haber and Modersitzki proposed the normalized gradient fields (NGF) [51, 54]:

$$\mathcal{D}^{\mathrm{NGF}}(T(\boldsymbol{x}+\boldsymbol{u}),R)=\int_{\Omega}1-(\nabla_n T(\boldsymbol{x}+\boldsymbol{u})\cdot\nabla_n R)^2\mathrm{d}\boldsymbol{x}, \tag{3.6}$$

where $\nabla_n T(\boldsymbol{x}+\boldsymbol{u})$ is defined by $\nabla T(\boldsymbol{x}+\boldsymbol{u})/|\nabla T(\boldsymbol{x}+\boldsymbol{u})|$ (assuming $\nabla T(\boldsymbol{x}+\boldsymbol{u})\neq 0$). (3.6) gives its minimal value 0 when the normalized gradients of the reference and the deformed template are linearly dependent.

### 3.2.4 Mutual Information (MI)

Mutual Information (MI) derived from the theory of information was introduced by Maes et al. [84] and Viola et al. [125] independently. There are several ways to define Mutual Information [104]. Here, we introduce the definition related to the Kullback-Leibler distance. For two distributions $p$ and $q$, the Kullback-Leibler distance is defined as

$$\int_{\Omega}p(x)\log\frac{p(x)}{q(x)}\mathrm{d}x. \tag{3.7}$$

It is a measure of the distance between two distributions. Analogous to the Kullback-Leibler measure, Mutual Information of the two images is defined by the following formulation:

$$\mathcal{D}^{\mathrm{MI}}(T(\boldsymbol{x}+\boldsymbol{u}),R)=-\int_{\mathbb{R}^2}p_{T,R}(t,r)\log\frac{p_{T,R}(t,r)}{p_T(t)p_R(r)}\mathrm{d}t\mathrm{d}r, \tag{3.8}$$

where $p_T, p_R$ are probability distributions of the grey values in $T$ and $R$ and $p_{T,R}$ is the joint probability distribution of the grey values. It is a measure of dependence between two images. In practice, we can use either a histogram-based density estimator or a Parzen-Window-based density estimator to estimate the joint probability $p_{T,R}$ [92]. Note that when $T$ and $R$ are independent, $\mathcal{D}^{MI}$ is 0. Hence, minimizing (3.8) can maximize the similarity between the given images.

*Remark* 3.1. There are still many kinds of fitting terms for multi-modality registration, for instance, the normalized gradient field (NGF) [69, 76, 109], edges sketching registration [2], and normalized gradient fitting (GT) [69, 117]. Recently [16] proposed a cross-correlation similarity measure based on reproducing kernel Hilbert spaces and found advantages over Mutual Information.

## 3.3 Regularizations

As mentioned before, only minimizing the fidelity term is ill-posed in the sense of Hadamard. To overcome this difficulty, regularization is indispensable, which can rule out irregular results according to prior information. In this section, we review some classic regularizers commonly used in image registration.

### 3.3.1 Linear Elastic Regularizer

The linear elastic regularizer proposed by Broit [9] in 1981 is defined by the following formulation:

$$\mathcal{R}^{\text{Lelas}}(\boldsymbol{u}) = \int_\Omega \frac{\mu}{4} \sum_{l,m=1}^{d} (\partial_{x_l} u_m + \partial_{x_m} u_l)^2 + \frac{\lambda}{2} (\nabla \cdot \boldsymbol{u})^2 \mathrm{d}\boldsymbol{x}, \tag{3.9}$$

where $\mu$ and $\lambda$ denote the so-called Láme constants [37]. Especially, this regularizer can penalize the linear affine deformation.

### 3.3.2 Diffusion Regularizer

Modersitzki and Fischer [38] proposed the diffusion regularizer based on $W^{1,2}$ semi-norm:

$$\mathcal{R}^{\text{Diff}}(\boldsymbol{u}) = \frac{1}{2} \int_\Omega \sum_{l=1}^{d} |\nabla u_l|^2 \mathrm{d}\boldsymbol{x}, \tag{3.10}$$

which aims to control the smoothness of the displacement. In reality, the diffusion regularizer is a special case of the linear elastic regularizer: when $\mu$ and $\lambda$ in (3.9) are 1 and $-1$ respectively, the linear elastic regularizer can degenerate to the diffusion regularizer.

### 3.3.3 Hyperelastic Regularizer

In 2004, Droske and Rumpf [33] first applied the hyperelastic regularizer to image registration. They consider a polyconvex energy functional:

$$\mathcal{R}^{\text{Hyper}_1}(\boldsymbol{y}) := \int_\Omega \alpha_l \|\nabla \boldsymbol{y}\|_2^2 + \alpha_s \|\text{cof} \nabla \boldsymbol{y}\|_2^2 + \alpha_v \phi_1(\det(\nabla \boldsymbol{y})) \mathrm{d}\boldsymbol{x}, \tag{3.11}$$

where $\phi_1(v) = \alpha_1 v^2 - \alpha_2 \ln v$ and $\text{cof} \nabla \boldsymbol{y}$ is the cofactor. Then $\mathcal{R}^{\text{Hyper}_1}(\boldsymbol{y})$ penalizes volume shrinkage and growth.

In 2013, Burger et al. [11] also used a polyconvex regularization term:

$$\mathcal{R}^{\text{Hyper}_2}(\boldsymbol{y}) := \int \alpha_l \text{length}(\boldsymbol{y}) + \alpha_s \text{surface}(\boldsymbol{y}) + \alpha_v \text{volume}(\boldsymbol{y}) \mathrm{d}\boldsymbol{x}. \tag{3.12}$$

Here,

$$\begin{aligned}
&\text{length}(\boldsymbol{y}) = \phi_l(\nabla \boldsymbol{y}), && \phi_l(X) = \|X - \text{Id}\|_2^2, \\
&\text{surface}(\boldsymbol{y}) = \phi_c(\text{cof} \nabla \boldsymbol{y}), && \phi_c(X) = \max\{\|X\|_2^2 - 3, 0\}^2, \\
&\text{volume}(\boldsymbol{y}) = \phi_2(\det(\nabla \boldsymbol{y})), && \phi_2(v) = ((v-1)^2/v)^2.
\end{aligned}$$

Here, since $\phi_2(v)$ also goes to $\infty$ when $v$ goes to 0 or $\infty$ and $\phi_2(v) = \phi_2(1/v)$, $\phi_2(v)$ controls the volume such that shrinkage and growth have the same price. Hence, $\mathcal{R}^{\text{Hyper}_2}$

also restricts the Jacobian determinant of the transformation $\boldsymbol{y}$ close to 1 which is too strong in some 2D applications [135].

This hyperelastic regularizer can help to get a diffeomorphic transformation since when $\det(\nabla \boldsymbol{y}) \to 0$, $\mathcal{R}^{\text{Hyper}}(\boldsymbol{y}) \to \infty$. For more details about the theory of hyperelastic material, please refer to [25, 26].

*Remark* 3.2. Here, we regularize the transformation $\boldsymbol{y}$. And when $d = 2$, the hyperelastic regularizer only involves length term and volume term.

### 3.3.4 Fluid Regularizer

Christensen et al. [18] proposed an effective viscous fluid model characterized by a spatial smoothing of the velocity field. For the viscous fluid model, the deformation is governed by the Navier-Stokes equation:

$$\eta \Delta \boldsymbol{v} + (\lambda + \eta) \nabla (\nabla \cdot \boldsymbol{v}) + \boldsymbol{F} = 0. \tag{3.13}$$

Here, $\eta$ and $\lambda$ are the viscosity coefficients, the term $\nabla^2 \boldsymbol{v}$ constraints the velocity field to vary smoothly, the term $\nabla(\nabla \cdot \boldsymbol{v})$ allows structures in the template to change in mass and $\boldsymbol{F}$ is the nonlinear deformation force field, which can be defined by $(T(\boldsymbol{x} + \boldsymbol{u}) - R)\nabla T(\boldsymbol{x} + \boldsymbol{u})$. In the Eulerian frame, to account for the difference between the velocity $\boldsymbol{v}$ and the time rate of change of the deformation $\boldsymbol{u}$ here, we use material derivative to provide the time rate of change:

$$\boldsymbol{v} = \partial_t \boldsymbol{u} + \boldsymbol{v} \cdot \nabla \boldsymbol{u}. \tag{3.14}$$

In [18], the condition $|\det(\nabla \boldsymbol{y})| \geq 0.5$ is checked at each iteration and if not satisfied, restarting the numerical solver is initiated so that a diffeomorphic transformation is obtained.

### 3.3.5 Total Variation Regularizer

The total variation technique was firstly used in image processing by Rudin, Osher, and Fatemi [107]. In [43], the total variation regularizer is extend to image registration:

$$\mathcal{R}^{\text{TV}}(\boldsymbol{u}) = \int_{\Omega} \sum_{l=1}^{d} |\nabla u_l| \mathrm{d}\boldsymbol{x}. \tag{3.15}$$

This regularizer allows steep gradients and discontinuities in the displacement field rather than the smooth displacement field derived by the diffusion regularizer (3.10).

### 3.3.6 Linear Curvature Regularizer

The linear curvature regularizer proposed by Fischer and Modersitzki [39, 40] is the first one of the curvature-type regularizers for image registration. The formulation of the linear curvature regularizer is as follows:

$$\mathcal{R}^{\mathrm{Lcurv}}(\boldsymbol{u}) = \int_{\Omega} \sum_{l=1}^{d} |\Delta u_l|^2 \mathrm{d}\boldsymbol{x}. \tag{3.16}$$

Note that $\mathcal{R}^{\mathrm{Lcurv}}$ has a non-trivial kernel which contains affine linear displacements. Hence, using this linear curvature regularizer does not penalize affine linear transformations and does not need an additional affine linear pre-registration step.

### 3.3.7 Henn and Witsch's Curvature Regularizer

Henn and Witsch [59–61] modified the linear curvature regularizer and proposed the following curvature regularizer:

$$\mathcal{R}^{\mathrm{HWcurv}}(\boldsymbol{u}) = \int_{\Omega} \sum_{l=1}^{2} (\Delta u_l - 2(\partial_{x_1x_1} u_l \partial_{x_2x_2} u_l - \partial_{x_1x_2} u_l \partial_{x_2x_1} u_l))^2 \mathrm{d}\boldsymbol{x}. \tag{3.17}$$

The kernel of this regularizer only contains the affine linear displacement and hence, this regularizer is invariant under planar rotation and translation.

### 3.3.8 Mean Curvature Regularizer

The mean curvature regularizer is given by the following formulation:

$$\mathcal{R}^{\mathrm{Mcurv}}(\boldsymbol{u}) = \frac{1}{2} \int_{\Omega} \sum_{l=1}^{2} |\kappa_M(u_l)|^2 \mathrm{d}\boldsymbol{x}, \tag{3.18}$$

which is proposed by Chumchob et al. [24] and $\kappa_M(u_l) = \nabla \cdot \frac{\nabla u_l}{\sqrt{1+|\nabla u_l|^2}}$. If we assume that $u_l, l = 1, 2$ is a surface represented by $(x_1, x_2, u_l(x_1, x_2))$, $\kappa_M(u_l)$ is the mean curvature of the surface $u_l$. In addition, there is a link between the linear curvature regularizer and the mean curvature regularizer: when $\nabla u_l \approx 0$, $\kappa_M(u_l) \approx \Delta u_l$ and $\mathcal{R}^{\mathrm{Mcurv}}(\boldsymbol{u}) \approx \mathcal{R}^{\mathrm{Lcurv}}(\boldsymbol{u})$.

### 3.3.9 Gaussian Curvature Regularizer

In 2015, Ibrahim et al. [71] proposed the Gaussian curvature regularizer:

$$\mathcal{R}^{\mathrm{Gcurv}}(\boldsymbol{u}) = \int_{\Omega} \sum_{l=1}^{2} |\kappa_G(u_l)| \, \mathrm{d}\boldsymbol{x}, \tag{3.19}$$

where $\kappa_G(u_l) = \frac{\partial_{x_1x_1} u_l \partial_{x_2x_2} u_l - \partial_{x_1x_2} u_l \partial_{x_2x_1} u_l}{(|\nabla u_l|^2+1)^2}$. In theory, this Gaussian curvature regularizer has advantages over the total variation regularizer and the mean curvature

regularizer and in numerical experiments, it can outperform the linear curvature regularizer and the mean curvature regularizer. In addition, when $\nabla u_l \approx 0$, $\kappa_G(u_l) \approx \partial_{x_1x_1}u_l\partial_{x_2x_2}u_l - \partial_{x_1x_2}u_l\partial_{x_2x_1}u_l$. So Henn and Witsch's curvature regularizer can be considered as an approximation of the sum of the squared principal curvatures $\kappa_{p_1}(u_l)$ and $\kappa_{p_2}(u_l)$:

$$\begin{aligned}\kappa_{p_1}^2(u_l) + \kappa_{p_2}^2(u_l) &= \left(\nabla \cdot \frac{\nabla u_l}{\sqrt{1+|\nabla u_l|^2}}\right)^2 - 2\frac{\partial_{x_1x_1}u_l\partial_{x_2x_2}u_l - \partial_{x_1x_2}u_l\partial_{x_2x_1}u_l}{(|\nabla u_l|^2+1)^2} \\ &\approx (\Delta u_l)^2 - 2(\partial_{x_1x_1}u_l\partial_{x_2x_2}u_l - \partial_{x_1x_2}u_l\partial_{x_2x_1}u_l).\end{aligned} \tag{3.20}$$

*Remark* 3.3. Here, we can find that the linear elastic regularizer, the diffusion regularizer, the hyperelastic regularizer, the fluid regularizer, the total variation regularizer, and the linear curvature regularizer are all suitable for 2D and 3D image registration. However, Henn and Witsch's curvature regularizer, the mean curvature regularizer and the Gaussian curvature regularizer are only applied to 2D image registration.

## 3.4 General Solution Framework

In the previous section, the image registration problem has been formulated as a variational problem (3.3). How to solve this variational problem (3.3) efficiently and effectively is always a difficult issue.

Note that (3.3) is an infinite-dimensional optimization problem. In general, this type of problem cannot be solved analytically, requiring, therefore, the use of numerical schemes. There are two main numerical approaches to solve infinite-dimensional optimization problems. The first approach, referred as first-optimize-then-discretize, consists of differentiating the objective function (3.3) to obtain its continuous Euler-Lagrange equation, discretizing these equations, and then solving the resulting finite-dimensional equations numerically. The second approach, referred as first-discretize-then-optimize, consists of discretizing the objective function (3.3) and then solving the resulting finite-dimensional optimization problem by some optimization algorithm.

### 3.4.1 First-Optimize-Then-Discretize

For the first-optimize-then-discretize, we try to find the solution of the Euler-Lagrange equation of (3.3):

$$\boldsymbol{f}(\boldsymbol{u}) + \alpha\mathcal{A}(\boldsymbol{u}) = 0 \tag{3.21}$$

subject to the appropriate boundary conditions. Here, the term $\boldsymbol{f}$ and the term $\mathcal{A}$ are usually derived from the Gâteaux derivative of the fidelity term $\mathcal{D}$ and the Gâteaux derivative of the regularizer term $\mathcal{R}$ respectively. In addition, $\boldsymbol{f}$ can be viewed as the external forces to match the given images and $\mathcal{A}$ can be viewed as the internal forces to remove the undesired deformation field $\boldsymbol{u}$.

As an example, let us consider the following classical diffusion model:

$$\min_{\boldsymbol{u}} \frac{1}{2}\int_{\Omega}(T(\boldsymbol{x}+\boldsymbol{u})-R)^2\mathrm{d}\boldsymbol{x}+\frac{\alpha}{2}\int_{\Omega}\sum_{l=1}^{d}|\nabla u_l|^2\mathrm{d}\boldsymbol{x}, \tag{3.22}$$

where the fidelity term is SSD and the regularizer is the diffusion regularizer. Its corresponding Euler-Lagrange equation is as follows:

$$(T(\boldsymbol{x}+\boldsymbol{u})-R)\nabla_{\boldsymbol{u}}T(\boldsymbol{x}+\boldsymbol{u})-\alpha\Delta\boldsymbol{u}=0 \tag{3.23}$$

subject to the natural boundary condition $\langle\nabla u_l,\boldsymbol{n}\rangle=0$ on $\partial\Omega$ and $l=1,...,d$. Compared with (3.21), the term $\boldsymbol{f}$ in (3.23) is $(T(\boldsymbol{x}+\boldsymbol{u})-R)\nabla_{\boldsymbol{u}}T(\boldsymbol{x}+\boldsymbol{u})$ and the term $\mathcal{A}$ in (3.23) is $-\Delta\boldsymbol{u}$. Particularly, there exists a fast implementation based on the so-called additive operator splitting (AOS) scheme [91, 127]. In [20], a fast solver was also developed for this model.

For solving (3.21), if $\boldsymbol{f}$ is nonlinear and $\mathcal{A}$ is linear, the semi-implicit time marching scheme can be defined by the following formulation:

$$\frac{\boldsymbol{u}(t^{k+1})-\boldsymbol{u}(t^k)}{\tau}=\boldsymbol{f}(\boldsymbol{u}(t^k))+\alpha\mathcal{A}(\boldsymbol{u}(t^{k+1})), \tag{3.24}$$

where $\tau$ is the time step length, $k\in\mathbb{N}_0$ and $\boldsymbol{u}(t)=\boldsymbol{u}(\boldsymbol{x},t)$. For more details, refer to [38–40, 59, 64, 91]. If $\boldsymbol{f}$ and $\mathcal{A}$ are both nonlinear, we can define the following fixed point iteration:

$$\boldsymbol{f}(\boldsymbol{u}^k)+\alpha\mathcal{A}[\boldsymbol{u}^k](\boldsymbol{u}^{k+1})=0 \tag{3.25}$$

where $\boldsymbol{f}$ and $\mathcal{A}$ are both linearized at the current approximation $\boldsymbol{u}^k$ and $k\in\mathbb{N}_0$. For more details, please refer to [43, 62].

### 3.4.2 First-Discretize-Then-Optimize

For the first-discretize-then-optimize, the first step is to discretize (3.3) to get a finite dimensional optimization problem:

$$\min_{U} J(U). \tag{3.26}$$

The standard iterative scheme of the optimization method is given by the following formulation:

$$U^{k+1}=U^k+\theta^k\delta U^k, \tag{3.27}$$

where $U^k$ is the current iteration point, $\theta^k$ is the step length and $\delta U^k$ is the search direction. Here, how to choose a good search direction $\delta U^k$ is very crucial. In the following chapters, we will give more details about the Gauss-Newton direction and several strategies to further speed up the optimization algorithm. For more details, please refer to [48, 49, 52, 58, 62, 63].

## 3.5 Cubic Spline Interpolation

In image registration, the evaluation of the deformed template image $T(\boldsymbol{x}+\boldsymbol{u})$ must involve interpolation because $\boldsymbol{x}+\boldsymbol{u}$ does not in general correspond with pixel points. Nearest neighbour interpolation is the easiest interpolation, but it leads to a function whose derivative may not be defined. In this thesis, the main solver algorithm is based on the first-discretize-then-optimize, and a continuously differentiable interpolation is necessary. Linear interpolation is continuous but not differentiable at the grid points. Cubic spline interpolation is twice continuously differentiable and satisfies our requirements. Hence, in this section, we present the details about cubic spline interpolation.

Let $K = \{x_0, ..., x_n\}$ be a set containing $n+1$ knots with $a = x_0 < \cdots < x_n = b$.

**Definition 3.4** (**Cubic Spline Interpolation**)**.** We say that a function $s \in C^2[a,b]$ is a cubic spline on $[a,b]$ if in each interval $[x_i, x_{i+1}]$, $s$ is a cubic polynomial $s_i, i \in \{0, ..., n-1\}$. In addition, we say that it is a cubic spline interpolation if $s(x_i) = y_i$ for given values $y_i$.

Based on the Definition 3.4, a cubic spline interpolation $s$ is a piecewise cubic polynomial:

$$s(x) = \begin{cases} a_0 + b_0(x-x_0) + c_0(x-x_0)^2 + d_0(x-x_0)^3 & \text{if } x_0 \le x \le x_1, \\ \vdots & \vdots \\ a_i + b_i(x-x_i) + c_i(x-x_i)^2 + d_i(x-x_i)^3 & \text{if } x_i \le x \le x_{i+1}, \\ \vdots & \vdots \\ a_{n-1} + b_{n-1}(x-x_{n-1}) + c_{n-1}(x-x_{n-1})^2 + d_{n-1}(x-x_{n-1})^3 & \text{if } x_{n-1} \le x \le x_n. \end{cases} \tag{3.28}$$

and it should satisfy the following conditions:

1. $s(x_j) = y_j$ for $j = 0, ..., n$,
2. $s_j(x_{j+1}) = s_{j+1}(x_{j+1})$ for $j = 0, ..., n-2$,
3. $s_j'(x_{j+1}) = s_{j+1}'(x_{j+1})$ for $j = 0, ..., n-2$,
4. $s_j''(x_{j+1}) = s_{j+1}''(x_{j+1})$ for $j = 0, ..., n-2$.

Here, we notice that $s(x)$ have $4n$ unknowns and $4n-2$ conditions. In order to find these $4n$ unknowns, we need two extra conditions. There exist two boundary conditions often used in the spline interpolation: one is called free or natural boundary conditions: $s''(x_0) = s''(x_n) = 0$ and the other one is called clamped boundary conditions: $s'(x_0) = y'(x_0)$ and $s'(x_n) = y'(x_n)$.

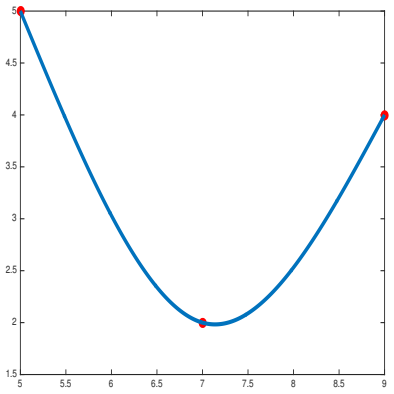



FIGURE 3.1: An example of a cubic spline interpolation with natural boundary conditions.

**Example 3.1.** *Figure 3.1 shows a cubic spline interpolation passing through* $(5,5),(7,2)$ *and* $(9,4)$ *with natural boundary conditions. The formula of this cubic spline interpolation is as follows:*

$$s(x)=\begin{cases}5-\frac{17}{8}(x-5)+\frac{5}{32}(x-5)^3 & \text{if } 5\le x\le 7,\\ 2-\frac{1}{4}(x-7)+\frac{15}{16}(x-7)^2-\frac{5}{32}(x-7)^3 & \text{if } 7\le x\le 9.\end{cases} \tag{3.29}$$

Another way to build a spline interpolation is to use B-splines and this idea can be easily extended to high dimensional cases. B-splines of order $n$ are basis functions for spline functions of the same order defined over the same knots. B-splines of order $n$ can be generated by the Cox-de Boor recursion formula [29]:

$$B_{i,n}:=\frac{x-x_i}{x_{i+n-1}-x_i}B_{i,n-1}(x)+\frac{x_{i+n}-x}{x_{i+n}-x_{i+1}}B_{i+1,n-1}(x), \tag{3.30}$$

where the B-spline of order 1 is defined by:

$$B_{i,1}(x)=\begin{cases}1 & \text{if } x_i\le x\le x_{i+1},\\ 0 & \text{otherwise.}\end{cases} \tag{3.31}$$

Then for a cubic spline interpolation $s(x)$, we can write it as a linear combination of B-splines of order 4 (cubic B-splines Figure 3.2):

$$s(x)=\sum_{i=0}^{n}c_iB_{i,4}(x). \tag{3.32}$$

According to the conditions $s(x_i)=y_i$, we can solve a linear system to get the coefficients $c_i$.

For high dimensional cases, we only consider 2D case here and other high dimensional cases can be extended similarly. Let $K=\{(x_{1,0},x_{2,0}),...(x_{1,n_1},x_{2,n_2})\}$ be a set containing

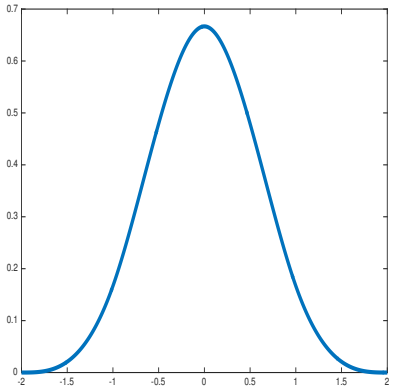



FIGURE 3.2: B-spline of Order 4 (cubic B-spline).

$(n_1+1)\times(n_2+1)$ knots. In order to find a cubic spline interpolation to pass through these $(n_1+1)\times(n_2+1)$ knots, this cubic spline interpolation can be written into the following formulation:

$$s(x_1,x_2)=\sum_{j=0}^{n_2}\sum_{i=0}^{n_1}c_{i,j}B_{i,4}(x_1)B_{j,4}(x_2). \tag{3.33}$$

Set $B_{n_1}=[B_{i,4}(x_{1,k})]_{k,i=0}^{n_1}$, $B_{n_2}=[B_{j,4}(x_{2,k})]_{k,j=0}^{n_2}$ and $B=B_{n_2}\otimes B_{n_1}$, where $\otimes$ denotes the Kronecker product. Then we can get the coefficients $c_{i,j}$ by solving the following linear system:

$$Bc=y, \tag{3.34}$$

where $c$ and $y$ are the vector forms of $[c_{i,j}]_{i,j=0}^{i=n_1,j=n_2}$ and $[y_{i,j}]_{i,j=0}^{i=n_1,j=n_2}$ according to the lexicographical ordering.

## 3.6 Multilevel Strategy

In practice, the multilevel strategy is often used in the image registration problem [52, 92]. We firstly coarsen the template $T$ and the reference $R$ by several levels. In the finest level, we set $T_L=T$ and $R_L=R$ and in the coarsest level, we set $T_l=T$ and $R_l=R$. Then we solve the image registration problem on the coarsest level to get a solution. By using an interpolation operator on the solution of the coarsest level, we get an initial guess for the next level. We repeat this process and obtain the final registration on the finest level. This multilevel strategy has several advantages: in the coarse level, only important patterns can be considered, and it is a standard technique used in order to avoid getting trapped into a meaningless local minimum; the computational speed is very fast because of fewer variables than on the fine level; the solution on the coarse level can be a good initial guess for the fine level.

In order to coarsen the given images, we average the adjacent cells: for 2D images, the each pixel $(i, j)$ of the image $I_\ell, \ell \in \{l, ..., L-1\}$ on the coarse level is

$$\frac{1}{4}(I_{\ell+1}^{2i-1,2j-1} + I_{\ell+1}^{2i-1,2j} + I_{\ell+1}^{2i,2j-1} + I_{\ell+1}^{2i,2j}); \tag{3.35}$$

for 3D images, the each pixel $(i, j, k)$ of the image $I_\ell, \ell \in \{l, ..., L-1\}$ on the coarse level is

$$\begin{aligned}\frac{1}{8}(&I_{\ell+1}^{2i-1,2j-1,2k-1} + I_{\ell+1}^{2i-1,2j,2k-1} + I_{\ell+1}^{2i,2j-1,2k-1} + I_{\ell+1}^{2i,2j,2k-1} \\ &+ I_{\ell+1}^{2i-1,2j-1,2k} + I_{\ell+1}^{2i-1,2j,2k} + I_{\ell+1}^{2i,2j-1,2k} + I_{\ell+1}^{2i,2j,2k}).\end{aligned} \tag{3.36}$$

As an illustration, Figure 3.3 shows the multilevel representation of a pair of the template $T$ and the reference $R$.[1]

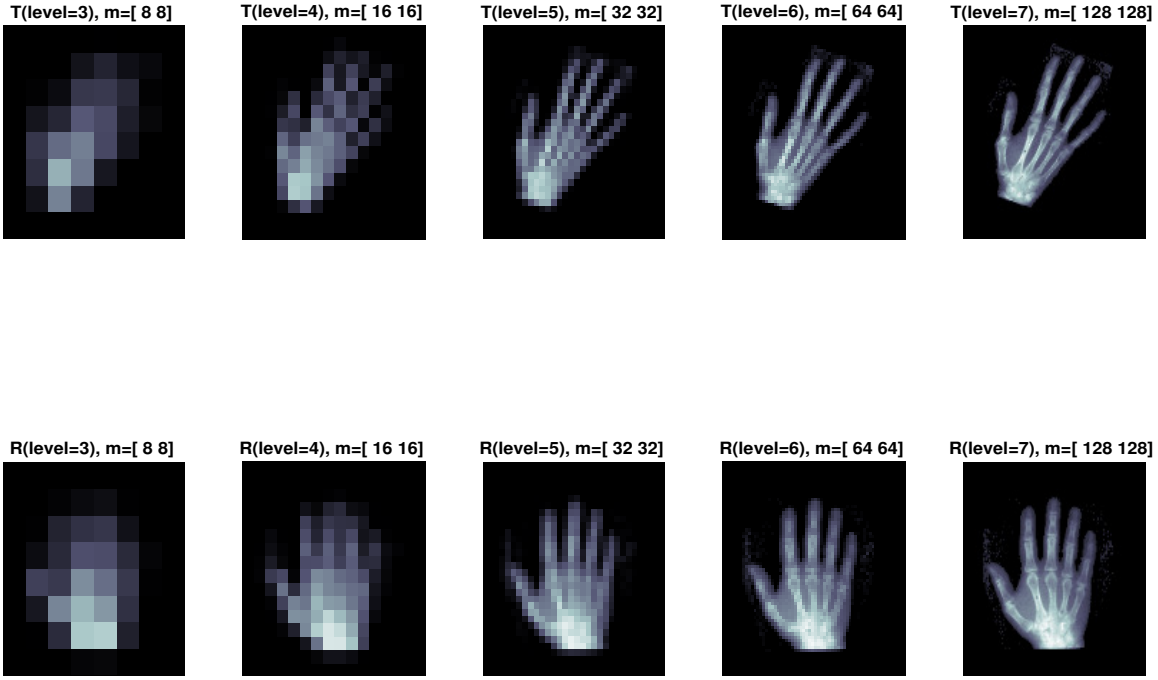



FIGURE 3.3: An example of the multilevel representation of a pair of 2D images.

## 3.7 Conclusion

In this chapter, we build the variational framework for image registration, and the relevant details are reviewed, including fidelity terms, regularizers, numerical algorithm, interpolation, and multilevel strategy.

However, when the deformation is large, many variational models, including the popular diffusion model, cannot ensure diffeomorphism. One common observation is that the fidelity error appears small while the obtained transformation is incorrect by way of mesh folding. Hence, recently, more and more researchers have focused on diffeomorphic

[1] https://github.com/C4IR/FAIR.m

image registration where the derived transformation is continuously differentiable, and it also has a continuously differentiable inverse. In the next chapter, we will propose a novel technique to ensure the diffeomorphism motivated by the Beltrami coefficient.

# Chapter 4

# A Novel Diffeomorphic Model for Image Registration and Its Algorithm

In this chapter, we first propose a new variational model with a special regularizer, based on the quasi-conformal theory, which can guarantee that the registration map is diffeomorphic [135]. We then propose an iterative method to solve the resulting nonlinear optimization problem and prove the convergence of the method. Numerical experiments can demonstrate that the new model can not only get a diffeomorphic registration even when the deformation is large, but also possess the accuracy in comparison with the current best models.

## 4.1 Introduction

Over the last decade, more and more researchers have focused on diffeomorphic image registration where folding measured by the local invertibility quantity $\det(\nabla \boldsymbol{y})$ is reduced or avoided. Here, $\boldsymbol{y}$ denotes the transformation in the registration model and $\det(\nabla \boldsymbol{y})$ is the Jacobian determinant of $\boldsymbol{y}$. Under desired assumptions, obtaining a one-to-one mapping is a natural choice as reviewed in [114].

In 2004, Haber and Modersitzki [50] proposed an image registration model imposing volume preserving constraints, by ensuring $\det(\nabla \boldsymbol{y})$ is close to 1. Although volume preservation is critical in some applications where some underlying (e.g., anatomical) structure is known to be incompressible [114], it is not required or reasonable in others. In a later work, the same authors [53] relaxed the constraint to allow $\det(\nabla \boldsymbol{y})$ to lie in a specific interval. Yanovsky et al. [132] applied the symmetric Kullback-Leibler distance to quantify $\det(\nabla \boldsymbol{y})$ to achieve a diffeomorphic mapping. Burger et al. [11] designed a

volume penalty term ensured that shrinkage and growth had the same cost in their variational functional. The constrained hierarchical parametric approach [94] ensures that the mapping is globally one-to-one and thus preserves topology in the deformed image. Sdika [111] introduced a regularizer to penalize the non-invertible transformation. In [124], Vercauteren et al. proposed an efficient non-parametric diffeomorphic image registration algorithm based on Thirion's demons algorithm [118]. In addition, a framework called Large Deformation Diffeomorphic Metric Mapping (LDDMM) can generate the diffeomorphic transformation for image registration [4, 34, 88, 122]. An entirely different framework proposed by Lam and Lui [77] obtains diffeomorphic registrations by constraining Beltrami coefficients of a quasi-conformal map $f = y_1(\boldsymbol{x}) + \boldsymbol{i}y_2(\boldsymbol{x})$, instead of controlling the map $\boldsymbol{y}(\boldsymbol{x})$ directly.

In this chapter, we aim to reformulate the Lam and Lui's Beltrami measure as a direct regularizer for controlling $\det(\nabla \boldsymbol{y})$ and to assess the effectiveness of the resulting variational models; though the idea applies to any commonly used models, we apply it to the diffusion model as one simple example. Our contributions are two-fold:

- We propose a new Beltrami coefficient based regularizer that is explicitly expressed in terms of $\det(\nabla \boldsymbol{y})$. This establishes a link between the Beltrami coefficient of the transformation and the quantity $\det(\nabla \boldsymbol{y})$.
- An effective, iterative scheme is presented, and numerical experimental results show that the new registration model has a good performance and produces a diffeomorphic mapping while remaining competitive to state-of-the-art models from non-Beltrami frameworks.

We remark that several interesting works that are concerned with reversible transformations (such as [17, 131]) may also benefit from this study.

The rest of the chapter is organized as follows. Section 4.2 briefly reviews how to get a diffeomorphic transformation for image registration. In Section 4.3, we propose a new regularizer and a new registration model. The effective discretization and numerical scheme are discussed in Section 4.4. The results of numerical experiments are shown in Section 4.5, and finally a summary is concluded in Section 4.6.

## 4.2 Diffeomorphic Transformation

In Section 3.1, we have built the variational framework for image registration:

$$\min_{\boldsymbol{u}} \mathcal{J}(\boldsymbol{u}) = \mathcal{D}(T(\boldsymbol{x}+\boldsymbol{u}), R) + \alpha \mathcal{R}(\boldsymbol{u}), \tag{4.1}$$

where $\mathcal{D}(T(\boldsymbol{x}+\boldsymbol{u}), R)$ is the distance measure, $\mathcal{R}(\boldsymbol{u})$ is the regularizer and $\alpha$ is a positive parameter to balance these two terms.

However, if we just solve (4.1) when $\mathcal{D}$ is SSD (3.4) and $\mathcal{R}$ is the diffusion regularizer (3.10), the obtained solution $\boldsymbol{u}$ or $\boldsymbol{y}$ is mathematically correct but often incorrect physically. This is due to no guarantee of mesh non-folding which is measured by $\det(\nabla \boldsymbol{y}) > 0$, i.e., a positive determinant of the local Jacobian matrix $\nabla \boldsymbol{y}$ of the transformation $\boldsymbol{y}$.

To achieve $\det(\nabla \boldsymbol{y}) > 0$, one can find several recent works that impose this constraint in some direct way. We review a few of such models before we present our new constraint. In the form of (4.1), the idea is to choose $\mathcal{R}_1(\cdot)$ as a penalty or a soft constraint to control $\det(\nabla \boldsymbol{y})$ in the following (note $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{u}$)

$$\min_{\boldsymbol{u}} \mathcal{J}(\boldsymbol{u}) = \mathcal{D}(T(\boldsymbol{x} + \boldsymbol{u}), R) + \alpha \mathcal{R}(\boldsymbol{u}) + \beta \mathcal{R}_1(\boldsymbol{y}). \tag{4.2}$$

**Volume Control**. In 2004, Haber and Modersitzki [50] used volume preserving constraint (area in 2D) for image registration, namely

$$\det(\nabla \boldsymbol{y}) = 1.$$

As a consequence, we can ensure that the transformation is diffeomorphic. However, volume preservation is not desirable when the anatomical structure is compressible in medical imaging.

**Slack Constraint**. Improving on [53], the constraint $\det(\nabla \boldsymbol{y}) = 1$ is relaxed and a slack constraint is proposed

$$a \leq \det(\nabla \boldsymbol{y}) \leq b,$$

where a positive interval $[a, b]$ is provided by the user as prior information in the specific application e.g. $[a, b] = [0.1, 2]$.

**Unbiased Transform**. In [132], according to information theory, $\det(\nabla \boldsymbol{y})$ is controlled by the symmetric Kullback-Leibler distance

$$\int_{\Omega} (\det(\nabla \boldsymbol{y}) - 1) \log(\det(\nabla \boldsymbol{y})) \mathrm{d}\boldsymbol{x}.$$

It can help to get an unbiased diffeomorphic transformation. This idea was tested with the fluid regularizer (first order).

**Optimal Mass Transport**. Let $\Omega_0$ and $\Omega_1$ be two subdomains of $\mathbb{R}^d$, with smooth boundaries, each with a positive density function, $\sigma_0$ and $\sigma_1$, respectively. Assume that

$$\int_{\Omega_0} \sigma_0 = \int_{\Omega_1} \sigma_1 \tag{4.3}$$

so that the same total mass is associated with $\Omega_0$ and $\Omega_1$. Consider a diffeomorphism $\boldsymbol{y}$ from $\Omega_0$ to $\Omega_1$ which maps one density to the other in the sense that

$$\sigma_0 = (\det(\nabla \boldsymbol{y}))\sigma_1 \circ \boldsymbol{y}, \tag{4.4}$$

which is called the mass preservation (MP) property. In order to find the optimal MP map, [56] defines the $L^p$ Kantorovich-Wasserstein metric as follows:

$$\min_{\boldsymbol{y}\in \mathrm{MP}} \int_{\Omega_0} \|\boldsymbol{y}-\boldsymbol{x}\|^p \sigma_0 \mathrm{d}\boldsymbol{x}. \tag{4.5}$$

An optimal MP map, when it exists, is an MP map which minimizes (4.5). Here, we notice that if the density is 1, the MP problem degenerates to the VP (volume preservation) problem.

**Balance of Shrinkage and Growth**. Geometrically $\det(\nabla \boldsymbol{y}) = 1$ implies volume preservation. Similarly $\det(\nabla \boldsymbol{y}) < 1$ implies shrinkage while $\det(\nabla \boldsymbol{y}) > 1$ implies growth. A function that treats the cases of shrinkage and growth identically is $\phi(v) = ((v-1)^2/v)^2$ since $\phi(1/v) = \phi(v)$. A volume penalty

$$\int_\Omega \left( \frac{(\det(\nabla \boldsymbol{y})-1)^2}{\det(\nabla \boldsymbol{y})} \right)^2 \mathrm{d}\boldsymbol{x} \tag{4.6}$$

is used in the hyperelastic model [11], which ensures that shrinkage and growth have the same price.

**LDDMM Framework**. The variational formulation of large deformation diffeomorphic metric mapping (LDDMM) [4, 34, 88, 114, 122] is defined by:

$$\begin{aligned} &\min_{\mathcal{T},v} \mathcal{D}(\mathcal{T}(\cdot,1),R) + \alpha \mathcal{R}(v) \\ &\text{s.t.} \quad \partial_t \mathcal{T}(\boldsymbol{x},t) + v(\boldsymbol{x},t)\cdot\nabla\mathcal{T}(\boldsymbol{x},t) = 0 \quad \text{and} \quad \mathcal{T}(\boldsymbol{x},0) = T, \end{aligned} \tag{4.7}$$

where $v : \Omega \times [0,1] \to \mathbb{R}^d$ is the velocity and $\mathcal{T} : \Omega \times [0,1] \to \mathbb{R}$ is a series of images. Here, LDDMM regularizes the velocity $v$ and we can compute its corresponding transformation $\boldsymbol{y}$ by using the information of $v$. When $v$ is sufficiently smooth, it can lead to a diffeomorphic transformation $\boldsymbol{y}$, namely $\det(\nabla \boldsymbol{y}) > 0$. However, since LDDMM involves the transport equation, the time $t$ is introduced, and the dimension of the original problem is increased. Hence, designing an efficient solver for LDDMM is very difficult.

**Diffeomorphic Demons (DDemons)**. [124] presents an efficient non-parametric diffeomorphic image registration algorithm based on Thirion's demons algorithm [118]. The basic idea is to adopt the optimization procedure underlaying the demons algorithm to a space of diffeomorphic transformations. Its iterations are as follows:

1. Given the current transformation $\boldsymbol{y}$, compute a correspondence update field $\boldsymbol{u}$ by minimizing $\|T(\boldsymbol{y}(\exp(\boldsymbol{u}))) - R\|^2 + \frac{\sigma_i^2}{\sigma_x^2}\|\boldsymbol{y} - \boldsymbol{y}(\exp(\boldsymbol{u}))\|^2$with respect to $\boldsymbol{u}$.
2. For a fluid-like regularization let $\boldsymbol{u} \leftarrow K_{\text{fluid}} \star \boldsymbol{u}$. The convolution kernel will typically be a Gaussian kernel.
3. Let $\boldsymbol{c} \leftarrow \boldsymbol{y}(\exp(\boldsymbol{u}))$.

4. For a diffusion-like regularization let $\boldsymbol{y} \leftarrow K_{\text{diff}} \star \boldsymbol{c}$ (else let $\boldsymbol{y} \leftarrow \boldsymbol{c}$). The convolution kernel will also typically be a Gaussian kernel.

**Beltrami Indirect Control**. In 2014, Lam and Lui [77] presented a novel approach in a Beltrami framework to obtain diffeomorphic registrations with large deformations using landmark and intensity information via quasi-conformal maps. Before introducing this model, we first describe some fundamental theories about the quasi-conformal map and Beltrami coefficient.

A complex map $z = x_1 + \boldsymbol{i}x_2 \longmapsto f(z) = y_1(x_1, x_2) + \boldsymbol{i}y_2(x_1, x_2)$ from a domain in $\mathbb{C}$ onto another domain is quasi-conformal if it has continuous partial derivatives and satisfies the following Beltrami equation:

$$\frac{\partial f}{\partial \bar{z}} = \mu(f)\frac{\partial f}{\partial z}, \tag{4.8}$$

for some complex-valued Lebesgue measurable $\mu$ [5] satisfying $\|\mu\|_\infty < 1$. Here $\mu = \mu(\boldsymbol{y}) \equiv f_{\bar{z}}/f_z$ is called the Beltrami coefficient explicitly computable from $\boldsymbol{y}$ since

$$\begin{cases} f_z = \dfrac{\partial f}{\partial z} \equiv \dfrac{1}{2}\Big(\dfrac{\partial f}{\partial x_1} - \boldsymbol{i}\dfrac{\partial f}{\partial x_2}\Big) = \dfrac{(y_1)_{x_1} + (y_2)_{x_2}}{2} + \boldsymbol{i}\dfrac{(y_2)_{x_1} - (y_1)_{x_2}}{2}, \\ f_{\bar{z}} = \dfrac{\partial f}{\partial \bar{z}} \equiv \dfrac{1}{2}\Big(\dfrac{\partial f}{\partial x_1} + \boldsymbol{i}\dfrac{\partial f}{\partial x_2}\Big) = \dfrac{(y_1)_{x_1} - (y_2)_{x_2}}{2} + \boldsymbol{i}\dfrac{(y_2)_{x_1} + (y_1)_{x_2}}{2}, \end{cases} \tag{4.9}$$

where $(y_1)_{x_1} = \partial y_1/\partial x_1$. Conversely $\boldsymbol{y} = \boldsymbol{y}^\mu$ can be computed for a given $\mu$ through solving $\mu(\boldsymbol{y}) = \mu$.

A quasi-conformal map is a homeomorphism (in particular one-to-one) and its first-order approximation takes small circles to small ellipses of bounded eccentricity [44]. As a special case, $\mu = 0$ means that the map $f$ is holomorphic and conformal, characterized by $f_{\bar{z}} = 0$ or $y_1, y_2$ satisfying the Cauchy-Riemann equations $(y_1)_{x_1} = (y_2)_{x_2}, (y_1)_{x_2} = -(y_2)_{x_1}$. For more details about quasi-conformal theory, please refer to [1, 44, 80].

Thus in the context of image registration, enforcing $\|\mu\|_\infty < 1$ provides the control for the transformation $\boldsymbol{y}$ and ensures homeomorphism. The quasi-conformal hybrid registration model (QCHR) in [77] is

$$\min_{\boldsymbol{y}} \int_\Omega |\nabla\mu|^2 + \alpha \int_\Omega |\mu|^2 + \beta \int_\Omega (T(\boldsymbol{y}) - R)^2 \tag{4.10}$$

subject to $\boldsymbol{y} = (y_1, y_2)$ satisfying

1). $\mu = \mu(\boldsymbol{y})$;

2). $\boldsymbol{y}(p_j) = q_j$ for $1 \le j \le m$ (landmark constraints);

3). $\|\mu(\boldsymbol{y})\|_\infty < 1$ (bijectivity);

which indirectly controls $\det(\nabla \boldsymbol{y})$ via the Beltrami coefficient, where $\mu(\boldsymbol{y})$ is the Beltrami coefficient of the transformation $\boldsymbol{y}$. The above model is solved by a penalty splitting method. It minimizes the following functional:

$$\int_\Omega |\nabla \nu|^2 + \alpha \int_\Omega |\nu|^p + \sigma \int_\Omega |\nu - \mu|^2 + \beta \int_\Omega (T(\boldsymbol{y}^\mu) - R)^2 \tag{4.11}$$

subject to the constraints that $\|\nu\|_\infty < 1$ and $\boldsymbol{y}^\mu$ is the quasi-conformal map with Beltrami coefficient $\mu$ satisfying $\boldsymbol{y}^\mu(p_j) = q_j$ for $1 \leq j \leq m$. Then in each iteration, it needs to solve the following two subproblems alternately:

$$\begin{aligned} \mu_{n+1} = &\arg\min \sigma \int_\Omega |\mu - \nu_n|^2 + \beta \int_\Omega (T(\boldsymbol{y}^\mu) - R)^2 \\ &\text{s.t.} \quad \boldsymbol{y}^\mu(p_j) = q_j \text{ for } 1 \leq j \leq m \end{aligned} \tag{4.12}$$

and

$$\nu_{n+1} = \arg\min \int_\Omega |\nabla \nu|^2 + \alpha \int_\Omega |\nu|^p + \sigma \int_\Omega |\nu - \mu_{n+1}|^2. \tag{4.13}$$

In addition, it also solves the equation $\mu(\boldsymbol{y}) = \mu$ by the linear Beltrami solver (LBS) [83] to find $\boldsymbol{y}$ and ensure that $\boldsymbol{y}$ matches the landmark constraints.

Thus, instead of controlling the Jacobian determinant of the transformation directly, controlling the Beltrami coefficient is also a good alternative providing the same but indirect control. However, since their algorithm [77] has to deal with two main unknowns (the transformation $\boldsymbol{y}$ and its Beltrami coefficient $\mu$), and one auxiliary unknown (the coefficient $\nu$) in a non-convex formulation, the increased cost, practical implementation, and convergence are real issues; for challenging problems, one cannot observe convergence and therefore, the full capability of the model is not realized.

We are motivated to reduce the unknowns and simplify their algorithm. Our solution is to reformulate the problem in the space of the primary variable $\boldsymbol{y}$ or $\boldsymbol{u}$, not in the transformed space of variables $\mu, \nu$. We make use of the explicit formula of $\mu = \mu(\boldsymbol{y})$. Working with the primal mapping $\boldsymbol{y}$ enables us to introduce the advantages of minimizing a Beltrami coefficient to the above reviewed variational framework (4.1), effectively unifying the two frameworks.

Hence, we propose a new regularizer based Beltrami coefficient and, in the numerical part, we can find that it is easy to be implemented. Moreover, the reformulated control regularizer can potentially be applied to a large class of variational models.

## 4.3 The Proposed Image Registration Model

In this section, we aim to present a new regularizer based on Beltrami coefficient, which can help to get a diffeomorphic transformation. Then combining the new regularizer

with the diffusion model (3.22), we present a novel model. Of course, combining with other models may be studied as well since the idea is the same.

For $f(z) = y_1(x_1, x_2) + \boldsymbol{i}y_2(x_1, x_2)$, according to the Beltrami equation (4.8) and the definitions (4.9), we have

$$\mu(f) = \frac{\partial f}{\partial \bar{z}} \Big/ \frac{\partial f}{\partial z} = \frac{((y_1)_{x_1} - (y_2)_{x_2}) + \boldsymbol{i}((y_2)_{x_1} + (y_1)_{x_2})}{((y_1)_{x_1} + (y_2)_{x_2}) + \boldsymbol{i}((y_2)_{x_1} - (y_1)_{x_2})}, \tag{4.14}$$

$$|\mu(f)|^2 = \frac{((y_1)_{x_1} - (y_2)_{x_2})^2 + ((y_2)_{x_1} + (y_1)_{x_2})^2}{((y_1)_{x_1} + (y_2)_{x_2})^2 + ((y_2)_{x_1} - (y_1)_{x_2})^2} = \frac{\|\nabla \boldsymbol{f}\|_2^2 - 2\det(\nabla \boldsymbol{f})}{\|\nabla \boldsymbol{f}\|_2^2 + 2\det(\nabla \boldsymbol{f})}, \tag{4.15}$$

where $\boldsymbol{f} = (y_1(x_1, x_2), y_2(x_1, x_2))$. Note $(y_1)_{x_1}(y_2)_{x_2} - (y_2)_{x_1}(y_1)_{x_2} = \det(\nabla \boldsymbol{f})$. So $\det(\nabla \boldsymbol{f})$ can be represented by the Beltrami coefficient $\mu(f)$

$$\det(\nabla \boldsymbol{f}) = |f_z|^2(1 - |\mu(f)|^2) \tag{4.16}$$

Clearly $\det(\nabla \boldsymbol{f}) > 0$ if $|\mu(f)| < 1$, and by the inverse function theorem, the map $f$ is locally bijective. We conclude that $f$ is a diffeomorphism if we assume that $\Omega$ is bounded and simply connected.

### 4.3.1 New Regularizer

Our new regularizer based on $|\mu(f)| < 1$ to control the transformation to get a diffeomorphic mapping is

$$\mathcal{R}_1(\boldsymbol{y}) = \int_\Omega \phi(|\mu|^2)\mathrm{d}\boldsymbol{x}, \quad |\mu|^2 = \frac{\|\nabla \boldsymbol{y}\|_2^2 - 2\det(\nabla \boldsymbol{y})}{\|\nabla \boldsymbol{y}\|_2^2 + 2\det(\nabla \boldsymbol{y})} \tag{4.17}$$

which clearly involves the Jacobian determinant of the transformation $\det(\nabla \boldsymbol{y})$ in a non-trivial way and we explore the choices of $\phi$ below.

*Remark* 4.1. Our new regularizer has two advantages: one is that the obtained transformation $\boldsymbol{y}$ do not need to possess $\det(\nabla \boldsymbol{y}) \to 1$; the other one is that we only compute the transformation and do not need to compute its Beltrami coefficient and introduce another auxiliary unknown as [77]. Also, from the numerical experiments, we can see that our new regularizer is easy to be implemented and obtains accurate and diffeomorphic transformations.

### 4.3.2 The Proposed Model

The above regularizer (4.17) providing a constraint on $\boldsymbol{y}$ is ready to be combined with an existing model. In the framework (4.2), using (4.17), the first version of our new model takes the form

$$\min_{\boldsymbol{y}} \frac{1}{2}\int_\Omega (T(\boldsymbol{y}) - R)^2 \mathrm{d}\boldsymbol{x} + \frac{\alpha}{2}\int_\Omega \sum_{l=1}^{2} |\nabla u_l|^2 \mathrm{d}\boldsymbol{x} + \beta \int_\Omega \phi(|\mu|^2)\mathrm{d}\boldsymbol{x} \tag{4.18}$$

where $\boldsymbol{u} = \boldsymbol{y}(\boldsymbol{x}) - \boldsymbol{x}$ is the deformation field and $\mu = \mu(\boldsymbol{y})$. To promote $|\mu(f)| < 1$, our first and simple choice is $\phi(v) = \phi_1(v) = \frac{1}{(v-1)^2}$, which forces (4.18) and $\phi(v)$ to reduce $v$, at the initial guess $v = 0$ when $\boldsymbol{u} = \boldsymbol{0}$, since $\phi_1(v) \to \infty$ when $v \to 1$.

*Remark* 4.2. From (4.10) and (4.18), we see that the QCHR model focuses on obtaining a smooth Beltrami coefficient, and our model focuses on the diffeomorphic transformation itself. There are major differences between the regularizer in QCHR model and our new regularizer: the former is characterized by the Beltrami coefficient $\mu$ directly and gradient of this Beltrami coefficient $\mu$, while the latter is characterized by the Beltrami coefficient indirectly in terms of the transformation $\boldsymbol{y}$ and the gradient of $\boldsymbol{u}$. Since $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{u}$ is our desired transformation, our direct regularizers such as $|\nabla \boldsymbol{u}|^2$ make more sense than indirect regularizers such as $|\nabla \mu|^2$.

However as long as $|\mu(f)| < 1$, we would not give a preference to forcing $|\mu(f)| \to 0$. To put some control on bias, similarly to [11], we are led to 2 more choices of a less unbiased function to modify $\mathcal{R}_1(\boldsymbol{y})$

- $\phi(v) = \phi_2(v) = \frac{v}{(v-1)^2}$: balance $|\mu(f)|$ between 0 and 1 as $\phi_2(v) = \phi_2(1/v)$;
- $\phi(v) = \phi_3(v) = \frac{v^2}{(v-1)^2}$: encourage $|\mu(f)| \to 0$ and $|\mu(f)| \neq 1$;

Below, we list first order derivatives and second order derivatives for the above different $\phi(v)$:

- $\phi_1'(v) = \frac{2}{(v-1)^3}$ and $\phi_1''(v) = \frac{6}{(v-1)^4}$;
- $\phi_2'(v) = -\frac{v+1}{(v-1)^2}$ and $\phi_2''(v) = \frac{2v+4}{(v-1)^4}$;
- $\phi_3'(v) = -\frac{2v}{(v-1)^3}$ and $\phi_3''(v) = \frac{4v+2}{(v-1)^4}$;

which will be used in subsequent solutions. With a general $\phi(v)$, the second version of our proposed model takes the form:

$$\min_{\boldsymbol{u}} \frac{1}{2} \int_\Omega (T(\boldsymbol{x} + \boldsymbol{u}) - R)^2 \mathrm{d}\boldsymbol{x} + \frac{\alpha}{2} \int_\Omega \sum_{l=1}^{2} |\nabla u_l|^2 \mathrm{d}\boldsymbol{x} + \beta \int_\Omega \phi(|\mu|^2) \mathrm{d}\boldsymbol{x}, \tag{4.19}$$

where $|\mu|^2 = \frac{(\partial_{x_1} u_1 - \partial_{x_2} u_2)^2 + (\partial_{x_1} u_2 + \partial_{x_2} u_1)^2}{(\partial_{x_1} u_1 + \partial_{x_2} u_2 + 2)^2 + (\partial_{x_1} u_2 - \partial_{x_2} u_1)^2}$ is written in component form ready for discretization, using $y_1 = x_1 + u_1(x_1, x_2)$, $y_2 = x_2 + u_2(x_1, x_2)$, and $\partial_{x_1} u_1 = \partial u_1 / \partial x_1$.

## 4.4 The Numerical Algorithm

In this section, we will present a numerical algorithm to solve model (4.19). We choose the first-discretize-then-optimize approach. Directly discretizing this variational model gives rise to a finite dimensional optimization problem. Then we use optimization methods to solve this resulting problem.

### 4.4.1 Discretization

We use finite differences to discretize model (4.19) on a unit square domain $\Omega = [0,1]^2$. In implementation, we employ the nodal grid and define a spatial partition $\Omega_h = \{\boldsymbol{x}^{i,j} \in \Omega \mid \boldsymbol{x}^{i,j} = (x_1^i, x_2^j) = (ih, jh), 0 \le i \le n, 0 \le j \le n\}$, where $h = \frac{1}{n}$ and the discrete domain consists of $n^2$ cells of size $h \times h$ (see Section 2.4.1). We discretize the displacement field $\boldsymbol{u}$ on the nodal grid, namely $\boldsymbol{u}^{i,j} = (u_1^{i,j}, u_2^{i,j}) = (u_1(x_1^i, x_2^j), u_2(x_1^i, x_2^j))$. For ease of presentation, according to the lexicographical ordering, we reshape

$$X = (x_1^0, ..., x_1^n, ..., x_1^0, ..., x_1^n, x_2^0, ..., x_2^0, ..., x_2^n, ..., x_2^n)^T \in \mathbb{R}^{2(n+1)^2 \times 1},$$

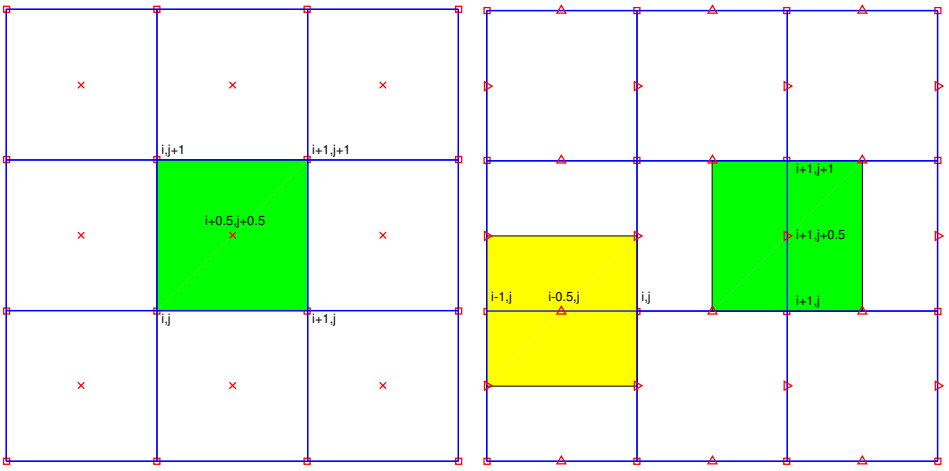
and

$$U = (u_1^{0,0}, ..., u_1^{n,0}, ..., u_1^{0,n}, ..., u_1^{n,n}, u_2^{0,0}, ..., u_2^{n,0}, ..., u_2^{0,n}, ..., u_2^{n,n})^T \in \mathbb{R}^{2(n+1)^2 \times 1}.$$

**Discretization of Term 1 in** (4.19)

According to the cell-centered partition in Figure 4.1(a) and mid-point rule, we get

$$\begin{aligned} \mathcal{D}(T(\boldsymbol{x}+\boldsymbol{u}), R) &:= \frac{1}{2}\int_\Omega (T(\boldsymbol{x}+\boldsymbol{u}(\boldsymbol{x})) - R(\boldsymbol{x}))^2 \mathrm{d}\boldsymbol{x} \\ &\approx \frac{h^2}{2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}(T(\boldsymbol{x}^{i+\frac{1}{2},j+\frac{1}{2}} + \boldsymbol{u}(\boldsymbol{x}^{i+\frac{1}{2},j+\frac{1}{2}})) - R(\boldsymbol{x}^{i+\frac{1}{2},j+\frac{1}{2}}))^2. \end{aligned} \tag{4.20}$$




(a) Illustration of cell-centered partition: Green cell denoted by $\Omega_{i,j}$. Nodal Grid □

(b) Partition for $\partial_x$ and $\partial_y$. The left yellow cell is $\Omega_{i,j}^{x_1}$ and the right green cell is $\Omega_{i,j}^{x_2}$.

FIGURE 4.1: Partition of domain $\Omega = \cup_{ij}\Omega_{i,j}$. Note that solutions $u_1$ and $u_2$ are defined at nodes.

Set $\vec{R} = R(PX) \in \mathbb{R}^{n^2 \times 1}$ as the discretized reference image and $\vec{T}(PX+PU) \in \mathbb{R}^{n^2 \times 1}$ as the discretized deformed template image, where $P \in \mathbb{R}^{2n^2 \times 2(n+1)^2}$ is an averaging matrix

for the transfer from the nodal grid representation of $U$ to the cell centered positions [50, 53].

Consequently, for SSD, we obtain the following discretization:

$$\mathcal{D}(T(\boldsymbol{x}+\boldsymbol{u}),R) \approx \frac{h^2}{2}(\vec{T}(PX+PU)-\vec{R})^T(\vec{T}(PX+PU)-\vec{R}). \tag{4.21}$$

**Discretization of Term 2 in** (4.19)

For the diffusion regularizer,

$$\mathcal{R}^{\text{Diff}}(\boldsymbol{u}) := \frac{\alpha}{2}\int_\Omega \sum_{l=1}^{2} |\nabla u_l|^2 \mathrm{d}\boldsymbol{x}, \tag{4.22}$$

according to the the partition in Figure 4.1(b) and mid-point rule, we have

$$\int_{\Omega_{i,j}^{x_1}} |\partial_{x_1} u_l|^2 \mathrm{d}\boldsymbol{x} \approx h^2(\partial_{x_1}^{i+\frac{1}{2},j} u_l)^2 \qquad 1 \le j \le n-1, \tag{4.23}$$

or at the boundary half-boxes

$$\int_{\Omega_{i,j}^{x_1}} |\partial_{x_1} u_l|^2 \mathrm{d}\boldsymbol{x} \approx \frac{h^2}{2}(\partial_{x_1}^{i+\frac{1}{2},j} u_l)^2 \qquad j=0,n. \tag{4.24}$$

And for $\int_{\Omega_{i,j}^{x_2}} |\partial_{x_2} u_l|^2 \mathrm{d}\boldsymbol{x},\ l=1,2$, we have similar results.

As designed, we use compact (short) difference schemes to compute the $\partial_{x_1} u_l$ and $\partial_{x_2} u_l,\ l=1,2$:

$$\partial_{x_1}^{i+\frac{1}{2},j} u_l \approx \frac{u_l^{i+1,j}-u_l^{i,j}}{h}, \qquad \partial_{x_2}^{i,j+\frac{1}{2}} u_l \approx \frac{u_l^{i,j+1}-u_l^{i,j}}{h}. \tag{4.25}$$

Then (4.22) can be rewritten in the following formulation:

$$\mathcal{R}^{\text{Diff}}(\boldsymbol{u}) \approx \frac{\alpha h^2}{2} U^T A^T G A U. \tag{4.26}$$

See Appendix 4.7.1 for details on $A$ and $G$.

*Remark* 4.3. Note that here the matrix $A$ is the discretized gradient matrix. So $A^TGA$ is the discretized Laplace matrix.

**Discretization of Term 3 in** (4.19)

For simplicity, denote $|\mu(\boldsymbol{y})| = |\mu(\boldsymbol{x}+\boldsymbol{u})|$ by $|\mu(\boldsymbol{u})|$. From (4.19), note that $\phi(|\mu(\boldsymbol{u})|^2)$ involves only first order derivatives and all $\boldsymbol{u}^{i,j}$ are available at vertex pixels. Thus it is convenient first to obtain approximations at all cell centers (e.g. at $V_5$ in Figure 4.2) and second to use local linear elements to facilitate first order derivatives. We

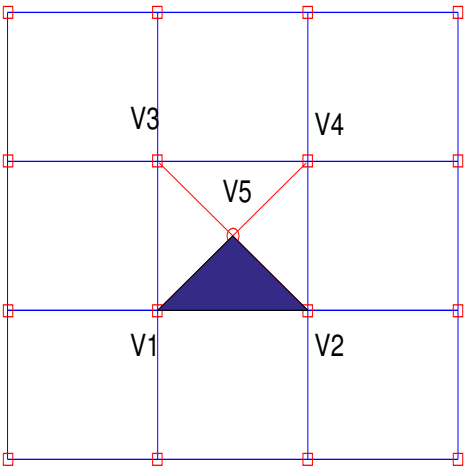



FIGURE 4.2: Partition of a cell, nodal point □ and center point ∘. $\triangle V_1V_2V_5$ is $\Omega_{i,j,k}$.

shall divide each cell (Figure 4.2) into 4 triangles. In each triangle, we construct two linear interpolation functions to approximate the $u_1$ and $u_2$. Consequently, all partial derivatives are locally constants or $\phi(|\mu(\boldsymbol{u})|^2)$ is constant in each triangle.

According to the partition in Figure 4.2, we get

$$\mathcal{R}^{\text{Beltrami}}(\boldsymbol{u}) := \beta \int_{\Omega} \phi(|\mu(\boldsymbol{u})|^2)\mathrm{d}\boldsymbol{x} = \beta \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{4} \int_{\Omega_{i,j,k}} \phi(|\mu(\boldsymbol{u})|^2)\mathrm{d}\boldsymbol{x}. \tag{4.27}$$

Set $\boldsymbol{L}^{i,j,k}(\boldsymbol{x}) = (L_1^{i,j,k}(\boldsymbol{x}), L_2^{i,j,k}(\boldsymbol{x})) = (a_1^{i,j,k}x_1 + a_2^{i,j,k}x_2 + b_1^{i,j,k}, a_3^{i,j,k}x_1 + a_4^{i,j,k}x_2 + b_2^{i,j,k})$, which is the linear interpolation for $\boldsymbol{u}$ in the $\Omega_{i,j,k}$. Note that $\partial_{x_1}L_1^{i,j,k} = a_1^{i,j,k}, \partial_{x_2}L_1^{i,j,k} = a_2^{i,j,k}, \partial_{x_1}L_2^{i,j,k} = a_3^{i,j,k}$ and $\partial_{x_2}L_2^{i,j,k} = a_4^{i,j,k}$. According to (4.19), the discretization of Beltrami regularizer can be written into following:

$$\mathcal{R}^{\text{Beltrami}}(\boldsymbol{u}) \approx \frac{\beta h^2}{4} \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{4} \phi\left(\frac{(a_1^{i,j,k} - a_4^{i,j,k})^2 + (a_2^{i,j,k} + a_3^{i,j,k})^2}{(a_1^{i,j,k} + a_4^{i,j,k} + 2)^2 + (a_2^{i,j,k} - a_3^{i,j,k})^2}\right). \tag{4.28}$$

To simplify (4.28), define 3 vectors $\vec{\boldsymbol{r}}(U), \vec{\boldsymbol{r}}^1(U), \vec{\boldsymbol{r}}^2(U) \in \mathbb{R}^{4n^2}$ by $\vec{\boldsymbol{r}}(U)_l = \vec{\boldsymbol{r}}^1(U)_l \times \vec{\boldsymbol{r}}^2(U)_l$, $\vec{\boldsymbol{r}}^1(U)_l = (a_1^{i,j,k} - a_4^{i,j,k})^2 + (a_2^{i,j,k} + a_3^{i,j,k})^2$, $\vec{\boldsymbol{r}}^2(U)_l = 1/[(a_1^{i,j,k} + a_4^{i,j,k} + 2)^2 + (a_2^{i,j,k} - a_3^{i,j,k})^2]$ where $l = (k-1)n^2 + (j-1)n + i \ \in [1, 4n^2]$. Hence, (4.28) becomes

$$\mathcal{R}^{\text{Beltrami}}(\boldsymbol{u}) \approx \frac{\beta h^2}{4} \boldsymbol{\phi}(\vec{\boldsymbol{r}}(U))e^T \tag{4.29}$$

where $\boldsymbol{\phi}(\vec{\boldsymbol{r}}(U)) = (\phi(\vec{\boldsymbol{r}}(U)_1), ..., \phi(\vec{\boldsymbol{r}}(U)_{4n^2}))$ denotes the pixel-wise discretization of $u_1, u_2$ at all cell centers, and $e = (1, ..., 1) \in \mathbb{R}^{4n^2}$. Here, $\vec{\boldsymbol{r}}(U)$ is the square of the discretized Beltrami coefficient; we rewrite it in a compact form in Appendix 4.7.2.

Finally, combining the above three parts (4.21), (4.26) and (4.29), we get the discretization formulation for model (4.19):

$$\min_U J(U) := \frac{h^2}{2}(\vec{T}(PX+PU)-\vec{R})^T(\vec{T}(PX+PU)-\vec{R}) + \frac{\alpha h^2}{2}U^TA^TGAU + \frac{\beta h^2}{4}\boldsymbol{\phi}(\vec{\boldsymbol{r}}(U))e^T. \tag{4.30}$$

*Remark* 4.4. Because of the nodal grid scheme, the partition in Figure 4.1 and 4.2 and the structure of the objective functional (4.19), we do not need to introduce the boundary condition leading to (4.30).

*Remark* 4.5. According to the definition of $\phi$ and $\vec{\boldsymbol{r}}(U)_l \geq 0$, each component of $\boldsymbol{\phi}(\vec{\boldsymbol{r}}(U))$ is non-negative and differentiable.

### 4.4.2 Optimization Method for The Discretized Problem (4.30)

In the numerical implementation, we choose a line search method to solve the resulting unconstrained optimization problem (4.30). To guarantee the search direction is a descent direction, we employ the Gauss-Newton direction as the standard direction involving non-definite Hessians do not generate a descent direction. In addition, in Section 2.6, we have mentioned that the Gauss-Newton approach presents two advantages: one is that we do not need to compute the second order term and it can save computation time; the other one is that this Gauss-Newton matrix is more important than the second term, either because of small second order derivatives or because of small residuals [97].

Next, we will investigate the details about the approximated Hessian $\hat{H}(U^k)$, step length $\theta^k$, stopping criteria, and multilevel strategy.

#### Approximated Hessian $\hat{H}$

We consider each of the three terms in $J(U)$ from (4.30) separately.

Firstly, we consider the discretized SSD

$$\frac{h^2}{2}(\vec{T}(PX+PU)-\vec{R})^T(\vec{T}(PX+PU)-\vec{R}). \tag{4.31}$$

Its gradient and Hessian are respectively

$$\begin{cases} d_1 &= h^2P^T\vec{T}_{\tilde{\boldsymbol{U}}}^T(\vec{T}(\tilde{\boldsymbol{U}})-\vec{R}) \in \mathbb{R}^{2(n+1)^2\times 1}, \\ H_1 &= h^2P^T(\vec{T}_{\tilde{\boldsymbol{U}}}^T\vec{T}_{\tilde{\boldsymbol{U}}} + \sum_{l=1}^{n^2}(\vec{T}(\tilde{\boldsymbol{U}})-\vec{R})_l\nabla^2(\vec{T}(\tilde{\boldsymbol{U}})-\vec{R})_l)P \in \mathbb{R}^{2(n+1)^2\times 2(n+1)^2}, \end{cases} \tag{4.32}$$

where $\tilde{\boldsymbol{U}} = PX+PU$ and $\vec{T}_{\tilde{\boldsymbol{U}}} = \frac{\partial \vec{T}(\tilde{\boldsymbol{U}})}{\partial \tilde{\boldsymbol{U}}}$ as the Jacobian of $\vec{T}$ with respect to $\tilde{\boldsymbol{U}}$.

For $H_1$, we cannot ensure that it is positive semi-definite. If it is not positive definite, we may not get a descent direction. So we omit the second order term of $H_1$ to obtain

the approximated Hessian of (4.31):

$$\hat{H}_1 = h^2 P^T (\vec{T}_{\tilde{\boldsymbol{U}}}^T \vec{T}_{\tilde{\boldsymbol{U}}}) P. \tag{4.33}$$

*Remark* 4.6. Evaluation of the deformed template image $T$ must involve interpolation because $\tilde{\boldsymbol{U}}$ does not in general correspond with pixel points; in our implementation, as with [92], we use B-splines interpolation to get $\vec{T}(\tilde{\boldsymbol{U}})$ (see Section 3.5).

Secondly, for the discretized diffusion regularizer $\frac{\alpha h^2}{2} U^T A^T G A U$, its gradient and Hessian are the following:

$$\begin{cases} d_2 &= \alpha h^2 A^T G A U \in \mathbb{R}^{2(n+1)^2 \times 1}, \\ H_2 &= \alpha h^2 A^T G A \in \mathbb{R}^{2(n+1)^2 \times 2(n+1)^2}. \end{cases} \tag{4.34}$$

Since $H_2$ is positive definite when $U$ is applied with Dirichlet boundary conditions, we do not approximate it.

Finally, for the discretized Beltrami term

$$\frac{\beta h^2}{4} \boldsymbol{\phi}(\vec{\boldsymbol{r}}(U)) e^T, \tag{4.35}$$

the gradient and the Hessian are as follows:

$$\begin{cases} d_3 &= \frac{\beta h^2}{4} \mathrm{d}\vec{\boldsymbol{r}}^T \mathrm{d}\boldsymbol{\phi}(\vec{\boldsymbol{r}}) \in \mathbb{R}^{2(n+1)^2 \times 1}, \\ H_3 &= \frac{\beta h^2}{4} (\mathrm{d}\vec{\boldsymbol{r}}^T \mathrm{d}^2 \boldsymbol{\phi}(\vec{\boldsymbol{r}}) \mathrm{d}\vec{\boldsymbol{r}} + \sum_{l=1}^{4n^2} [\mathrm{d}\boldsymbol{\phi}(\vec{\boldsymbol{r}})]_l \nabla^2 \vec{\boldsymbol{r}}_l) \in \mathbb{R}^{2(n+1)^2 \times 2(n+1)^2} \end{cases} \tag{4.36}$$

where $\mathrm{d}\boldsymbol{\phi}(\vec{\boldsymbol{r}}) = (\phi'(\vec{\boldsymbol{r}}_1), ..., \phi'(\vec{\boldsymbol{r}}_{4n^2}))^T$ is the vector of derivatives of $\boldsymbol{\phi}$ at all cell centers,

$$\begin{cases} \mathrm{d}\vec{\boldsymbol{r}} &= \mathrm{diag}(\vec{\boldsymbol{r}}^1) \mathrm{d}\vec{\boldsymbol{r}}^2 + \mathrm{diag}(\vec{\boldsymbol{r}}^2) \mathrm{d}\vec{\boldsymbol{r}}^1, \\ \mathrm{d}\vec{\boldsymbol{r}}^1 &= 2\mathrm{diag}(A_1 U) A_1 + 2\mathrm{diag}(A_2 U) A_2, \\ \mathrm{d}\vec{\boldsymbol{r}}^2 &= -\mathrm{diag}(\vec{\boldsymbol{r}}^2 \odot \vec{\boldsymbol{r}}^2)[2\mathrm{diag}(A_3 U + 2) A_3 + 2\mathrm{diag}(A_4 U) A_4], \end{cases} \tag{4.37}$$

$\odot$ denotes a Hadamard product, $\mathrm{d}\vec{\boldsymbol{r}}, \mathrm{d}\vec{\boldsymbol{r}}^1, \mathrm{d}\vec{\boldsymbol{r}}^2$ are the Jacobian of $\vec{\boldsymbol{r}}, \vec{\boldsymbol{r}}^1, \vec{\boldsymbol{r}}^2$ with respect to $U$ respectively, $[\mathrm{d}\boldsymbol{\phi}(\vec{\boldsymbol{r}})]_l$ is the $l$th component of $\mathrm{d}\boldsymbol{\phi}(\vec{\boldsymbol{r}})$ and $\mathrm{d}^2\boldsymbol{\phi}(\vec{\boldsymbol{r}})$ is the Hessian of $\boldsymbol{\phi}$ with respect to $\vec{\boldsymbol{r}}$, which is a diagonal matrix whose $i$th diagonal element is $\phi''(\vec{\boldsymbol{r}}_i)$, $1 \leq i \leq 4n^2$. Here $\mathrm{diag}(v)$ is a diagonal matrix with $v$ on its main diagonal. More details about $\vec{\boldsymbol{r}}^1$, $\vec{\boldsymbol{r}}^2$, $A_1$, $A_2$, $A_3$ and $A_4$ are shown in Appendix 4.7.2 and some illustration of our notation is given in Appendix 4.7.3.

To extract a positive semi-definite part out of (4.36), we omit the second order term and obtain the approximated Hessian as

$$\hat{H}_3 = \frac{\beta h^2}{4} \mathrm{d}\vec{\boldsymbol{r}}^T \mathrm{d}^2 \boldsymbol{\phi}(\vec{\boldsymbol{r}}) \mathrm{d}\vec{\boldsymbol{r}}. \tag{4.38}$$

Therefore for functional $J(U)$ in (4.30) with any choice of $\phi$, we obtain its gradient

$$d_J = d_1 + d_2 + d_3 \tag{4.39}$$

and approximated Hessian:

$$\hat{H} = \hat{H}_1 + H_2 + \hat{H}_3. \tag{4.40}$$

**Search Direction**

At each iteration, using (4.39) and (4.40), we need to solve the Gauss-Newton system to find the search direction of (4.30):

$$\hat{H}\delta U = -d_J, \tag{4.41}$$

where $\delta U$ is the search direction. In our implementation, we use MINRES with diagonal preconditioning to solve this linear system [3, 101].

**Step Length**

We use the standard Armijo strategy with backtracking to find a suitable step length $\theta$ (see Section 2.6.1). In the implementation, we also need to check that $\vec{\boldsymbol{r}}(U)$ (4.52) is smaller than 1. Recall that $\vec{\boldsymbol{r}}(U)$ is the norm square of the discretized Beltrami term. As a safe guard, we choose T0 = $10^{-8}$ and Tol = $10^{-12}$ as the lower bound of the step

length $\theta$ and $\theta\|\delta U\|$ [11, 73, 97, 115]. The algorithm is summarized in Algorithm 9.

**Algorithm 9:** Armijo Line Search: $(U, \mathrm{ID}) \leftarrow \mathrm{ALS}(U, \delta U)$

Step 1: Initialisation;
Set ID $= 0$, $\theta = 1$, Tol$= 10^{-12}$, T0 $= 10^{-8}$ and $\eta = 10^{-4}$. Compute $J(U)$ and $d_J$;
Step 2: Feasibility checking;
**while** $\theta\|\delta U\| \geq Tol$ **do**
    $U^{new} = U + \theta\delta U$;
    **if** $||\vec{\boldsymbol{r}}(U^{new})|| \leq 1$ **then**
        If $\theta \geq$ T0, exit this while loop and go to Step 3, else if $\theta <$ T0, go to Step 4;
    **end**
    Reduce the factor $\theta$ by $\theta = \theta/2$;
**end**
Step 3: Line Search;
**while** $\theta\|\delta U\| \geq Tol$ **do**
    Compute $J(U^{new})$;
    **if** $J(U^{new}) < J(U) + \theta\eta {d_J}^T \delta U$ **then**
        If $\theta \geq$ T0, exit this algorithm with $U = U^{new}$, else if $\theta <$ T0, go to Step 4;
    **end**
    Reduce the factor $\theta$ by $\theta = \theta/2$;
    $U^{new} = U + \theta\delta U$;
**end**
Step 4: Set ID $= 1$ and $U = U^{new}$.

### Stopping Criteria

Here, we adopt the stopping criteria as in [92]:

(1.a) $\|J(U^{k+1}) - J(U^k)\| \leq \tau_J(1 + \|J(U^0)\|)$,

(1.b) $\|U^{k+1} - U^k\| \leq \tau_W(1 + \|U^0 + X\|)$,

(1.c) $\|d_J\| \leq \tau_G(1 + \|J(U^0)\|)$,

(2) $\|d_J\| \leq$ eps,

(3) $k \geq$ MaxIter.

Here, eps is the machine precision and MaxIter is the maximal number of outer iterations. We set $\tau_J = 10^{-3}$, $\tau_W = 10^{-2}$, $\tau_G = 10^{-2}$ and MaxIter$= 500$. If any one of (1) (2) or (3) is satisfied, the iterations are terminated. Hence, a Gauss-Newton numerical scheme with

Armijo line search can be developed. The resulting Gauss-Newton numerical scheme by using Armijo line search is summarized in Algorithm 10.

**Algorithm 10:** Gauss-Newton scheme by using Armijo line search for Image Registration: $(U, \text{ID}) \leftarrow \text{GNAIRA}(\alpha, \beta, U^0, T, R)$

Step 1: Set $k = 0$ at the solution point $U^k = U^0$;
Step 2: For (4.30), compute the energy functional $J(U^k)$, its gradient $d_J^k$ and the approximated Hessian $\hat{H}^k$ by (4.40);
**while** *"none of the listed* 3 *stopping criteria are satisfied"* **do**
    Solve the Gauss-Newton equation: $\hat{H}^k \delta U^k = -d_J^k$;
    $(U^{k+1}, \text{ID}) \leftarrow \text{ALS}(U^k, \delta U^k)$ by Algorithm 9;
    **if** $ID = 1$ **then**
        Exit this algorithm;
    **else**
        $k = k + 1$;
        Compute $J(U^k)$, $d_J^k$ and $\hat{H}^k$;
    **end**
**end**

Next, we discuss the global convergence result of Algorithm 10 for our reformulated problem (4.30). Firstly, we review some relevant theorems.

**Theorem 4.7** ([73])**.** *For the unconstrained optimization problem*

$$\min_U J(U)$$

*let an iterative sequence be defined by* $U^{k+1} = U^k + \theta \delta U^k$*, where* $\delta U^k = -(\hat{H}^k)^{-1} d_J(U^k)$ *and* $\theta$ *is obtained by Algorithm 9. Assume that three conditions are met: (i).* $d_J$ *be Lipschitz continuous; (ii). the matrices* $\hat{H}^k$ *are SPD; (iii). there exist constants* $\bar{\kappa}$ *and* $\zeta$ *such that the condition number* $\kappa(\hat{H}^k) \leq \bar{\kappa}$ *and the norm* $||\hat{H}^k|| \leq \zeta$ *for all* $k$*. Then either* $J(U^k)$ *is unbounded from below or*

$$\lim_{k \to \infty} d_J(U^k) = 0 \tag{4.42}$$

*and hence any limit point of the sequence of iterates is a stationary point.*

*Remark* 4.8. In the following, we will prove our convergence result under the Dirichlet boundary condition (namely, the boundary is fixed) and this condition is needed to prove the symmetric and positive definite (SPD) property of the approximated Hessians. In practical implementation, such a condition is not required as confirmed by experiments.

In addition, define an important set $\mathcal{U} := \{U \mid \vec{\boldsymbol{r}}(U)_l \leq 1 - \epsilon, 1 \leq l \leq 4n^2\}$ for small $\epsilon$. So $U \in \mathcal{U}$ means that the transformation is diffeomorphic. Under suitable $\beta$, we assume that each $U^k$ generated by Algorithm 10 is in $\mathcal{U}$.

Secondly we stage a simple lemma that is needed shortly for studying $\hat{H}^k$.

**Lemma 4.9.** *Let a matrix be comprised of* 3 *submatrices* $H = H_1 + H_2 + H_3$. *If* $H_1$ *and* $H_2$ *are symmetric and positive semi-definite and* $H_3$ *is SPD, then* $H$ *is SPD with* $\lambda_{h_3} \leq \lambda_h$, *where* $\lambda_{h_3}$ *and* $\lambda_h$ *are the minimum eigenvalues of* $H_3$ *and* $H$ *separately.*

*Proof.* According to Rayleigh quotient, we can find a vector $v$ such that

$$\lambda_h = \frac{v^T H v}{v^T v}. \tag{4.43}$$

Then we have

$$\lambda_{h_3} \leq \frac{v^T H_1 v}{v^T v} + \frac{v^T H_2 v}{v^T v} + \frac{v^T H_3 v}{v^T v} = \frac{v^T H v}{v^T v} = \lambda_h, \tag{4.44}$$

which completes the proof. □

**Theorem 4.10.** *Assume that* $T$ *and* $R$ *are twice continuously differentiable. For* (4.30), *when* $\phi = \phi_1, \phi_2$ *or* $\phi_3$, *by using Algorithm 10, we obtain*

$$\lim_{k\to\infty} d_J(U^k) = 0 \tag{4.45}$$

*and hence any limit point of the sequence of iterates produced by Algorithm 10 is a stationary point.*

*Proof.* It suffices to show that Algorithm 10 satisfies the requirements of Theorem 4.7. Recall $\vec{\boldsymbol{r}}(U)$ and we can see that it is continuous. Here, we use the Dirichlet boundary condition and we can assume that $\|U\|$ is bounded. Then $\vec{\boldsymbol{r}}(U)$ is a continuous mapping from a compact set to $\mathbb{R}^{4n^2\times 1}$ and $\vec{\boldsymbol{r}}(U)$ is proper. So for some small $\epsilon > 0, \mathcal{U}$ is compact.

Firstly, we show that in $\mathcal{U}$, $d_J$ of (4.30) is Lipschitz continuous. When $\phi = \phi_1, \phi_2$ or $\phi_3$, the term $\boldsymbol{\phi}(\vec{\boldsymbol{r}}(U))e^T$ in the (4.30) is twice continuously differentiable with respect to $U \in \mathcal{U}$. In addition, $T$ and $R$ are twice continuously differentiable. So (4.30) is twice continuously differentiable with respect to $U \in \mathcal{U}$ and $d_J$ is Lipschitz continuous.

Secondly, we show that in $\mathcal{U}$, $\hat{H}^k = \hat{H}_1^k + H_2^k + \hat{H}_3^k$ is SPD. By the construction of $\hat{H}_1^k$ and $\hat{H}_3^k$, they are symmetric positive semi-definite. $H_2^k$ is symmetric positive definite under the Dirichlet boundary condition. Consequently, $\hat{H}^k$ is SPD.

Thirdly, we show that both $\kappa(\hat{H}^k)$ and $\|\hat{H}^k\|$ are bounded. We notice that in each iteration, $H_2^k = \alpha h^2 A^T G A$ is constant and we can set $\|H_2^k\| = \zeta_2$. For $\hat{H}_1^k = h^2 P^T (\vec{T}_{\tilde{\boldsymbol{U}}}^T \vec{T}_{\tilde{\boldsymbol{U}}}) P$, we get its upper bound $\zeta_1$ because $T$ is twice continuously differentiable and $\mathcal{U}$ is compact. For $\phi = \phi_1, \phi_2$ or $\phi_3$, $\phi$ is twice continuously differentiable with respect to $U \in \mathcal{U}$, then we have $\|\hat{H}_3^k\| \leq \frac{\beta h^2}{4} \|\mathrm{d}\vec{\boldsymbol{r}}^T\| \|\mathrm{d}^2 \boldsymbol{\phi}(\vec{\boldsymbol{r}})\| \|\mathrm{d}\vec{\boldsymbol{r}}\| \leq \zeta_3$. Hence, we have

$$\|\hat{H}^k\| \leq \|\hat{H}_1^k\| + \|H_2^k\| + \|\hat{H}_3^k\| \leq \zeta_1 + \zeta_2 + \zeta_3. \tag{4.46}$$

So set $\zeta = \zeta_1+\zeta_2+\zeta_3$ and $\|\hat{H}^k\| \leq \zeta$. Set $\sigma$ as the minimum eigenvalue of $H_2^k$. According to Lemma 4.9, the smallest eigenvalue $\lambda_{\min}$ of $\hat{H}^k$ should be larger than $\sigma$. The largest eigenvalue $\lambda_{\max}$ of $\hat{H}^k$ should be smaller than $\zeta$ due to $\lambda_{\max} \leq \|\hat{H}^k\|$. So set $\bar{\kappa} = \frac{\zeta}{\sigma}$ and the conditional number of $\hat{H}^k$ is smaller than $\bar{\kappa}$.

Finally, we can find that (4.30) has lower bound 0. So by applying Theorem 4.7, we finish the proof. □

**Multi-Level Strategy**

The multilevel strategy is a standard technique to provide the initial guess in image registration illustrated in Section 3.6. The multilevel scheme representing our main algorithm is summarized below where $I_H^h$ is an interpolation operator based on bi-linear interpolation techniques and $I_h^H$ is a restriction operator for transferring information to a coarser level.

**Algorithm 11:** Multilevel Image Registration: $U \leftarrow \text{MLIR}(\alpha, \beta, U^0, T, R)$

Step 1: Compute the largest possible number L of levels based on size of $T, R$;
    Define the coarsest level as level $l$;
    Work out the multilevel representation of given images $R$ and $T$;
        $R_L = R, T_L = T$;
        $R_{L-1} = I_h^H R_L$,$T_{L-1} = I_h^H T_L$;
        ...;
        $R_l = I_h^H R_{l+1}, T_l = I_h^H T_{l+1}$;
Step 2: Set the initial guess on the coarsest level;
        $U_L = U^0, \quad U_\ell^0 = I_h^H U_{\ell+1}^0,\ \ell = L-1, ..., l$;
Step 3: Apply Algorithm 10 on the coarsest level $l$ with $U_l^0$;
  $(U_l, \text{ID}) \leftarrow \text{GNAIRA}(\alpha, \beta, U_l^0, T_l, R_l)$;
**if** $ID = 1$ **then**
  Exit this algorithm;
**end**
**for** $\ell = 2 : L$ **do**
  Interpolate the solution from a coarser level $U_\ell^0 = I_H^h U_{\ell-1}$;
  Apply Algorithm 10 on level $\ell$: $(U_\ell, \text{ID}) \leftarrow \text{GNAIRA}(\alpha, \beta, U_\ell^0, T_\ell, R_\ell)$;
  **if** $ID = 1$ **then**
    Exit this algorithm;
  **end**
**end**

## 4.5 Numerical Results

In this section, we will give some numerical results to illustrate the performance of our model (4.19). We hope to achieve 3 aims:

**1).** Which choice of $\phi$ is the best for our model (4.19)?

**2).** We wish to compare with the current state-of-the-art methods (with codes listed for readers' benefit) in the literature for good diffeomorphic mappings:

(a) Hyperelastic Model [11]: code from http://www.siam.org/books/fa06/

(b) LDDMM [88]: code from https://github.com/C4IR/FAIR.m/tree/master/add-ons/LagLDDMM

(c) Diffeomorphic Demons (DDemons) [124]: code from http://www.insight-journal.org/browse/publication/154

(d) QCHR [77]; the code provided by the author Dr. Kam Chu Lam.

All of the tests are performed on a PC with 3.40 GHz Intel(R) Core(TM) i7-4770 microprocessor and with installed memory (RAM) of 32 GB.

**3).** Most importantly, we like to test and highlight the advantages of our new model.

Let $\boldsymbol{y}$ be the final transformation obtained by a particular model for registering two given images $T, R$. We use the following three measures to quantify the performance of this model and use them for later comparisons:

**(i).** Re_SSD (the relative Sum of Squared Differences) which is given by

$$\text{Re\_SSD} = \frac{\|T(\boldsymbol{y}) - R\|^2}{\|T - R\|^2}; \tag{4.47}$$

**(ii).** $\min \det(\nabla \boldsymbol{y})$ and $\max \det(\nabla \boldsymbol{y})$ that are the minimum and the maximum of the Jacobian determinant of this transformation;

**(iii).** Jaccard similarity coefficient (JSC) as defined by

$$\text{JSC} = \frac{|DT_r \cap R_r|}{|DT_r \cup R_r|}, \tag{4.48}$$

where $DT_r$ and $R_r$ represent respectively the segmented regions of interest (e.g. certain image feature such as an organ) in the deformed template (after registration) and the reference. Hence, JSC is the ratio of the intersection of $DT_r$ and $R_r$ to the union of $DT_r$ and $R_r$ [75]. JSC=1 shows that a perfect alignment of the segmentation boundary and JSC=0 indicates that the segmented regions have no overlap after registration.

In practice, we scale the intensity value of $T$ and $R$ to $[0, 255]$. Here, we state a strategy for choosing the parameters. For our model (4.19), $\alpha$ should be related to energy $\mathcal{D}(\boldsymbol{u}_0)$ where $\boldsymbol{u}_0$ is the initial guess for the displacement, and $\beta$ should be related to $\alpha$. Empirically, we set $\alpha \in [\alpha_1, \alpha_2]$, where $\alpha_1 = 0.5\mathcal{D}(\boldsymbol{u}_0)10^{-2}$ and $\alpha_2 = 2\mathcal{D}(\boldsymbol{u}_0)10^{-2}$. Respectively

for $\phi = \phi_1$, $\phi_2$, $\phi_3$, we set $\beta \in [3\alpha, 5\alpha]$, $[0.5\alpha, 2\alpha]$ and $[\alpha, 5\alpha]$. For simplicity, we denote the model (4.19) with $\phi_1$, $\phi_2$ and $\phi_3$ by New 1, New 2 and New 3 respectively.

It should be noted that a good registration result should produce a small Re_SSD, be diffeomorphic and yield a large JSC value for a region of interest.

### 4.5.1 Example 1 — Improvement over The Diffusion Model

In this example, we test a pair of real medical images, X-ray Hands of resolution $128 \times 128$. Figure 4.3 (a-b) show the template and the reference. We compare our model with the diffusion model and study the improvement over it. In the implementation, for both models, we use a five-step multilevel strategy.

We conduct two experiments using different parameters:

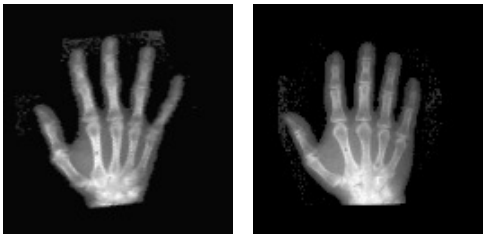
**i). Fixed parameters.** Our first choice uses fixed parameters. For New 1-3, we set $\beta = 7$, $\beta = 1$ and $\beta = 9$ respectively, and fix $\alpha = 2$. To be fair, we also choose $\alpha = 2$ for the diffusion model. In this case, Figure 4.3 show the deformed templates $T(\boldsymbol{y})$ from 4 models. From it, we can see that all four models can produce visually satisfactory results. To differentiate them, we have to check the quantitative measures from Table 4.1. We can notice that the transformation obtained by the diffusion model is non-diffeomorphic due to $\min \det(\nabla \boldsymbol{y}) < 0$ (i.e. mesh folded, though visually pleasing and the Re_SSD is small). Figure 4.4 illustrates the transformation $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{u}$ locally at its folding point. In contrast, our New 1-3 can generate diffeomorphic transformations.

**ii). Optimized parameters.** The second choice uses the fine-tuned parameters for the diffusion model. We test $\alpha \in [1, 500]$ and find the smallest $\alpha = 430$ with which the diffusion model generates a diffeomorphic transformation. Then for our model, we also set $\alpha = 430$ (which is not optimized to favor the former) and set $\beta = 5$ for New 1-3 (to test the robustness of our model). Table 4.1 shows the detailed results for this second test. From it, we can see that the Re_SSD and JSC of our model are similar to the diffusion model. And the transformations obtained by New 1-3 are all diffeomorphic while the diffusion model is only diffeomorphic with the help of an optimized $\alpha$. This shows that our model possesses the robustness (in the sense of not requiring optimized $\alpha$) with the help of a positive $\beta$.

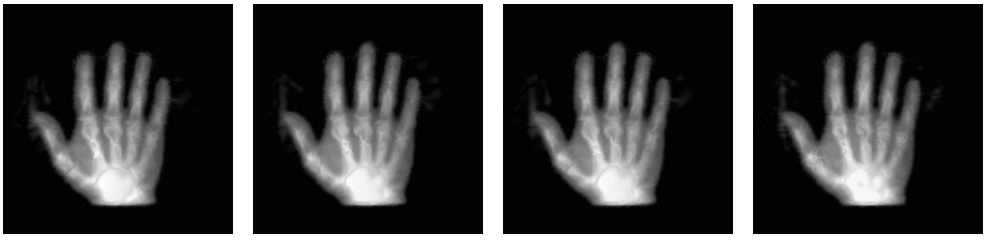
Hence, this example demonstrates that our New 1-3 are robust and can all help to get an accurate and diffeomorphic transformation.

### 4.5.2 Example 2 — Test of Large Deformation and Comparison of Models

As known, if the underlying deformation is small, it is generally believed that most models can deliver diffeomorphic transformations. This belief is true if one keeps increasing $\alpha$, which in turn compromises the registration quality by increasing Re_SSD (as seen in



(a) Template (b) Reference



(c) $T(\boldsymbol{y})$ by New 1 (d) $T(\boldsymbol{y})$ by New 2 (e) $T(\boldsymbol{y})$ by New 3 (f) $T(\boldsymbol{y})$ by Diffusion model

FIGURE 4.3: Example 1 results of Hand to Hand registration ($\alpha = 2$): in the top row, there are the template and reference. In the second row, there are the deformed templates obtained by model (4.19) and the diffusion model separately. Though the last column is visually fine, the transformation is not correct – see Table 4.1.

TABLE 4.1: Example 1 — Comparison of the new model (New 1-3) with the diffusion model based a fixed $\alpha$ and an optimized $\alpha$ for the latter. Clearly the latter model can produce an incorrect result if not tuned while New 1-3 are less sensitive to $\alpha$ with the help of $\beta$.

| | Resolution | Re_SSD | $\min \det(\nabla \boldsymbol{y})$ | $\max \det(\nabla \boldsymbol{y})$ | JSC | time (s) |
|---|---|---|---|---|---|---|
| First Test $\alpha = 2$ | | | | | | |
| New 1 | $128 \times 128$ | 1.85% | 0.0032 | 20.1606 | 99.20% | 33.3 |
| New 2 | $128 \times 128$ | 1.27% | 0.0003 | 33.2371 | 99.54% | 19.9 |
| New 3 | $128 \times 128$ | 1.62% | 0.0014 | 24.4540 | 99.26% | 30.9 |
| Diffusion Model | $128 \times 128$ | 0.90% | $-36.7964$ | 72.2924 | 98.41% | 12.1 |
| Second Test $\alpha = 430$ | | | | | | |
| New 1 | $128 \times 128$ | 7.83% | 0.1337 | 4.8247 | 98.28% | 2.5 |
| New 2 | $128 \times 128$ | 7.80% | 0.1300 | 4.8364 | 98.28% | 2.5 |
| New 3 | $128 \times 128$ | 7.78% | 0.1260 | 4.8472 | 98.36% | 2.5 |
| Diffusion Model | $128 \times 128$ | 7.75% | 0.0066 | 4.8278 | 98.30% | 0.9 |

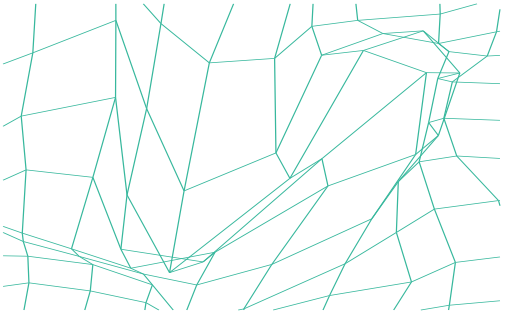


FIGURE 4.4: Zooming in the transformation (obtained by the diffusion model) where there is folding.

2 tests of $\alpha$ in Example 1 where the larger $\alpha = 430$ achieves diffeomorphism for diffusion with a worse Re_SSD value).

Therefore to test the capability of a registration model, we need to take an example that requires large deformation. To this end, we consider Example 2 – a classic synthetic example consisting of **a Disc** and **a C shape** of resolution $128 \times 128$ as shown in Figure 4.5 (a-b). We compare our 3 models (New 1-3) with 5 other models: the hyperelastic model, LDDMM, DDemons, QCHR, and the diffusion model in registration quality and performance. For this example, we use a five-step multilevel strategy for our model, the hyperelastic model, and the diffusion model. For LDDMM and QCHR, we use a three-step multilevel strategy. We use a one-step multilevel strategy for DDemons as we find that multilevel does not improve the results.

Following our stated strategy for choosing the parameter for our models, we set $\beta = 80, 120, 100$ for New 1-3 respectively and fix $\alpha = 70$. To be consistent, we also set $\alpha = 70$ for the diffusion model. For the hyperelastic model, LDDMM and QCHR, we set respectively $\{\alpha_l = 100, \alpha_s = 0, \alpha_v = 18\}$, $\alpha = 400$ and $\{\alpha = 0.1, \beta = 1\}$ as used in the literature [11, 77, 88] for the same example. For the parameters of DDemons, we try to optimize the parameters $\{\sigma_s, \sigma_g\}$ in the domain $[0.5, 5] \times [0.5, 5]$ and take the optimal choice $\{\sigma_s = 1.5, \sigma_g = 3.5\}$.

We now present the comparative results. Figure 4.5 (c-j) show that except for the diffusion model, all the other models can produce the accepted registered results. Especially, our model and LDDMM are slightly better than the hyperelastic model, DDemons and QCHR. It is pleasing to see that the new model produces equally good results for this challenging example. From Table 4.2, we see that our New 1-3, hyperelastic model, LDDMM, DDemons, and QCHR produce $\min \det(\nabla \boldsymbol{y}) > 0$, i.e., the transformations obtained by these five models are diffeomorphic, but the diffusion model fails again with $\min \det(\nabla \boldsymbol{y}) < 0$.

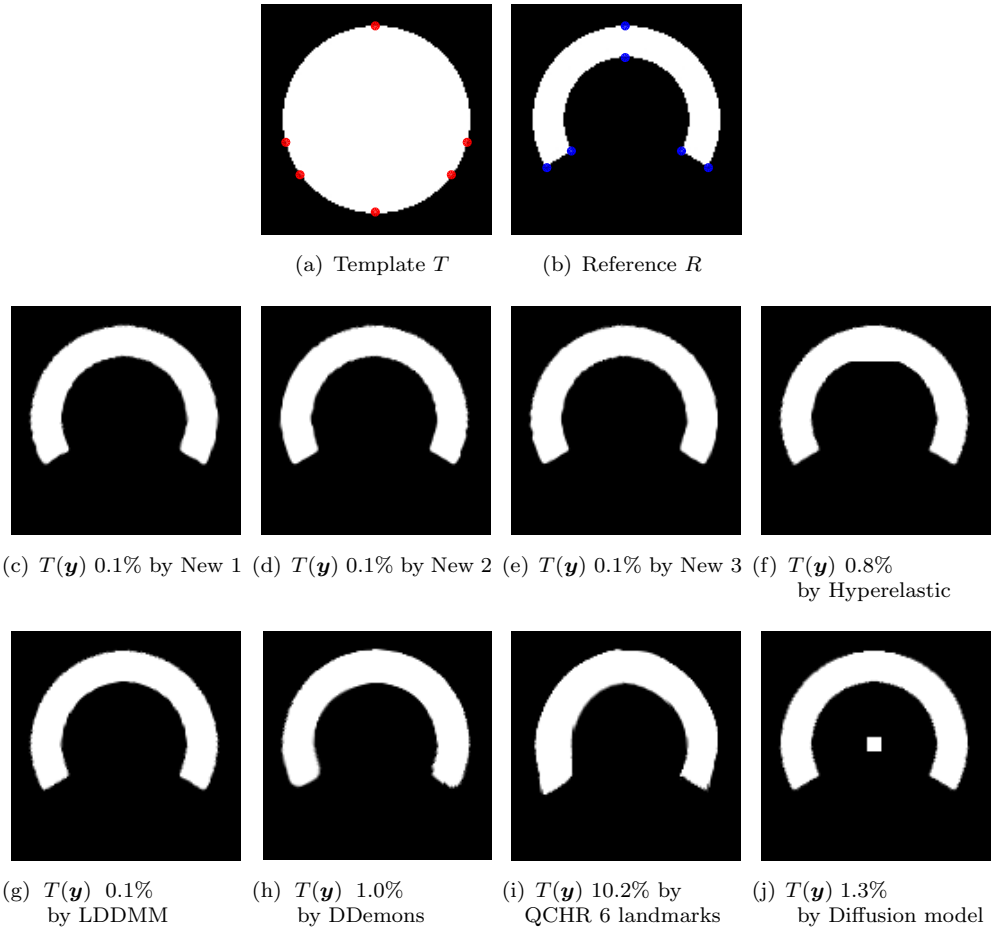
Because New 1-3 are motivated by the QCHR model, we now discuss the results of these two types of models. On the one hand, according to Table 4.2, we can find that our model takes less time. This is because, as we have mentioned, the algorithm for QCHR needs to solve alternatively two subproblems (including several linear systems) in each iteration. Its convergence cannot be guaranteed. However, our model only needs to solve one linear system in each iteration. In addition, we employ the Gauss-Newton method, which can be superlinearly convergent under the appropriate conditions. As we have also remarked, the QCHR algorithm can have convergence problems. This is now illustrated in Figure 4.6 where we plot the relative residual of our model (New 3) and the relative residual of QCHR. We observe that New 3 decreases to below $10^{-2}$ though not monotonically, but the relative residual of QCHR does not decrease and is over 0.1.

On the other hand, we can compare the obtained solutions' quality by checking the energy functionals. Using the same QCHR functional, the QCHR solution for Example 2 has the value 702.7 while the transformation obtained by New 3 gives the value 72.3, which is much smaller. This indicates that the result obtained by the QCHR algorithm

TABLE 4.2: Example 2 — Comparison of the new model (New 1-3) with 5 other models.

| | Resolution | Re_SSD | $\min\det(\nabla \boldsymbol{y})$ | $\max\det(\nabla \boldsymbol{y})$ | JSC | time (s) |
|---|---|---|---|---|---|---|
| New 1 | $128 \times 128$ | 0.06% | 0.0053 | 21.8478 | 96.11% | 5.0 |
| New 2 | $128 \times 128$ | 0.07% | 0.0028 | 18.9933 | 96.13% | 6.4 |
| New 3 | $128 \times 128$ | 0.05% | 0.0108 | 25.0695 | 96.00% | 3.2 |
| Hyperelastic Model | $128 \times 128$ | 0.81% | 0.2426 | 5.8530 | 94.84% | 1.6 |
| LDDMM | $128 \times 128$ | 0.07% | 0.1050 | 12.1524 | 95.09% | 14.7 |
| DDemons | $128 \times 128$ | 1.02% | $1.3 \times 10^{-7}$ | 8.2326 | 94.24% | 67.8 |
| QCHR Model | $128 \times 128$ | 10.17% | 0.0004 | 69.3977 | 81.88% | 16.8 |
| Diffusion Model | $128 \times 128$ | 1.25% | $-10.1612$ | 162.5034 | 94.21% | 0.3 |

is not accurate. This is consistent with the fact that the Re_SSD and JSC of New 3 are also better than QCHR. Both discussions reach the same conclusion: the QCHR algorithm cannot obtain the minimizer of the original QCHR functional.



(a) Template $T$ (b) Reference $R$

(c) $T(\boldsymbol{y})$ 0.1% by New 1 (d) $T(\boldsymbol{y})$ 0.1% by New 2 (e) $T(\boldsymbol{y})$ 0.1% by New 3 (f) $T(\boldsymbol{y})$ 0.8% by Hyperelastic

(g) $T(\boldsymbol{y})$ 0.1% by LDDMM (h) $T(\boldsymbol{y})$ 1.0% by DDemons (i) $T(\boldsymbol{y})$ 10.2% by QCHR 6 landmarks (j) $T(\boldsymbol{y})$ 1.3% by Diffusion model

FIGURE 4.5: Example 2 results of Disc to C. The percentage value shows Re_SSD error. In the top row, there are the template and reference. In the second and third row, there are the deformed templates obtained by New 1-3 and 5 other models separately. The landmarks in the template and reference are only used for QCHR and the last result (j) by Diffusion is evidently not correct.

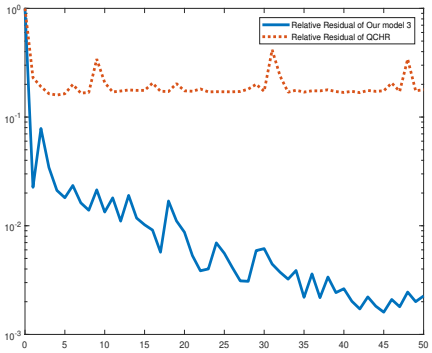



FIGURE 4.6: Example 2 Relative Residual of New 3 and QCHR: The solid line indicates the relative residual of New 3. And the dotted line shows the relative residual of the second subproblem in QCHR. Here, we can find that in the same 50 iterations, the relative residual of New 3 is decreasing to below $10^{-2}$. However, the relative residual of QCHR is not decreasing and over 0.1. Hence, the convergence of the algorithm for QCHR cannot be guaranteed.

### 4.5.3 Example 3 — Comparison of Models for A Challenging Test

Here, we illustrate the fact that area preservation between images can become unnecessary and trying to enforce it (as in the hyperelastic model) can fail to register an image. We choose the particular template and reference images, as shown in Figure 4.7 (a-b), having significantly different areas in their main features – here the area of 'Disc' is much larger than 'C'. The resolution of the images is $512 \times 512$. We test the performance of New 1-3 and other models. In this example, we use a seven-step multilevel strategy for New 1-3, the hyperelastic model, and the diffusion model. For LDDMM and QCHR, we use a five-step multilevel strategy. We use a single level for DDemons (since multilevels do not help).

In choosing the parameters for all the models to register this example, we first follow our strategy to set $\beta = 250, 50, 100$ for New 1-3 respectively and fix $\alpha = 50$. To be consistent, we also set $\alpha = 50$ for the diffusion model. For the hyperelastic model, we also set $\alpha_l = 50$ because it contains the diffusion term, and take $\alpha_s = 0$. For the third parameter $\alpha_v$ in the hyperelastic model, we test it in the range $[55, 150]$ and choose its optimal value $\alpha_v = 75$. For LDDMM and QCHR, we set the default value $\alpha = 400$ and $\{\alpha = 0.1, \beta = 1\}$ as the previous example. For the parameters of DDemons, we test the parameters $\{\sigma_s, \sigma_g\}$ in the domain $[0.5, 5] \times [0.5, 5]$ and choose its optimal choice $\{\sigma_s = 2, \sigma_g = 5\}$. Hence we would expect the hyperelastic model and DDemons to perform well.

The test results for Example 3 are presented in Table 4.3 and Figure 4.7. Although all models except for the diffusion model produce diffeomorphic transformations, we can see visually that only 3 models (our New 2-3 and LDDMM) produce acceptable results, also confirmed by the table:

TABLE 4.3: Example 3 — Comparison of the new model (New 1-3) with 5 other models.

| | Resolution | Re_SSD | $\min\det(\nabla \boldsymbol{y})$ | $\max\det(\nabla \boldsymbol{y})$ | JSC | time (s) |
|---|---|---|---|---|---|---|
| New 1 | $512 \times 512$ | 3.06% | 0.0332 | 38.2864 | 79.08% | 293.0 |
| New 2 | $512 \times 512$ | 0.23% | 0.0033 | 64.1928 | 96.73% | 273.2 |
| New 3 | $512 \times 512$ | 0.07% | 0.0065 | 59.8003 | 97.82% | 139.1 |
| Hyperelastic Model | $512 \times 512$ | 3.87% | 0.4587 | 6.9885 | 75.5% | 36.6 |
| LDDMM | $512 \times 512$ | 0.41% | 0.0184 | 40.2544 | 95.05% | 218.3 |
| DDemons | $512 \times 512$ | 2.83% | $9.6 \times 10^{-6}$ | 34.8529 | 80.56% | $> 5000$ |
| QCHR Model | $512 \times 512$ | 0.94% | 0.0227 | 4.3830 | 91.47% | 260.0 |
| Diffusion Model | $512 \times 512$ | 0.65% | $-14.8361$ | 350.4944 | 93.81% | 3.3 |

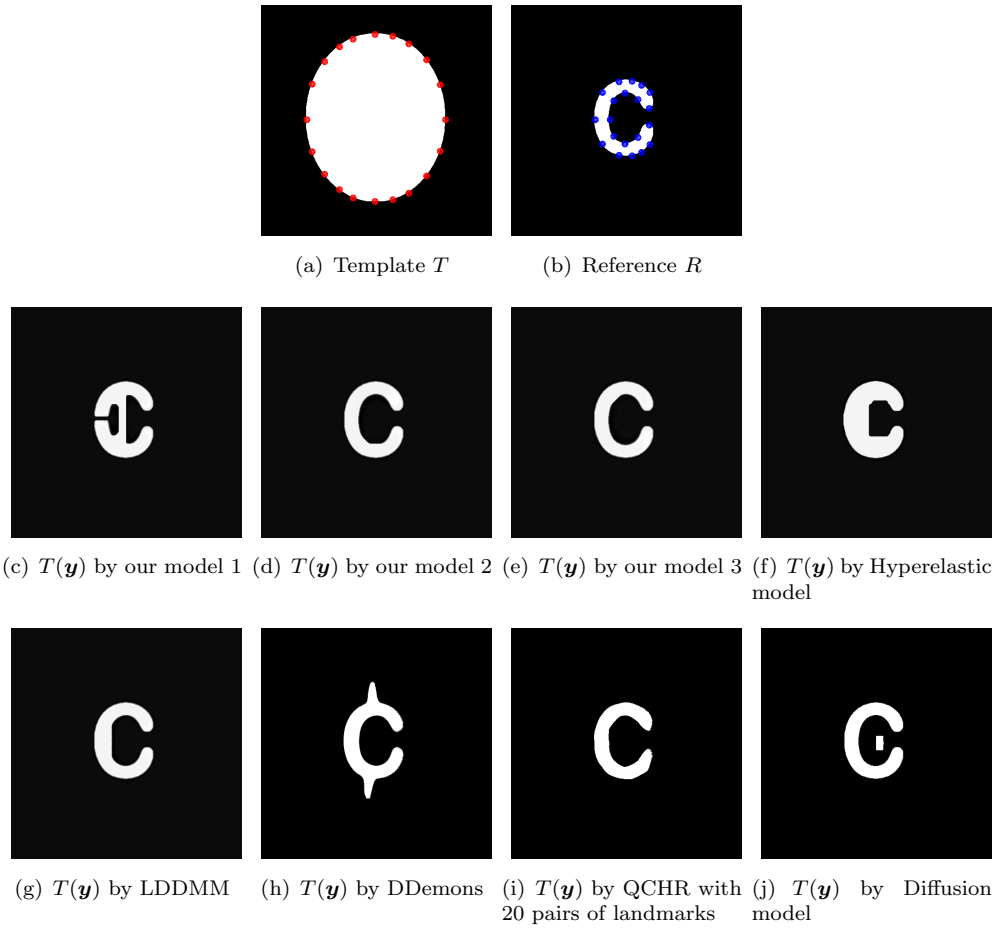
- The badly deformed template generated by our New 1 shows that the model lacks robustness;
- The hyperelastic model, though producing a diffeomorphic transform, fails (despite using an optimized parameter) because this model including a regularization term $(\det(\nabla \boldsymbol{y}) - 1)^4/(\det(\nabla \boldsymbol{y}))^2$ tends to preserve area. If we do not optimize parameters for the hyperelastic model, our tests show that its results are even worse.
- In the previous example, we have pointed out that QCHR needs more computing time and, from Table 4.3, we see that the time for QCHR is about two times as long as our New 3;
- The DDemons is trapped in a local minimum, and its CPU time is also excessive ($> 5000$ seconds). We also try to apply a multilevel strategy to DDemons, but for this example, the result is not satisfied. The Re_SSD, JSC and CPU time of our New 3 are all slightly better than the second best LDDMM;
- Both Tables 4.2 and 4.3 show that the diffusion model produces solutions having a negative Jacobian determinant (folding) which might be viewed non-physical; this model is included only for reference.

Hence, our model has advantages over other models for large deformation registrations not requiring preserving area.

As remarked, the landmarks supplied to QCHR can severely affect the results. Figure 4.8 shows three sets of an increasing number of landmarks for Examples 2-3. We observe that more landmarks lead to better results in terms of JSC values.

### 4.5.4 Example 4 — Comparison of The New Model with Other Models

In the final test, we test a pair of anonymized CT images in resolution $512 \times 512$ from the Royal Liverpool University Hospital. Figure 4.9 (a-b) show the template and the reference. The template was taken in September 2016, and the reference was taken in May 2016. We want to compare the changes of our interested regions of the abdominal aortic aneurysm with stents inserted inside them (with cross sections shown as two white 'circles' in images in Figure 4.9 (a-b)) during these four months. In addition, the



(a) Template $T$ (b) Reference $R$

(c) $T(\boldsymbol{y})$ by our model 1 (d) $T(\boldsymbol{y})$ by our model 2 (e) $T(\boldsymbol{y})$ by our model 3 (f) $T(\boldsymbol{y})$ by Hyperelastic model

(g) $T(\boldsymbol{y})$ by LDDMM (h) $T(\boldsymbol{y})$ by DDemons (i) $T(\boldsymbol{y})$ by QCHR with 20 pairs of landmarks (j) $T(\boldsymbol{y})$ by Diffusion model

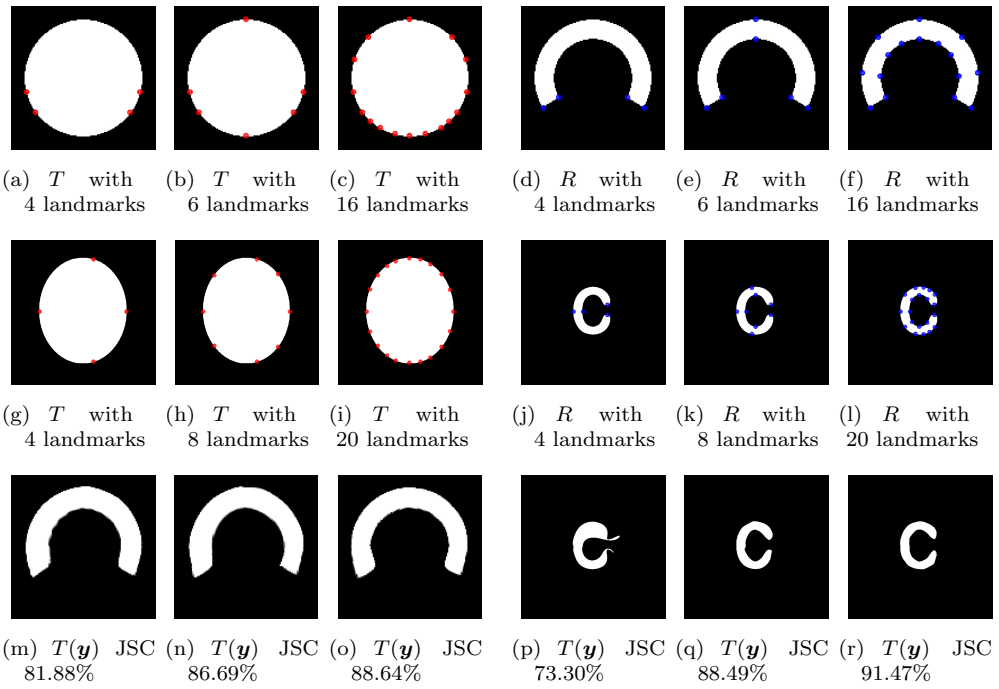
FIGURE 4.7: Example 3 results of a large Disc to a small letter C: in the top row, there are the template and reference. In the second and third row, there are the deformed templates obtained by model (4.19) and other models separately. The landmarks in the template and reference are only used for QCHR.

interested region is used to compute JSC. The small white region on top of the images helps us to identify the correct slice to compare.

Here, following the previous example, we use the same multilevel strategy: a seven-step multilevel strategy for our model, the hyperelastic model, and the diffusion model, a five-step multilevel strategy for LDDMM and QCHR and a one-step multilevel strategy for DDemons.

Following our strategy for choosing the parameter of our model, we set $\alpha = 20$ and set $\beta = 100, 40, 75$ with New 1-3 respectively. For the diffusion model and LDDMM, we test $\alpha$ from $[100, 2000]$ and set the optimal value 1300 and 500 respectively. For the hyperelastic model, we set $\{\alpha_l = 20, \alpha_s = 0, \alpha_v = 50\}$. We use the default value $\{\alpha = 0.1, \beta = 1\}$ for QCHR. For the parameters of DDemons, we test the parameters $\{\sigma_s, \sigma_g\}$ in the domain $[0.5, 5] \times [0.5, 5]$ and choose $\{\sigma_s = 4, \sigma_g = 4.5\}$.

With the optimized parameters, all the models in this example generate diffeomorphic transformations as seen from Table 4.4. DDemons and QCHR for this example are



(a) $T$ with 4 landmarks (b) $T$ with 6 landmarks (c) $T$ with 16 landmarks (d) $R$ with 4 landmarks (e) $R$ with 6 landmarks (f) $R$ with 16 landmarks

(g) $T$ with 4 landmarks (h) $T$ with 8 landmarks (i) $T$ with 20 landmarks (j) $R$ with 4 landmarks (k) $R$ with 8 landmarks (l) $R$ with 20 landmarks

(m) $T(\boldsymbol{y})$ JSC 81.88% (n) $T(\boldsymbol{y})$ JSC 86.69% (o) $T(\boldsymbol{y})$ JSC 88.64% (p) $T(\boldsymbol{y})$ JSC 73.30% (q) $T(\boldsymbol{y})$ JSC 88.49% (r) $T(\boldsymbol{y})$ JSC 91.47%

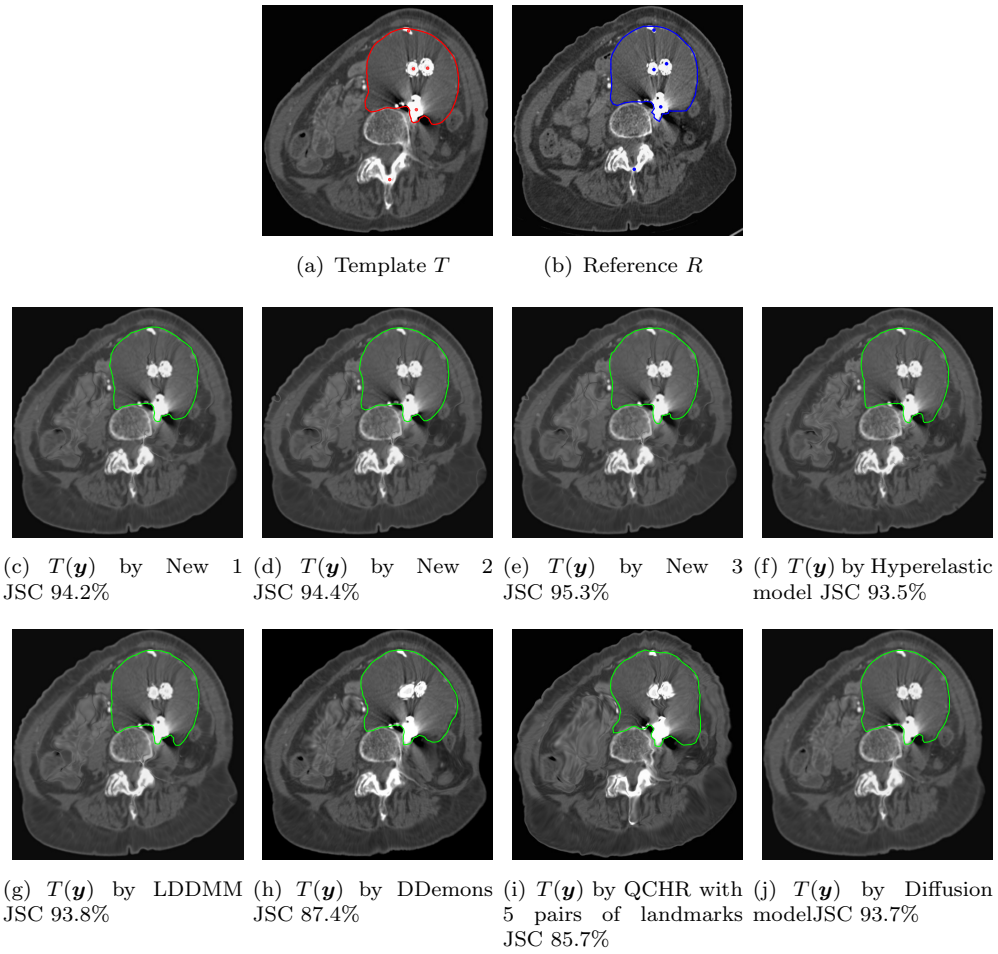
FIGURE 4.8: Tests of QCHR with different landmarks: Example 2 (row 1) and Example 3 (row 2). On the left 3 columns of row 3, we show the registered templates for row 1. On the right 3 columns of row 3, we show the registered templates for row 2. Here, we can see that the accuracy of QCHR improves with the increase of landmarks.

TABLE 4.4: Example 4 — Comparison of the new model (New 1-3) with 5 other models

| | Resolution | Re_SSD | $\min \det(\nabla \boldsymbol{y})$ | $\max \det(\nabla \boldsymbol{y})$ | JSC |
|---|---|---|---|---|---|
| New 1 | $512 \times 512$ | 4.75% | 0.0124 | 52.6802 | 94.19% |
| New 2 | $512 \times 512$ | 3.49% | 0.0068 | 46.6383 | 94.39% |
| New 3 | $512 \times 512$ | 3.47% | 0.0051 | 49.9309 | 95.34% |
| Hyperelastic Model | $512 \times 512$ | 4.44% | 0.4181 | 3.6192 | 93.51% |
| LDDMM | $512 \times 512$ | 5.18% | 0.0319 | 20.8164 | 93.79% |
| DDemons | $512 \times 512$ | 18.89% | 0.1846 | 2.6309 | 87.40% |
| QCHR Model | $512 \times 512$ | 26.71% | 0.0481 | 16.2555 | 85.68% |
| Diffusion Model | $512 \times 512$ | 10.02% | 0.0342 | 7.3450 | 93.65% |

not as good as other models because they give worse Re_SSD and JSC. A worse JSC means the interested regions obtained by these two methods have significant differences from the reference (Figure 4.9 (h-i)). The diffusion model obtains good JSC. However, its deformed template is a bit far (overall) from the reference (since Re_SSD= 10.02%). The other two models (Hyperelastic, LDDMM) generate good Re_SSD and JSC. However, our models produce the lowest Re_SSD and the best JSC. Hence, for this example of real images, our model is competitive to state-of-the-art methods. Though there is a broad agreement between Re_SSD and JSC, one has to combine with segmentation models to ensure a strict agreement.

*Remark* 4.11. According to the above four examples, our New 1 is not robust while New 2-3 can both generate accurate and diffeomorphic transformations. However, we recommend New 3 as the first choice because of the least computing time and the best



(a) Template $T$ (b) Reference $R$

(c) $T(\boldsymbol{y})$ by New 1 JSC 94.2% (d) $T(\boldsymbol{y})$ by New 2 JSC 94.4% (e) $T(\boldsymbol{y})$ by New 3 JSC 95.3% (f) $T(\boldsymbol{y})$ by Hyperelastic model JSC 93.5%

(g) $T(\boldsymbol{y})$ by LDDMM JSC 93.8% (h) $T(\boldsymbol{y})$ by DDemons JSC 87.4% (i) $T(\boldsymbol{y})$ by QCHR with 5 pairs of landmarks JSC 85.7% (j) $T(\boldsymbol{y})$ by Diffusion modelJSC 93.7%

FIGURE 4.9: Example 4 results of a pair of CT images: the template $T$ and the reference $R$ in the top row. The contours show the regions of interest. In the second and third rows, we show the deformed templates obtained by 8 models. The 5 landmarks in the template and the reference are only used by QCHR.

quality, and New 2 as the second choice.

## 4.6 Conclusion

Controlling mesh folding is a key issue in image registration models to ensure local invertibility. Many existing models either do not impose any further controls on the underlying transformation beyond smoothness (so potentially generating unrealistic or non-physical transformations or mapping) or impose a *direct* (often strongly biased, e.g., towards area or volume preservation) control on some explicit function of the measure $\det(\nabla \boldsymbol{y})$. This chapter introduces a novel, unbiased, and robust regularizer, which is reformulated from Beltrami coefficient framework to ensure a diffeomorphic transformation. Moreover, we find that a direct approach (our New 1) from this Beltrami reformulation provides an alternative, but less competitive method but further refinements (especially our New 3) of this new regularizer can give rise to more robust models than the existing methods. We highly recommend our model New 3 i.e. (4.19) with $\phi = \phi_3$.

In designing optimization methods for solving the resulting highly nonlinear variational model, we give a suitable approximation of the exact Hessian matrix, which is necessary to derive a convergent iterative method. Our test results can show that the new model (New 1-3, especially New 3) is competitive with the state-of-the-art models. The main advantage lies in robustness.

Here, in the numerical implementation, we choose the Gauss-Newton method. For this method, in each iteration, the main computational cost is from solving the linear system. Especially for the huge dimensional problem, solving the linear system should be expensive. Motivated by this point, in the next chapter, we design a novel two-step Gauss-Newton method, in which a second step is computed within each iteration by minimizing a quadratic approximation of the objective function over a certain 2D subspace. Numerical results on image registration problems show that the proposed methods can outperform the standard multilevel Gauss-Newton method.

## 4.7 Appendix

### 4.7.1 Computation of Matrices $A$ and $G$ in (4.26)

Set $B = I_2 \otimes I_{n+1} \otimes \partial_n^{1,h} \in \mathbb{R}^{2n(n+1)\times 2(n+1)^2}$, $C = I_2 \otimes \partial_n^{1,h} \otimes I_{n+1} \in \mathbb{R}^{2n(n+1)\times 2(n+1)^2}$,

$$\partial_n^{1,h} = \frac{1}{h}\begin{bmatrix} -1 & 1 & & \\ & -1 & 1 & \\ & \dots & \dots & \dots \\ & & -1 & 1 \\ & & & -1 \quad 1 \end{bmatrix} \in \mathbb{R}^{n,n+1}, \quad A = \begin{bmatrix} B \\ C \end{bmatrix} \in \mathbb{R}^{4n(n+1)\times 2(n+1)^2},$$

where $\otimes$ denotes a Kronecker product. To represent the difference between interior and boundary pixels, we need to introduce a diagonal matrix

$$G = \begin{bmatrix} G_1 & 0 & 0 & 0 \\ 0 & G_2 & 0 & 0 \\ 0 & 0 & G_1 & 0 \\ 0 & 0 & 0 & G_2 \end{bmatrix} \in \mathbb{R}^{4n(n+1)\times 4n(n+1)}, \tag{4.49}$$

where $G_1$ and $G_2$ are diagonal matrices. For $G_1$, $G_{1_{i+1+jn,i+1+jn}} = 1$ if $0 \le i \le n-1, 1 \le j \le n-1$ or $\frac{1}{2}$ if $0 \le i \le n-1, j = 0, n$. Similarly, for $G_2$, $G_{2_{i+1+j(n+1),i+1+j(n+1)}} = 1$ if $1 \le i \le n-1, 0 \le j \le n-1$ or $\frac{1}{2}$ if $i = 0, n, 0 \le j \le n-1$.

### 4.7.2 Computation of The Vector $\vec{r}(U)$ in (4.29)

We demonstrate how to build the linear interpolation $\boldsymbol{L}$ in $\triangle V_1V_2V_5$, in Figure 4.2.

First of all, denote the 3 vertices of this triangle by $V_1 = \boldsymbol{x}^{1,1}$, $V_2 = \boldsymbol{x}^{2,1}$ and $V_5 = \boldsymbol{x}^{1.5,1.5}$. Set $\boldsymbol{L}(V_1) = (u_1^{1,1}, u_2^{1,1})$, $\boldsymbol{L}(V_2) = (u_1^{2,1}, u_2^{2,1})$ at the vertex pixels, and $\boldsymbol{L}(V_5) =$

$(u_1^{1.5,1.5}, u_2^{1.5,1.5})$ at the cell centre (approximated values). Here the linear approximations are $\boldsymbol{L}(x_1, x_2) = (a_1x_1 + a_2x_2 + b_1, a_3x_1 + a_4x_2 + b_2)$.

After substituting $V_1, V_2$ and $V_5$ into $\boldsymbol{L}$, we get

$$\begin{pmatrix} x_1^1 - x_1^{1.5} & x_2^1 - x_2^{1.5} \\ x_1^2 - x_1^{1.5} & x_2^1 - x_2^{1.5} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} u_1^{1,1} - u_1^{1.5,1.5} \\ u_1^{2,1} - u_1^{1.5,1.5} \end{pmatrix},$$

$$\begin{pmatrix} x_1^1 - x_1^{1.5} & x_2^1 - x_2^{1.5} \\ x_1^2 - x_1^{1.5} & x_2^1 - x_2^{1.5} \end{pmatrix} \begin{pmatrix} a_3 \\ a_4 \end{pmatrix} = \begin{pmatrix} u_2^{1,1} - u_2^{1.5,1.5} \\ u_2^{2,1} - u_2^{1.5,1.5} \end{pmatrix}.$$

Then

$$\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \frac{1}{\det} \begin{pmatrix} x_2^1 - x_2^{1.5} & -x_2^1 + x_2^{1.5} \\ -x_1^2 + x_1^{1.5} & x_1^1 - x_1^{1.5} \end{pmatrix} \begin{pmatrix} u_1^{1,1} - u_1^{1.5,1.5} \\ u_1^{2,1} - u_1^{1.5,1.5} \end{pmatrix}, \tag{4.50}$$

$$\begin{pmatrix} a_3 \\ a_4 \end{pmatrix} = \frac{1}{\det} \begin{pmatrix} x_2^1 - x_2^{1.5} & -x_2^1 + x_2^{1.5} \\ -x_1^2 + x_1^{1.5} & x_1^1 - x_1^{1.5} \end{pmatrix} \begin{pmatrix} u_2^{1,1} - u_2^{1.5,1.5} \\ u_2^{2,1} - u_2^{1.5,1.5} \end{pmatrix}, \tag{4.51}$$

where $\det = \begin{vmatrix} x_1^1 - x_1^{1.5} & x_2^1 - x_2^{1.5} \\ x_1^2 - x_1^{1.5} & x_2^1 - x_2^{1.5} \end{vmatrix}$.

According to (4.50) and (4.51), we can formulate two matrices $D1 \in \mathbb{R}^{4n^2 \times (n+1)^2}$ and $D2 \in \mathbb{R}^{4n^2 \times (n+1)^2}$ such that

$$\begin{aligned} \boldsymbol{a}_1 - \boldsymbol{a}_5 &= [D1, -D2]U = A_1U \in \mathbb{R}^{4n^2 \times 1}, \\ \boldsymbol{a}_4 + \boldsymbol{a}_2 &= [D2, D1]U = A_2U \in \mathbb{R}^{4n^2 \times 1}, \\ \boldsymbol{a}_1 + \boldsymbol{a}_5 &= [D1, D2]U = A_3U \in \mathbb{R}^{4n^2 \times 1}, \\ \boldsymbol{a}_4 - \boldsymbol{a}_2 &= [D2, -D1]U = A_4U \in \mathbb{R}^{4n^2 \times 1}. \end{aligned}$$

Here, $\boldsymbol{a}_\theta = (a_\theta^1, ..., a_\theta^{4n^2})^T, \theta = 1, 2, 4, 5$, where $a_\theta^l = a_\theta^{i,j,k}$ and $l = (k-1)n^2 + (j-1)n + i$.

Next using the Hadamard product $\odot$, we get a compact form for

$$\begin{cases} \vec{\boldsymbol{r}}^1(U) & = A_1U \odot A_1U + A_2U \odot A_2U, \\ \vec{\boldsymbol{r}}^2(U) & = 1/((A_3U + 2) \odot (A_3U + 2) + A_4U \odot A_4U), \\ \vec{\boldsymbol{r}}(U) & = \vec{\boldsymbol{r}}^1 \odot \vec{\boldsymbol{r}}^2 \quad \in \mathbb{R}^{4n^2 \times 1}. \end{cases} \tag{4.52}$$

### 4.7.3 Computing The Gradient and Approximated Hessian of The Term (4.35)

Here, as an example, we set $n = 2$ and $\phi = \phi_1$ to compute the gradient and approximated Hessian of the discretized Beltrami term (4.35).

Because of $n = 2$, we have

$$U = (u_1^{0,0}, ..., u_1^{2,0}, ..., u_1^{0,2}, ..., u_1^{2,2}, u_2^{0,0}, ..., u_2^{2,0}, ..., u_2^{0,2}, ..., u_2^{2,2})^T \in \mathbb{R}^{18 \times 1}.$$

From (4.50)-(4.51), we can formulate two matrices $D1, D2 \in \mathbb{R}^{16\times 9}$ respectively by:

$$\begin{bmatrix}
-2 & 2 & & & & & & & \\
 & -2 & 2 & & & & & & \\
 & & & -2 & 2 & & & & \\
 & & & & -2 & 2 & & & \\
-1 & 1 & & -1 & 1 & & & & \\
 & -1 & 1 & & -1 & 1 & & & \\
 & & & -1 & 1 & & -1 & 1 & \\
 & & & & -1 & 1 & & -1 & 1 \\
 & & & -2 & 2 & & & & \\
 & & & & -2 & 2 & & & \\
 & & & & & & -2 & 2 & \\
 & & & & & & & -2 & 2 \\
-1 & 1 & & -1 & 1 & & & & \\
 & -1 & 1 & & -1 & 1 & & & \\
 & & & -1 & 1 & & -1 & 1 & \\
 & & & & -1 & 1 & & -1 & 1
\end{bmatrix},
\begin{bmatrix}
-1 & -1 & & 1 & 1 & & & & \\
 & -1 & -1 & & 1 & 1 & & & \\
 & & & -1 & -1 & & 1 & 1 & \\
 & & & & -1 & -1 & & 1 & 1 \\
 & -2 & & & 2 & & & & \\
 & & -2 & & & 2 & & & \\
 & & & & -2 & & & 2 & \\
 & & & & & -2 & & & 2 \\
-1 & -1 & & 1 & 1 & & & & \\
 & -1 & -1 & & 1 & 1 & & & \\
 & & & -1 & -1 & & 1 & 1 & \\
 & & & & -1 & -1 & & 1 & 1 \\
-2 & & & 2 & & & & & \\
 & -2 & & & 2 & & & & \\
 & & & -2 & & & 2 & & \\
 & & & & -2 & & & 2 &
\end{bmatrix}.$$

Then we can build $A_1, A_2, A_3$ and $A_4$ and compute $\vec{\boldsymbol{r}}^1, \vec{\boldsymbol{r}}^2$ and $\vec{\boldsymbol{r}}$ by (4.52). According to (4.37), we have $\mathrm{d}\vec{\boldsymbol{r}} \in \mathbb{R}^{16\times 18}$.

When $\phi(v) = \phi_1(v)$, we have $\phi_1'(v) = \frac{2}{(v-1)^3}$, $\phi_1''(v) = \frac{6}{(v-1)^4}$ and so $\mathrm{d}\boldsymbol{\phi}(\vec{\boldsymbol{r}}) = (\frac{2}{(\vec{\boldsymbol{r}}_1-1)^3}, ..., \frac{2}{(\vec{\boldsymbol{r}}_{16}-1)^3})^T$ in (4.36). In (4.38), the $i$th diagonal element is $[\mathrm{d}^2\boldsymbol{\phi}(\vec{\boldsymbol{r}})]_{ii} = \frac{6}{(\vec{\boldsymbol{r}}_i-1)^4}$, $1 \le i \le 16$. Similarly, when $\phi(v) = \phi_2$, $\mathrm{d}\boldsymbol{\phi}(\vec{\boldsymbol{r}}) = (\frac{-\vec{\boldsymbol{r}}_1-1}{(\vec{\boldsymbol{r}}_1-1)^2}, ..., \frac{-\vec{\boldsymbol{r}}_{16}-1}{(\vec{\boldsymbol{r}}_{16}-1)^2})^T$ and $[\mathrm{d}^2\boldsymbol{\phi}(\vec{\boldsymbol{r}})]_{ii} = \frac{2\vec{\boldsymbol{r}}_i+4}{(\vec{\boldsymbol{r}}_i-1)^4}$. And when $\phi(v) = \phi_3$, $\mathrm{d}\boldsymbol{\phi}(\vec{\boldsymbol{r}}) = (\frac{-2\vec{\boldsymbol{r}}_1}{(\vec{\boldsymbol{r}}_1-1)^3}, ..., \frac{-2\vec{\boldsymbol{r}}_{16}}{(\vec{\boldsymbol{r}}_{16}-1)^3})^T$ and $[\mathrm{d}^2\boldsymbol{\phi}(\vec{\boldsymbol{r}})]_{ii} = \frac{4\vec{\boldsymbol{r}}_i+2}{(\vec{\boldsymbol{r}}_i-1)^4}$.

Hence, we can get $d_3$ in (4.36) and $\hat{H}_3$ in (4.38).

# Chapter 5

# Improved Optimization Methods for Image Registration Problems

In this chapter, we propose new multilevel optimization methods for minimizing continuously differentiable functions obtained by discretizing models for image registration problems [15]. These multilevel schemes rely on a novel two-step Gauss-Newton method, in which a second step is computed within each iteration by minimizing a quadratic approximation of the objective function over a specific 2D subspace. Numerical results on image registration problems show that the proposed methods can outperform the standard multilevel Gauss-Newton method.

## 5.1 Introduction

In this chapter, following from Section 3.4, we reconsider first-discretize-then-optimize and propose two simple techniques to improve the performance of the multilevel Gauss-Newton method on image registration problems. The first technique consists of the possible use of a second step within each iteration of the Gauss-Newton method. This step is computed by minimizing a quadratic approximation of the objective function over a 2D subspace. This subspace is spanned by the steepest descent direction and the L-BFGS direction with respect to the current point given by the Gauss-Newton step. If such a subspace step provides any decrease in the objective function, it is accepted; otherwise, it is discarded. The second technique is a modification of the standard coarse-to-fine multilevel strategy. At each level, instead of using the interpolated solution of the previous level directly as the initial point, we try to find a better initial point by minimizing a quadratic approximation of the objective function over the subspace spanned by the interpolated solutions of all the previous levels. If this new point results in a decrease of the objective function value, it is accepted as the new initial point; otherwise, we proceed as in the standard coarse-to-fine approach.

This chapter is organized as follows. In Section 5.2, we describe the methods resulting from the two proposed techniques. We also present a convergence analysis for these schemes. In Section 5.3, we report the results of extensive numerical experiments showing the effectiveness of our new methods. Finally, in Section 5.4, we summarize the contribution of this chapter.

## 5.2 Optimization Methods

In this section, we present the optimization methods resulting from the use of our two novel subspace techniques, which are inspired by [134]. For clarity, we briefly review the standard multilevel Gauss-Newton method used in Section 4.4.2.

### 5.2.1 Multilevel Gauss-Newton Method

Consider the optimization problem

$$\min_{\boldsymbol{u}\in\mathcal{U}} \mathcal{J}(\boldsymbol{u}), \tag{5.1}$$

where $\mathcal{J}$ is a functional from an infinite-dimensional vector space $\mathcal{U}$ to $\mathbb{R}$. Let $\mathcal{U}_l$ be a finite-dimensional subspace of $\mathcal{U}$ with basis $\{\phi_{l,j}\}_{j=1}^{n_l}$ at grid level $l$, where $n_l$ is the dimension of $\mathcal{U}_l$. By definition, this means that given any $\boldsymbol{u}_l \in \mathcal{U}_l$, there exists a vector $u_l = (u_{l,1}, ..., u_{l,n_l}) \in \mathbb{R}^{n_l}$ such that

$$\boldsymbol{u}_l = \sum_{j=1}^{n_l} u_{l,j}\phi_{l,j}. \tag{5.2}$$

Suppose that we have nested spaces $\mathcal{U}_{N_0} \subset \cdots \subset \mathcal{U}_{N-1} \subset \mathcal{U}_N \subset \mathcal{U}$. For each level $l$, we shall consider the discrete functional $J_l : \mathbb{R}^{n_l} \to \mathbb{R}$ given by

$$J_l(u_l) = \mathcal{J}(\boldsymbol{u}_l), \tag{5.3}$$

where $\boldsymbol{u}_l$ is computed by (5.2). Thus, on each level $l$, the discretized version of (5.1) is

$$\min_{u_l\in\mathbb{R}^{n_l}} J_l(u_l). \tag{5.4}$$

In the first-discretize-then-optimize approach, our goal is to obtain an approximated solution of (5.2) by solving iteratively its discrete version (5.4) for $l = N$. Given an initial guess $u_l^0$ for the solution of (5.4), Newton method generates a sequence $\{u_l^k\}$ by the rule $u_l^{k+1} = u_l^k + \theta_l^k p_l^k$, with

$$\nabla^2 J_l(u_l^k)p_l^k = -\nabla J_l(u_l^k), \tag{5.5}$$

where $\nabla J_l(u_l^k)$ and $\nabla^2 J_l(u_l^k)$ are the gradient and the Hessian of $J_l$ at $u_l^k$ respectively. However, in many situations, the structure of the objective $J_l$ gives

$$\nabla^2 J_l(u_l^k) = \hat{H}_l^k + A_l^k, \tag{5.6}$$

where $\hat{H}_l^k \in \mathbb{R}^{n_l \times n_l}$ is a symmetric and positive definite matrix that is easily computed, while $A_l^k$ is difficult to compute. For these cases, the common approach is the Gauss-Newton method, where the sequence $\{u_l^k\}$ is defined similarly but, in contrast to (5.5), $p_l^k$ is obtained by solving the linear system

$$\hat{H}_l^k p_l^k = -\nabla J_l(u_l^k). \tag{5.7}$$

If the step length $\theta_l^k$ is computed by the Armijo line search, we have Algorithm 12.

**Algorithm 12:** Gauss-Newton Method: $u_l^* = GN(l, u_l^0)$

Compute $\nabla J_l(u_l^0)$ and $\hat{H}_l^0 \approx \nabla^2 J_l(u_l^0)$;
Set $\eta = 10^{-4}$ and $k = 0$;
**while** *the stopping criteria is not satisfied* **do**
  Compute $p_l^k$ by solving $\hat{H}_l^k p_l^k = -\nabla J_l(u_l^k)$;
  Find the smallest integer $i_k \geq 0$ such that $\theta_l^k = (0.5)^{i_k}$ satisfies $J_l(u_l^k + \theta_l^k p_l^k) \leq J_l(u_l^k) + \eta \theta_l^k \nabla J_l(u_l^k)^T p_l^k$;
  $u_l^{k+1} = u_l^k + \theta_l^k p_l^k$;
  Compute $\nabla J_l(u_l^{k+1})$ and $\hat{H}_l^{k+1} \approx \nabla^2 J_l(u_l^0)$;
  $i = i + 1$;
**end**
$u_l^* = u_l^k$.

To provide the initial guess and speed up the algorithm, the multilevel Gauss-Newton method can be summarized in the following way (Algorithm 13).

**Algorithm 13:** Multilevel Gauss Newton Method

Set $u_{N_0} = (0, ..., 0) \in \mathbb{R}^{n_{N_0}}$ and $l = N_0$ (coarsest level);
**while** $l < N$ **do**
  Compute $u_l^* = GN(l, u_l^0)$;
  $u_{l+1}^0 = I_l^{l+1} u_l^*$;
  $l = l + 1$;
**end**
$u_N^* = GN(l, u_N^0)$.

For this method, in each iteration, the main computational cost is from solving the linear system. Especially for huge dimensional problems, solving the linear system is likely to be expensive. Motivated by this point, we design a novel two-step Gauss-Newton method, in which a second step is computed within each iteration by minimizing a

quadratic approximation of the objective function over a certain 2D subspace spanned by the steepest descent direction and the L-BFGS direction.

### 5.2.2 Two-Step Gauss-Newton Method

In order to enhance the performance of the Gauss-Newton method, we consider the use of a second step after the Gauss-Newton step within each iteration. If we obtain any reduction in the objective function value, the new step is accepted. Otherwise, the new step is rejected. Since we are dealing with large-scale problems, this additional step must be cheap to compute. Therefore, we propose the following subspace procedure. Denote by $\tilde{u}_l^{k+1}$ the Gauss-Newton iterate computed at step 6 of Algorithm 12, that is

$$\tilde{u}_l^{k+1} = u_l^k + \theta_l^k p_l^k, \quad \text{with } \hat{H}_l^k p_l^k = -\nabla J_l(u_l^k). \tag{5.8}$$

Let $\tilde{H}_l^{k+1}$ be the Gauss-Newton approximation to $\nabla^2 J_l(\tilde{u}_l^{k+1})$ and consider the quadratic model of $J_l$ around $\tilde{u}_l^{k+1}$:

$$m_l(\tilde{u}_l^{k+1} + p) = J_l(\tilde{u}_l^{k+1}) + \nabla J_l(\tilde{u}_l^{k+1})^T p + \frac{1}{2} p^T \hat{H}_l^{k+1} p. \tag{5.9}$$

We compute the second step $\tilde{p}_l^{k+1}$ by minimizing $m_l(\tilde{u}_l^{k+1} + p)$ over the subspace

$$\mathcal{V} = \text{span}\{p_{l,SD}^{k+1}, p_{l,QN}^{k+1}\}, \tag{5.10}$$

where $p_{l,SD}^{k+1} = -\nabla J_l(\tilde{u}_l^{k+1})$ and $p_{l,QN}^{k+1} = -B_l^{k+1} \nabla J_l(\tilde{u}_l^{k+1})$ with $B_l^{k+1}$ being the approximation to $(\nabla^2 J_l(\tilde{u}_l^{k+1}))^{-1}$ given by the L-BFGS formula [82]. More specifically,

$$\tilde{p}_l^{k+1} = c_1 p_{l,SD}^{k+1} + c_2 p_{l,QN}^{k+1}, \tag{5.11}$$

where $c = (c_1, c_2)^T \in \mathbb{R}^2$ is a solution of the quadratic minimization problem

$$\min_{c \in \mathbb{R}^2} (g_l^{k+1})^T c + \frac{1}{2} c^T \mathcal{Q}_l^{k+1} c, \tag{5.12}$$

with

$$g_l^{k+1} = \begin{pmatrix} \nabla J_l(\tilde{u}_l^{k+1})^T p_{l,SD}^{k+1} \\ \nabla J_l(\tilde{u}_l^{k+1})^T p_{l,QN}^{k+1} \end{pmatrix} \tag{5.13}$$

and

$$\mathcal{Q}_l^{k+1} = \begin{pmatrix} (p_{l,SD}^{k+1})^T \tilde{H}_l^{k+1} p_{l,SD}^{k+1} & (p_{l,SD}^{k+1})^T \tilde{H}_l^{k+1} p_{l,QN}^{k+1} \\ (p_{l,QN}^{k+1})^T \tilde{H}_l^{k+1} p_{l,SD}^{k+1} & (p_{l,QN}^{k+1})^T \tilde{H}_l^{k+1} p_{l,QN}^{k+1} \end{pmatrix}. \tag{5.14}$$

Problem (5.12) is equivalent to the $2 \times 2$ linear system

$$\mathcal{Q}_l^{k+1} c = -g_l^{k+1}, \tag{5.15}$$

which makes the computation of the second step $\tilde{p}_l^{k+1}$ in (5.11) very cheap. If $J_l(\tilde{u}_l^{k+1} + \tilde{p}_l^{k+1}) < J_l(\tilde{u}_l^{k+1})$, then we accept the new step and we define $u_l^{k+1} = \tilde{u}_l^{k+1} + \tilde{p}_l^{k+1}$.

Otherwise, we reject the new step and we define $u_l^{k+1} = \tilde{u}_l^{k+1}$. The resulting two-step Gauss-Newton method can be summarized as follows.

**Algorithm 14:** Two-Step Gauss Newton Method: $u_l^* = 2SGN(l, u_l^0)$

Compute $\nabla J_l(u_l^0)$ and $\hat{H}_l^0 \approx \nabla^2 J_l(u_l^0)$;
Set $B_l^0 = I$, $m = 3$, $\eta = 10^{-4}$ and $k = 0$;
**while** *the stopping criteria is not satisfied* **do**
  Compute $p_l^k$ by solving $\hat{H}_l^k p_l^k = -\nabla J_l(u_l^k)$;
  Find the smallest integer $i_k \geq 0$ such that $\theta_l^k = (0.5)^{i_k}$ satisfies $J_l(u_l^k + \theta_l^k p_l^k) \leq J_l(u_l^k) + \eta\theta_l^k \nabla J_l(u_l^k)^T p_l^k$;
  $\tilde{u}_l^{k+1} = u_l^k + \theta_l^k p_l^k$;
  Compute $\nabla J_l(\tilde{u}_l^{k+1})$ and $\tilde{H}_l^{k+1} \approx \nabla^2 J_l(\tilde{u}_l^{k+1})$;
  (L-BFGS Direction) Compute $p_{l,QN}^{k+1}$ by Algorithm 5;
  (Second Step) Let $p_{l,SD}^{k+1} = -\nabla J_l(\tilde{u}_l^{k+1})$, compute $c = (c_1, c_2)^T \in \mathbb{R}^2$ by solving (5.15) and then set $\tilde{p}_l^{k+1} = c_1 p_{l,SD}^{k+1} + c_2 p_{l,QN}^{k+1}$;
  **if** $J_l(\tilde{u}_l^{k+1} + \hat{p}_l^{k+1}) < J_l(\tilde{u}_l^{k+1})$ **then**
    $u_l^{k+1} = \tilde{u}_l^{k+1} + \tilde{p}_l^{k+1}$;
    Compute $J_l(u_l^{k+1})$ and $\hat{H}_l^{k+1} \approx \nabla^2 J_l(u_l^{k+1})$;
  **else**
    $u_l^{k+1} = \tilde{u}_l^{k+1}$;
    $\hat{H}_l^{k+1} = \tilde{H}_l^{k+1}$;
  **end**
  $k = k + 1$;
**end**
$u_l^* = u_l^k$.

Finally, if at step 3 and 7 of Algorithm 13, we replace the Gauss-Newton method by our new two-step Gauss-Newton method, we obtain the multilevel algorithm below.

**Algorithm 15:** Multilevel Two-Step Gauss Newton Method

Set $u_{N_0} = (0, ..., 0) \in \mathbb{R}^{n_{N_0}}$ and $l = N_0$ (coarsest level);
**while** $l < N$ **do**
  Compute $u_l^* = 2SGN(l, u_l^0)$;
  $u_{l+1}^0 = I_l^{l+1} u_l^*$;
  $l = l + 1$;
**end**
$u_N^* = 2SGN(l, u_N^0)$.

### 5.2.3 Convergence Analysis

The analysis of Algorithm 12 and 14 in a constrained setting can be done in a unified framework. Consider the finite-dimensional optimization problem

$$\min_{u\in\mathbb{R}^n} J(u), \quad \text{s.t. } u \in \mathcal{U}, \tag{5.16}$$

where $J : \mathbb{R}^n \to \mathbb{R}$ is a differentiable function and $\mathcal{U} \in \mathbb{R}^n$ is an open set. Clearly, problem (5.16) may have no solution. Thus, we seek for the iterative method that generates sequences $\{u^k\} \in \mathcal{U}$ of feasible points such that $\{J(u^k)\}$ is monotonically decreasing. By incorporating the constraint within the Armijo line search in Algorithms 12 and 14 (and omitting the level index $l$), the resulting algorithms can be seen as particular cases of the following framework.

**Algorithm 16:** Feasible Direction Method

Given $u^0 \in \mathcal{U}$, $B^0 \in \mathbb{R}^{n\times n}$ symmetric and positive definite;
Set $\eta \in (0, 1)$ and $k = 0$;
**while** *the stopping criteria is not satisfied* **do**
  Compute $p^k = -B^k \nabla J(u^k)$;
  Find the smallest integer $i_k \geq 0$ such that $\theta^k = (0.5)^{i_k}$ satisfies $J(u^k + \theta^k p^k) \leq J(u^k) + \eta\theta^k \nabla J(u^k)^T p^k$ and $u^k + \theta^k p^k \in U$;
  $\tilde{u}^{k+1} = u^k + \theta^k p^k$;
  Find $u^{k+1} \in U$ such that $J(u^{k+1}) < J(\tilde{u}^{k+1})$ and choose $B^{k+1} \in \mathbb{R}^{n\times n}$ symmetric and positive definite;
  $k = k + 1$;
**end**
$u^* = u^k$.

*Remark* 5.1. In Algorithm 16, $B^k$ is the inverse of the Gauss-Newton matrix, that is, $B^k = (\hat{H}^k)^{-1}$. To better see the correspondence between Algorithm 16 and Algorithm 12 and 14, note that in Algorithm 12, we set $u^{k+1} = \tilde{u}^{k+1}$ for all $k$, while in Algorithms 14, we may have $u^{k+1} \neq \tilde{u}^{k+1}$ if the second step is successful.

We shall study the worst-case complexity and global convergence properties of Algorithm 16. By worst-case complexity, we mean an upper bound on the maximum number of iterations that Algorithm 16 may take to find an approximated critical point of $J$ or a point close to the boundary of the feasible set. Our analysis is an adaptation of the analysis of Nesterov [96] for the gradient method. Consider the following assumptions:

A1 The objective $J : \mathbb{R}^n \to \mathbb{R}$ is differentiable and $\nabla J : \mathbb{R}^n \to \mathbb{R}$ is $L$-Lipschitz:

$$\|\nabla J(w) - \nabla J(u)\| \leq L\|w - u\|, \forall w, u \in \mathbb{R}^n. \tag{5.17}$$

A2 The set $L(u^0) = \{u \in \mathbb{R}^n | J(u) \leq J(u^0)\}$ is compact.

A3 There exist constants $c_1 \geq c_0 > 0$ such that the largest eigenvalue $\lambda_1$ and smallest eigenvalue $\lambda_0$ of $B^k$ satisfy the following condition:

$$c_0 \leq \lambda_0 \leq \lambda_1 \leq c_1. \tag{5.18}$$

The next lemma shows that if $\nabla J(u^k) \neq 0$, then there exists $i_k \geq 0$ such that the step 5 in the Algorithm 16 is well-defined. The proof is based on elementary analysis arguments and it is included here for completeness.

**Lemma 5.2.** *Suppose that A1 holds. Given $\bar{u} \in U, B \in \mathbb{R}^{n\times n}$ positive definite and $\eta \in (0,1)$, let $p = -B\nabla J(\bar{u})$. If $\nabla J(\bar{u}) \neq 0$, then there exists $\delta > 0$ such that*

$$J(\bar{u} + \theta p) \leq J(\bar{u}) + \eta\theta \nabla J(u)^T p \text{ and } \bar{u} + \theta p \in U, \tag{5.19}$$

*for all $\theta \in [0, \delta)$.*

*Proof.* Since $U$ is an open set, there exists $\epsilon > 0$ such that

$$\|u - \bar{u}\| \leq \epsilon \Longrightarrow u \in U. \tag{5.20}$$

Thus, if we consider $u = \bar{u} + \theta p$, it follows that

$$0 \leq \theta \leq \frac{\epsilon}{\|p\|} \Longrightarrow \bar{u} + \theta p \in U. \tag{5.21}$$

Let us denote $\delta_1 = \epsilon/\|p\|$. On the other hand, as $J$ is differentiable and $\eta \in (0,1)$, we have

$$\begin{aligned} \lim_{\theta\to 0} \frac{J(\bar{u}+\theta p) - J(\bar{u})}{\theta} &= \nabla J(\bar{u})^T p = -\nabla J(\bar{u})^T B \nabla J(\bar{u}) \\ &< -\eta \nabla J(\bar{u})^T B \nabla J(\bar{u}) \\ &= \eta \nabla J(\bar{u})^T p. \end{aligned} \tag{5.22}$$

Hence, there exists $\delta_2 > 0$ such that

$$\frac{J(\bar{u}+\theta p) - J(\bar{u})}{\theta} < \eta \nabla J(\bar{u})^T p, \tag{5.23}$$

for all $\theta \in (0, \delta_2)$. Therefore,

$$J(\bar{u} + \theta p) \leq J(\bar{u}) + \eta\theta \nabla J(\bar{u})^T p, \ \forall \theta \in [0, \delta_2). \tag{5.24}$$

Finally, if we take $\delta = \min\{\delta_1, \delta_2\}$, it follows from (5.21) and (5.24) that

$$J(\bar{u} + \theta p) \leq J(\bar{u}) + \eta\theta \nabla J(u)^T p \text{ and } \bar{u} + \theta p \in U, \ \forall \theta \in [0, \delta),$$

and the proof is complete. □

The lemma below gives a lower bound for the sequence $\{\theta^k\}$ and will be crucial to establish a lower bound for the functional decrease obtained in the consecutive iterations of Algorithm 16. Its proof is an adaptation of the proof of Lemma 11.1.1 in [115].

**Lemma 5.3.** *Suppose that A1 holds. Then, for all $k$, we have*

$$\theta^k \geq \min\left\{1, \frac{1-\eta}{L}\left(-\frac{\nabla J(u^k)^T p^k}{\|p^k\|^2}\right), \frac{\Gamma(u^k)}{2\|p^k\|}\right\}, \tag{5.25}$$

*where , for all $u \in \mathcal{U}$,*

$$\Gamma(u) = \inf_{w \notin \mathcal{U}} \|u - w\|. \tag{5.26}$$

*Proof.* If $i_k = 0$, then $\theta^k = 1$ and so (5.25) holds. Thus, suppose that $i_k > 0$. If $u^k + 2\theta^k p^k \in \mathcal{U}$, then from the definition of $i_k$, we know that $u^k + 2\theta^k p^k = u^k + (0.5)^{i_k - 1} p^k$ does not satisfy the inequality (5.19). Thus,

$$J(u^k + 2\theta^k p^k) > J(u^k) + 2\eta\theta^k \nabla J(u^k)^T p^k. \tag{5.27}$$

Since $\nabla J$ is $L-$Lipschitz, it follows that

$$J(u^k + 2\theta^k p^k) \leq J(u^k) + 2\theta^k \nabla J(u^k)^T p^k + 2L(\theta^k)^2 \|p^k\|^2. \tag{5.28}$$

Then, combining (5.27) and (5.28), we have

$$J(u^k) + 2\eta\theta^k \nabla J(u^k)^T p^k < J(u^k) + 2\theta^k \nabla J(u^k)^T p^k + 2L(\theta^k)^2 \|p^k\|^2 \tag{5.29}$$

and

$$\theta^k > -\frac{1-\eta}{L}\left(\frac{\nabla J(u^k)^T p^k}{\|p^k\|^2}\right) \tag{5.30}$$

and so, (5.25) also holds.

Finally, if $u^k + 2\theta^k p^k \notin \mathcal{U}$, it follows from the definition of $\Gamma(u^k)$ that

$$2\theta^k \|p^k\| \geq \Gamma(u^k). \tag{5.31}$$

Thus, $\theta^k \geq \Gamma(u^k)/2\|p^k\|$, and once again (5.25) holds. □

Now, we are in position to establish a worst-case complexity bound for Algorithm 16.

**Theorem 5.4.** *Suppose that A1-A3 hold and let $\{u^k\}$ be a sequence generated by Algorithm 16 such that*

$$\Gamma(u^k) > \epsilon \quad \text{and} \quad \|\nabla J(u^k)\| > \epsilon, \quad \text{for } k = 0, ..., T-1, \tag{5.32}$$

*for a given precision $\epsilon > 0$. Then, $J(u)$ is bounded from below by some $J_{\text{low}}$ and we must have*

$$T \leq \left( \frac{J(u^0) - J_{\text{low}}}{\kappa_c} \right) \epsilon^{-2}, \tag{5.33}$$

*where*

$$\kappa_c = \min \left\{ \eta c_0, \frac{\eta(1-\eta)c_0^2}{L c_1^2}, \frac{\eta c_0}{2c_1} \right\}. \tag{5.34}$$

*Proof.* By step 7 of Algorithm 16, we have $J(u^{k+1}) < J(\tilde{u}^{k+1})$. Thus, combining (5.19) and the lower bound for $\theta^k$ in (5.25), we obtain the following lower bound for the decrease of the function value in the consecutive iterations:

$$\begin{aligned} &J(u^k) - J(u^{k+1}) \geq J(u^k) - J(\tilde{u}^{k+1}) \geq \eta\theta^k(-\nabla J(u^k)^T p^k) \\ &\geq \eta \min \left\{ -\nabla J(u^k)^T p^k, \frac{1-\eta}{L} \left( -\frac{\nabla J(u^k)^T p^k}{\|p^k\|} \right)^2, \frac{\Gamma(u^k)}{2} \left( -\frac{\nabla J(u^k)^T p^k}{\|p^k\|} \right) \right\}. \end{aligned} \tag{5.35}$$

On the other hand, from A3 it follows that

$$\|p^k\| = \| - B^k \nabla J(u^k)\| \leq \|B^k\| \|\nabla J(u^k)\| \leq c_1 \|\nabla J(u^k)\|, \tag{5.36}$$

and

$$-\nabla J(u^k)^T p^k = \nabla J(u^k)^T B^k \nabla J(u^k) \geq c_0 \|\nabla J(u^k)\|^2. \tag{5.37}$$

Hence,

$$-\frac{\nabla J(u^k)^T p^k}{\|p^k\|} \geq \frac{c_0 \|\nabla J(u^k)\|^2}{c_1 \|\nabla J(u^k)\|} = \frac{c_0}{c_1} \|\nabla J(u^k)\|. \tag{5.38}$$

Then combining (5.35) with (5.37), (5.38) and (5.32), we obtain

$$\begin{aligned} J(u^k) - J(u^{k+1}) &\geq \eta \min \left\{ c_0 \|\nabla J(u^k)\|^2, \frac{(1-\eta)c_0^2}{L c_1^2} \|\nabla J(u^k)\|^2, \frac{c_0}{2c_1} \Gamma(u^k) \|\nabla J(u^k)\| \right\} \\ &\geq \min \left\{ \eta c_0, \frac{\eta(1-\eta)c_0^2}{L c_1^2}, \frac{\eta c_0}{2c_1} \right\} \min \left\{ \|\nabla J(u^k)\|^2, \Gamma(u^k) \|\nabla J(u^k)\| \right\} \\ &= \kappa_c \min \left\{ \|\nabla J(u^k)\|^2, \Gamma(u^k) \|\nabla J(u^k)\| \right\} \\ &> \kappa_c \epsilon^2, \quad \text{for} \quad k = 0, ..., T-1. \end{aligned} \tag{5.39}$$

From A2, it follows that $J$ has a global minimizer on $\mathbb{R}^n$. Thus, there exists $J_{\text{low}}$ such that $J(u^k) \geq J_{\text{low}}$ for all $k$. Therefore,

$$J(u^0) - J_{\text{low}} \geq J(u^0) - J(u^T) = \sum_{k=0}^{T-1} J(u^k) - J(u^{k+1}) \geq \sum_{k=0}^{T-1} \kappa_c \epsilon^2 = T \kappa_c \epsilon^2. \tag{5.40}$$

Then we have

$$T \leq \left( \frac{J(u^0) - J_{\text{low}}}{\kappa_c} \right) \epsilon^{-2}, \tag{5.41}$$

and the proof is complete. □

*Remark* 5.5. Theorem 5.4 means that given $\epsilon > 0$, Algorithm 16 takes at most $\mathcal{O}(\epsilon^{-2})$ iterations to generate a point $u^T \in \mathcal{U}$ such that

$$\Gamma(u^T) \leq \epsilon \quad \text{or} \quad \|\nabla J(u^T)\| \leq \epsilon. \tag{5.42}$$

For $\mathcal{U} = \mathbb{R}^2$, this bound agrees in order with known complexity bounds for first-order methods [12, 47, 96]. In any case, by (5.39), we have

$$J(u^T) < J(u^{T-1}) < \cdots J(u^1) < J(u^0). \tag{5.43}$$

Finally, from inequality (5.39), we can establish the following global convergence result.

**Theorem 5.6.** *Suppose that A1-A3 hold. Then, given $u^0 \in \mathcal{U}$, the sequence $\{u^k\} \subset \mathcal{U}$ generated by Algorithm 16 from $u^0$ admits a subsequence that converges either to a point in the boundary of $\mathcal{U}$ or to a critical point of $J$ in $\mathcal{U}$.*

*Proof.* Let us denote the closure of $\mathcal{U}$ by $\bar{\mathcal{U}}$. Note that $\{u^k\} \subset L(u^0)$. Thus, by A2, the sequence $\{u^k\}$ is bounded and, therefore, it admits a convergent subsequence $\{u^{k_j}\}$, with $u^{k_j} \to \bar{u} \in \bar{\mathcal{U}}$. Since $J$ is continuous, we also have $J(u^{k_j}) \to J(\bar{u})$ as $j$ goes to infinity. Thus, sequence $\{J(u^k)\}$ is monotonically decreasing and admits a convergent subsequence. Hence, $\{J(u^k)\}$ must be convergent, which implies that

$$\lim_{j\to\infty} J(u^k) - J(u^{k+1}) = 0. \tag{5.44}$$

Thus, by applying the Squeeze Theorem on inequality (5.39), we conclude that

$$\lim_{j\to\infty} \nabla J(u^{k_j}) = 0 \quad \text{or} \quad \lim_{j\to\infty} \Gamma(u^{k_j})\nabla J(u^{k_j}) = 0. \tag{5.45}$$

On the other hand, as $\nabla J$ and $\Gamma$ are continuous functions, we have

$$\lim_{j\to\infty} \nabla J(u^{k_j}) = \nabla J(\bar{u}) \quad \text{or} \quad \lim_{j\to\infty} \Gamma(u^{k_j}) = \Gamma(\bar{u}). \tag{5.46}$$

Then, combining (5.45) and (5.46), it follows that

$$\nabla J(\bar{u}) = 0 \quad \text{or} \quad \Gamma(\bar{u}) = 0, \tag{5.47}$$

that is, the limit point $\bar{u}$ is either a point in the boundary of $\mathcal{U}$ or a critical point of $J$ in $\mathcal{U}$. □

### 5.2.4 Subspace Multilevel Technique

In the standard coarse-to-fine multilevel strategy, the initial point $u_{l+1}^0$ for level $l+1$ is computed using only the solution $u_l^*$ of the previous level. To allow the finding of a better initial point, we propose the use of all the previous solutions $u_l^*, u_{l-1}^*, ..., u_{N_0}^*$ by employing again the subspace technique. Given $N_0 \leq z \leq w \leq N$, let us denote by $I_z^w$ the prolongation operator from level $z$ to level $w$. We set $\tilde{u}_{l+1}^0 = I_l^{l+1} u_l^*$ and we compute $\nabla J_{l+1}(\tilde{u}_{l+1}^0)$ and $\tilde{H}_{l+1}^0 \approx \nabla^2 J_{l+1}(\tilde{u}_{l+1}^0)$. The we obtain a search direction $\tilde{p}_{l+1}^0$ by solving the subspace quadratic problem

$$\begin{aligned} &\min_p J_{l+1}(\tilde{u}_{l+1}^0) + \nabla J_{l+1}(\tilde{u}_{l+1}^0)^T p + \frac{1}{2} p^T \tilde{H}_{l+1}^0 p, \\ &\text{s.t. } p \in \mathcal{V}_{l+1}^0 \in \mathbb{R}^{l-N_0+1}, \end{aligned} \tag{5.48}$$

where $\mathcal{V}_{l+1}^0 = \text{span}\left\{ I_{N_0}^{l+1} u_{N_0}^*, ..., I_l^{l+1} u_l^* \right\}$. As in the two-step Gauss-Newton method, $\tilde{p}_{l+1}^0$ can be easily computed by solving a small scale linear system. If $J_{l+1}(\tilde{u}_{l+1}^0 + \tilde{p}_{l+1}^0) < J_{l+1}(\tilde{u}_{l+1}^0)$, we define the initial point for level $l+1$ as $u_{l+1}^0 = \tilde{u}_{l+1}^0 + \tilde{p}_{l+1}^0$. Otherwise, we set $u_{l+1}^0 = \tilde{u}_{l+1}^0$. The corresponding modification of Algorithm 13 can be summarized in the following way.

**Algorithm 17:** Subspace Multilevel Gauss Newton Method

1 Set $u_{N_0} = (0, ..., 0) \in \mathbb{R}^{n_{N_0}}$ and $l = N_0$ (coarsest level);
2 **while** $l < N$ **do**
3 Compute $u_l^* = GN(l, u_l^0)$;
4 Use (5.48) to compute $u_{l+1}^0$;
5 $l = l + 1$;
6 **end**
7 $u_N^* = GN(l, u_N^0)$.

Finally, if at step 3 and 7 of Algorithm 17 we replace the Gauss-Newton method by our new two-step Gauss-Newton method, we obtain the subspace multilevel algorithm below.

**Algorithm 18:** Subspace Multilevel Two-Step Gauss Newton Method

1 Set $u_{N_0} = (0, ..., 0) \in \mathbb{R}^{n_{N_0}}$ and $l = N_0$ (coarsest level);
2 **while** $l < N$ **do**
3 Compute $u_l^* = 2SGN(l, u_l^0)$;
4 Use (5.48) to compute $u_{l+1}^0$;
5 $l = l + 1$;
6 **end**
7 $u_N^* = 2SGN(l, u_N^0)$.

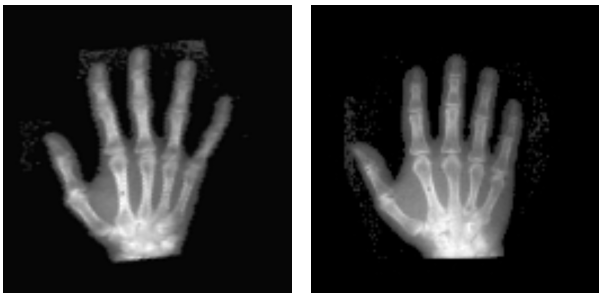
## 5.3 Numerical Experiments

In order to investigate the numerical performance of the proposed methods, we have tested implementations of the following algorithms on the hyperelastic model (3.12):

- The standard multilevel Gauss-Newton algorithm (i.e., Algorithm 13). We shall refer to this code as GN (from Gauss-Newton).
- The multilevel two-step Gauss-Newton algorithm (i.e., Algorithm 15). We shall refer to this code as TS (from two-step).
- The subspace multilevel Gauss-Newton algorithm (i.e., Algorithm 17). We shall refer to this code as SIG (from subspace initial guess).
- The subspace multilevel two-step Gauss-Newton algorithm (i.e., Algorithm 18). We shall refer to this code as HYBRID since it can be viewed as a combination of TS and SIG.

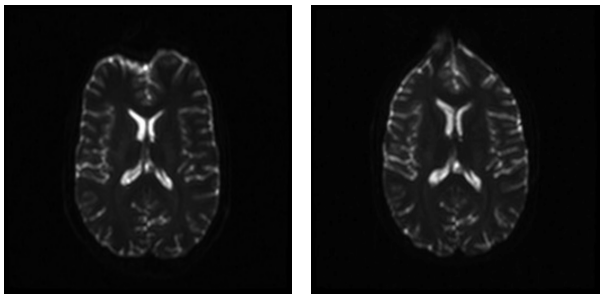
The algorithms are coded in Matlab (R2017a) languages, and the tests are performed on a PC with 3.20 GHz Intel(R) Core(TM) i5-5600 microprocessor, and with installed memory (RAM) of 8.00 GB. On all codes, the Gauss-Newton linear system is solved by the conjugate gradient (CG) method with diagonal preconditioner. We stop the execution of the CG method when the relative residual becomes smaller 0.1 or when the maximum of 50 iterations is reached.

The codes are applied to image registration problems corresponding to 20 pairs of images (reference, template): ten pairs of medical images (Figure. 5.1—5.10), and ten pairs of medical images (Figure. 5.11—5.20). To evaluate the performance of the codes for several problem sizes, we consider four different resolutions: 128×128, 256×256, 512×512 and $1024 \times 1024$. Specifically, we use the Matlab package FAIR as the basis for our tests (see details in [92]). In all codes, the constraint $\det(\nabla \boldsymbol{y}) > 0$, for $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x})$, is handled within the Armijo line search, that is, to be accepted, a trial step must provide a sufficient decrease in the objective and the resulting point must be feasible with respect to the referred constraint (see Algorithm 16).



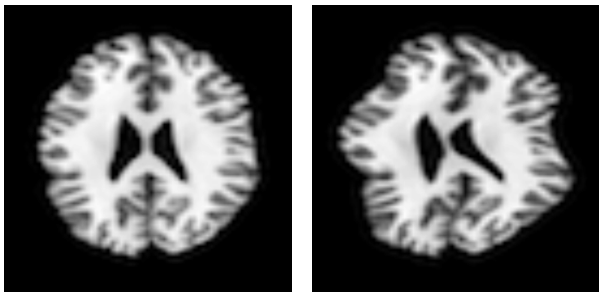
(a) Reference (b) Template

FIGURE 5.1: Problem hand.



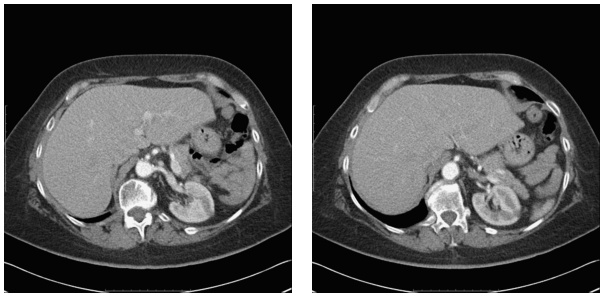
(a) Reference (b) Template

FIGURE 5.2: Problem EPslice.



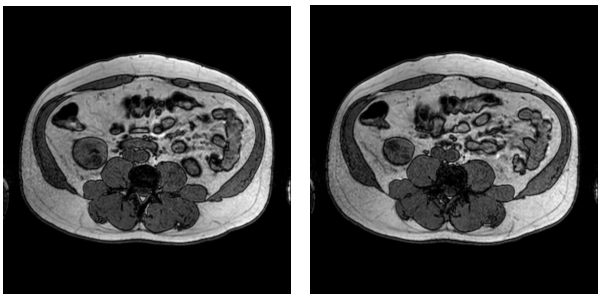
(a) Reference (b) Template

FIGURE 5.3: Problem brain.



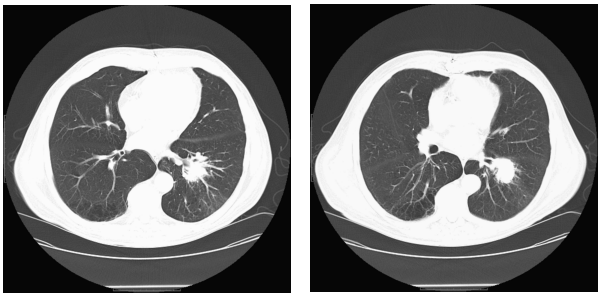
(a) Reference (b) Template

FIGURE 5.4: Problem CT.



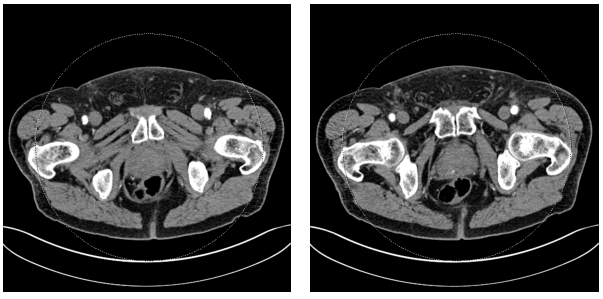
(a) Reference (b) Template

FIGURE 5.5: Problem MRI.



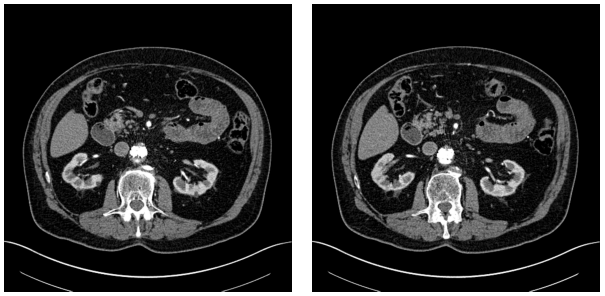
(a) Reference (b) Template

FIGURE 5.6: Problem lung.



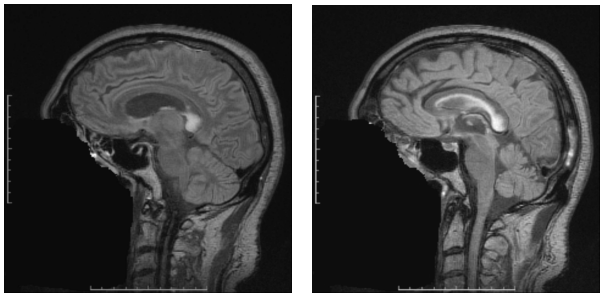
(a) Reference (b) Template

FIGURE 5.7: Problem CT1.



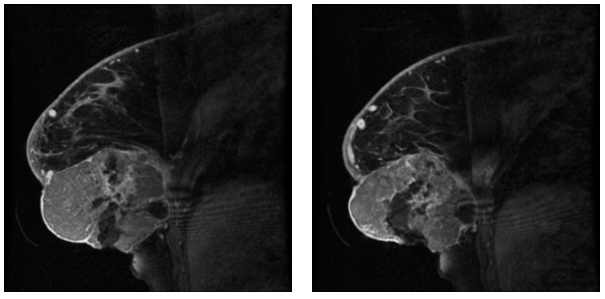
(a) Reference (b) Template

FIGURE 5.8: Problem CT2.



(a) Reference (b) Template

FIGURE 5.9: Problem MRI2.



(a) Reference (b) Template

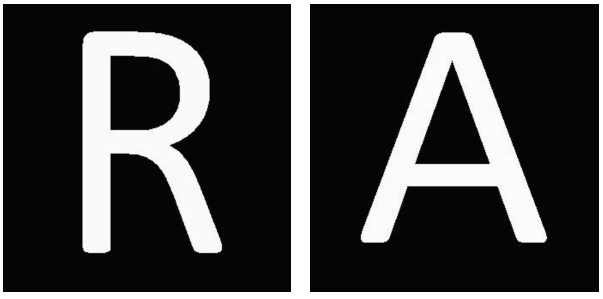
FIGURE 5.10: Problem breast.

(a) Reference (b) Template

FIGURE 5.11: Problem circle to c.

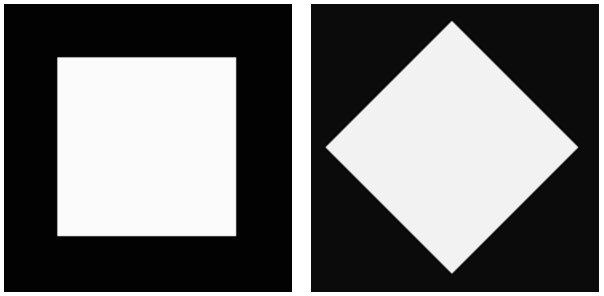
(a) Reference (b) Template

FIGURE 5.12: Problem c to circle.



(a) Reference (b) Template

FIGURE 5.13: Problem A to R.



(a) Reference (b) Template

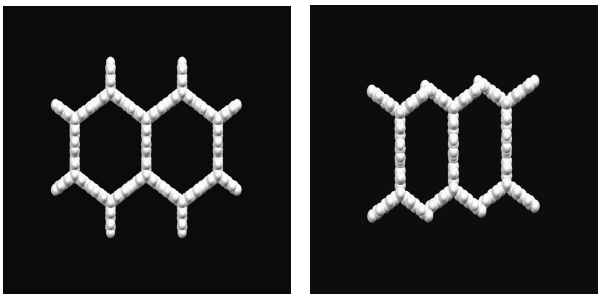
FIGURE 5.14: Problem square to square.

(a) Reference (b) Template

FIGURE 5.15: Problem Lena.

(a) Reference (b) Template

FIGURE 5.16: Problem circle to square.



(a) Reference (b) Template

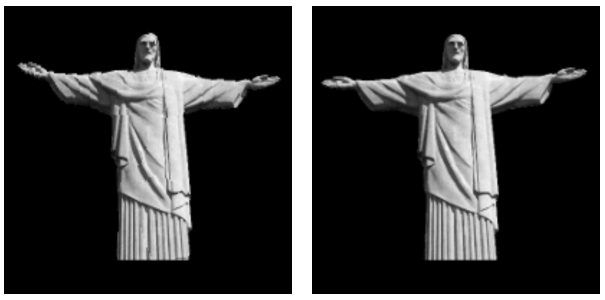
FIGURE 5.17: Problem molecule.

(a) Reference (b) Template

FIGURE 5.18: Problem F to F.

(a) Reference (b) Template

FIGURE 5.19: Problem circle to I.



(a) Reference (b) Template

FIGURE 5.20: Problem Rio.

The results reported below summarize more than 21 hours of numerical implementation. Problems and results for the resolution $128 \times 128$ are given in Table 5.1, where "TIME" represents the time in seconds taken by the code to solve the corresponding problem, "IT" represents the number of nonlinear iterations performed to reach the solution, "FE" represents the number function evaluations performed, and "TOTAL" provides the sum of the values in the corresponding column of the table, where the total time is given in seconds.

From Table 5.1, we see that TS, SIG, and HYBRID are better than GN in terms of the total time. The fastest code is HYBRID, which outperforms GN on 16 problems (9 of them corresponding to medical images).

| | GN | | | SIG | | | TS | | | HYBRID | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Problem | Time | IT | FE | Time | IT | FE | Time | IT | FE | Time | IT | FE |
| 1. Hand | 2.1 | 31 | 75 | 1.8 | 31 | 80 | 2.1 | 30 | 120 | 1.6 | 27 | 114 |
| 2. EPLslice | 11.4 | 82 | 216 | 8.6 | 71 | 190 | 5.0 | 52 | 214 | 4.7 | 47 | 210 |
| 3. Brain | 1.5 | 52 | 116 | 1.5 | 49 | 116 | 1.3 | 29 | 114 | 1.3 | 29 | 110 |
| 4. CT | 5.9 | 51 | 116 | 5.6 | 48 | 116 | 3.9 | 42 | 166 | 3.7 | 40 | 162 |
| 5. MRI | 3.3 | 56 | 123 | 3.3 | 56 | 129 | 2.7 | 38 | 152 | 2.7 | 38 | 156 |
| 6. Lung | 3.3 | 40 | 95 | 3.7 | 45 | 112 | 2.5 | 30 | 123 | 3.5 | 36 | 153 |
| 7. CT1 | 8.9 | 82 | 202 | 7.4 | 79 | 200 | 4.7 | 51 | 206 | 3.9 | 47 | 194 |
| 8. CT2 | 4.1 | 36 | 82 | 3.1 | 33 | 82 | 2.7 | 28 | 112 | 2.2 | 25 | 106 |
| 9. MRI2 | 10.2 | 95 | 206 | 9.4 | 91 | 204 | 5.3 | 57 | 228 | 5.5 | 58 | 238 |
| 10. Breast | 5.1 | 53 | 117 | 5.0 | 53 | 123 | 3.5 | 37 | 147 | 3.7 | 38 | 157 |
| 11. Circle to C | 1.4 | 45 | 103 | 1.6 | 48 | 114 | 1.4 | 37 | 131 | 1.3 | 37 | 146 |
| 12. C to Circle | 2.7 | 46 | 119 | 2.8 | 49 | 129 | 4.5 | 87 | 356 | 2.5 | 40 | 175 |
| 13. A to R | 0.8 | 39 | 88 | 0.7 | 41 | 99 | 0.8 | 23 | 94 | 0.6 | 19 | 84 |
| 14. Square to Square | 0.6 | 19 | 46 | 0.6 | 19 | 54 | 0.5 | 14 | 60 | 0.5 | 12 | 61 |
| 15. Lena | 0.6 | 18 | 46 | 0.7 | 18 | 52 | 0.7 | 17 | 68 | 0.6 | 16 | 70 |
| 16. Circle to Square | 0.2 | 12 | 34 | 0.4 | 14 | 44 | 0.2 | 11 | 44 | 0.3 | 13 | 56 |
| 17. Molecule | 0.9 | 23 | 56 | 1.0 | 22 | 60 | 1.1 | 22 | 87 | 1.1 | 20 | 84 |
| 18. F to F | 0.9 | 41 | 95 | 1.0 | 43 | 104 | 0.8 | 25 | 102 | 0.8 | 25 | 106 |
| 19. Circle to I | 1.5 | 25 | 60 | 1.2 | 28 | 73 | 1.3 | 22 | 90 | 1.3 | 19 | 80 |
| 20. Rio | 0.6 | 15 | 40 | 0.4 | 14 | 44 | 0.5 | 14 | 56 | 0.5 | 13 | 58 |
| Total | 65.6 | 861 | 2035 | 59.7 | 852 | 2125 | 45.5 | 666 | 2670 | 42.5 | 598 | 2511 |

TABLE 5.1: Results for resolution $128 \times 128$.

Table 5.2 shows the results for resolution $256 \times 256$. Codes TS and HYBRID are better than GN in terms of total time. The fastest code is HYBRID, which outperforms GN on 11 problems (7 of them corresponding to medical images).

| | GN | | | SIG | | | TS | | | HYBRID | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Problem | Time | IT | FE | Time | IT | FE | Time | IT | FE | Time | IT | FE |
| 1. Hand | 6.4 | 35 | 85 | 7.1 | 35 | 92 | 9.9 | 36 | 144 | 9.2 | 33 | 140 |
| 2. EPLslice | 110.1 | 172 | 480 | 120.6 | 176 | 500 | 88.0 | 123 | 548 | 83.2 | 120 | 534 |
| 3. Brain | 4.6 | 61 | 136 | 4.9 | 56 | 134 | 5.1 | 35 | 128 | 4.9 | 34 | 132 |
| 4. CT | 175.9 | 201 | 470 | 169.8 | 185 | 440 | 79.4 | 146 | 577 | 84.8 | 146 | 588 |
| 5. MRI | 45.4 | 92 | 212 | 37.2 | 84 | 200 | 24.3 | 63 | 254 | 22.4 | 59 | 246 |
| 6. Lung | 29.2 | 61 | 155 | 41.9 | 74 | 196 | 50.2 | 66 | 289 | 31.1 | 61 | 264 |
| 7. CT1 | 54.8 | 115 | 286 | 53.8 | 112 | 287 | 35.6 | 81 | 330 | 32.9 | 73 | 305 |
| 8. CT2 | 40.5 | 89 | 192 | 38.3 | 73 | 168 | 25.5 | 60 | 240 | 28.2 | 61 | 250 |
| 9. MRI2 | 60.2 | 136 | 305 | 60.1 | 132 | 304 | 42.4 | 97 | 391 | 43.7 | 99 | 407 |
| 10. Breast | 36.6 | 77 | 176 | 28.8 | 75 | 177 | 26.8 | 59 | 241 | 20.4 | 56 | 233 |
| 11. Circle to C | 7.0 | 51 | 117 | 6.3 | 54 | 130 | 7.2 | 42 | 148 | 4.9 | 41 | 166 |
| 12. C to Circle | 9.8 | 53 | 135 | 10.9 | 54 | 142 | 10.1 | 91 | 374 | 8.3 | 44 | 193 |
| 13. A to R | 2.5 | 41 | 94 | 3.3 | 44 | 109 | 4.5 | 28 | 114 | 2.9 | 22 | 99 |
| 14. Square to Square | 1.9 | 21 | 52 | 6.6 | 26 | 73 | 3.1 | 17 | 72 | 4.4 | 16 | 79 |
| 15. Lena | 2.2 | 20 | 52 | 2.6 | 20 | 60 | 2.6 | 19 | 76 | 2.8 | 18 | 80 |
| 16. Circle to Square | 2.3 | 15 | 42 | 2.5 | 19 | 56 | 0.4 | 12 | 48 | 0.9 | 13 | 62 |
| 17. Molecule | 5.2 | 32 | 76 | 5.7 | 27 | 74 | 6.1 | 26 | 101 | 5.2 | 24 | 101 |
| 18. F to F | 3.8 | 46 | 107 | 3.7 | 47 | 116 | 3.4 | 28 | 116 | 3.9 | 28 | 122 |
| 19. Circle to I | 6.6 | 32 | 76 | 5.9 | 32 | 83 | 5.5 | 26 | 102 | 5.7 | 23 | 96 |
| 20. Rio | 1.4 | 17 | 46 | 1.7 | 16 | 52 | 1.8 | 16 | 64 | 2.0 | 15 | 68 |
| Total | 606.6 | 1367 | 3294 | 611.6 | 1341 | 3393 | 432.0 | 1071 | 4367 | 401.9 | 986 | 4165 |

TABLE 5.2: Results for resolution $256 \times 256$.

Table 5.3 shows the results for resolution $512 \times 512$. Codes TS, SIG, and HYBRID are better than GN in terms of total time. In this case, the fastest code is HYBRID, which outperforms GN on 13 problems (9 of them corresponding to medical images).

| | GN | | | SIG | | | TS | | | HYBRID | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Problem | Time | IT | FE | Time | IT | FE | Time | IT | FE | Time | IT | FE |
| 1. Hand | 37.6 | 39 | 95 | 32.8 | 38 | 102 | 25.9 | 38 | 152 | 26.3 | 35 | 150 |
| 2. EPLslice | 220.3 | 190 | 525 | 332.3 | 207 | 586 | 286.2 | 149 | 666 | 214.0 | 137 | 617 |
| 3. Brain | 18.0 | 64 | 144 | 15.1 | 58 | 142 | 15.0 | 37 | 148 | 15.7 | 36 | 144 |
| 4. CT | 2827.4 | 524 | 1430 | 1700 | 370 | 988 | 824.3 | 253 | 1065 | 446.2 | 202 | 832 |
| 5. MRI | 303.3 | 132 | 309 | 298.5 | 117 | 286 | 209.3 | 96 | 391 | 286.9 | 105 | 436 |
| 6. Lung | 175.8 | 77 | 201 | 134.3 | 93 | 246 | 425.7 | 109 | 500 | 243.8 | 88 | 389 |
| 7. CT1 | 385.1 | 158 | 394 | 346.1 | 155 | 396 | 224.8 | 110 | 443 | 248.2 | 109 | 454 |
| 8. CT2 | 1083.4 | 208 | 537 | 1044.4 | 189 | 511 | 480.1 | 125 | 533 | 576.9 | 137 | 607 |
| 9. MRI2 | 339.4 | 167 | 388 | 270.0 | 157 | 371 | 204.3 | 125 | 507 | 249.0 | 130 | 545 |
| 10. Breast | 221.2 | 100 | 234 | 230.6 | 98 | 239 | 195.5 | 85 | 347 | 151.1 | 77 | 319 |
| 11. Circle to C | 31.2 | 54 | 125 | 57.2 | 68 | 162 | 60.9 | 51 | 172 | 58.9 | 51 | 197 |
| 12. C to Circle | 37.9 | 56 | 144 | 39.9 | 57 | 150 | 47.3 | 97 | 401 | 47.4 | 49 | 210 |
| 13. A to R | 26.2 | 49 | 112 | 13.3 | 46 | 117 | 17.8 | 30 | 121 | 14.5 | 24 | 111 |
| 14. Square to Square | 16.7 | 25 | 62 | 17.9 | 27 | 78 | 26.3 | 20 | 84 | 20.7 | 18 | 86 |
| 15. Lena | 17.2 | 22 | 58 | 19.1 | 22 | 68 | 18.3 | 21 | 84 | 19.7 | 20 | 90 |
| 16. Circle to Square | 11.4 | 20 | 54 | 13.3 | 23 | 66 | 2.6 | 13 | 52 | 3.7 | 14 | 66 |
| 17. Molecule | 74.3 | 40 | 95 | 69.7 | 46 | 116 | 74.5 | 39 | 145 | 64.7 | 37 | 150 |
| 18. F to F | 12.2 | 48 | 113 | 7.1 | 48 | 122 | 6.6 | 29 | 122 | 7.6 | 29 | 128 |
| 19. Circle to I | 32.4 | 43 | 100 | 53.1 | 42 | 105 | 45.3 | 33 | 128 | 64.3 | 31 | 125 |
| 20. Rio | 9.5 | 19 | 52 | 9.6 | 18 | 60 | 11.8 | 18 | 72 | 12.5 | 17 | 78 |
| Total | 5880.7 | 2035 | 5172 | 4704.6 | 1879 | 4911 | 3202.6 | 1478 | 6133 | 2772.3 | 1346 | 5734 |

TABLE 5.3: Results for resolution $512 \times 512$.

Finally, Table 5.4 shows the results for resolution $1024 \times 1024$. Once again, TS, SIG, and HYBRID are better than GN in terms of total time. The fastest code is TS, which outperforms GN on 12 problems (6 of them corresponding to medical images).

The improved performances of TS and HYBRID over GN are better highlighted in Tables 5.5— 5.7, which shows the reduction in the total time provided by the new methods.

| | GN | | | SIG | | | TS | | | HYBRID | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Problem | Time | IT | FE | Time | IT | FE | Time | IT | FE | Time | IT | FE |
| 1. Hand | 119.9 | 41 | 101 | 121.2 | 40 | 110 | 105.3 | 40 | 160 | 58.2 | 36 | 156 |
| 2. EPLslice | 539.9 | 198 | 550 | 589.5 | 216 | 612 | 518.2 | 155 | 693 | 654.2 | 150 | 677 |
| 3. Brain | 49.8 | 65 | 148 | 91.7 | 63 | 156 | 52.9 | 38 | 152 | 46.7 | 37 | 152 |
| 4. CT | 3884.8 | 556 | 1504 | 2299.4 | 390 | 1035 | 2478.2 | 297 | 1257 | 1526.3 | 235 | 965 |
| 5. MRI | 1048.2 | 166 | 384 | 1094.8 | 136 | 337 | 908.2 | 120 | 486 | 696.0 | 118 | 489 |
| 6. Lung | 929.0 | 102 | 258 | 1074.6 | 125 | 318 | 1615.3 | 140 | 650 | 2520.2 | 144 | 653 |
| 7. CT1 | 1246.3 | 181 | 446 | 1040.6 | 174 | 440 | 906.2 | 133 | 532 | 1050.3 | 143 | 585 |
| 8. CT2 | 6148.7 | 330 | 899 | 4924.8 | 287 | 796 | 3758.3 | 198 | 881 | 3865.9 | 215 | 982 |
| 9. MRI2 | 947.9 | 189 | 442 | 809.6 | 170 | 409 | 998.3 | 152 | 621 | 1088 | 154 | 652 |
| 10. Breast | 515.3 | 107 | 254 | 477.5 | 105 | 259 | 760.1 | 100 | 414 | 1120.6 | 104 | 440 |
| 11. Circle to C | 235.5 | 61 | 141 | 538.6 | 93 | 214 | 126.1 | 53 | 178 | 192.4 | 56 | 210 |
| 12. C to Circle | 245.3 | 61 | 156 | 239.8 | 65 | 168 | 181.1 | 100 | 414 | 204.9 | 54 | 226 |
| 13. A to R | 124.2 | 58 | 132 | 117.9 | 51 | 129 | 89.2 | 34 | 132 | 105.3 | 29 | 123 |
| 14. Square to Square | 57.8 | 26 | 66 | 106.7 | 29 | 84 | 68.0 | 21 | 88 | 64.2 | 19 | 90 |
| 15. Lena | 140.3 | 26 | 68 | 109.3 | 25 | 78 | 135.8 | 24 | 96 | 92.2 | 22 | 100 |
| 16. Circle to Square | 69.6 | 23 | 62 | 57.1 | 27 | 76 | 43.4 | 14 | 56 | 34.1 | 15 | 70 |
| 17. Molecule | 364.3 | 47 | 111 | 374.9 | 53 | 132 | 830.6 | 69 | 242 | 662.0 | 59 | 223 |
| 18. F to F | 60.4 | 51 | 121 | 63.2 | 51 | 132 | 70.5 | 31 | 130 | 81.3 | 32 | 142 |
| 19. Circle to I | 73.2 | 44 | 104 | 139.8 | 44 | 111 | 181.7 | 39 | 146 | 234.1 | 38 | 149 |
| 20. Rio | 90.9 | 22 | 60 | 69.2 | 20 | 68 | 70.6 | 20 | 80 | 78.9 | 19 | 88 |
| Total | 16,017.9 | 2354 | 6007 | 14,340 | 2164 | 5664 | 13,898 | 1778 | 7408 | 14,376.6 | 1679 | 7172 |

TABLE 5.4: Results for resolution $1024 \times 1024$.

| Resolution | Time GN | Time SIG | Reduction |
|---|---|---|---|
| $128 \times 128$ | 65.6 | 59.7 | 9.0% |
| $256 \times 256$ | 606.5 | 611.6 | – |
| $512 \times 512$ | 5880.7 | 4704.6 | 20.0% |
| $1024 \times 1024$ | 16,917.9 | 14,340 | 15.2% |

TABLE 5.5: Comparison of the total time (in seconds) to solve all 20 problems for each resolution between GN and SIG.

| Resolution | Time GN | Time TS | Reduction |
|---|---|---|---|
| $128 \times 128$ | 65.6 | 45.5 | 30.6% |
| $256 \times 256$ | 606.5 | 432.0 | 28.7% |
| $512 \times 512$ | 5880.7 | 3202.6 | 45.5% |
| $1024 \times 1024$ | 16,917.9 | 13,898 | 17.8% |

TABLE 5.6: Comparison of the total time (in seconds) to solve all 20 problems for each resolution between GN and TS.

As mentioned above, codes TS and HYBRID behave much better when we consider only medical images. In terms of the total time, this difference of performance is shown on Tables 5.8—5.10.

Additional information about the codes can be obtained by using the Performance Profile, which is a tool for benchmarking and comparing optimization software [31]. More specifically, let $t_{p,s}$ denotes the time to solve problem $p$ by solver $s$. The performance ratio is defined as $r_{p,s} = \frac{t_{p,s}}{t_p^*}$, where $t_p^*$ is the lowest time required to solve problem $p$ among all solvers that are being compared. Clearly, $r_{p,s} \geq 1$ for all $p$ and $s$. The performance profile for each code $s$ is defined as

$$\rho_s(\tau) = \frac{\text{number of problems for which } r_{p,s} \leq \tau}{\text{total number of problems}}. \tag{5.49}$$

| Resolution | Time GN | Time HYBRID | Reduction |
|---|---|---|---|
| $128 \times 128$ | 65.6 | 42.5 | 35.2% |
| $256 \times 256$ | 606.5 | 401.9 | 33.7% |
| $512 \times 512$ | 5880.7 | 2772.3 | 52.8% |
| $1024 \times 1024$ | 16,917.9 | 14,376.6 | 15.0% |

TABLE 5.7: Comparison of the total time (in seconds) to solve all 20 problems for each resolution between GN and HYBRID.

| Resolution | Time GN | Time SIG | Reduction |
|---|---|---|---|
| $128 \times 128$ | 55.7 | 49.4 | 11.3% |
| $256 \times 256$ | 563.6 | 562.6 | 0.2% |
| $512 \times 512$ | 5611.6 | 4404.5 | 21.5% |
| $1024 \times 1024$ | 15,456 | 12,524 | 18.9% |

TABLE 5.8: Comparison of the total time (in seconds) to solve all 10 problems with medical images for each resolution between GN and SIG.

| Resolution | Time GN | Time TS | Reduction |
|---|---|---|---|
| $128 \times 128$ | 55.7 | 33.7 | 39.4% |
| $256 \times 256$ | 563.6 | 387.1 | 31.3% |
| $512 \times 512$ | 5611.6 | 2891.1 | 48.4% |
| $1024 \times 1024$ | 15,456 | 12,100.9 | 21.7% |

TABLE 5.9: Comparison of the total time (in seconds) to solve all 10 problems with medical images for each resolution between GN and TS.

| Resolution | Time GN | Time HYBRID | Reduction |
|---|---|---|---|
| $128 \times 128$ | 55.7 | 32.9 | 41.0% |
| $256 \times 256$ | 563.6 | 360.8 | 36.0% |
| $512 \times 512$ | 5611.6 | 2458.3 | 56.2% |
| $1024 \times 1024$ | 15,456 | 12,627.6 | 18.3% |

TABLE 5.10: Comparison of the total time (in seconds) to solve all 10 problems with medical images for each resolution between GN and HYBRID.

Therefore, the value $\rho_s(\tau)$ represents the percentage of problems solved by algorithm $s$ with a cost most $\tau$ times worse than that of the best algorithm. This means that, for a given value of $\tau$, the best solver is the one with the highest value of $\rho_s(\tau)$. In particular, $\rho_s(1)$ gives the percentage of problems for which solver $s$ is the best.

Figure 5.21—5.23 show the performance profiles for codes GN and HYBRID taking as references all 20 problems with medical images and resolutions of $128 \times 128$, $256 \times 256$ and $512 \times 512$ (combined results of Table 5.1—5.3). As expected, we can see that in this set of problems, code HYBRID is significantly more efficient than GN in terms of CPU time and the number of iterations. It is interesting to notice that GN outperforms HYBRID in terms of the number of function evaluations. However, the effect is compensated by the time that HYBRID saves in the solution of a smaller number of Gauss-Newton linear system (which is equal to the number of iterations).

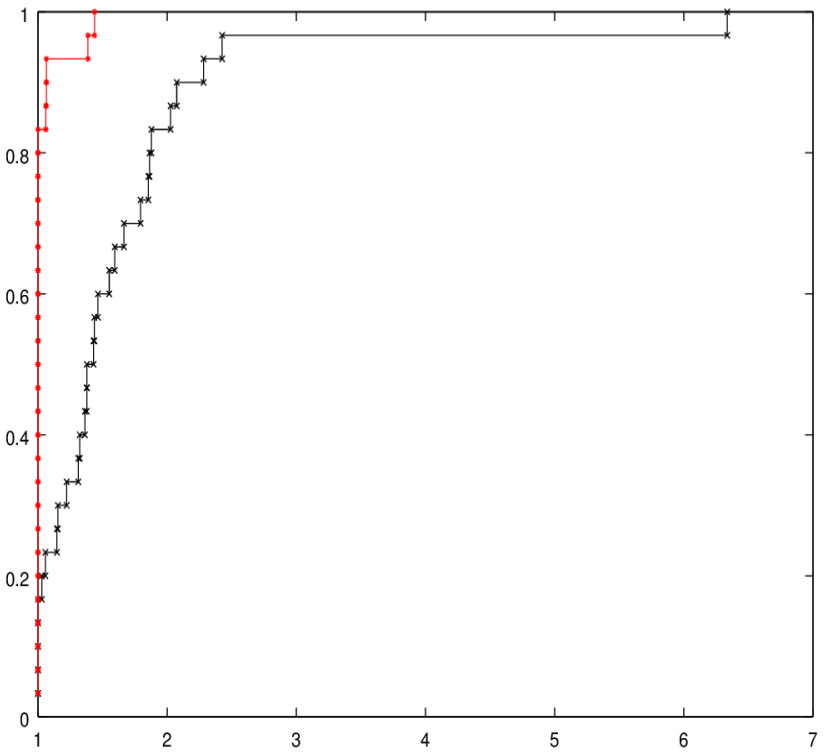



FIGURE 5.21: Performance Profile based on CPU Time for the set of 20 problems with medical images and resolutions of $128 \times 128$ and $256 \times 256$. The red line and black line represents HYBRID and GN respectively.

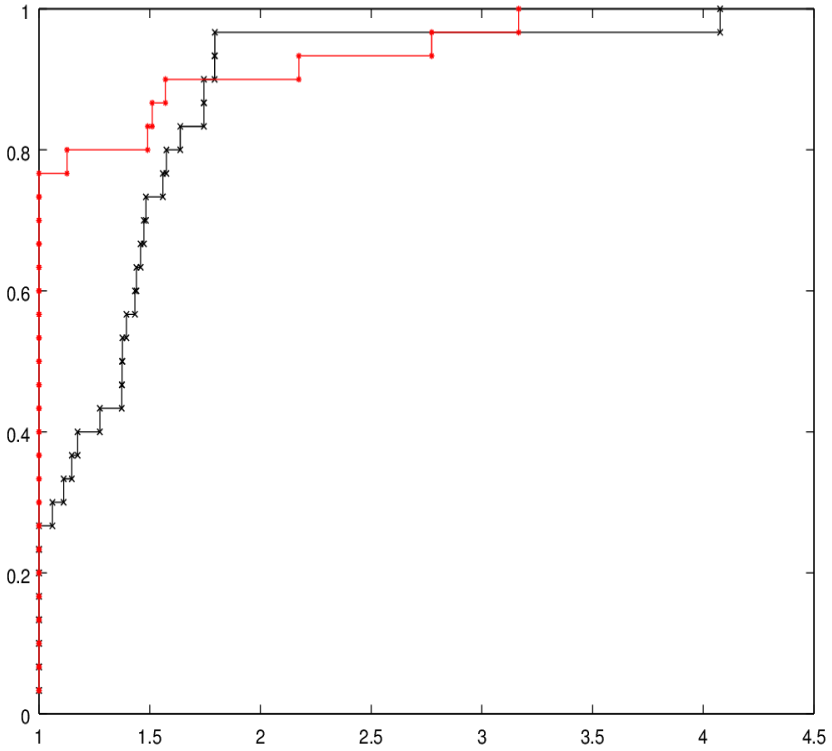



FIGURE 5.22: Performance Profile based on Number of Iterations for the set of 20 problems with medical images and resolutions of $128 \times 128$, $256 \times 256$ and $512 \times 512$. The red line and black line represents HYBRID and GN respectively.

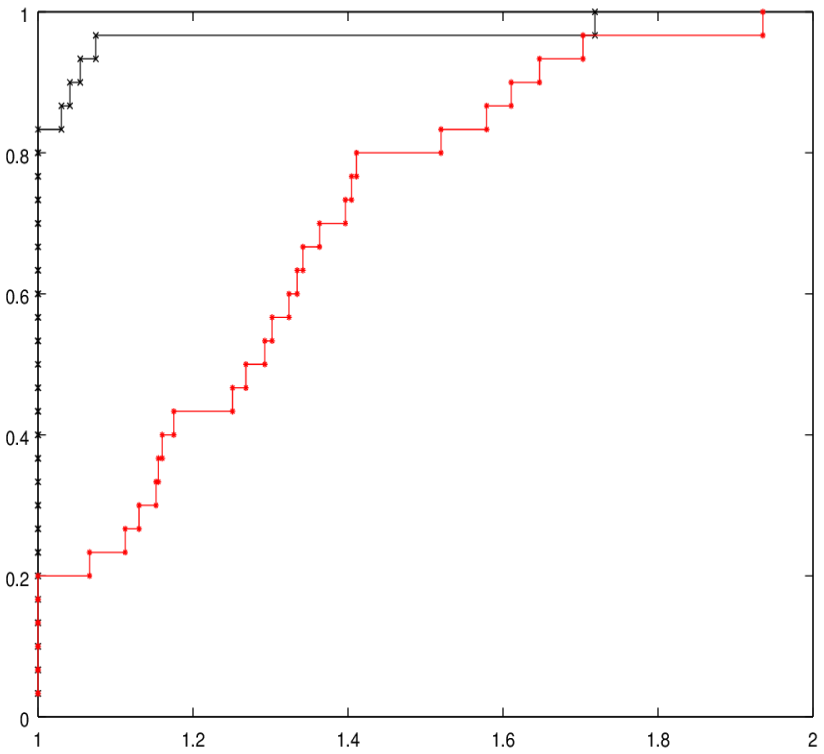
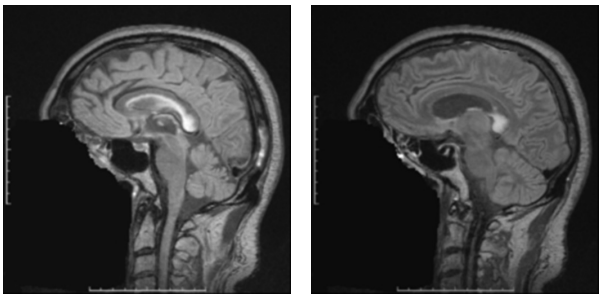
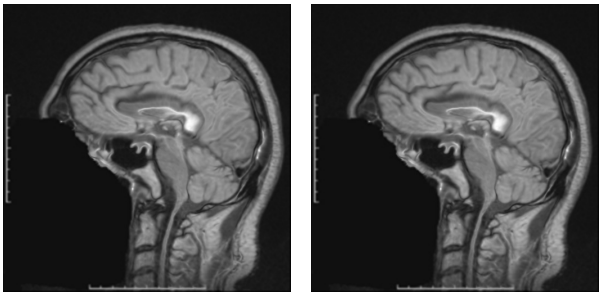



FIGURE 5.23: Performance Profile based on Number of Function Evaluations for the set of 20 problems with medical images and resolutions of $128 \times 128$, $256 \times 256$ and $512 \times 512$. The red line and black line represents HYBRID and GN respectively.

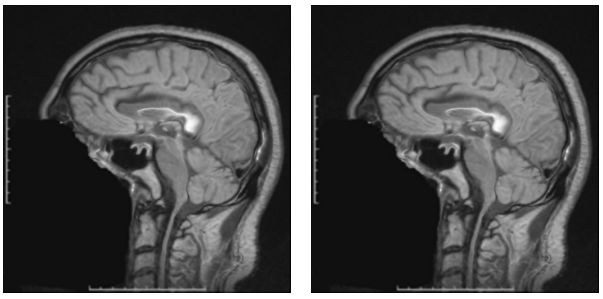
As an example, Figure 5.24 shows the registered images obtained by all codes applied to problem MRI2 with resolution $512 \times 512$.



(a) Template Image. (b) Reference Image.



(c) $T(\boldsymbol{y})$ obtained by code **GN** in 468.11 seconds. (d) $T(\boldsymbol{y})$ obtained by code **SIG** in 326.04 seconds.



(e) $T(\boldsymbol{y})$ obtained by code **TS** in 250.82 seconds. (f) $T(\boldsymbol{y})$ obtained by code **HYBRID** in 314.29 seconds.

FIGURE 5.24: Registered images for problem MRI2 with resolution $512 \times 512$.

We also tested the codes GN and HYBRID on four 3D problems from [92] (such as the Brain Problem illustrated on Figure 5.25 and 5.26). The results are in Table 5.11.

| | GN | | | HYBRID | | |
|---|---|---|---|---|---|---|
| Problem | Time | IT | FE | Time | IT | FE |
| 1. Brain | 1435 | 26 | 60 | 1249 | 25 | 97 |
| 2. Knee | 937 | 16 | 40 | 698 | 13 | 56 |
| 3. Phantom | 105 | 15 | 37 | 243 | 16 | 68 |
| 4. Mice | 62 | 28 | 65 | 48 | 17 | 68 |
| Total | 2539 | 85 | 202 | 2238 | 71 | 289 |

TABLE 5.11: Results for 3D problems.

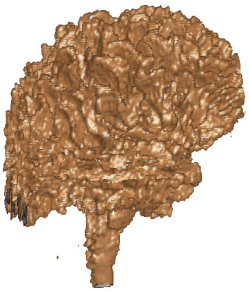
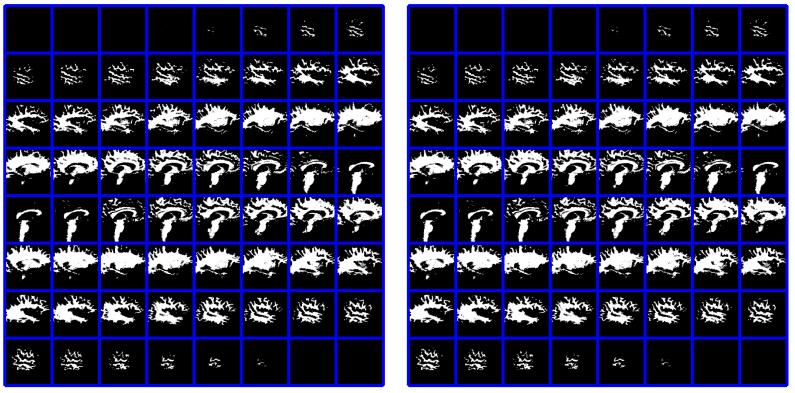


FIGURE 5.25: 3D brain problem.



(a) Template (b) Reference

FIGURE 5.26: Template and Reference for the 3D brain problem.

Once again, HYBRID outperformed GN. However, it seems that the gain of HYBRID over GN deteriorates when the problems become larger. One possible explanation is that for larger problems, the computational cost to compute function and gradient evaluation becomes comparable with the cost to solve the Gauss-Newton problem. In this case, the saving obtained by performing a smaller number of iterations may be not enough to compensate for the additional time used to evaluate the objective function and its gradients.

Finally, it is worth to mention that the methods proposed in this work can be applied to general smooth optimization problems. Notice that the critical component of the codes HYBRID and TS is the Algorithm 14 embedded in them. To evaluate the performance of this algorithm on a different class of problems, we apply it on a set of 10 test problems from [93] (without the multilevel step). The results on Table 5.12 show that the gain obtained with Algorithm 14 over the standard Gauss-Newton method is not restricted to image registration problems.

| | Gauss-Newton | | Algorithm 14 | |
|---|---|---|---|---|
| Problem | IT | FE | IT | FE |
| 1. Extended Rosenbrock $(100, 100)$ | 50 | 180 | 32 | 129 |
| 2. Extended Rosenbrock $(500, 500)$ | 49 | 177 | 27 | 110 |
| 3. Extended Powell Singular $(100, 100)$ | 23 | 47 | 26 | 79 |
| 4. Extended Powell Singular $(500, 500)$ | 24 | 49 | 26 | 79 |
| 5. Penalty I $(100, 101)$ | 99 | 667 | 25 | 154 |
| 6. Penalty II $(500, 501)$ | 101 | 670 | 27 | 154 |
| 7. Variably Dimensioned $(100, 100)$ | 48 | 93 | 28 | 81 |
| 8. Discrete Integral Equation $(100, 100)$ | 15 | 31 | 02 | 06 |
| 9. Broyden Tridiagonal $(100, 100)$ | 20 | 41 | 11 | 32 |
| 10. Broyden Banded $(100, 100)$ | 24 | 49 | 15 | 45 |
| Total | 453 | 2004 | 219 | 869 |

TABLE 5.12: Results for MGH problems.

## 5.4 Conclusion

In this chapter, we propose a two-step Gauss-Newton method for smooth unconstrained optimization and a modified coarse-to-fine multilevel scheme. Both methods rely on elementary subspace techniques, and they aim to solve image registration problems by the first-discretize-then-optimize approach. Numerical experiments are performed on a diverse set of 20 pairs of images (Reference, Template) considering four different resolutions. The results obtained correspond to more than 21 hours of numerical computation time. For registration problems with the resolution of $128 \times 128$, $256 \times 256$ and $512 \times 512$, a hybrid method of our two new subspace methods outperforms the standard multilevel Gauss-Newton method, reducing the total running time in 52.8% for problems with the resolution of $512 \times 512$. The advantage of the new methods over the Gauss-Newton scheme is even bigger when we consider the registration of medical images. For example, in our set of 10 problems from medical images with the resolution of $512 \times 512$, our hybrid method is faster than the multilevel Gauss-Newton method on nine problems, reducing the total running time in 56.2%. These results are very encouraging.

In Chapter 4, we propose a new regularizer based on the Beltrami concept to control the 2D transformation. However, since the Beltrami coefficient has no definition in 3D space, we cannot directly apply the notion of the Beltrami coefficient to 3D image registration. Hence, in the next chapter, we present two new formulations to generalize the Beltrami concept in 3D space and then define our new 3D registration models.

# Chapter 6

# An Efficient Iterative Algorithm for A Beltrami Based 3D Diffeomorphic Model

In this chapter, we first present two new formulations to measure the Beltrami concept in 3D space and then define our new registration models, recommending, in particular, the second model. We propose a converging Gauss-Newton iterative method to solve the resulting nonlinear optimization problems. Numerical experiments show that the new models can produce more accurate diffeomorphic transformations than competing state-of-the-art registration models.

## 6.1 Introduction

In Chapter 4, we have mentioned that in image registration, folding will appear when the deformation is large if we impose no constraint on the transformation and few models have built-in capabilities to impose such constraints. Controlling $\det(\nabla \boldsymbol{y}) > 0$ is an effective way but for some applications, it is very difficult for users to decide its upper bound and lower bound. Another effective way to avoid the folding is to control the Beltrami coefficient [77, 135]. The quasi-conformal theory shows that if the infinity norm of the Beltrami coefficient is smaller than 1, the corresponding mapping is diffeomorphic. Normally, the Beltrami coefficient is defined in the complex space and for 2D image registration, we can consider the transformation as a complex mapping and control its Beltrami coefficient to get a diffeomorphic transformation. However, since the Beltrami coefficient has no definition in 3D space, we cannot directly apply the notion of the Beltrami coefficient to 3D image registration.

In this chapter, we propose two new formulations to measure the norm of the Beltrami coefficient in 3D space motivated by our previous work [135]. Combining the classical

diffusion model, we propose new registration models that can cope with large deformation registration problems and generate diffeomorphic transformations. An effective numerical scheme is presented and numerical experimental results also, show that the new registration models can obtain good performances and accurate transformations and can be competitive with the other state-of-the-art registration models.

The rest of the chapter is organized as follows. In Section 6.2, we briefly review related works and then propose our new regularizers for 3D image registration. In Section 6.3, the new registration models are proposed and the details about the numerical implementation, including discretization and optimization method, are illustrated. Numerical experiment results are shown in Section 6.4, and finally a conclusion is summarized in Section 6.5.

## 6.2 Related Works and New Regularizers for 3D Image Registration

In this section, we first review related works. Then based on the notion of the Beltrami coefficient and our previous work [135], we propose our new regularizers for 3D image registration.

### 6.2.1 Related Works

There exist many 3D models, though not as many as 2D models, which can produce diffeomorphic transformations for image registration. Before proposing our new regularizers for 3D image registration, we briefly review one related model.

#### LLL Model

In 2016, Lee, Lam and Lui in [79] extended the notion of the standard conformality distortion $K(f)$ for a mapping in $\mathbb{R}^2$ to $\mathbb{R}^n (n \geq 3)$. Then they proposed a variational model involving $K(f)$ to deal with the landmark-matching problem in higher dimensional spaces. Before presenting the notion of the conformality distortion in $\mathbb{R}^n$ and Lee, Lam and Lui's model (LLL model), we first review the notion of the conformality distortion.

Any real-linear map from $\mathbb{C}$ to $\mathbb{C}$ has the complex form

$$f(z) = az + b\bar{z}, \tag{6.1}$$

with complex constants $a$ and $b$. For orientation preserving $f$, the determinant is $|a|^2 - |b|^2 > 0$ and the formula can be rewritten as

$$f(z) = a(z + \mu\bar{z}), \tag{6.2}$$

where the complex number$\mu = b/a$ is the Beltrami coefficient and $|\mu| < 1$. In this form, $f$ is the stretch map $S(z) = z + \mu\bar{z}$ postcomposed by a multiplication which is conformal

and consists of a rotation through the angle $\arg a$ and magnification by the factor $|a|$. The distortion caused by $f$ is expressed by $\mu$ and from it we can find angles of directions of maximal magnification and maximal shrinking: the angle of maximal magnification is $(\arg \mu)/2$ with magnifying factor $1 + |\mu|$ and the angle of maximal shrinking is the orthogonal angle $(\arg \mu - \pi)/2$ with shrinking factor $1 - |\mu|$. Then we can define by $K$ the dilatation:

$$K(f) = \frac{1 + |\mu|}{1 - |\mu|}. \tag{6.3}$$

Here, $K(f)$ expresses the ratio of the largest singular value of the Jacobian determinant of $f$ divided by the smallest singular value.

In the LLL work, the above definition of the conformality distortion $K(f)$ of a mapping $f$ is generalized to the $n$ dimensional space and is defined by

$$K(\boldsymbol{f}) := \begin{cases} \dfrac{1}{n}\left(\dfrac{\|\nabla \boldsymbol{f}\|_F^2}{\det(\nabla \boldsymbol{f})^{2/n}}\right) & \text{if } \det(\nabla \boldsymbol{f}) > 0, \\ +\infty & \text{otherwise} \end{cases} \tag{6.4}$$

where $\boldsymbol{f}(x_1, ..., x_n) = (y_1(x_1, ..., x_n), ..., y_n(x_1, ..., x_n))$ and $\nabla \boldsymbol{f}$ is a $n \times n$ Jacobian matrix. Combining the conformality distortion, the LLL model is defined by

$$\begin{aligned} &\min_{\boldsymbol{y}} \|K(\boldsymbol{y})\|_1 + \frac{\alpha}{2}\|\Delta \boldsymbol{y}\|_2^2 \\ &\text{s.t. } \boldsymbol{y}(p_i) = q_i, \quad 1 \le i \le m, \end{aligned} \tag{6.5}$$

where $p_i$ and $q_i$ are the given landmark points. Here, the first term controls the minimal conformality distortion and the second term keeps the smoothness of the mapping. To implement alternative minimization iterations in the numerical solution [79], an auxiliary (matrix) variable $\boldsymbol{s} = \nabla \boldsymbol{y}$ is introduced. In 3D case, the LLL model (6.5) takes the following equivalent form:

$$\begin{aligned} &\min_{\boldsymbol{y}} \frac{1}{3}\left(\frac{\|\nabla \boldsymbol{y}\|_F^2}{\det(\boldsymbol{s})^{2/3}}\right) + \frac{\alpha}{2}\|\Delta \boldsymbol{y}\|_2^2 \\ &\text{s.t. } \boldsymbol{s} = \nabla \boldsymbol{y}, \ \det(\boldsymbol{s}) > 0 \text{ and } \boldsymbol{y}(p_i) = q_i \ \ 1 \le i \le m, \end{aligned} \tag{6.6}$$

Then an alternating direction method with Lagrangian multipliers can be applied to solve (6.6). Note that this model is designed for landmark registration. Below, we consider adapting (6.6) to an intensity registration framework.

### 6.2.2 Motivation and New Regularizers for 3D Image Registration

In the LLL model, the conformality distortion is restricted. According to (6.3), controlling the norm of the Beltrami coefficient should achieve the same aim, and, in the 2D case, this idea has been confirmed in [135]. Hence, we want to extend the standard definition of the norm of the Beltrami coefficient to 3D space.

**Properties of 2D Regularizer $|\mu(f)|^2$**

Through investigating $|\mu(f)|^2$ in (4.15) , we can obtain the following lemma:

**Lemma 6.1.** *The 2D Beltrami regularizer* $|\mu(f)|^2$ (4.15) *possesses the following properties:*

*P1 If* $|\mu(f)|^2 = 0$*, then we have the singular values of* $\det(\nabla \boldsymbol{f})$ *are equal;*

*P2* $0 \leq |\mu(f)|^2 \leq \infty$*;*

*P3* $0 \leq |\mu(f)|^2 < 1 \Leftrightarrow \det(\nabla \boldsymbol{f}) > 0$*;*

*P4* $|\mu(f)|^2 = 1 \Leftrightarrow \det(\nabla \boldsymbol{f}) = 0$*;*

*P5* $1 < |\mu(f)|^2 \leq \infty \Leftrightarrow \det(\nabla \boldsymbol{f}) < 0$*;*

*P6 If the singular values of* $\det(\nabla \boldsymbol{f})$ *are equal, then when* $\det(\nabla \boldsymbol{f}) > 0$, $|\mu(f)|^2 = 0$ *and when* $\det(\nabla \boldsymbol{f}) < 0$, $|\mu(f)|^2$ *is* $\infty$.

*Proof.* For P1, if $|\mu(f)|^2 = 0$, according to (4.15), $\|\nabla \boldsymbol{f}\|_F^2 = 2\det(\nabla \boldsymbol{f})$. Hence, $\det(\nabla \boldsymbol{f})$ is non-negative. Set $\sigma_1$ and $\sigma_2$ as the singular values of $\det(\nabla \boldsymbol{f})$. Then we have $\|\nabla \boldsymbol{f}\|_F^2 = \sigma_1^2 + \sigma_2^2$ and $\det(\nabla \boldsymbol{f}) = \sigma_1\sigma_2$. So $(\sigma_1 - \sigma_2)^2 = 0$ and we have $\sigma_1 = \sigma_2$.

For P2, since the numerator and the denominator of (4.15) are both non-negative, we have $0 \leq |\mu(f)|^2 \leq \infty$.

P3-P5 follow from (4.15).

For P6, if $\sigma_1 = \sigma_2$, when $\det(\nabla \boldsymbol{f}) > 0$, we have $\|\nabla \boldsymbol{f}\|_F^2 - 2\det(\nabla \boldsymbol{f}) = (\sigma_1 - \sigma_2)^2$ and $\|\nabla \boldsymbol{f}\|_F^2 + 2\det(\nabla \boldsymbol{f}) = (\sigma_1 + \sigma_2)^2$, then $|\mu(f)|^2 = 0$. But when $\det(\nabla \boldsymbol{f}) < 0$, since $\det(\nabla \boldsymbol{f}) = -\sigma_1\sigma_2$, we have $\|\nabla \boldsymbol{f}\|_F^2 - 2\det(\nabla \boldsymbol{f}) = (\sigma_1 + \sigma_2)^2$ and $\|\nabla \boldsymbol{f}\|_F^2 + 2\det(\nabla \boldsymbol{f}) = (\sigma_1 - \sigma_2)^2$, then $|\mu(f)|^2 = \infty$. □

Since the Beltrami regularizer $|\mu(f)|^2$ gives good performances in 2D image registration [135], better than other 2D regularizers enforcing $\det(\nabla \boldsymbol{f}) > 0$, we next generalize it to 3D using a similar quantity. Then we propose our new regularizers for 3D image registration, motivated by similar properties and the desire to obtain an analogous regularizer better than competing ones in 3D.

**New Regularizers for 3D Image Registration**

Our starting point in generalizing to the 3D case is to construct the 3D quantity $\mathcal{N}_1(\boldsymbol{f})$ below that aims to satisfy some algebraic properties similar to P1-P6 of Lemma 6.1. Since enforcing $\det(\nabla \boldsymbol{f}) > 0$ implicitly is the ultimate objective, linking this quantity to $\det(\nabla \boldsymbol{f})$ is the key. Then after $\mathcal{N}_1$, we propose an improved variant $\mathcal{N}_2$ and a further quality $\mathcal{N}_3$ based on the LLL work.

Based on the above discussions, we first propose our new regularizer 1:

**Definition 6.2.** If the map $\boldsymbol{f}(x_1, x_2, x_3) = (y_1(x_1, x_2, x_3), y_2(x_1, x_2, x_3), y_3(x_1, x_2, x_3))$ is continuously differentiable, then we define

$$\mathcal{N}_1(\boldsymbol{f}) = \frac{\|\nabla \boldsymbol{f}\|_F^2 - 3(\det(\nabla \boldsymbol{f}))^{2/3}}{\|\nabla \boldsymbol{f}\|_F^2 + 3(\det(\nabla \boldsymbol{f}))^{2/3}} \tag{6.7}$$

as the new regularizer for a 3D map $\boldsymbol{f}$.

Then, we can have the following lemma:

**Lemma 6.3.** *Regularizer $\mathcal{N}_1$ from* (6.7) *possesses the following properties:*

*P1* $\mathcal{N}_1(\boldsymbol{f}) = 0 \Leftrightarrow$ *the singular values of* $\det(\nabla \boldsymbol{f})$ *are equal;*

*P2* $0 \leq \mathcal{N}_1(\boldsymbol{f}) \leq 1$*;*

*P3* $0 \leq \mathcal{N}_1(\boldsymbol{f}) < 1 \Leftrightarrow \det(\nabla \boldsymbol{f}) \neq 0$*;*

*P4* $\mathcal{N}_1(\boldsymbol{f}) = 1 \Leftrightarrow \det(\nabla \boldsymbol{f}) = 0$.

*Proof.* For P1, if $\mathcal{N}_1(\boldsymbol{f}) = 0$, according to (6.7), we have $\|\nabla \boldsymbol{f}\|_F^2 = 3(\det(\nabla \boldsymbol{f}))^{2/3}$. Set $\sigma_1, \sigma_2$ and $\sigma_3$ as the singular values of $\det(\nabla \boldsymbol{f})$. Since $\|\nabla \boldsymbol{f}\|_F^2 = \sum_{i=1}^3 \sigma_i^2$ and $(\det(\nabla \boldsymbol{f}))^2 = \Pi_{i=1}^3 \sigma_i^2$, we have $\sum_{i=1}^3 \sigma_i^2 = 3(\Pi_{i=1}^3 \sigma_i^2)^{1/3}$. Then due to the inequality of arithmetic and geometric means, we have $\sigma_1 = \sigma_2 = \sigma_3$. If the singular values of $\det(\nabla \boldsymbol{f})$ are equal, we have $\|\nabla \boldsymbol{f}\|_F^2 = 3(\det(\nabla \boldsymbol{f}))^{2/3}$ and $\mathcal{N}_1 = 0$.

For P2, according to the inequality of arithmetic and geometric means, the numerator is always non-negative, and the denominator is always positive. So we have $0 \leq \mathcal{N}_1(\boldsymbol{f}) \leq 1$.

P3-P4 can be verified using the above formula (6.7). □

However $\mathcal{N}_1(\boldsymbol{f})$ is not entirely satisfactory, because P5 and P6 (of 2D) from Lemma 6.1 do not hold. Since the term $3(\det(\nabla \boldsymbol{f}))^{2/3}$ is always positive, for P5, when $\det \nabla \boldsymbol{f} < 0$, we have $0 \leq \mathcal{N}_1(\boldsymbol{f}) < 1$ and for P6, if the singular values of $\det(\nabla \boldsymbol{f})$ are equal, whether $\det(\nabla \boldsymbol{f})$ is positive or negative, $\mathcal{N}_1(\boldsymbol{f})$ is always 0. These two missing properties suggest that we need to make further modifications. Below, we propose our new regularizer 2 to overcome this issue and to possess all the six properties.

**Definition 6.4.** If the map $\boldsymbol{f}(x_1, x_2, x_3) = (y_1(x_1, x_2, x_3), y_2(x_1, x_2, x_3), y_3(x_1, x_2, x_3))$ is continuous differentiable, then we define

$$\mathcal{N}_2(\boldsymbol{f}) = \frac{\|\nabla \boldsymbol{f}\|_F - \sqrt{3}(\det(\nabla \boldsymbol{f}))^{1/3}}{\|\nabla \boldsymbol{f}\|_F + \sqrt{3}(\det(\nabla \boldsymbol{f}))^{1/3}} \tag{6.8}$$

as another new regularizer for a 3D map $\boldsymbol{f}$.

Then we have the following lemma for (6.8), which is similar to Lemma 6.1:

**Lemma 6.5.** *Regularizer* $\mathcal{N}_2$ *from* (6.8) *possesses the following properties:*

*P1* *If* $\mathcal{N}_2(\boldsymbol{f}) = 0$, *then we have the singular values of* $\det(\nabla \boldsymbol{f})$ *are equal;*

*P2* $0 \leq \mathcal{N}_2(\boldsymbol{f}) \leq \infty$;

*P3* $0 \leq \mathcal{N}_2(\boldsymbol{f}) < 1 \Leftrightarrow \det(\nabla \boldsymbol{f}) > 0$;

*P4* $\mathcal{N}_2(\boldsymbol{f}) = 1 \Leftrightarrow \det(\nabla \boldsymbol{f}) = 0$;

*P5* $1 < \mathcal{N}_2(\boldsymbol{f}) \leq \infty \Leftrightarrow \det(\nabla \boldsymbol{f}) < 0$;

*P6* *If the singular values of* $\det(\nabla \boldsymbol{f})$ *are equal, then when* $\det(\nabla \boldsymbol{f}) > 0$, $\mathcal{N}_2(\boldsymbol{f}) = 0$ *and when* $\det(\nabla \boldsymbol{f}) < 0$, $\mathcal{N}_2(\boldsymbol{f})$ *is* $\infty$;

*Proof.* Since $\mathcal{N}_2(\boldsymbol{f})$ involves the term $(\det(\nabla \boldsymbol{f}))^{1/3}$ which can be positive when $\det(\nabla \boldsymbol{f})$ is positive or negative when $\det(\nabla \boldsymbol{f})$ is negative, we can obtain these 6 properties just following the proofs of Lemma 6.1 and Lemma 6.3. □

Until now, aiming to enforce the constraint $\det(\nabla \boldsymbol{f}) > 0$, we have proposed two new regularizers $\mathcal{N}_1(\boldsymbol{f})$ and $\mathcal{N}_2(\boldsymbol{f})$ from generalizing 2D results.

Finally, making use of (6.4) from the LLL work, we shall consider the following regularizer for a map $\boldsymbol{f}$ also:

$$\mathcal{N}_3(\boldsymbol{f}) = \frac{1}{3}\left(\frac{\|\nabla \boldsymbol{f}\|_F^2}{\det(\nabla \boldsymbol{f})^{2/3}}\right). \tag{6.9}$$

Similarly, we have the following lemma about the properties of $\mathcal{N}_3(\boldsymbol{f})$.

**Lemma 6.6.** *Regularizer* $\mathcal{N}_3$ *from* (6.9) *possesses the following properties:*

*P1* $\mathcal{N}_3(\boldsymbol{f}) = 1 \Leftrightarrow$ *the singular values of* $\det(\nabla \boldsymbol{f})$ *are equal;*

*P2* $1 \leq \mathcal{N}_3(\boldsymbol{f}) \leq \infty$;

*P3* $\mathcal{N}_3(\boldsymbol{f}) = \infty \Leftrightarrow \det(\nabla \boldsymbol{f}) = 0$.

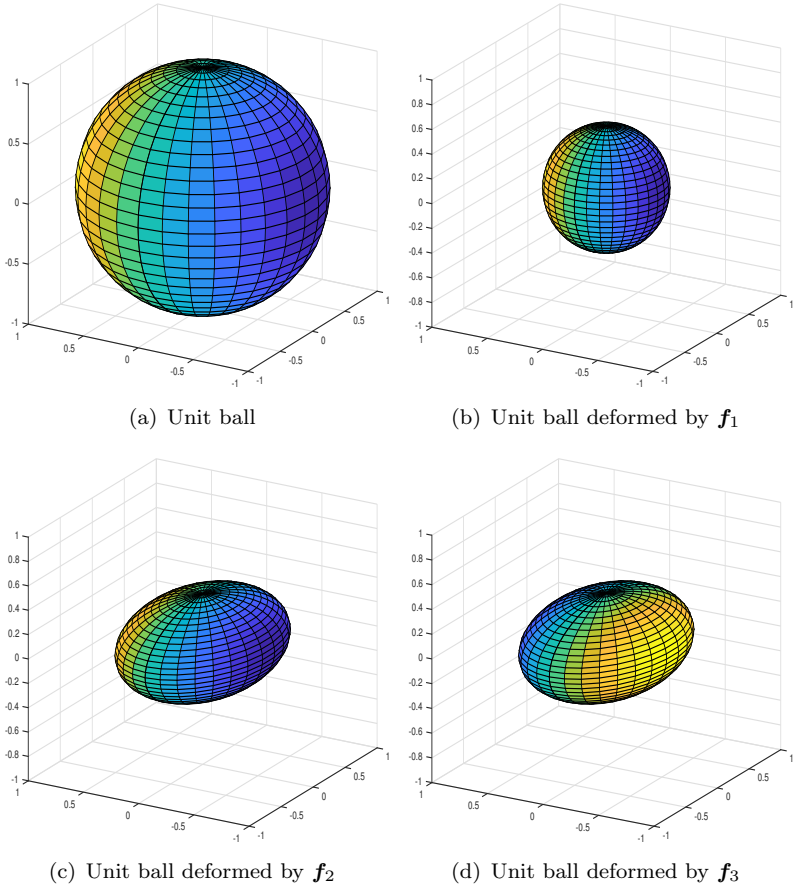
*Proof.* Just follow the proofs of Lemma 6.3. □

Now, we briefly discuss the geometric properties of our proposed regularizers. By the definitions, the roles of $\mathcal{N}_1$ and $\mathcal{N}_2$ in 3D are similar to $|\mu|$ in 2D. Then according to (6.3), we can define their corresponding dilatations: $K_1 = \frac{1+\mathcal{N}_1}{1-\mathcal{N}_1}$ and $K_2 = \frac{1+\mathcal{N}_2}{1-\mathcal{N}_2}$. Here, we can find that $K_1$ and $\mathcal{N}_3$ are identical. Although $\mathcal{N}_3$ is derived from (6.4), $K$ is equal to $\mathcal{N}_3$ only when $\det(\nabla \boldsymbol{f}) > 0$, which means that $\mathcal{N}_1$ has a link with $K$ only when $\det(\nabla \boldsymbol{f}) > 0$. Furthermore, we consider the following three simple example functions: $\boldsymbol{f}_1 = (0.5x_1, 0.5x_2, 0.5x_3)$, $\boldsymbol{f}_2 = (0.9x_1, 0.6x_2, 0.4x_3)$ and $\boldsymbol{f}_3 = (0.9x_1, -0.6x_2, 0.4x_3)$ (Figure 6.1). In order to emphasize the link between 2D and 3D, we project the above three functions into $x_1, x_2-$plane and define $\bar{\boldsymbol{f}}_1 = (0.5x_1, 0.5x_2)$, $\bar{\boldsymbol{f}}_2 = (0.9x_1, 0.6x_2)$ and

$\bar{\boldsymbol{f}}_3 = (0.9x_1, -0.6x_2)$. Just according to the above mentioned definitions, we have the following results:

- For $\bar{\boldsymbol{f}}_1$, $|\mu| = 0$ and $K = 1$. Then for $\boldsymbol{f}_1$, we have $\mathcal{N}_1 = 0$ and $K_1 = 1$, $\mathcal{N}_2 = 0$ and $K_2 = 1$, and $K = 1$.
- For $\bar{\boldsymbol{f}}_2$, $|\mu| = 0.04$ and $K = 1.0833$. Then for $\boldsymbol{f}_2$, we have $\mathcal{N}_1 = 0.1037$ and $K_1 = 1.2315$, $\mathcal{N}_2 = 0.0520$ and $K_2 = 1.1097$, and $K = 1.2315$.
- For $\bar{\boldsymbol{f}}_3$, $|\mu| = 25$ and $K = -1.0833$. Then for $\boldsymbol{f}_2$, we have $\mathcal{N}_1 = 0.1037$ and $K_1 = 1.2315$, $\mathcal{N}_2 = 19.2280$ and $K_2 = -1.1097$, and $K = \infty$.

Here, we can note that when $\det(\nabla \boldsymbol{f}) > 0$, $\mathcal{N}_1$ and $\mathcal{N}_2$ in 3D are consistent with $|\mu|$ in 2D but when $\det(\nabla \boldsymbol{f}) < 0$, only $\mathcal{N}_2$ in 3D is consistent with $|\mu|$ in 2D, which actually has been asserted by Lemma 6.1, 6.3 and 6.5. And we will also see that $\mathcal{N}_2$ is a better option than $\mathcal{N}_1$.



(a) Unit ball (b) Unit ball deformed by $\boldsymbol{f}_1$

(c) Unit ball deformed by $\boldsymbol{f}_2$ (d) Unit ball deformed by $\boldsymbol{f}_3$

FIGURE 6.1: Illustration of the distortion and dilatation in 3D space. Here, we can note that $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are orientation-preserving but $\boldsymbol{f}_3$ is not orientation-preserving (from the color).

In the next section, we will build our new models motivated by these three new regularizers and present the details about its numerical algorithm.

## 6.3 New Models for 3D Image Registration and Their Numerical Algorithms

In this section, we first formulate our new models to deal with the 3D image registration problem. Then we give the details about the numerical algorithm, such as discretization, optimization method, stopping criteria and multilevel strategy.

### 6.3.1 New Models for 3D Image Registration

Here, we propose the following three new variational models for 3D image registration:

**New Model 1 (NM1)**

$$\min_{\boldsymbol{u}} \frac{1}{2}\int_{\Omega}(T(\boldsymbol{y})-R(\boldsymbol{x}))^2\mathrm{d}\boldsymbol{x}+\frac{\alpha}{2}\int_{\Omega}\sum_{l=1}^{3}|\nabla u_l|^2\mathrm{d}\boldsymbol{x}+\beta\int_{\Omega}\varphi(\mathcal{N}_1(\boldsymbol{y}))\mathrm{d}\boldsymbol{x}, \tag{6.10}$$

where $\boldsymbol{y}=\boldsymbol{y}(\boldsymbol{x})=\boldsymbol{x}+\boldsymbol{u}$, and

**New Model 2 (NM2)**

$$\min_{\boldsymbol{u}} \frac{1}{2}\int_{\Omega}(T(\boldsymbol{y})-R(\boldsymbol{x}))^2\mathrm{d}\boldsymbol{x}+\frac{\alpha}{2}\int_{\Omega}\sum_{l=1}^{3}|\nabla u_l|^2\mathrm{d}\boldsymbol{x}+\beta\int_{\Omega}\varphi(\mathcal{N}_2(\boldsymbol{y}))\mathrm{d}\boldsymbol{x}, \tag{6.11}$$

where $\varphi(v)=v^2/((v-1)^2+\epsilon)$ is chosen and $\epsilon$ is a small number (such as $10^{-8}$) to promote $\mathcal{N}_1<1$ and $\mathcal{N}_2<1$ i.e. $\det\nabla\boldsymbol{y}>0$. Compared with $\phi(v)=v^2/(v-1)^2$ used in Chapter 4, here, $\varphi$ is infinitely continuously differentiable and bounded by $1/\epsilon$ which can help to build the existence of the solution.

**New Model 3 (NM3)**

$$\min_{\boldsymbol{u}} \frac{1}{2}\int_{\Omega}(T(\boldsymbol{y})-R(\boldsymbol{x}))^2\mathrm{d}\boldsymbol{x}+\frac{\alpha}{2}\int_{\Omega}\sum_{l=1}^{3}|\nabla u_l|^2\mathrm{d}\boldsymbol{x}+\beta\int_{\Omega}\varphi(\mathcal{N}_3(\boldsymbol{y}))\mathrm{d}\boldsymbol{x}, \tag{6.12}$$

where $\varphi(v)=v^2$ because we promote $\mathcal{N}_3<\infty$. Here, NM3 can be considered as a reasonable modified LLL model under our framework, which is mainly used to make a comparison in the later test since the $\mathcal{N}_3$ comes from LLL model.

We can notice that these three new models only differ in the third term of how to control the Jacobian determinant indirectly by restricting the norm of the Beltrami coefficient. The key message is that the resulting transformation $\boldsymbol{y}$ will be diffeomorphic under the suitable boundary condition, such as Dirichlet boundary condition.

### 6.3.2 Mathematical Analysis of The Proposed Models

Most registration models are nonconvex with respect to $\boldsymbol{u}$ and consequently, if solutions exist, there are local minimizers or solutions are generally not unique. Below we prove the existence of a solution for the problem (6.10) and the result can be easily extended to (6.11). Before stating the main result, we first consider the concept of the Carathéodory function.

**Definition 6.7.** Let $\Omega \subset \mathbb{R}^3$ be an open set and let $f : \Omega \times \mathbb{R}^n \times \mathbb{R}^{3\times n} \to [0, +\infty)$. Then $f$ is a Carathéodory function if:

1. $f(\boldsymbol{x}, \cdot, \cdot)$ is continuous for almost every $\boldsymbol{x} \in \Omega$.
2. $f(\boldsymbol{x}, \boldsymbol{u}, \psi)$ is measurable in $\boldsymbol{x}$ for every $(\boldsymbol{u}, \psi) \in \mathbb{R}^n \times \mathbb{R}^{3\times n}$.

**Lemma 6.8** ([137])**.** *Let $\Omega \subset \mathbb{R}^3$ be an open set and $f : \Omega \times \mathbb{R}^n \times \mathbb{R}^{3\times n} \to [0, +\infty)$ satisfies the following assumptions:*

*(i) $f$ is a Carathéodory function.*

*(ii) $f(\boldsymbol{x}, \boldsymbol{u}, \psi)$ is quasi-convex with respect to $\psi$.*

*(iii) $0 \leq f(\boldsymbol{x}, \boldsymbol{u}, \psi) \leq a(\boldsymbol{x}) + C(|\boldsymbol{u}|^2 + |\psi|^2)$ where $a(\boldsymbol{x}) \in L^1(\Omega)$, $C > 0$.*

*Then $\mathcal{J}(\boldsymbol{u})$ is weak lower semi-continuous (denoted by wlsc) in $W^{1,2}(\Omega)$.*

To analyze the proposed model (6.10), we first consider the solution space where $\mathcal{W} = \{\boldsymbol{u} \in W^{1,2}(\Omega) : |\int_\Omega \boldsymbol{u}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}| \leq |\Omega|\,(C_1 + \mathrm{diam}(\Omega))\}$. Then, it is convenient to rewrite the energy $\mathcal{J}(\cdot)$ by merging all terms under one integral in the following form:

$$\mathcal{J}(\boldsymbol{u}) = \int_\Omega f(\boldsymbol{x}, \boldsymbol{u}, \nabla \boldsymbol{u})\mathrm{d}\boldsymbol{x}, \tag{6.13}$$

where $f(\boldsymbol{x}, \boldsymbol{u}, \psi) = \dfrac{\alpha}{2}|\psi|^2 + (T(\boldsymbol{x} + \boldsymbol{u}) - R)^2 + \beta\varphi(\mathcal{N}_1(\boldsymbol{x} + \boldsymbol{u}))$.

To apply Lemma 6.8, we assume that the $T$ and $R$ are continuous and bounded by a constant $c > 0$. Then, we have the following result:

**Lemma 6.9.** *The energy functional $\mathcal{J}(\cdot)$ is wlsc in $\mathcal{W}$.*

*Proof.* We verify that the functional $f(\cdot)$ fulfils the assumptions in Lemma 6.8:

i) Since the $T$ and $R$ are continuous and $\boldsymbol{u} \in \mathcal{W}$, the function $f(\cdot)$ is a Carathéodory function.

ii) It is easy to check that $f(\boldsymbol{x}, \boldsymbol{u}, \psi)$ are convex with respect to $\psi$, clearly implying that it is quasi-convex.

iii) Since the $T$ and $R$ are bounded by $c$, we have:

$$\begin{aligned} f(\boldsymbol{x},\boldsymbol{u},\psi) &= \frac{\alpha}{2}|\psi|^2 + (T(\boldsymbol{x}+\boldsymbol{u}) - R)^2 + \beta\varphi(\mathcal{N}_1(\boldsymbol{x}+\boldsymbol{u})) \\ &\leq \frac{\alpha}{2}|\psi|^2 + 4c^2 + \frac{\beta}{\epsilon} \\ &\leq \frac{\alpha}{2}(|\boldsymbol{u}|^2 + |\psi|^2) + 4c^2 + \frac{\beta}{\epsilon}. \end{aligned} \tag{6.14}$$

Then, the function $f(\cdot)$ fulfils the condition (iii) of Lemma 6.8 with $a(\boldsymbol{x}) \equiv 4c^2 + \frac{\beta}{\epsilon}$ which implies that the energy $\mathcal{J}(\cdot)$, is *wlsc* in $\mathcal{W}$. □

We are now ready to prove the existence of a solution for the minimization model (6.10). Based on Lemma 6.8 and Lemma 6.9, we have the following result:

**Theorem 6.10.** *The minimization problem* (6.10) *admits at least one solution in the space* $\mathcal{W}$.

*Proof.* Here, we just follow the steps in Section 2.2.2. Since $\mathcal{J}(\boldsymbol{u})$ has a lower bound 0, there exist a minimizing sequence $(\boldsymbol{u}_n)_{n\in\mathbb{N}} \subset \mathcal{W}$ of $\mathcal{J}(\cdot)$ , i.e.,

$$\lim_{n\to\infty} \mathcal{J}(\boldsymbol{u}_n) = \inf_{\boldsymbol{u}\in\mathcal{W}} \mathcal{J}(\boldsymbol{u}).$$

Using the generalized Poincaré inequality, there exist two constants $C, K \in \mathbb{R}$ such that

$$\mathcal{J}(\boldsymbol{u}) \geq C\|\boldsymbol{u}\|^2_{W^{1,2}} + K. \tag{6.15}$$

The inequality (6.15) guarantees that the sequence $(\boldsymbol{u}_n)_{n\in\mathbb{N}}$ is uniformly bounded in $\mathcal{W}$. Thus, there exists a subsequence, still denoted $(\boldsymbol{u}_n)_{n\in\mathbb{N}}$, such that $\lim_{n\to\infty} \boldsymbol{u}_n = \boldsymbol{u}$ weakly in $\mathcal{W}$. Using the weak lower semi-continuity of $\mathcal{J}(\cdot)$, we obtain that the limit $\boldsymbol{u}$ is a minimizer of $\mathcal{J}(\cdot)$. □

### 6.3.3 The Numerical Algorithm

Here, we consider using first-discretize-then-optimize method to solve our proposed models (6.10), (6.11) and (6.12). Firstly, we choose a suitable discrete scheme to discretize the variational models (6.10), (6.11) and (6.12) to derive finite-dimensional optimization problems. Then we choose optimization methods to solve the resulting unconstrained optimization problems.

#### Discretization

Here, we extend the 2D case in Section 4.4.1 to the 3D case. We discretize our proposed models (6.10), (6.11) and (6.12) on the spatial domain $\Omega = [0,\omega_1] \times [0,\omega_2] \times [0,\omega_3]$. In the implementation, we employ the nodal grid (Figure 6.2) and define a spatial partition $\Omega_h^n = \{\boldsymbol{x}^{i,j,k} \in \Omega | \boldsymbol{x}^{i,j,k} = (x_1^i, x_2^j, x_3^k) = (ih_1, jh_2, kh_3), 0 \leq i \leq n_1, 0 \leq j \leq n_2, 0 \leq k \leq$

$n_3\}$, where $h_l = \frac{\omega_l}{n_l}, 1 \leq l \leq 3$ and the discrete domain consists of $n_1 n_2 n_3$ cells of size $h_1 \times h_2 \times h_3$. We discretize the displacement field $\boldsymbol{u}$ on the nodal grid, namely $\boldsymbol{u}^{i,j,k} = (u_1^{i,j,k}, u_2^{i,j,k}, u_3^{i,j,k}) = (u_1(x_1^i, x_2^j, x_3^k), u_2(x_1^i, x_2^j, x_3^k), u_3(x_1^i, x_2^j, x_3^k))$. The transformation field $\boldsymbol{y}$ is also discretized on the nodal grid as the same line. In order to simplify the presentation, we denote $h = h_1 h_2 h_3$, $N = n_1 n_2 n_3$ and $N_1 = (n_1 + 1)(n_2 + 1)(n_3 + 1)$ and according to the lexicographical ordering, we reshape

$$X = (x_1^0, ..., x_1^{n_1}, x_2^0, ..., x_2^{n_2}, x_3^0, ..., x_3^{n_3})^T \in \mathbb{R}^{3N_1 \times 1},$$

$$U = (u_1^{0,0,0}, ..., u_1^{n_1,n_2,n_3}, u_2^{0,0,0}, ..., u_2^{n_1,n_2,n_3}, u_3^{0,0,0}, ..., u_3^{n_1,n_2,n_3})^T \in \mathbb{R}^{3N_1 \times 1},$$

and

$$Y = (y_1^{0,0,0}, ..., y_1^{n_1,n_2,n_3}, y_2^{0,0,0}, ..., y_2^{n_1,n_2,n_3}, y_3^{0,0,0}, ..., y_3^{n_1,n_2,n_3})^T \in \mathbb{R}^{3N_1 \times 1}.$$

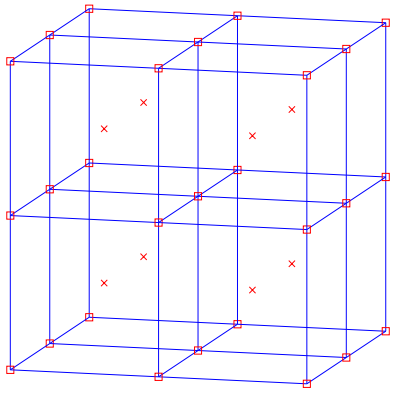


FIGURE 6.2: Partition of the domain $\Omega$. Nodal grid $\square$ and cell-centered grid $\times$.

**Discretization of SSD in** (6.10)**,** (6.11) **and** (6.12)

Just following Section 4.4.1, we directly have the following approximation for SSD:

$$\mathcal{D}(T(\boldsymbol{x} + \boldsymbol{u}), R) \approx \frac{h}{2}(\vec{T}(PU + PX) - \vec{R})^T(\vec{T}(PU + PX) - \vec{R}). \tag{6.16}$$

where $P \in \mathbb{R}^{3N \times 3N_1}$ is an averaging matrix from the nodal grid to the cell-centered grid.

**Discretization of The Diffusion Regularizer in** (6.10)**,** (6.11) **and** (6.12)

For the diffusion regularizer in (6.10), (6.11) and (6.12),

$$\mathcal{R}^{\text{Diff}}(\boldsymbol{u}) := \frac{\alpha}{2} \int_\Omega \sum_{l=1}^{3} |\nabla u_l|^2 \mathrm{d}\boldsymbol{x}, \tag{6.17}$$

based on the forward difference and mid-point rule, we have the following approximation:

$$\mathcal{R}^{\text{Diff}}(\boldsymbol{u}) \approx \frac{\alpha h}{2} \sum_{l=1}^{3} \sum_{i=0}^{n_1-1} \sum_{j=0}^{n_2-1} \sum_{k=0}^{n_3-1} \left( \frac{u_l^{i+1,j,k} - u_l^{i,j,k}}{h_1} \right)^2 + \left( \frac{u_l^{i,j+1,k} - u_l^{i,j,k}}{h_2} \right)^2 + \left( \frac{u_l^{i,j,k+1} - u_l^{i,j,k}}{h_3} \right)^2. \tag{6.18}$$

Then (6.18) can be rewritten into the following formulation:

$$\mathcal{R}^{\text{Diff}}(\boldsymbol{u}) \approx \frac{\alpha h}{2} U^T A^T A U, \tag{6.19}$$

where $A$ is shown in Appendix 6.6.1.

**Discretization of New Regularizers in** (6.10)**,** (6.11) **and** (6.12)

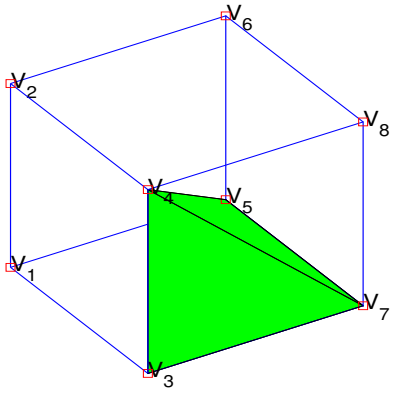



FIGURE 6.3: Partition of a voxel. $V_1, ..., V_8$ are vertices.

Here, our new regularizer $\mathcal{N}_1(\boldsymbol{y})$ only involves first order derivatives. Hence, we divide each voxel into 6 tetrahedrons $(V_3V_7V_4V_5, V_3V_1V_4V_5, V_4V_1V_2V_5, V_7V_4V_5V_8, V_4V_5V_8V_6, V_4V_2V_5V_6)$ and in each tetrahedron, we use three linear interpolation functions to approximate $y_1$, $y_2$ and $y_3$ (Figure 6.3).

According to this partition, we can get

$$\mathcal{R}_l(\boldsymbol{y}) = \beta \int_\Omega \varphi(\mathcal{N}_l(\boldsymbol{y}))\mathrm{d}\boldsymbol{x} = \beta \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{k=1}^{n_3} \sum_{m=1}^{6} \int_{\Omega_{i,j,k,m}} \varphi(\mathcal{N}_l(\boldsymbol{y}))\mathrm{d}\boldsymbol{x},$$

where $l \in \{1, 2, 3\}$ and $\Omega_{i,j,k,m}$ represents a tetrahedron.

Set $\boldsymbol{L}^{i,j,k,m}(\boldsymbol{x}) = (L_1^{i,j,k,m}(\boldsymbol{x}), L_2^{i,j,k,m}(\boldsymbol{x}), L_3^{i,j,k,m}(\boldsymbol{x})) = (a_1^{i,j,k,m}x_1 + a_2^{i,j,k,m}x_2 + a_3^{i,j,k,m}x_3 + b_1^{i,j,k,m}, a_4^{i,j,k,m}x_1 + a_5^{i,j,k,m}x_2 + a_6^{i,j,k,m}x_3 + b_2^{i,j,k,m}, a_7^{i,j,k,m}x_1 + a_8^{i,j,k,m}x_2 + a_9^{i,j,k,m}x_3 + b_3^{i,j,k,m})$, which is the linear interpolation for $\boldsymbol{y}$ in the $\Omega_{i,j,k,m}$. Note that $\partial_{x_1} L_1^{i,j,k,m} = a_1^{i,j,k,m}, \partial_{x_2} L_1^{i,j,k,m} = a_2^{i,j,k,m}, \partial_{x_3} L_1^{i,j,k,m} = a_3^{i,j,k,m}, \partial_{x_1} L_2^{i,j,k,m} = a_4^{i,j,k,m}, \partial_{x_2} L_2^{i,j,k,m} = a_5^{i,j,k,m}, \partial_{x_3} L_2^{i,j,k,m} = a_6^{i,j,k,m}, \partial_{x_1} L_3^{i,j,k,m} = a_7^{i,j,k,m}, \partial_{x_2} L_3^{i,j,k,m} = a_8^{i,j,k,m}, \partial_{x_3} L_3^{i,j,k,m} = a_9^{i,j,k,m}$.

Hence, in each tetrahedron $\Omega_{i,j,k,m}$, we have $|\nabla \boldsymbol{y}|_F^2 \approx \sum_{l=1}^{9}(a_l^{i,j,k,m})^2$ and $\det(\nabla \boldsymbol{y}) \approx a_1^{i,j,k,m}a_5^{i,j,k,m}a_9^{i,j,k,m}+a_2^{i,j,k,m}a_6^{i,j,k,m}a_7^{i,j,k,m}+a_4^{i,j,k,m}a_8^{i,j,k,m}a_3^{i,j,k,m}-a_2^{i,j,k,m}a_4^{i,j,k,m}a_9^{i,j,k,m}-a_1^{i,j,k,m}a_6^{i,j,k,m}a_8^{i,j,k,m}-a_3^{i,j,k,m}a_5^{i,j,k,m}a_7^{i,j,k,m}$.

Here, we construct $B_l, 1 \le l \le 9$:

$$\begin{aligned} &B_1 = [M_1, 0, 0], \quad B_4 = [0, M_1, 0], \quad B_7 = [0, 0, M_1], \\ &B_2 = [M_2, 0, 0], \quad B_5 = [0, M_2, 0], \quad B_8 = [0, 0, M_2], \\ &B_3 = [M_3, 0, 0], \quad B_6 = [0, M_3, 0], \quad B_9 = [0, 0, M_3], \end{aligned} \tag{6.20}$$

where $M_1$, $M_2$ and $M_3$ are the discrete operators of $\partial_{x_1}$, $\partial_{x_2}$ and $\partial_{x_3}$ respectively and how to construct them is shown in Appendix 6.6.2.

Then we denote by $B_lY\ (a_l^{1,1,1,1}, ..., a_l^{n_1,n_2,n_3,6})^T \in \mathbb{R}^{6N\times 1}$ and set

$$\begin{aligned} \vec{\boldsymbol{q}}^1(Y) &= \sum_{l=1}^{9} B_lY \odot B_lY, \\ \vec{\boldsymbol{q}}^2(Y) &= B_1Y \odot B_5Y \odot B_9Y + B_2Y \odot B_6Y \odot B_7Y + B_4Y \odot B_8Y \odot B_3Y \\ &\quad - B_2Y \odot B_4Y \odot B_9Y - B_1Y \odot B_6Y \odot B_8Y - B_3Y \odot B_5Y \odot B_7Y, \\ \vec{\boldsymbol{r}}_1^1(Y) &= \vec{\boldsymbol{q}}^1(Y) - 3(\vec{\boldsymbol{q}}^2(Y))^{2/3}, \\ \vec{\boldsymbol{r}}_1^2(Y) &= 1./(\vec{\boldsymbol{q}}^1(Y) + 3(\vec{\boldsymbol{q}}^2(Y))^{2/3}), \\ \vec{\boldsymbol{r}}_1(Y) &= \vec{\boldsymbol{r}}_1^1(Y) \odot \vec{\boldsymbol{r}}_1^2(Y), \\ \vec{\boldsymbol{r}}_2^1(Y) &= (\vec{\boldsymbol{q}}^1(Y))^{1/2} - \sqrt{3}(\vec{\boldsymbol{q}}^2(Y))^{1/3}, \\ \vec{\boldsymbol{r}}_2^2(Y) &= 1./((\vec{\boldsymbol{q}}^1(Y))^{1/2} + \sqrt{3}(\vec{\boldsymbol{q}}^2(Y))^{1/3}), \\ \vec{\boldsymbol{r}}_2(Y) &= \vec{\boldsymbol{r}}_2^1(Y) \odot \vec{\boldsymbol{r}}_2^2(Y), \\ \vec{\boldsymbol{r}}_3(Y) &= \vec{\boldsymbol{q}}^1(Y)./(3(\vec{\boldsymbol{q}}^2(Y))^{2/3}), \end{aligned} \tag{6.21}$$

where $\odot$ denotes the Hadamard product of two matrices and ./ denotes the component-wise division. Then we have the following approximation:

$$\mathcal{R}_l(\boldsymbol{y}) \approx \frac{\beta h}{6}\boldsymbol{\varphi}(\vec{\boldsymbol{r}}_l(Y))e^T, \tag{6.22}$$

where $\boldsymbol{\varphi}(\vec{\boldsymbol{r}}_l(Y)) = (\varphi(\vec{\boldsymbol{r}}_l(Y)_1), ..., \varphi(\vec{\boldsymbol{r}}_l(Y)_{6N})),\ l \in \{1, 2, 3\}$.

Finally, combining formula (6.16), (6.19) and (6.22), we get the discretized formulation for (6.10), (6.11) and (6.12):

$$\min_U J_l(U) = \frac{h}{2}(\vec{T}(PY) - \vec{R})^T(\vec{T}(PY) - \vec{R}) + \frac{\alpha h}{2}U^TA^TAU + \frac{\beta h}{6}\boldsymbol{\varphi}(\vec{\boldsymbol{r}}_l(Y))e^T, \tag{6.23}$$

where $PY = PU + PX, l \in \{1, 2, 3\}$.

*Remark* 6.11. In the implementation, we impose the Dirichlet boundary condition,

namely, $\boldsymbol{u}(\boldsymbol{x}) = 0$ when $\boldsymbol{x} \in \partial\Omega$, which means that we assume that the transformation is deformed in the interior region and fixed on the boundary. This is a suitable assumption in image registration because the main information is usually located in the interior region, especially for the medical imaging. If there is information near the boundary, we can assume that this boundary can be moving and choose the Neumann boundary condition.

**Optimization Method**

Here, in the numerical implementation, we choose a Gauss-Newton algorithm with a line search method to solve the resulting unconstrained optimization problems (6.23). The most important part is to approximate the Hessian and solve the Gauss-Newton system.

Next, presenting how to construct the approximated Hessian $\hat{H}^k$, we only investigate the relevant details about solving Model 1 since solving Model 2 and Model 3 follow the same lines.

Directly following Section 4.4.2, the gradients and approximated Hessians of the discretized SSD and the discretized diffusion regularizer are respectively:

$$\begin{cases} d_1 &= hP^T\vec{T}_{\tilde{\boldsymbol{U}}}^T(\vec{T}(\tilde{\boldsymbol{U}}) - \vec{R}), \\ \hat{H}_1 &= hP^T\vec{T}_{\tilde{\boldsymbol{U}}}^T\vec{T}_{\tilde{\boldsymbol{U}}}P, \end{cases} \tag{6.24}$$

and

$$\begin{cases} d_2 &= \alpha hA^TAU, \\ H_2 &= \alpha hA^TA. \end{cases} \tag{6.25}$$

The gradient and the approximated Hessian of the discretized new regularizer are as follows:

$$\begin{cases} d_3 &= \frac{\beta h}{6}\mathrm{d}\vec{\boldsymbol{r}}_1^T\mathrm{d}\boldsymbol{\varphi}(\vec{\boldsymbol{r}}_1), \\ \hat{H}_3 &= \frac{\beta h}{6}\mathrm{d}\vec{\boldsymbol{r}}_1^T\mathrm{d}^2\boldsymbol{\varphi}(\vec{\boldsymbol{r}}_1)\mathrm{d}\vec{\boldsymbol{r}}_1, \end{cases} \tag{6.26}$$

where $\mathrm{d}\boldsymbol{\varphi}(\vec{\boldsymbol{r}}_1) = (\varphi'((\vec{\boldsymbol{r}}_1)_1), ..., \varphi'((\vec{\boldsymbol{r}}_1)_{6N}))^T$ is the vector of derivatives of $\boldsymbol{\varphi}$ at all tetrahedrons,

$$\begin{aligned}
\mathrm{d}\vec{\boldsymbol{r}}_1 &= \mathrm{diag}(\vec{\boldsymbol{r}}_1^1)\mathrm{d}\vec{\boldsymbol{r}}_1^2 + \mathrm{diag}(\vec{\boldsymbol{r}}_1^2)\mathrm{d}\vec{\boldsymbol{r}}_1^1,\\
\mathrm{d}\vec{\boldsymbol{r}}_1^1 &= \mathrm{d}\vec{\boldsymbol{q}}^1 - 2\mathrm{diag}(1./\vec{\boldsymbol{q}}^2)\mathrm{d}\vec{\boldsymbol{q}}^2,\\
\mathrm{d}\vec{\boldsymbol{r}}_1^2 &= -\mathrm{diag}(\vec{\boldsymbol{r}}_1^2 \odot \vec{\boldsymbol{r}}_1^2)[\mathrm{d}\vec{\boldsymbol{q}}^1 + 2\mathrm{diag}(1./\vec{\boldsymbol{q}}^2)\mathrm{d}\vec{\boldsymbol{q}}^2],\\
\mathrm{d}\vec{\boldsymbol{q}}^1 &= 2\textstyle\sum_{l=1}^{9} \mathrm{diag}(B_l Y)B_l,\\
\mathrm{d}\vec{\boldsymbol{q}}^2 &= \mathrm{diag}(B_5 Y \odot B_9 Y - B_6 Y \odot B_8 Y)B_1\\
&+ \mathrm{diag}(B_6 Y \odot B_7 Y - B_4 Y \odot B_9 Y)B_2\\
&+ \mathrm{diag}(B_4 Y \odot B_8 Y - B_5 Y \odot B_7 Y)B_3\\
&+ \mathrm{diag}(B_8 Y \odot B_3 Y - B_2 Y \odot B_9 Y)B_4\\
&+ \mathrm{diag}(B_1 Y \odot B_9 Y - B_3 Y \odot B_7 Y)B_5\\
&+ \mathrm{diag}(B_2 Y \odot B_7 Y - B_1 Y \odot B_8 Y)B_6\\
&+ \mathrm{diag}(B_2 Y \odot B_6 Y - B_3 Y \odot B_5 Y)B_7\\
&+ \mathrm{diag}(B_4 Y \odot B_3 Y - B_1 Y \odot B_6 Y)B_8\\
&+ \mathrm{diag}(B_1 Y \odot B_5 Y - B_2 Y \odot B_4 Y)B_9,
\end{aligned} \tag{6.27}$$

$\odot$ denotes the Hadamard product, $\mathrm{d}\vec{\boldsymbol{r}}_1, \mathrm{d}\vec{\boldsymbol{r}}_1^1, \mathrm{d}\vec{\boldsymbol{r}}_1^2, \mathrm{d}\vec{\boldsymbol{q}}^1, \mathrm{d}\vec{\boldsymbol{q}}^2$ are the Jacobian of $\vec{\boldsymbol{r}}_1, \vec{\boldsymbol{r}}_1^1, \vec{\boldsymbol{r}}_1^2, \vec{\boldsymbol{q}}^1, \vec{\boldsymbol{q}}^2$ with respect to $U$ respectively, $\mathrm{d}^2\boldsymbol{\varphi}(\vec{\boldsymbol{r}}_1)$ is the Hessian of $\boldsymbol{\varphi}$ with respect to $\vec{\boldsymbol{r}}_1$, which is a diagonal matrix whose $i$th diagonal element is $\varphi''((\vec{\boldsymbol{r}}_1)_i),\ 1 \le i \le 6N$. Here, $\mathrm{diag}(v)$ is a diagonal matrix with $v$ on its main diagonal.

So the Gauss-Newton system is

$$\hat{H}\delta U = -d_J, \tag{6.28}$$

where $\hat{H} = \hat{H}_1 + H_2 + \hat{H}_3$ and $d_J = d_1 + d_2 + d_3$. In each iteration, we choose MINRES to solve this resulting Gauss-Newton system [3, 101]. In the implementation, the maximum number of inner iterations of MINRES is set to 50 and the tolerance for the relative residual is set to 0.1. Except for the diagonal preconditioner, we also consider a preconditioner, which is a band matrix $L$ shown in Figure 6.4. By using the Cholesky decomposition and two back substitutions, the computational cost of solving $Lx = b$ is only $\mathcal{O}(n)$. Here, we provide a matrix-free version which can speed up the algorithm since we do not need to formulate and store the matrix. In the Appendix 6.6.3 and 6.6.4, we illustrate the details about how to compute the matrix-vector product $\hat{H}v$, the diagonal of $\hat{H}$ and the preconditioner $L$.

For the step length, we use the Armijo strategy with backtracking to find a suitable step length $\theta$, which has been summarized in Algorithm 9. In addition, when the change in the objective function, the norm of the update and the norm of the gradient are all sufficiently small, the iterations are terminated. Hence, a Gauss-Newton scheme with Armijo line search can be developed (Algorithm 19). For the resulting Gauss-Newton scheme by using Armijo line search, we have the following global convergence result.

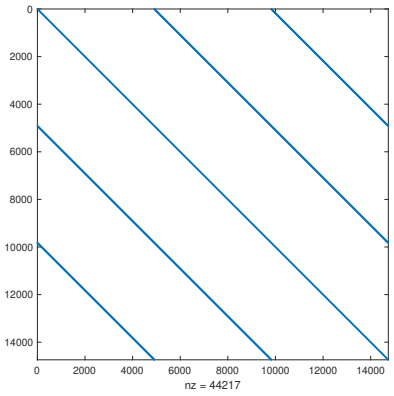



FIGURE 6.4: The structure of the preconditioner $L$. $L$ is composed of the diagonals of blocks of the approximated Hessian $\hat{H}$. The dimension of the problem is 14739 ($3 \times 17 \times 17 \times 17$).

**Theorem 6.12.** *Let $T$ and $R$ be twice continuously differentiable. If each $U^k$ generated by Algorithm 19 is in the $\mathcal{U}$, we obtain*

$$\lim_{k\to\infty} d_J(U^k) = 0 \tag{6.29}$$

*and hence any limit point of the sequence of iterates generated by Algorithm 19 is a stationary point. Here,*

$$\mathcal{U} = \{U \mid (\vec{\boldsymbol{r}}_1(U))_l \leq 1 - \epsilon, 1 \leq l \leq 6N\} \tag{6.30}$$

*and $\epsilon$ is a small constant.*

*Proof.* According to Theorem 4.7, we just need to prove that the following conditions are satisfied:

1). $d_J$ is Lipschitz continuous;

2). For all $k$, $\hat{H}^k$ is symmetric and positive definite;

3). There exist constants $\bar{\kappa}$ and $\zeta$ such that the condition number $\kappa(\hat{H}^k) \leq \bar{\kappa}$ and the norm $\|\hat{H}^k\| \leq \zeta$ for all $k$;

4). $J_1(U)$ has a lower bound.

Here, we use the Dirichlet boundary conditions and assume that $\|U\|$ is bounded. Then $\vec{\boldsymbol{r}}_1(U)$ is a continuous mapping from a compact set to $\mathbb{R}^{6N\times 1}$. Hence, for some small $\epsilon > 0$, $\mathcal{U}$ is compact. Then just follow the proof of Theorem 4.10.

□

*Remark* 6.13. Theorem 6.12 is also valid for Model 2 and we only need to change $\mathcal{U}$ into

$$\mathcal{U} = \{U \mid (\vec{\boldsymbol{r}}_2(U))_l \leq 1 - \epsilon, 1 \leq l \leq 6N\}. \tag{6.31}$$

**Algorithm 19:** Gauss-Newton scheme by using Armijo line search for Image Registration: $(U, \text{ID}) \leftarrow \text{GNAIRA}(\alpha, \beta, U^0, T, R)$

Step 1: Set $k = 0$ at the solution point $U^k = U^0$;
Step 2: For (6.23), compute the energy functional $J(U^k)$, its gradient $d_J^k$ and the approximated Hessian $\hat{H}^k$;
**while** *"the stopping criteria is not satisfied"* **do**
    Solve the Gauss-Newton equation: $\hat{H}^k \delta U^k = -d_J^k$;
    $(U^{k+1}, \text{ID}) \leftarrow \text{ALS}(U^k, \delta U^k)$ by Algorithm 9;
    **if** $ID = 1$ **then**
        Exit this algorithm;
    **else**
        $k = k + 1$;
        Compute $J(U^k)$, $d_J^k$ and $\hat{H}^k$;
    **end**
**end**

For providing reliable initial guesses, the multi-level strategy is also used in the numerical implementation.

## 6.4 Numerical Experiments

In this section, we demonstrate the performance of our new models (6.10) and (6.11) by two 3D examples. Specifically we shall compare these models:

- NM1 from (6.10);
- NM2 from (6.11);
- NM3 LLL from (6.12);
- Hyper1 from (3.11);
- Hyper2 from (3.12);
- LDDMM from (4.7).

All the numerical experiments are run in Matlab 2018a on a PC with 3.40 GHz Intel(R) Core(TM) i7-4770 microprocessor and 32 GB of memory. As a comparison, we compare our models (6.10) and (6.11) with the state-of-the-art methods, the hyperelastic models (Hyper1 and Hyper2)[11, 33], LDDMM [88] and modified LLL model (NM3).

For the choice of the parameters of Hyper1 and Hyper2, we just use the default parameters $\alpha_l = 100$ (length regularizer), $\alpha_s = 10$ (surface regularizer) and $\alpha_v = 100$ (volume regularizer) [88]. In addition, for Hyper1, we also set $\alpha_1 = 0.5$ and $\alpha_2 = 1$. For NM1, NM2, and NM3, we first fix $\alpha = 100$ to be consistent with Hyper1 and Hyper2 and the

choices of $\beta$ will be discussed later. For LDDMM, in these two examples, we set the parameter $\alpha = 400$ to control the smoothness of the velocity; this value was found to be an empirically optimal choice.

### 6.4.1 Example 1

In this example, we construct a synthetic example (a big ball and a small collapsed ball) to highlight the advantage of our models (6.10) and (6.11) over the other models. Figure 6.5 shows the template and the reference. Here, the dimension of the given images is $64 \times 64 \times 64$ and the domain of the images is $[0, 64]^3$. In the implementation, we employ a four-step multilevel strategy for all methods and discretize the images by using regular meshes with $8 \times 8 \times 8$, $16 \times 16 \times 16$, $32 \times 32 \times 32$ and $64 \times 64 \times 64$ respectively. On the finest level, the number of the unknowns in this example is 823875. For the choices of $\beta$ of NM1, NM2 and NM3, we set $\beta = 1.5 \times 10^3$, $2.7 \times 10^3$ and 100 respectively. For LDDMM, we set $N_t = 10$ as the number of time step for computing the characteristic and $n_t = 1$ as the number of cells in space-time grid [88].

Figure 6.6 shows the deformed templates obtained by these six models and Table 6.1 gives the corresponding measurements. Using the symbol $>$ to denote 'better than', the comparisons may be summarized as follows

- **Visual differences**. From Figure 6.6, we can see that NM1, NM2, NM3, and LDDMM have all generated visually acceptable deformed templates (similar to the reference), but Hyper1 and Hyper2 have not. That is,

$$\text{NM1, NM2, NM3 and LDDMM} \quad > \quad \text{Hyper1 and Hyper2.}$$

- **Error (accuracy) differences**. Column 2 of Table 6.1 shows the relative errors of six models to inform accuracies of this example. Hyper1 and Hyper2 are less satisfactory than all others. Precisely, we see that

$$\text{NM2} > \text{NM1} > \text{NM3} > \text{LDDMM} > \text{Hyper1} > \text{Hyper2.}$$

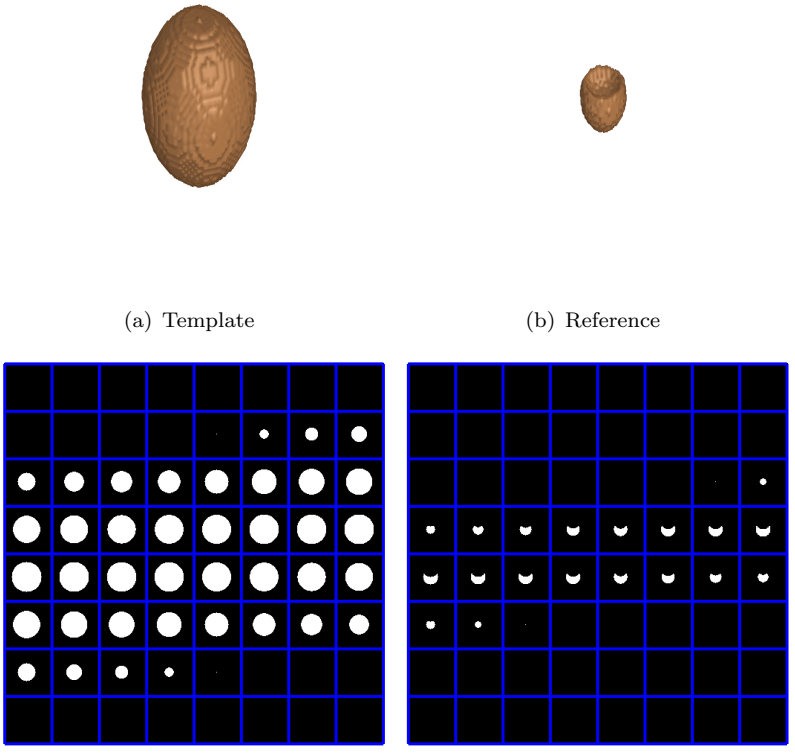
- **Diffeomorphisms differences**. Columns $3-4$ of Table 6.1 show the minimum and maximum of the Jacobian determinant of the transformation obtained by each model. Although we only require $\min \det(\nabla \boldsymbol{y}) > 0$ to ensure a diffeomorphic transformation and in this regard all six models are satisfactory, we can notice that the range of the Jacobian determinant of the transformations obtained by NM1, NM2, NM3, and LDDMM are larger than Hyper1 and Hyper2 since the latter explicitly aims for 1 which is not a reasonable condition in this example.

- **Solution speed differences**. Columns $5-6$ of Table 6.1 show the CPU times and iterations of these six models. We see that

  $$\text{NM1, NM2, NM3 and Hyper1} \quad > \quad \text{Hyper2 and LDDMM.}$$

  Here, for LDDMM, since the deformation is large, the main part of its computing time is spent on computing the characteristic of the transport equation accurately.

Hence, for the large deformation problems where volume preservation is not required, our new models NM1 and NM2 can show the advantages over other models, and NM2 may be recommended.



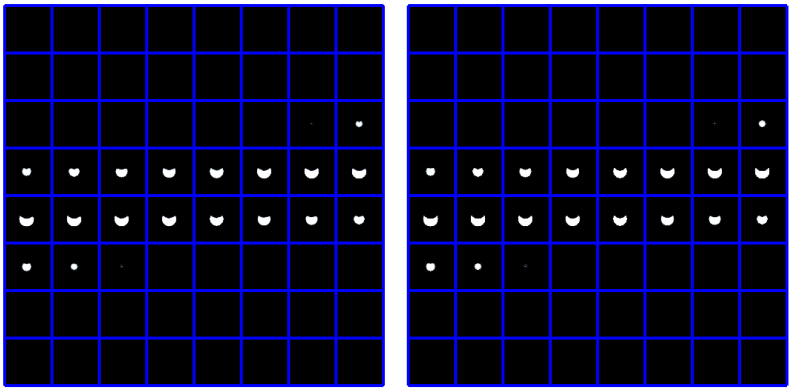
(a) Template (b) Reference

(c) The slices of Template (d) The slices of Reference

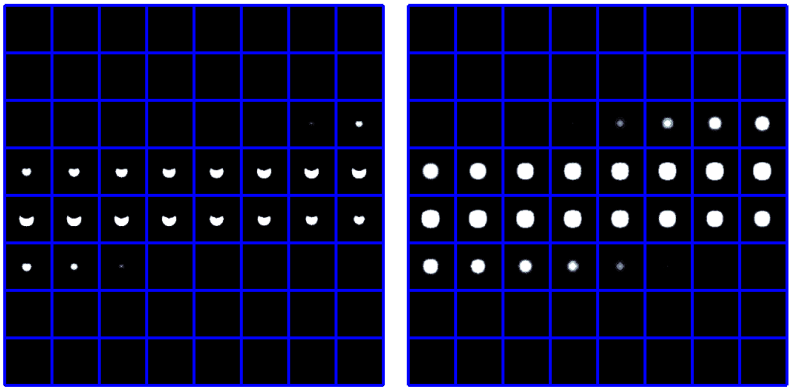
FIGURE 6.5: Example 1: the left column and the right column show the template and the reference respectively.

TABLE 6.1: Example 1 — Comparison of the new models with Hyper1, Hyper2 and LDDMM.

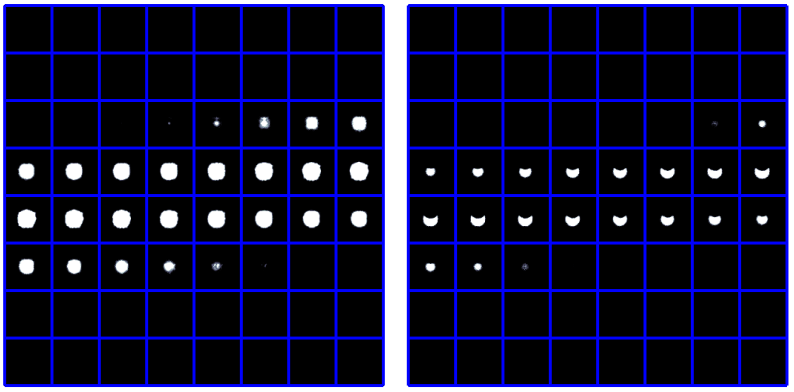
| | Re_SSD | $\min \det \nabla(\boldsymbol{y})$ | $\max \det \nabla(\boldsymbol{y})$ | time (s) | Iterations on each level |
|---|---|---|---|---|---|
| NM1 | 0.10% | 0.1557 | 33.6275 | 13.8 | $11, 3, 3, 3$ |
| NM2 | 0.08% | 0.1593 | 47.9578 | 13.2 | $12, 4, 3, 3$ |
| NM3 | 0.13% | 0.1521 | 36.9822 | 17.3 | $11, 4, 3, 4$ |
| Hyper1 | 15.4% | 0.1630 | 5.3095 | 61.0 | $5, 3, 4, 3$ |
| Hyper2 | 19.7% | 0.1968 | 5.0623 | 115.0 | $9, 6, 3, 4$ |
| LDDMM | 0.55% | $1.18e^{-4}$ | 38.1383 | 1286.8 | $11, 4, 2, 3$ |



(a) $T(\boldsymbol{y}_{\text{NM1}})(0.10\%)$ (b) $T(\boldsymbol{y}_{\text{NM2}})(0.08\%)$



(c) $T(\boldsymbol{y}_{\text{NM3}})(0.13\%)$ (d) $T(\boldsymbol{y}_{\text{Hyper1}})(15.4\%)$



(e) $T(\boldsymbol{y}_{\text{Hyper2}})(19.7\%)$ (f) $T(\boldsymbol{y}_{\text{LDDMM}})(0.55\%)$

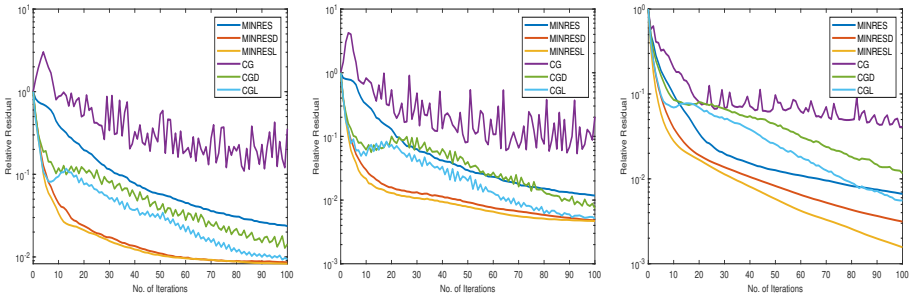
FIGURE 6.6: The results of Example 1: the top row shows the deformed templates obtained by NM1 (left) and NM2 (right). The second row shows the deformed templates obtained by NM3 (left) and Hyper1 (right). The bottom row shows the deformed template obtained by Hyper2 (left) and LDDMM (right). The percentage represents the relative error.

### 6.4.2 Test of The Convergence and Parameters' Sensitivity

Here, we use Example 1 to investigate the preconditioner, convergence of the algorithm and the parameters' sensitivity of our new models.

We first investigate the preconditioner mentioned in section 6.3.3 for NM1, NM2 and

NM3. Since the Gauss-Newton matrix in each iteration is symmetric and positive definite, we also compare the performance of the conjugate gradient method (CG). From Figure 6.7, we can find that for these three models, MINRES is better than CG. In addition, when we apply the diagonal preconditioner and $L$ preconditioner, they can accelerate the convergence significantly. Here, MINRES with $L$ preconditioner can give the best convergence performance among these solvers. Further, from Table 6.2, we can still find that MINRES with $L$ preconditioner uses the least number of iterations and computational time to reach the termination in NM2 and NM3. In NM1, the number of iterations and computational time derived by MINRES with $L$ preconditioner and $CG$ with preconditioner are very similar. Hence, MINRES with $L$ preconditioner is an effective solver for solving the Gauss-Newton system in the proposed new models.




(a) NM1 (b) NM2 (c) NM3

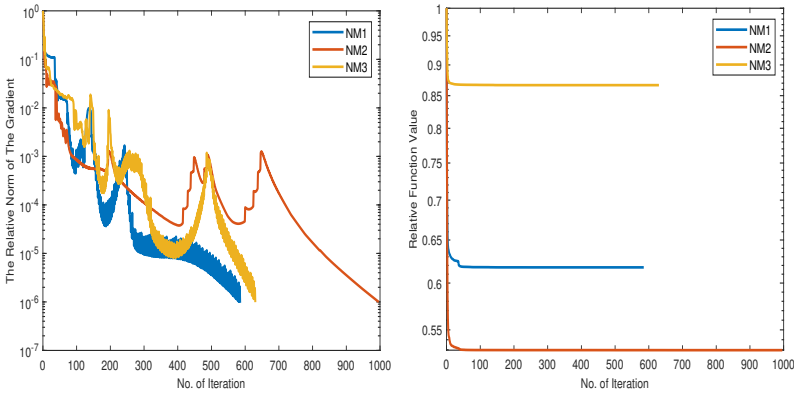
FIGURE 6.7: The performance of different solvers with different preconditioners for NM1, NM2 and NM3 in Example 1. Here, MINRES, MINRESD and MINRESL represent that the solver is MINRES without preconditioner, with diagonal preconditioner and with $L$ preconditioner respectively. And CG, CGD and CGL represent that the solver is conjugate gradient method without preconditioner, with diagonal preconditioner and with $L$ preconditioner respectively.

TABLE 6.2: The number of iterations needed to reach the termination for different solver with different preconditioner in Example 1.

| | NM1 | | NM2 | | NM3 | |
|---|---|---|---|---|---|---|
| | No. of Iter | Time(s) | No. of Iter | Time(s) | No. of Iter | Time(s) |
| MINRES | 34 | 8.2 | 22 | 5.6 | 10 | 4.5 |
| MINRESD | 6 | 1.8 | 4 | 1.4 | 5 | 2.6 |
| MINRESL | 5 | 1.7 | 3 | 1.2 | 4 | 2.3 |
| CG | 46 | 11.6 | 49 | 11.6 | 18 | 7.5 |
| CGD | 13 | 3.4 | 6 | 1.8 | 9 | 4.0 |
| CGL | 5 | 1.6 | 4 | 1.4 | 5 | 2.7 |

We next illustrate the convergence of the algorithm, for NM1, NM2, and NM3. Forcing the algorithm to run until the relative norms of the gradients reach $10^{-6}$ (note the algorithm can satisfy the stopping criteria in only several iterations with a large tolerance e.g. $10^{-2}$), Figure 6.8 shows the relative norm of the gradient from the first order condition, as shown in Figure 6.8 (a), and the relative energy functional values (Figure 6.8 (b)). We see that the relative norm of the gradient of NM1, NM2, and NM3 are reduced to $10^{-6}$. Clearly, the algorithm for NM1, NM2, and NM3 is convergent, as predicted

by Theorem 6.12. The convergence is not monotone, which is the usual behaviour of an optimization approach for a nonconvex problem [97].

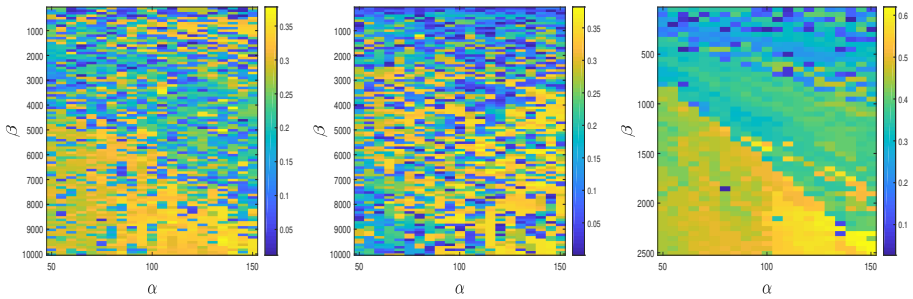



(a) Relative norm of the gradient of NM1-3

(b) Relative function values of NM1-3

FIGURE 6.8: The relative norm of the gradient and relative function values of NM1, NM2 and NM3 in Example1.

We finally test the sensitivity of parameters $\alpha$ and $\beta$ in NM1, NM2 and NM3. We test $\alpha$ and $\beta$ in these specific domains: $[50, 150] \times [100, 10^4]$, $[50, 150] \times [100, 10^4]$ and $[50, 150] \times [50, 2500]$ respectively for NM1, NM2 and NM3 with a view to identify (if possible) a range of parameters that lead to stable results. According to Figure 6.9, we can see that for these three models, they all generate diffeomorphic transformations under the specific parameter region. Hence, the three new models are robust with respect to the diffeomorphism. Furthermore, Figure 6.10 (b) shows that NM2 can give best Re_SSD which is not sensitive with respect to the parameters. At the same time, Figure 6.10 (a,c) show that the Re_SSD generated by NM1 and NM3 will change dramatically when the parameters change. Hence, NM2 is the best model among these three new models in terms of parameters' sensitivity.

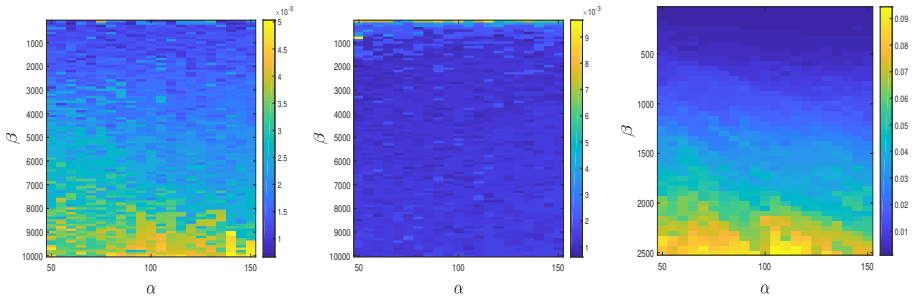



(a) NM1's mindet($\nabla(\boldsymbol{y})$)

(b) NM2's mindet($\nabla(\boldsymbol{y})$)

(c) NM3's mindet($\nabla(\boldsymbol{y})$)

FIGURE 6.9: Parameters' sensitivity test of NM1, NM2 and NM3 (with regard to diffeomorphism, using Example 1). Here, for the three models, they all generate diffeomorphic transformations in the specific parameter domain. Hence, they are robust with respect to the diffeomorphism.




(a) NM1's Re_SSD (bad) (b) NM2's Re_SSD (good) (c) NM3's Re_SSD (bad)

FIGURE 6.10: Test of sensitivity of relative residuals of NM1, NM2 and NM3 in their respective feasible regions from Figure 6.9. Here, we can observe that the relative residual generated by NM2 is not sensitive with respect to the parameters. However, for NM1 and NM3, when the parameters change, the relative residual generated by them will change dramatically. Hence, NM2 is the best model in terms of parameters' sensitivity.

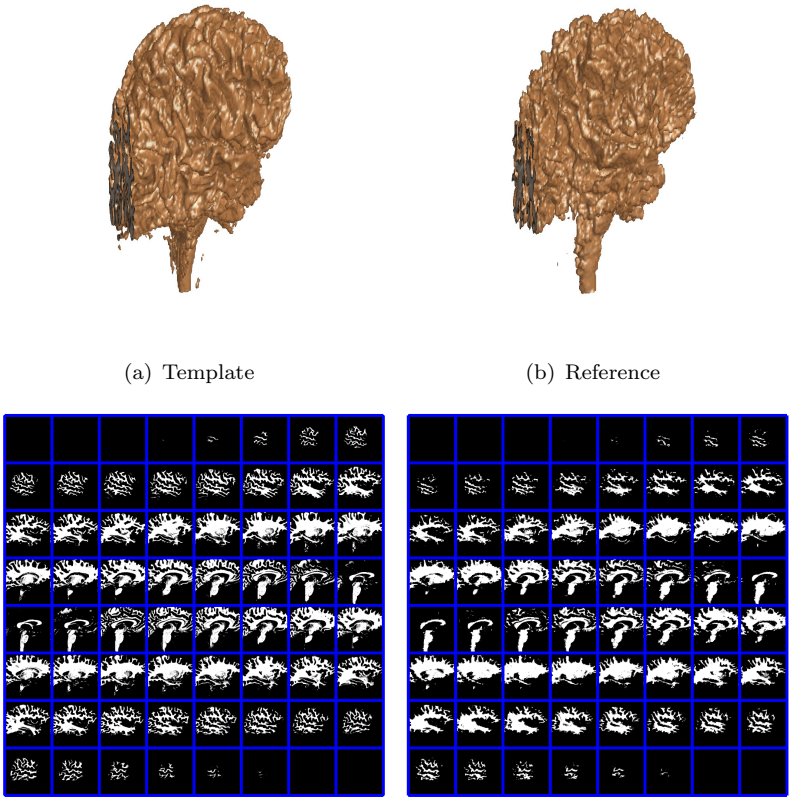
### 6.4.3 Example 2

We illustrate the performance of our recommended model NM2 in registering a pair of 3D real-life images. For completeness, we also compare it with the other five models (NM1, NM3, Hyper1-2, LDDMM). We choose the human brain images from the data accompanying the software FAIR [92]. The template and the corresponding reference are shown in Figure 6.11. The size of the given images is $128 \times 64 \times 128$ and the domain of the images is $[0, 20] \times [0, 10] \times [0, 20]$. In the implementation, for all six models, we employ a four-step multilevel strategy which is to discretize the images in the following different resolutions: $16 \times 8 \times 16$, $32 \times 16 \times 32$, $64 \times 32 \times 64$ and $128 \times 64 \times 128$. The number of unknowns on the finest level in this example is 3244995, making the task a large scale computing problem. Here, for the parameters for NM1, NM2 and NM3, we choose respectively $\beta = 1.5 \times 10^3$, $2.5 \times 10^3$ and 100 respectively. For LDDMM, we set $N_t = 2$ as the number of time step for computing the characteristic and $n_t = 1$ as the number of cells in space-time grid [88].

Figure 6.12 shows the deformed templates obtained by these models and Table 6.3 gives the corresponding quantitative measurements.

Similar to Example 1 results, we observe that although the deformed templates obtained by these six methods are visually good and the resulting transformations are all diffeomorphic (since the minimums of the Jacobian determinant of the transformations are positive), NM2 gives the smallest relative residual and NM1, NM2, NM3, and Hyper1 need much fewer iterations than Hyper2 and LDDMM with the total running times being about half or less of Hyper2 and LDDMM. In addition, the relative residual of Hyper1 is much larger than NM2 and the speed is also much slower than the newly proposed models.

Hence, this example demonstrates that our new model NM2 can be more advantageous than (and competitive to) the state-of-the-art models, Hyper1, Hyper2, LDDMM and NM3 (LLL) in terms of the computational time and the accuracy.



(a) Template (b) Reference

(c) The slices of Template (d) The slices of Reference

FIGURE 6.11: Example 2: the left column and the right column show the template and the reference respectively.

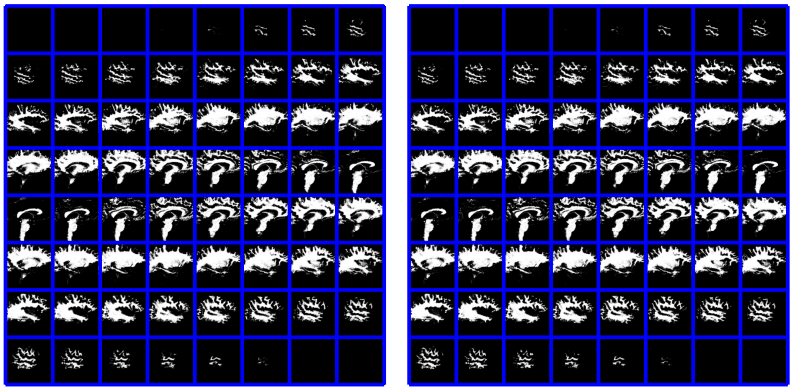
TABLE 6.3: Example 2 — Comparison of the new models with Hyper1, Hyper2 and LDDMM.

| | Resolution | Re_SSD | $\min \det \nabla(\boldsymbol{y})$ | $\max \det \nabla(\boldsymbol{y})$ | time (s) | Itertions on each level |
|---|---|---|---|---|---|---|
| NM1 | $128 \times 64 \times 128$ | 11.09% | 0.0177 | 30.8891 | 268.2 | $6, 10, 14, 16$ |
| NM2 | $128 \times 64 \times 128$ | 8.39% | 0.0033 | 77.1879 | 317.1 | $9, 11, 13, 13$ |
| NM3 | $128 \times 64 \times 128$ | 10.61% | 0.0200 | 85.0400 | 254.9 | $5, 8, 10, 12$ |
| Hyper1 | $128 \times 64 \times 128$ | 24.7% | 0.1223 | 2.6815 | 471.2 | $3, 4, 5, 6$ |
| Hyper2 | $128 \times 64 \times 128$ | 17.1% | 0.0960 | 2.7777 | 1447.6 | $11, 8, 14, 23$ |
| LDDMM | $128 \times 64 \times 128$ | 24.9% | 0.0570 | 10.5899 | 1254.1 | $27, 17, 11, 9$ |

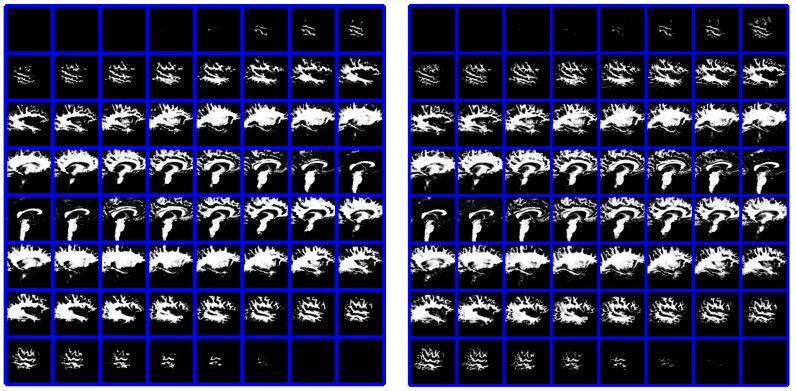
From tests done above and other experiments we conducted, we highly recommend the users to choose NM2 in terms of faster speed and higher accuracy.

## 6.5 Conclusion

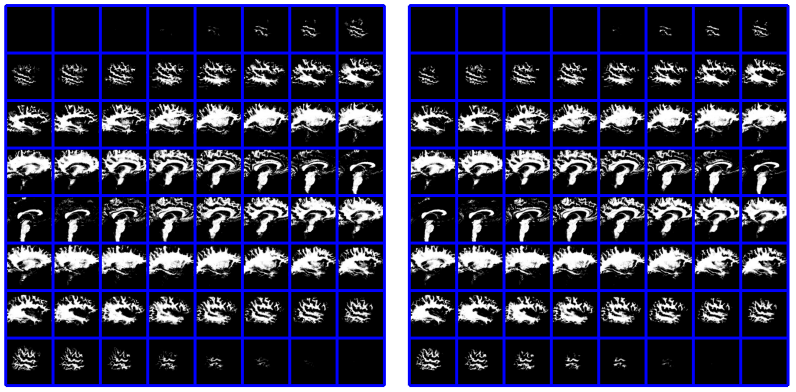
Computing a non-folding transformation in image registration is very important in many applications, such as medical imaging. The visual comparison is not a reliable way to



(a) $T(\boldsymbol{y}_{\text{NM1}})(11.09\%)$ (b) $T(\boldsymbol{y}_{\text{NM2}})(8.39\%)$



(c) $T(\boldsymbol{y}_{\text{NM3}})(10.61\%)$ (d) $T(\boldsymbol{y}_{\text{Hyper1}})(24.7\%)$



(e) $T(\boldsymbol{y}_{\text{Hyper2}})(17.1\%)$ (f) $T(\boldsymbol{y}_{\text{LDDMM}})(24.9\%)$

FIGURE 6.12: The results of Example 2: the top row shows the deformed templates obtained by NM1 (left) and NM2 (right). The second row shows the deformed templates obtained by NM3 (left) and Hyper1 (right). The bottom row shows the deformed template obtained by Hyper2 (left) and LDDMM (right). The percentage represents the relative error.

assess effectiveness. To achieve this aim, many models control the Jacobian determinant of the transformation explicitly which includes the state-of-the-art registration models. However, for some registration problems requiring larger deformation, controlling the Jacobian determinant of the transformation and forcing it close to 1 is not always reasonable; this can be seen from large fitting errors, though the underlying transformations

are diffeomorphic. This chapter explores the alternative method of generalizing the 2D Beltrami coefficient in quasi-conformal theory to 3D. Among the three new models, we have demonstrated that NM2 is the best choice (partly because the underlying 3D regularizer inherits all essential properties of a 2D Beltrami regularizer). To solve the new models efficiently, we design a converging Gauss-Newton scheme. The numerical experiments illustrate that our new models can have advantages over the hyperelastic models, LDDMM and NM3 (LLL). According to the performance of the running time and accuracy, we highly recommend NM2 as the first choice.

## 6.6 Appendix

### 6.6.1 Computation of $A$ in (6.19)

To simplify the formulation (6.18), we build a matrix A:

$$A = \begin{pmatrix} D1 & & \\ D2 & & \\ D3 & & \\ & D1 & \\ & D2 & \\ & D3 & \\ & & D1 \\ & & D2 \\ & & D3 \end{pmatrix}, \tag{6.32}$$

where $D_1 = I_{(n_3+1)} \otimes I_{(n_2+1)} \otimes \partial_{n_1}^{1,h_1}$, $D_2 = I_{(n_3+1)} \otimes \partial_{n_2}^{1,h_2} \otimes I_{(n_1+1)}$, $D_3 = \partial_{n_3}^{1,h_3} \otimes I_{(n_2+1)} \otimes I_{(n_1+1)}$ and

$$\partial_{n_l}^{1,h_l} = \frac{1}{h_l} \begin{pmatrix} -1 & 1 & & \\ & \cdot & \cdot & \\ & & -1 & 1 \end{pmatrix} \in \mathbb{R}^{n_l, n_l+1}, \quad 1 \le l \le 3. \tag{6.33}$$

Here, $\otimes$ indicates Kronecker product.

### 6.6.2 Computation of $M_1$, $M_2$ and $M_3$ in (6.20)

We first investigate the linear approximation $L(x_1, x_2, x_3) = a_1x_1 + a_2x_2 + a_3x_3 + b$ in the tetrahedron $V_3V_4V_5V_7$ (Figure 6.3). Denote these 4 vertices of this tetrahedron by $V_3 = \boldsymbol{x}^{1,1,1}$, $V_4 = \boldsymbol{x}^{2,2,2}$, $V_5 = \boldsymbol{x}^{3,3,3}$ and $V_7 = \boldsymbol{x}^{4,4,4}$. Set $L(\boldsymbol{x}^{1,1,1}) = y^{1,1,1}$, $L(\boldsymbol{x}^{2,2,2}) = y^{2,2,2}$, $L(\boldsymbol{x}^{3,3,3}) = y^{3,3,3}$ and $L(\boldsymbol{x}^{4,4,4}) = y^{4,4,4}$. Substituting $V_3, V_4$, $V_5$ and $V_7$ into $L$, we get

$$\begin{pmatrix} x_1^1 & x_2^1 & x_3^1 & 1 \\ x_1^2 & x_2^2 & x_3^2 & 1 \\ x_1^3 & x_2^3 & x_3^3 & 1 \\ x_1^4 & x_2^4 & x_3^4 & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ b \end{pmatrix} = \begin{pmatrix} y^{1,1,1} \\ y^{2,2,2} \\ y^{3,3,3} \\ y^{4,4,4} \end{pmatrix}. \tag{6.34}$$

Then eliminating $b$, we obtain

$$\begin{pmatrix} x_1^1 - x_1^4 & x_2^1 - x_2^4 & x_3^1 - x_1^4 \\ x_1^2 - x_2^4 & x_2^2 - x_2^4 & x_3^2 - x_2^4 \\ x_1^3 - x_3^4 & x_2^3 - x_2^4 & x_3^3 - x_3^4 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} y^{1,1,1} - y^{4,4,4} \\ y^{2,2,2} - y^{4,4,4} \\ y^{3,3,3} - y^{4,4,4} \end{pmatrix}. \tag{6.35}$$

Set

$$C = \begin{pmatrix} x_1^1 - x_1^4 & x_2^1 - x_2^4 & x_3^1 - x_1^4 \\ x_1^2 - x_2^4 & x_2^2 - x_2^4 & x_3^2 - x_2^4 \\ x_1^3 - x_3^4 & x_2^3 - x_2^4 & x_3^3 - x_3^4 \end{pmatrix}. \tag{6.36}$$

Then we have

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \frac{1}{\det} \begin{pmatrix} C_{11} & C_{21} & C_{31} \\ C_{12} & C_{22} & C_{32} \\ C_{13} & C_{23} & C_{33} \end{pmatrix} \begin{pmatrix} y^{1,1,1} - y^{4,4,4} \\ y^{2,2,2} - y^{4,4,4} \\ y^{3,3,3} - y^{4,4,4} \end{pmatrix}, \tag{6.37}$$

where det is the determinant of $C$ and $C_{ij}$ is the $(i,j)$ cofactor of $C$. Since the domain $\Omega$ has been divided into $N$ voxels, in order to find all $a_1$ in the tetrahedron with the same position of each voxel, we can make it as the following way:

$$\begin{pmatrix} a_1^1 \\ \vdots \\ a_1^N \end{pmatrix} = \frac{1}{\det}(C_{11}(E_3Y - E_7Y) + C_{21}(E_4Y - E_7Y) + C_{31}(E_5Y - E_7Y)), \tag{6.38}$$

where $E_l, l \in \{3,4,5,7\}$ is a matrix which extracts the corresponding positions of the vertices. Set $G_1 = \frac{1}{\det}(C_{11}(E_3 - E_7) + C_{21}(E_4 - E_7) + C_{31}(E_5 - E_7))$. For other 5 tetrahedrons, we can also build $G_l, l \in \{2, ..., 6\}$. Then we get

$$M_1 = \begin{pmatrix} G_1 \\ \vdots \\ G_6 \end{pmatrix}. \tag{6.39}$$

Similarly, we can obtain $M_2$ and $M_3$.

### 6.6.3 Computation of The Matrix-Vector Product $\hat{H}v$

Recall that $\hat{H} = \hat{H}_1 + H_2 + \hat{H}_3$ and we have $\hat{H}v = \hat{H}_1 v + H_2 v + \hat{H}_3 v$.

Firstly, for $\hat{H}_1 v = hP^T \vec{T}_{\tilde{\mathbf{U}}}^T \vec{T}_{\tilde{\mathbf{U}}} Pv$, we need to compute $v_1 = Pv$, $v_2 = \vec{T}_{\tilde{\mathbf{U}}} v_1$, $v_3 = \vec{T}_{\tilde{\mathbf{U}}}^T v_2$ and $\hat{H}_1 v = P^T v_3$. Since P is an averaging matrix from the nodal grid to the cell-centered grid, then as an example, the first component of $Pv$ is

$$\begin{aligned} (Pv)_1 =& \frac{1}{8}((v)_1 + (v)_2 + (v)_{1+n_1} + (v)_{2+n_1} + (v)_{1+(n_1+1)(n_2+1)} \\ &+ (v)_{2+(n_1+1)(n_2+1)} + (v)_{1+n_1+(n_1+1)(n_2+1)} + (v)_{2+n_1+(n_1+1)(n_2+1)}). \end{aligned} \tag{6.40}$$

$\vec{T}_{\tilde{\mathbf{U}}}$ has the following structure:

$$\vec{T}_{\tilde{\mathbf{U}}} = [\mathrm{diag}(w_1), \mathrm{diag}(w_2), \mathrm{diag}(w_3)]. \tag{6.41}$$

Then we have $\vec{T}_{\tilde{\mathbf{U}}} v_1 = \Sigma_{l=1}^{3} w_l \odot v_{1l}$ and $\vec{T}_{\tilde{\mathbf{U}}}^T v_2 = ((w_1 \odot v_2)^T, (w_2 \odot v_2)^T, (w_3 \odot v_2)^T)^T$, where $v_1 = (v_{11}^T, v_{12}^T, v_{13}^T)^T$. Similarly, it is easy to implement $P^T v_3$.

Secondly, in order to compute $H_2 v = \alpha h A^T A v$ and recall (6.32), we just consider how $D_l$ and $D_l^T, l \in \{1, 2, 3\}$ multiply a vector. For simplicity, we only investigate the details of $D_1$. Since $D_1 = I_{(n_3+1)} \otimes I_{(n_2+1)} \otimes \partial_{n_1}^{1,h_1}$, $(\partial_{n_1}^{1,h_1} v')_l = ((v')_{l+1} - (v')_l)/h_1, l \in \{1, ..., n_1\}$ and

$$(\partial_{n_1}^{1,h_1})^T v' = \begin{cases} -(v')_1/h_1; \\ ((v')_{l-1} - (v')_l)/h_1, l \in \{2, ...n_1\}; \\ (v')_{n_1}/h_1; \end{cases} \tag{6.42}$$

we can fast implement the multiplication of $D_1$ and $D_1^T$ with a vector.

Finally, because $\hat{H}_3 = \frac{\beta h}{6} \mathrm{d}\vec{\boldsymbol{r}}_1^T \mathrm{d}^2\boldsymbol{\varphi}(\vec{\boldsymbol{r}}_1)\mathrm{d}\vec{\boldsymbol{r}}_1$ and $\mathrm{d}^2\boldsymbol{\varphi}(\vec{\boldsymbol{r}}_1)$ is a diagonal matrix, we only need to consider computing $\mathrm{d}\vec{\boldsymbol{r}}_1 v$ and $\mathrm{d}\vec{\boldsymbol{r}}_1^T v'$. According to the (6.21), substituting $\mathrm{d}\vec{\boldsymbol{r}}_1^1$, $\mathrm{d}\vec{\boldsymbol{r}}_1^2$, $\mathrm{d}\vec{\boldsymbol{q}}^1$ and $\mathrm{d}\vec{\boldsymbol{q}}^2$ into $\mathrm{d}\vec{\boldsymbol{r}}_1$, we have

$$\mathrm{d}\vec{\boldsymbol{r}}_1 = \sum_{l=1}^{9} \Lambda_l B_l, \tag{6.43}$$

where

$$\begin{aligned}
\Lambda_1 &= 2\Gamma_1 \mathrm{diag}(B_1 Y) + 2\Gamma_2 \mathrm{diag}(B_5 Y \odot B_9 Y - B_6 Y \odot B_8 Y), \\
\Lambda_2 &= 2\Gamma_1 \mathrm{diag}(B_2 Y) + 2\Gamma_2 \mathrm{diag}(B_6 Y \odot B_7 Y - B_4 Y \odot B_9 Y), \\
\Lambda_3 &= 2\Gamma_1 \mathrm{diag}(B_3 Y) + 2\Gamma_2 \mathrm{diag}(B_4 Y \odot B_8 Y - B_5 Y \odot B_7 Y), \\
\Lambda_4 &= 2\Gamma_1 \mathrm{diag}(B_4 Y) + 2\Gamma_2 \mathrm{diag}(B_8 Y \odot B_3 Y - B_2 Y \odot B_9 Y), \\
\Lambda_5 &= 2\Gamma_1 \mathrm{diag}(B_5 Y) + 2\Gamma_2 \mathrm{diag}(B_1 Y \odot B_9 Y - B_3 Y \odot B_7 Y), \\
\Lambda_6 &= 2\Gamma_1 \mathrm{diag}(B_6 Y) + 2\Gamma_2 \mathrm{diag}(B_2 Y \odot B_7 Y - B_1 Y \odot B_8 Y), \\
\Lambda_7 &= 2\Gamma_1 \mathrm{diag}(B_7 Y) + 2\Gamma_2 \mathrm{diag}(B_2 Y \odot B_6 Y - B_3 Y \odot B_5 Y), \\
\Lambda_8 &= 2\Gamma_1 \mathrm{diag}(B_8 Y) + 2\Gamma_2 \mathrm{diag}(B_4 Y \odot B_3 Y - B_1 Y \odot B_6 Y), \\
\Lambda_9 &= 2\Gamma_1 \mathrm{diag}(B_9 Y) + 2\Gamma_2 \mathrm{diag}(B_1 Y \odot B_5 Y - B_2 Y \odot B_4 Y),
\end{aligned} \tag{6.44}$$

$\Gamma_1 = \mathrm{diag}(-\mathrm{d}\vec{\boldsymbol{r}}_1^1 \odot \mathrm{d}\vec{\boldsymbol{r}}_1^2 \odot \mathrm{d}\vec{\boldsymbol{r}}_1^2 + \mathrm{d}\vec{\boldsymbol{r}}_1^2)$ and $\Gamma_2 = \mathrm{diag}((-\mathrm{d}\vec{\boldsymbol{r}}_1^1 \odot \mathrm{d}\vec{\boldsymbol{r}}_1^2 \odot \mathrm{d}\vec{\boldsymbol{r}}_1^2 - \mathrm{d}\vec{\boldsymbol{r}}_1^2)./\vec{\boldsymbol{q}}^2)$. Furthermore, because of (6.20), (6.43) can be reformulated into the following formulation:

$$\mathrm{d}\vec{\boldsymbol{r}}_1 = [\Lambda_1 M_1 + \Lambda_2 M_2 + \Lambda_3 M_3, \Lambda_4 M_1 + \Lambda_5 M_2 + \Lambda_6 M_3, \Lambda_7 M_1 + \Lambda_8 M_2 + \Lambda_9 M_3]. \tag{6.45}$$

Hence, we only need to compute $M_l v_k$, where $l, k \in \{1, 2, 3\}$ and $v = (v_1^T, v_2^T, v_3^T)^T$. For simplification, we only consider $M_1 v_1$. Recall that (6.39) and we can get

$$M_1 v_1 = \begin{pmatrix} G_1 v_1 \\ \vdots \\ G_6 v_1 \end{pmatrix}. \tag{6.46}$$

Since $G_l, l \in \{1, ..., 6\}$ is just the linear combination of the matrix $E_l, l \in \{1, ..., 8\}$, finally we only compute $E_l v_1, l \in \{1, ..., 8\}$ which is very easy to be implemented.

Similarly, in order to compute $\mathrm{d}\vec{\boldsymbol{r}}_1^T v'$, we only need to compute $M_l^T v', l \in \{1, 2, 3\}$ and it can be decomposed to compute $E_l^T v'_k, l \in \{1, ..., 8\}$ and $k \in \{1, ..., 6\}$, where $v' = ((v'_1)^T, ..., (v'_6)^T)^T$.

### 6.6.4 The Diagonal of $\hat{H}$

According to the structure of $\hat{H}_1$, the diagonal of $\hat{H}_1$ is $h(P^T \odot P^T)\varsigma$, where $\varsigma$ is the diagonal of $\vec{T}_{\tilde{\mathbf{U}}}^T \vec{T}_{\tilde{\mathbf{U}}}$.

The diagonal of $H_2$ is $\alpha h(A^T \odot A^T)e$, where $e$ is a vector whose components are all equal to 1.

From (6.45) and $\hat{H}_3 = \frac{\beta h}{6} \mathrm{d}\vec{\boldsymbol{r}}_1^T \mathrm{d}^2\boldsymbol{\varphi}(\vec{\boldsymbol{r}}_1)\mathrm{d}\vec{\boldsymbol{r}}_1$, the diagonal of $\hat{H}_3$ is $\frac{\beta h}{6}(\varsigma_1^T, \varsigma_2^T, \varsigma_3^T)^T$, where

$$\begin{aligned}
\varsigma_1 &= \text{the diagonal of } \ (\Lambda_1 M_1 + \Lambda_2 M_2 + \Lambda_3 M_3)^T \mathrm{d}^2\boldsymbol{\varphi}(\vec{\boldsymbol{r}}_1)(\Lambda_1 M_1 + \Lambda_2 M_2 + \Lambda_3 M_3), \\
\varsigma_2 &= \text{the diagonal of } \ (\Lambda_4 M_1 + \Lambda_5 M_2 + \Lambda_6 M_3)^T \mathrm{d}^2\boldsymbol{\varphi}(\vec{\boldsymbol{r}}_1)(\Lambda_4 M_1 + \Lambda_5 M_2 + \Lambda_6 M_3), \\
\varsigma_3 &= \text{the diagonal of } \ (\Lambda_7 M_1 + \Lambda_8 M_2 + \Lambda_9 M_3)^T \mathrm{d}^2\boldsymbol{\varphi}(\vec{\boldsymbol{r}}_1)(\Lambda_7 M_1 + \Lambda_8 M_2 + \Lambda_9 M_3).
\end{aligned} \tag{6.47}$$

Now we only need to compute the diagonal of $M_{i_1}^T \Lambda_{j_1} \mathrm{d}^2\boldsymbol{\varphi}(\vec{\boldsymbol{r}}_1)\Lambda_{j_2} M_{i_2}$, where $i_1, i_2 \in \{1, 2, 3\}$ and $j_1, j_2 \in \{1, 2, 3\}, \{4, 5, 6\}$ or $\{7, 8, 9\}$. Since $\Lambda_{j_1} \mathrm{d}^2\boldsymbol{\varphi}(\vec{\boldsymbol{r}}_1)\Lambda_{j_2}$ is a diagonal matrix and set $\varsigma$ as the diagonal of $\Lambda_{j_1} \mathrm{d}^2\boldsymbol{\varphi}(\vec{\boldsymbol{r}}_1)\Lambda_{j_2}$, then the diagonal of $M_{i_1}^T \Lambda_{j_1} \mathrm{d}^2\boldsymbol{\varphi}(\vec{\boldsymbol{r}}_1)\Lambda_{j_2} M_{i_2}$ is $(M_{i_1}^T \odot M_{i_2}^T)\varsigma$ which is very easy to be implemented following **Appendix** 6.6.3.

The structure of the preconditioner $L$ is

$$\begin{pmatrix} \mathrm{diag}(\hat{H}_{11}) & \mathrm{diag}(\hat{H}_{12}) & \mathrm{diag}(\hat{H}_{13}) \\ \mathrm{diag}(\hat{H}_{21}) & \mathrm{diag}(\hat{H}_{22}) & \mathrm{diag}(\hat{H}_{23}) \\ \mathrm{diag}(\hat{H}_{31}) & \mathrm{diag}(\hat{H}_{32}) & \mathrm{diag}(\hat{H}_{33}) \end{pmatrix}. \tag{6.48}$$

Since $\hat{H}$ is symmetric and we have got the diagonal of $\hat{H}$, we only need to compute $\mathrm{diag}(\hat{H}_{12})$, $\mathrm{diag}(\hat{H}_{13})$ and $\mathrm{diag}(\hat{H}_{23})$. Actually, they are also computed easily just following the above mentioned steps.

# Chapter 7

# Conclusion and Future Research

This thesis describes a new framework for diffeomorphic image registration based on Beltrami coefficients and introduces a fast numerical algorithm based on a subspace strategy.

## 7.1 Conclusion

Firstly, motivated by the Beltrami concept, we propose a novel regularizer based on the Beltrami coefficient. Just combining this new regularizer with the diffusion model, we establish a novel diffeomorphic registration model. By applying the first-discretize-then-optimize method, we propose an iterative method to solve the resulting nonlinear optimization problem and prove the convergence of the method. Numerical experiments demonstrate that the new model can not only get a diffeomorphic registration even when the deformation is large, but also possess good accuracy in comparison with the current best models. Because there is no definition of the Beltrami coefficient in 3D space, we cannot directly generalize our idea to 3D image registration. Hence, we define a modulus, possessing the same properties as the 2D proposed regularizer, to measure the 'Beltrami-like' coefficient in 3D space. Then we build a new framework for 2D and 3D diffeomorphic image registration.

Secondly, in order to speed up the Gauss-Newton method, motivated by the subspace strategy, we propose a two-step Gauss-Newton method. This technique consists of the possible use of a second step within each iteration of the Gauss-Newton method. In addition, at each level, we try to find a better initial point by minimizing a quadratic approximation of the objective function over the subspace spanned by the interpolated solutions of all the previous levels rather than using the interpolated solution of the previous level directly as the initial point. These techniques show better numerical performance compared with the standard Gauss-Newton method.

In conclusion, this thesis focuses on the 2D and 3D diffeomorphic models and their fast algorithms for image registration problems.

## 7.2 Future Research

In the future, we could further extend the ideas in this thesis, such as, building a 3D 'Beltrami-like' coefficient, generalizing our framework to the multimodality image registration, considering other regularizers, designing fast algorithms (considering ADMM and the multigrid method) and applying deep learning to image registration.

**3D 'Beltrami-like' Coefficient.**

From Beltrami equation (4.8), we have the explicit relation between a Beltrami coefficient and a quasi-conformal mapping. However, since there is no definition of the Beltrami coefficient in 3D space, in Chapter 6, we define a modulus to measure the 'Beltrami-like' coefficient in 3D space. Although we have the modulus of 3D 'Beltrami-like' coefficient, building this 3D 'Beltrami-like' coefficient is hard because of the ill-posedness. Recently, the quaternion has successfully been applied to colour image processing [35, 103]. If we consider combining the quaternion with the 3D space, then the quaternion will potentially provide a way to build this 3D 'Beltrami-like' coefficient.

**Multimodality Image Registration**

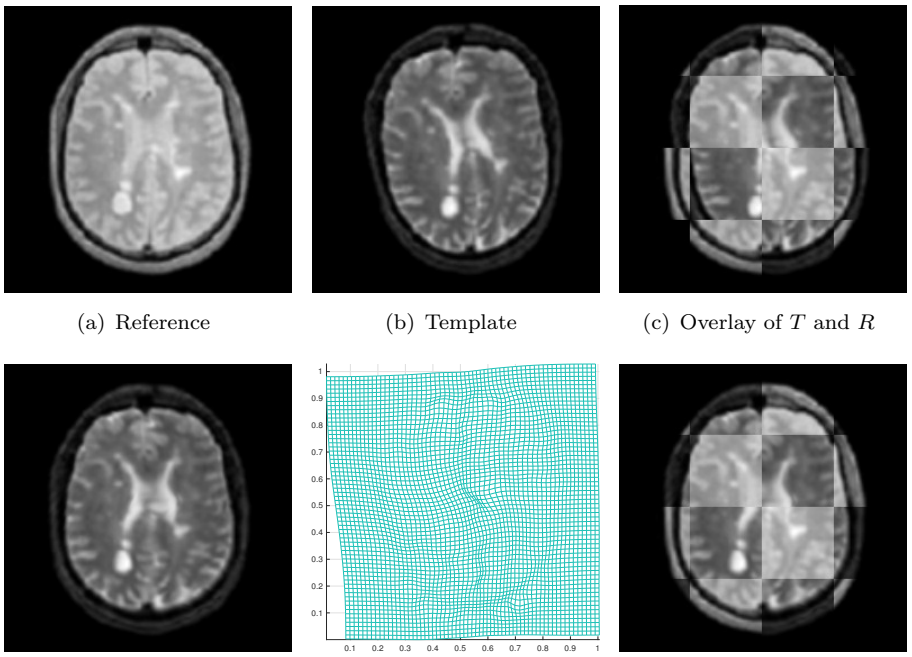
Since the images are taken from different machineries (such as CT/PET), the key point in the multimodality image registration is to design an effective fitting term. Hence, generalizing our framework

$$\min_{\boldsymbol{u}\in\mathcal{U}} \mathcal{J}(\boldsymbol{u}) = \mathcal{D}(T(\boldsymbol{x}+\boldsymbol{u}), R) + \alpha\mathcal{R}(\boldsymbol{u}) + \beta\mathcal{C}(\boldsymbol{y}) \tag{7.1}$$

to the multimodality image registration only needs to replace SSD with a suitable measure, where $\mathcal{R}$ is the regularizer and $\mathcal{C}$ is the control term mentioned in Chapter 4 and 6. In Chapter 3, we have reviewed the most widely used fitting terms in the multimodality image registration: mutual information and normalized gradient fields. For mutual information, how to accurately estimate probability distributions of the grey values is difficult and for normalized gradient fields, [117] has pointed that it will not work well when the gradients are null or very weak. Hence, in our preliminary work, we choose the 'gradient field difference' (GF) and 'Triangular Measure' (TM) proposed in [117] as fitting terms:

$$\begin{aligned} \mathcal{D}^{GF}(\boldsymbol{u}) &= \int_\Omega |\nabla_n T(\boldsymbol{x}+\boldsymbol{u}) - \nabla_n R|^2 \mathrm{d}\boldsymbol{x}, \\ \mathcal{D}^{TM}(\boldsymbol{u}) &= \int_\Omega (|\nabla T(\boldsymbol{x}+\boldsymbol{u})| + |\nabla R| - |\nabla T(\boldsymbol{x}+\boldsymbol{u}) + \nabla R|)^2 \mathrm{d}\boldsymbol{x}. \end{aligned} \tag{7.2}$$

We have applied this generalizing model to a pair of MRI images (T1 and T2) (Figure 7.1 (a, b)), whose dimension is $128 \times 128$. From Figure 7.1, we observe that this generalizing model shows a visually satisfied deformed template and the resulting transformation is diffeomorphic. How to design a better fitting term and how to tune the parameters in the variational model still need to be considered carefully.



(a) Reference (b) Template (c) Overlay of $T$ and $R$

(d) $T(\boldsymbol{y})$ (e) $\boldsymbol{y}$ $\min \det(\nabla \boldsymbol{y}) = 0.0235$ (f) Overlay of $T(\boldsymbol{y})$ and $R$

FIGURE 7.1: A pair of MRI images (T1 and T2). The resulting transformation is diffeomorphic and the deformed template is also visually satisfied.

### Regularizers

In this thesis, we mainly consider the diffusion regularizer. We could further investigate other regularizers reviewed in Chapter 3, which have been used in image registration, or some other regularizers, such as infimal convolution regularizer and total generalized variation regularizer, which have been used in image processing but not used in image registration. Through investigating the geometric properties of the regularizers, we could design new regularizers or decide which regularizer is likely to be useful in our variational framework for any specific application.

TABLE 7.1: Measurements of 2D Brain example.

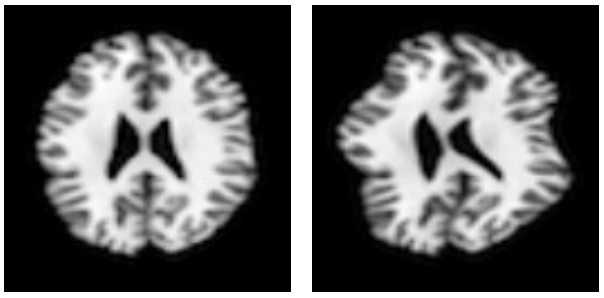
| $\alpha$ | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
|---|---|---|---|---|---|---|---|---|---|---|
| GN | | | | | | | | | | |
| time (s) | 5.66 | 0.85 | 0.74 | 0.83 | 0.76 | 0.66 | 0.64 | 0.86 | 0.70 | 0.91 |
| Re_SSD | 0.75% | 0.30% | 0.34% | 0.41% | 0.46% | 0.52% | 0.56% | 0.62% | 0.67% | 0.72% |
| ADMM | | | | | | | | | | |
| time (s) | 0.28 | 0.33 | 0.44 | 0.40 | 0.38 | 0.35 | 0.35 | 0.39 | 0.44 | 0.52 |
| Re_SSD | 0.37% | 0.43% | 0.42% | 0.48% | 0.55% | 0.63% | 0.66% | 0.69% | 0.74% | 0.75% |

**ADMM**

For the joint variational model (3.3), just following Section 2.6.2, ADMM can be applied as follows:

$$\begin{aligned} \boldsymbol{z}^{k+1} &:= \text{argmin } \mathcal{D}(\boldsymbol{x}+\boldsymbol{z}) + (\lambda^k)^T(\boldsymbol{z}-\boldsymbol{u}^k) + \frac{1}{2\sigma}\|\boldsymbol{z}-\boldsymbol{u}^k\|_2^2, \\ \boldsymbol{u}^{k+1} &:= \text{argmin } \alpha\mathcal{R}(\boldsymbol{u}) + (\lambda^k)^T(\boldsymbol{z}^{k+1}-\boldsymbol{u}) + \frac{1}{2\sigma}\|\boldsymbol{z}^{k+1}-\boldsymbol{u}\|_2^2, \\ \lambda^{k+1} &:= \lambda^k + \frac{1}{\sigma}(\boldsymbol{z}^{k+1}-\boldsymbol{u}^{k+1}), \end{aligned} \tag{7.3}$$

where $\lambda$ is the Lagrangian multiplier and $\sigma > 0$ is a penalty parameter. By exploiting the structure of the fitting term and the regularization term and with the help of the penalty parameter, the speed of ADMM may be faster than standard Gauss-Newton method. For example, if the fitting term is SSD, in each iteration, for the subproblem $\boldsymbol{z}$, we only need to solve a linear system whose coefficient matrix is a tridiagonal matrix in 2D (a pentadiagonal matrix in 3D) and which can be solved by $\mathcal{O}(n)$ operations; if the regularization term is the diffusion term, finding the solution of the subproblem for $\boldsymbol{u}$ only requires to solve the discrete Laplace equation which can be solved fast by iterative solvers. In our preliminary work, we first apply ADMM to the diffusion model and Table 7.1 shows the results of a pair of MRI images (Figure 7.2). From Table 7.1, we observe that under different parameters $\alpha$, ADMM is much faster than GN and keeps the quality simultaneously. However, there still exist a lot of work to do, such as, how we give criteria to choose an efficient value for the penalty parameter $\sigma$ and how we extend the linear constraint to a nonlinear constraint when dealing with the Jacobian determinant of the transformation.



(a) Reference (b) Template

FIGURE 7.2: A pair of MRI images.

**Multigrid Method.**

In 2000, Nash [95] proposed the multigird approach for the discretized optimization problem. The basic idea is to find the suitable search direction in the coarse grid. Given an initial optimization problem on the fine grid

$$\min_{u^h} f^h(u^h) \tag{7.4}$$

and an initial guess $u_0^h$, one iteration of this algorithm consists of:

- If this is the coarsest grid, solve

$$\min_{u^h} f^h(u^h). \tag{7.5}$$

- Otherwise, apply $N_0$ iteration of an optimization algorithm to the original problem, to obtain $u_1^h$.
- Compute

$$\begin{aligned} u_0^H &= I_h^H u_1^h \\ g_1^h &= \nabla f^h(u_1^h) \\ g_0^H &= \nabla f^H(u_0^H) \\ \bar{v} &= g_0^H - I_h^H g_1^h \end{aligned} \tag{7.6}$$

- Apply the multigrid method, with initial guess $u_0^H$ to

$$\min_{u^H} f^H(u^H) - \bar{v}^T u^H \tag{7.7}$$

  and let $u_1^H$ be the result.
- Compute $e^h = I_H^h(u_1^H - u_0^H)$.
- Perform a line search to obtain $u_2^h \leftarrow u_1^h + \alpha e^h$.
- Apply $N_1$ iterations of an optimization algorithm to the original problem, with initial guess $u_2^h$ to obtain $u_3^h$.

In the implementation, the core part is to ensure that $e^h$ is a descent direction. This is true when the each individual optimization problem is convex, the multigrid subproblems are solved 'accurately enough' and the interpolation and restriction operators satisfy: $I_H^h = C_{h,H}(I_h^H)^T$, where $C_{h,H}$ is some positive constant that may depend on $h$ and $H$.

However, for the nonconvex problem, although we can get the descent direction by modifying the subproblem or employing the line search strategy [95, 128], the performance for the hyperelastic model is not very good according to our testing experiments. We

may consider applying the multigrid method based on first-optimize-then-discretize and then motivated by [119], design a good smoother to speed up the computation time.

### Deep Learning

Applying deep learning to image registration is a new trend [113, 116, 130]. Through combining deep learning with our variational framework, we could construct a network for image registration. When the network is trained well, computing the deformed template and the transformation will be very fast.

# Bibliography


[1] Lars Valerian Ahlfors and Clifford J Earle, *Lectures on quasiconformal mappings*, van Nostrand Princeton, 1966.

[2] Angel Angelov and Marcus Wagner, *Multimodal image registration by elastic matching of edge sketches via optimal control*, Journal of Industrial & Management Optimization **10** (2014), no. 2, 567–590.

[3] Richard Barrett, Michael W Berry, Tony F Chan, James Demmel, June Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk Van der Vorst, *Templates for the solution of linear systems: building blocks for iterative methods*, vol. 43, Siam, 1994.

[4] M Faisal Beg, Michael I Miller, Alain Trouvé, and Laurent Younes, *Computing large deformation metric mappings via geodesic flows of diffeomorphisms*, International journal of computer vision **61** (2005), no. 2, 139–157.

[5] Lipman Bers, *Quasiconformal mappings, with applications to differential equations, function theory and topology*, Bulletin of the American Mathematical Society **83** (1977), no. 6, 1083–1100.

[6] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver, *Visual navigation for mobile robots: A survey*, Journal of intelligent and robotic systems **53** (2008), no. 3, 263–296.

[7] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al., *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends® in Machine learning **3** (2011), no. 1, 1–122.

[8] Stephen Boyd and Lieven Vandenberghe, *Convex optimization*, Cambridge university press, 2004.

[9] Chaim Broit, *Optimal registration of deformed images*, Ph.D. thesis, University of Pennsylvania, 1981.

[10] Lisa Gottesfeld Brown, *A survey of image registration techniques*, ACM computing surveys (CSUR) **24** (1992), no. 4, 325–376.

[11] Martin Burger, Jan Modersitzki, and Lars Ruthotto, *A hyperelastic regularization energy for image registration*, SIAM Journal on Scientific Computing **35** (2013), no. 1, B132–B148.

[12] Coralia Cartis, Nicholas IM Gould, and Ph L Toint, *On the complexity of steepest descent, newton's and regularized newton's methods for nonconvex unconstrained optimization problems*, Siam journal on optimization **20** (2010), no. 6, 2833–2852.

[13] Caihua Chen, Bingsheng He, Yinyu Ye, and Xiaoming Yuan, *The direct extension of admm for multi-block convex minimization problems is not necessarily convergent*, Mathematical Programming **155** (2016), no. 1-2, 57–79.

[14] Ke Chen, *Matrix preconditioning techniques and applications*, vol. 19, Cambridge University Press, 2005.

[15] Ke Chen, Geovani Nunes Grapiglia, Jinyun Yuan, and Daoping Zhang, *Improved optimization methods for image registration problems*, Numerical Algorithms **80** (2019), no. 2, 305–336.

[16] Yunmei Chen, Jiangli Shi, Murali Rao, and Jin-Seop Lee, *Deformable multi-modal image registration by maximizing rényi's statistical dependence measure*, Inverse Problems & Imaging **9** (2015), no. 1, 79–103.

[17] Yunmei Chen and Xiaojing Ye, *Inverse consistent deformable image registration*, The Legacy of Alladi Ramakrishnan in the Mathematical Sciences, Springer, 2010, pp. 419–440.

[18] Gary E Christensen, Richard D Rabbitt, Michael I Miller, et al., *Deformable templates using large deformation kinematics*, IEEE transactions on image processing **5** (1996), no. 10, 1435–1447.

[19] Gary Edward Christensen, *Deformable shape models for anatomy*, Ph.D. thesis, Washington University Saint Louis, USA, 1994.

[20] N. Chumchob and Ke Chen, *A robust multigrid approach for variational image registration models*, Journal of Computational and Applied Mathematics **236** (2011), 653–674.

[21] Noppadol Chumchob and Ke Chen, *A robust affine image registration method*, International Journal of Numerical Analysis and Modeling **6** (2009), no. 2, 311–334.

[22] ———, *A variational approach for discontinuity-preserving image registration*, East-West Journal of Mathematics (2010), 266–282.

[23] ———, *Improved variational image registration model and a fast algorithm for its numerical approximation*, Numerical Methods for Partial Differential Equations **28** (2012), no. 6, 1966–1995.

[24] Noppadol Chumchob, Ke Chen, and Carlos Brito-Loeza, *A fourth-order variational image registration model and its fast multigrid algorithm*, Multiscale Modeling & Simulation **9** (2011), no. 1, 89–128.

[25] Philippe G Ciarlet, *Three-dimensional elasticity*, vol. 20, Elsevier, 1988.

[26] ———, *Linear and nonlinear functional analysis with applications*, vol. 130, Siam, 2013.

[27] Bernard Dacorogna, *Direct methods in the calculus of variations*, vol. 78, Springer Science & Business Media, 2007.

[28] ———, *Introduction to the calculus of variations*, World Scientific Publishing Company, 2014.

[29] Carl De Boor, *A practical guide to splines*, vol. 27, Springer-Verlag New York, 1978.

[30] Christian Demant, Bernd Streicher-Abel, and Carsten Garnica, *Industrial image processing: Visual quality control in manufacturing*, Springer Science & Business Media, 2013.

[31] Elizabeth D Dolan and Jorge J Moré, *Benchmarking optimization software with performance profiles*, Mathematical programming **91** (2002), no. 2, 201–213.

[32] Marc Droske and Wolfgang Ring, *A mumford–shah level-set approach for geometric image registration*, SIAM journal on Applied Mathematics **66** (2006), no. 6, 2127–2148.

[33] Marc Droske and Martin Rumpf, *A variational approach to nonrigid morphological image registration*, SIAM Journal on Applied Mathematics **64** (2004), no. 2, 668–687.

[34] Paul Dupuis, Ulf Grenander, and Michael I Miller, *Variational problems on flows of diffeomorphisms for image matching*, Quarterly of applied mathematics (1998), 587–600.

[35] Todd A Ell, Nicolas Le Bihan, and Stephen J Sangwine, *Quaternion fourier transforms for signal and image processing*, John Wiley & Sons, 2014.

[36] Ernie Esser, Xiaoqun Zhang, and Tony F Chan, *A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science*, SIAM Journal on Imaging Sciences **3** (2010), no. 4, 1015–1046.

[37] Kang Feng and Zhong-Ci Shi, *Mathematical theory of elastic structures*, Springer Science & Business Media, 2013.

[38] Bernd Fischer and Jan Modersitzki, *Fast diffusion registration*, Contemporary Mathematics **313** (2002), 117–128.

[39] ———, *Curvature based image registration*, Journal of Mathematical Imaging and Vision **18** (2003), no. 1, 81–85.

[40] ———, *A unified approach to fast image registration and a new curvature based registration technique*, Linear Algebra and its applications **380** (2004), 107–124.

[41] ———, *Ill-posed medicine—an introduction to image registration*, Inverse Problems **24** (2008), no. 3, 034008.

[42] Charles Fox, *An introduction to the calculus of variations*, Courier Corporation, 1987.

[43] Claudia Frohn-Schauf, Stefan Henn, and Kristian Witsch, *Multigrid based total variation image registration*, Computing and Visualization in Science **11** (2008), no. 2, 101–113.

[44] Frederick P Gardiner and Nikola Lakic, *Quasiconformal teichmüller theory*, no. 76, American Mathematical Soc., 2000.

[45] Ali Gholipour, Nasser Kehtarnavaz, Richard Briggs, Michael Devous, and Kaundinya Gopinath, *Brain functional localization: a survey of image registration techniques*, IEEE transactions on medical imaging **26** (2007), no. 4, 427–451.

[46] Tom Goldstein and Stanley Osher, *The split bregman method for l1-regularized problems*, SIAM journal on imaging sciences **2** (2009), no. 2, 323–343.

[47] Geovani Nunes Grapiglia, Jinyun Yuan, and Ya-xiang Yuan, *Nonlinear stepsize control algorithms: Complexity bounds for first-and second-order optimality*, Journal of Optimization Theory and Applications **171** (2016), no. 3, 980–997.

[48] Eldad Haber, Stefan Heldmann, and Jan Modersitzki, *Adaptive mesh refinement for nonparametric image registration*, SIAM journal on scientific computing **30** (2008), no. 6, 3012–3027.

[49] ———, *A computational framework for image-based constrained registration*, Linear Algebra and its Applications **431** (2009), no. 3-4, 459–470.

[50] Eldad Haber and Jan Modersitzki, *Numerical methods for volume preserving image registration*, Inverse problems **20** (2004), no. 5, 1621.

[51] ———, *Intensity gradient based registration and fusion of multi-modal images*, International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2006, pp. 726–733.

[52] ———, *A multilevel method for image registration*, SIAM Journal on Scientific Computing **27** (2006), no. 5, 1594–1607.

[53] ———, *Image registration with guaranteed displacement regularity*, International Journal of Computer Vision **71** (2007), no. 3, 361–372.

[54] ———, *Intensity gradient based registration and fusion of multi-modal images*, Methods of information in medicine **46** (2007), no. 3, 292–299.

[55] Jacques Hadamard, *Sur les problèmes aux dérivées partielles et leur signification physique*, Princeton university bulletin (1902), 49–52.

[56] Steven Haker, Lei Zhu, Allen Tannenbaum, and Sigurd Angenent, *Optimal mass transport for registration and warping*, International Journal of computer vision **60** (2004), no. 3, 225–240.

[57] Bingsheng He and Xiaoming Yuan, *On the o(1/n) convergence rate of the douglas–rachford alternating direction method*, SIAM Journal on Numerical Analysis **50** (2012), no. 2, 700–709.

[58] Stefan Henn, *A levenberg–marquardt scheme for nonlinear image registration*, BIT Numerical Mathematics **43** (2003), no. 4, 743–759.

[59] ———, *A multigrid method for a fourth-order diffusion equation with application to image processing*, SIAM Journal on Scientific Computing **27** (2005), no. 3, 831–849.

[60] ———, *A full curvature based algorithm for image registration*, Journal of Mathematical Imaging and Vision **24** (2006), no. 2, 195–208.

[61] ———, *A translation and rotation invariant gauss–newton like scheme for image registration*, BIT Numerical Mathematics **46** (2006), no. 2, 325–344.

[62] Stefan Henn and Kristian Witsch, *Iterative multigrid regularization techniques for image matching*, SIAM journal on scientific computing **23** (2001), no. 4, 1077–1093.

[63] ———, *Multimodal image registration using a variational approach*, SIAM Journal on Scientific Computing **25** (2004), no. 4, 1429–1447.

[64] ———, *Image registration based on multiscale energy information*, Multiscale Modeling & Simulation **4** (2005), no. 2, 584–609.

[65] Magnus R Hestenes, *Multiplier and gradient methods*, Journal of optimization theory and applications **4** (1969), no. 5, 303–320.

[66] Magnus R Hestenes and Eduard Stiefel, *Methods of conjugate gradients for solving linear systems*, Journal of Research of the National Bureau of Standards **49** (1952), no. 6, 409–436.

[67] Derek LG Hill, Philipp G Batchelor, Mark Holden, and David J Hawkes, *Medical image registration*, Physics in medicine & biology **46** (2001), no. 3, R1.

[68] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal, *Fundamentals of convex analysis*, Springer Science & Business Media, 2012.

[69] Erlend Hodneland, Arvid Lundervold, Jarle Rørvik, and Antonella Z Munthe-Kaas, *Normalized gradient fields for nonlinear motion correction of dce-mri time series*, Computerized Medical Imaging and Graphics **38** (2014), no. 3, 202–210.

[70] Mingyi Hong and Zhi-Quan Luo, *On the linear convergence of the alternating direction method of multipliers*, Mathematical Programming **162** (2017), no. 1-2, 165–199.

[71] Mazlinda Ibrahim, Ke Chen, and Carlos Brito-Loeza, *A novel variational model for image registration using gaussian curvature*, Geometry, Imaging and Computing **1** (2014), no. 4, 417–446.

[72] Jurgen Jost, Jürgen Jost, and Xianqing Li-Jost, *Calculus of variations*, vol. 64, Cambridge University Press, 1998.

[73] Carl T Kelley, *Iterative methods for optimization*, vol. 18, Siam, 1999.

[74] CT Kelley, *Iterative methods for linear and nonlinear equations*, Frontiers in applied mathematics **16** (1995), 575–601.

[75] Arno Klein, Jesper Andersson, Babak A Ardekani, John Ashburner, Brian Avants, Ming-Chang Chiang, Gary E Christensen, D Louis Collins, James Gee, Pierre Hellier, et al., *Evaluation of 14 nonlinear deformation algorithms applied to human brain mri registration*, Neuroimage **46** (2009), no. 3, 786–802.

[76] Lars König and Jan Rühaak, *A fast and accurate parallel algorithm for non-linear image registration using normalized gradient fields*, Biomedical Imaging (ISBI), 2014 IEEE 11th International Symposium on, IEEE, 2014, pp. 580–583.

[77] Ka Chun Lam and Lok Ming Lui, *Landmark-and intensity-based registration with large deformations via quasi-conformal maps*, SIAM Journal on Imaging Sciences **7** (2014), no. 4, 2364–2392.

[78] Serge Lang, *Introduction to linear algebra*, Springer Science & Business Media, 2012.

[79] Yin Tat Lee, Ka Chun Lam, and Lok Ming Lui, *Landmark-matching transformation with large deformation via n-dimensional quasi-conformal maps*, Journal of Scientific Computing **67** (2016), no. 3, 926–954.

[80] Olli Lehto and Kalle I Virtanen, *Quasiconformal mappings in the plane*, vol. 126, Springer New York, 1973.

[81] Hava Lester and Simon R Arridge, *A survey of hierarchical non-linear medical image registration*, Pattern recognition **32** (1999), no. 1, 129–149.

[82] Dong C Liu and Jorge Nocedal, *On the limited memory bfgs method for large scale optimization*, Mathematical programming **45** (1989), no. 1-3, 503–528.

[83] Lok Ming Lui, Ka Chun Lam, Tsz Wai Wong, and Xianfeng Gu, *Texture map and video compression using beltrami representation*, SIAM Journal on Imaging Sciences **6** (2013), no. 4, 1880–1902.

[84] Frederik Maes, Andre Collignon, Dirk Vandermeulen, Guy Marchal, and Paul Suetens, *Multimodality image registration by maximization of mutual information*, IEEE transactions on Medical Imaging **16** (1997), no. 2, 187–198.

[85] JB Antoine Maintz and Max A Viergever, *A survey of medical image registration*, Medical Image Analysis **2** (1998), no. 1, 1–36.

[86] Andreas Mang and George Biros, *An inexact newton–krylov algorithm for constrained diffeomorphic image registration*, SIAM journal on imaging sciences **8** (2015), no. 2, 1030–1069.

[87] ———, *Constrained h^1-regularization schemes for diffeomorphic image registration*, SIAM journal on imaging sciences **9** (2016), no. 3, 1154–1194.

[88] Andreas Mang and Lars Ruthotto, *A lagrangian gauss–newton–krylov solver for mass-and intensity-preserving diffeomorphic image registration*, SIAM Journal on Scientific Computing **39** (2017), no. 5, B860–B885.

[89] Calvin R Maurer and J Michael Fitzpatrick, *A review of medical image registration*, Interactive image-guided neurosurgery **17** (1993).

[90] Tim McInerney and Demetri Terzopoulos, *Deformable models in medical image analysis: a survey*, Medical image analysis **1** (1996), no. 2, 91–108.

[91] Jan Modersitzki, *Numerical methods for image registration*, Oxford University Press, 2004.

[92] ———, *Fair: flexible algorithms for image registration*, vol. 6, SIAM, 2009.

[93] Jorge J Moré, Burton S Garbow, and Kenneth E Hillstrom, *Testing unconstrained optimization software*, ACM Transactions on Mathematical Software (TOMS) **7** (1981), no. 1, 17–41.

[94] Oliver Musse, Fabrice Heitz, and Jean Paul Armspach, *Topology preserving deformable image matching using constrained hierarchical parametric models*, IEEE Transactions on Image Processing **10** (2001), no. 7, 1081–1093.

[95] Stephen G Nash, *A multigrid approach to discretized optimization problems*, Optimization Methods and Software **14** (2000), no. 1-2, 99–116.

[96] Yurii Nesterov, *Introductory lectures on convex optimization: A basic course*, vol. 87, Springer Science & Business Media, 2013.

[97] Jorge Nocedal and Stephen J Wright, *Numerical optimization 2nd*, 2006.

[98] Francisco PM Oliveira and Joao Manuel RS Tavares, *Medical image registration: a review*, Computer methods in biomechanics and biomedical engineering **17** (2014), no. 2, 73–93.

[99] Stanley Osher, Martin Burger, Donald Goldfarb, Jinjun Xu, and Wotao Yin, *An iterative regularization method for total variation-based image restoration*, Multiscale Modeling & Simulation **4** (2005), no. 2, 460–489.

[100] Christopher C Paige, *Computational variants of the lanczos method for the eigenproblem*, IMA Journal of Applied Mathematics **10** (1972), no. 3, 373–381.

[101] Christopher C Paige and Michael A Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM journal on numerical analysis **12** (1975), no. 4, 617–629.

[102] Konstantinos Papafitsoros and Carola-Bibiane Schönlieb, *A combined first and second order variational approach for image reconstruction*, Journal of mathematical imaging and vision **48** (2014), no. 2, 308–338.

[103] Soo-Chang Pei and Ching-Min Cheng, *Color image processing by using binary quaternion-moment-preserving thresholding technique*, IEEE Transactions on Image Processing **8** (1999), no. 5, 614–628.

[104] Josien PW Pluim, JB Antoine Maintz, and Max A Viergever, *Mutual-information-based registration of medical images: a survey*, IEEE transactions on medical imaging **22** (2003), no. 8, 986–1004.

[105] Michael JD Powell, *A method for nonlinear constraints in minimization problems*, Optimization (1969), 283–298.

[106] Torsten Rohlfing, Calvin R Maurer Jr, David A Bluemke, and Michael A Jacobs, *Volume-preserving nonrigid registration of mr breast images using free-form deformation with an incompressibility constraint*, IEEE Transactions on Medical Imaging **22** (2003), no. 6, 730–741.

[107] Leonid I Rudin, Stanley Osher, and Emad Fatemi, *Nonlinear total variation based noise removal algorithms*, Physica D: nonlinear phenomena **60** (1992), no. 1-4, 259–268.

[108] Daniel Rueckert, Luke I Sonoda, Carmel Hayes, Derek LG Hill, Martin O Leach, and David J Hawkes, *Nonrigid registration using free-form deformations: application to breast MR images*, IEEE Transactions on Medical Imaging **18** (1999), no. 8, 712–721.

[109] Jan Rühaak, Lars König, Marc Hallmann, Nils Papenberg, Stefan Heldmann, Hanno Schumacher, and Bernd Fischer, *A fully parallel algorithm for multimodal image registration using normalized gradient fields*, Biomedical Imaging (ISBI), 2013 IEEE 10th International Symposium on, IEEE, 2013, pp. 572–575.

[110] Lars Ruthotto, *Hyperelastic image registration: Theory, numerical methods, and applications*, Ph.D. thesis, University of Münster, 2012.

[111] Michaël Sdika, *A fast nonrigid image registration with constraints on the Jacobian using large scale constrained optimization*, IEEE Transactions on Medical Imaging **27** (2008), no. 2, 271–281.

[112] Ramtin Shams, Parastoo Sadeghi, Rodney A Kennedy, and Richard I Hartley, *A survey of medical image registration on multicore and the gpu*, IEEE Signal Processing Magazine **27** (2010), no. 2, 50–60.

[113] Dinggang Shen, Guorong Wu, and Heung-Il Suk, *Deep learning in medical image analysis*, Annual review of biomedical engineering **19** (2017), 221–248.

[114] Aristeidis Sotiras, Christos Davatzikos, and Nikos Paragios, *Deformable medical image registration: A survey*, IEEE Transactions on Medical Imaging **32** (2013), no. 7, 1153–1190.

[115] Wenyu Sun and Ya-Xiang Yuan, *Optimization theory and methods: nonlinear programming*, vol. 1, Springer Science & Business Media, 2006.

[116] Anis Theljani and Ke Chen, *An unsupervised deep learning method for inverse-consistent multi-modality registration*, 23rd Conference on Medical Image Understanding and Analysis.

[117] Anis Theljani and Ke Chen, *An augmented lagrangian method for solving a new variational model based on gradients similarity measures and high order regulariation for multimodality registration*, Inverse Problems & Imaging **13** (2019), no. 2, 309–335.

[118] J-P Thirion, *Image matching as a diffusion process: an analogy with Maxwell's demons*, Medical Image Analysis **2** (1998), no. 3, 243–260.

[119] Tony Thompson and Ke Chen, *A more robust multigrid algorithm for diffusion type registration models*, Journal of Computational and Applied Mathematics **361** (2019), 502–527.

[120] Andrei Nikolaevich Tikhonov, *On the solution of ill-posed problems and the method of regularization*, Doklady Akademii Nauk, vol. 151, Russian Academy of Sciences, 1963, pp. 501–504.

[121] Lloyd N Trefethen and David Bau III, *Numerical linear algebra*, vol. 50, Siam, 1997.

[122] Alain Trouvé, *Diffeomorphisms groups and pattern matching in image analysis*, International journal of computer vision **28** (1998), no. 3, 213–221.

[123] B Van Brunt, *The calculus of variations. 2004*.

[124] Tom Vercauteren, Xavier Pennec, Aymeric Perchant, and Nicholas Ayache, *Diffeomorphic demons: Efficient non-parametric image registration*, NeuroImage **45** (2009), no. 1, S61–S72.

[125] Paul Viola and William M Wells III, *Alignment by maximization of mutual information*, International journal of computer vision **24** (1997), no. 2, 137–154.

[126] Curtis R Vogel, *Computational methods for inverse problems*, SIAM, 2002.

[127] Joachim Weickert, B Romeny, and M Viergever, *Efficient and reliable schemes for nonlinear diffusion filtering*, IEEE Transactions on Image Processing **7** (1998), no. 3, 398–410.

[128] Zaiwen Wen and Donald Goldfarb, *A line search multigrid method for large-scale nonlinear optimization*, SIAM Journal on Optimization **20** (2009), no. 3, 1478–1503.

[129] Chen Xing and Peihua Qiu, *Intensity-based image registration by nonparametric local smoothing*, IEEE Transactions on Pattern Analysis and Machine Intelligence **33** (2011), no. 10, 2081–2092.

[130] Xiao Yang, Roland Kwitt, Martin Styner, and Marc Niethammer, *Quicksilver: Fast predictive image registration–a deep learning approach*, NeuroImage **158** (2017), 378–396.

[131] Xuan Yang, Jihong Pei, and Jingli Shi, *Inverse consistent non-rigid image registration based on robust point set matching*, Biomedical engineering online **13** (2014), no. 2, S2.

[132] I Yanovsky, P Thompson, S Osher, and A Leow, *Large deformation unbiased diffeomorphic nonlinear image registration: Theory and implementation*, UCLA CAM Report (2006), 06–71.

[133] Wotao Yin, Stanley Osher, Donald Goldfarb, and Jerome Darbon, *Bregman iterative algorithms for $\ell_1$-minimization with applications to compressed sensing*, SIAM Journal on Imaging sciences **1** (2008), no. 1, 143–168.

[134] Ya-xiang Yuan, *A review on subspace methods for nonlinear optimization*, Proceedings of the International Congress of Mathematics, 2014, pp. 807–827.

[135] Daoping Zhang and Ke Chen, *A novel diffeomorphic model for image registration and its algorithm*, Journal of Mathematical Imaging and Vision **60** (2018), no. 8, 1261–1283.

[136] Jianping Zhang and Ke Chen, *Variational image registration by a total fractional-order variation model*, Journal of Computational Physics **293** (2015), 442–461.

[137] XP Zhou, *Weak lower semicontinuity of a functional with any order*, Journal of mathematical analysis and applications **221** (1998), no. 1, 217–237.

[138] Barbara Zitova and Jan Flusser, *Image registration methods: a survey*, Image and vision computing **21** (2003), no. 11, 977–1000.