Image Analysis Based on Differential Operators with Applications to Brain MRIs

by

ZICONG ZHOU

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2019

To my wife, Mei, son, Wenyuan and parents,

who have been always supporting and encouraging me.

ACKNOWLEDGEMENTS

First of all, I give my most sincere praises to my God and Savior Jesus Christ. His wisdom, guidance and disciplines have kept enlightening me so that I could humbly seek and learn to enjoy the journey of Mathematics in my life.

I would like to thank my academic advisor, Professor Dr. Guojun Liao, who serves the chair of my dissertation committee, for his constantly patience and trusts in me. Dr. Liao shares his mathematical insights to motivate me. He collaborates with me to build the research experiences. He coaches me by the collaborative work and connects me to broader research fields.

I thank the members of my dissertation committee Dr. Ren-cang Li, Dr. Benito Chen and Dr. Andrzej Korzeniowski for their interests and advices for my research and sacrificing time to serve in the committee. With many constructive discussions and invaluable comments, I have got to know how build stronger confidence in my future researches. I thank the department Chair, Dr. Jianzhong Su, for his kind suggestions and supports.

I am grateful for the community of Math Department in University of Texas at Arlington. In Dr. Gaik Ambartsoumian and Dr. Andrzej Korzeniowski's classes, I got to enjoy the beauty of analytical mathematics. In Dr. Hristo Kojouharov, Dr. Ren-cang Li and Dr. Chaoqun Liu's classes, I got to establish the foundations in scientific computing. And Dr. Li Wang kindly share her workstation server so that I got to realize some heavy simulations.

In this wonderful community of UTA math department, I met many interesting people and made several good friends who have helped and shared with me during my study in UTA. Xi Chen, Mohammad Javad Latifi Jebelli, Yinlin Dong, Gul Karaduman, Junwei Sun, Slghi Choi and many more, I am thankful for their friendships, partnerships and companion.

Finally, I would like to express my gratitude to my parents, wife and son. My family has always been part of the original and ultimate reasons for me to be who I am, what I am doing and where I am heading to. They are the true given fortunes and blessings in my life.

June 28, 2019

ABSTRACT

Image Analysis Based on Differential Operators

with Applications to Brain MRIs

Zicong Zhou, Ph.D.

The University of Texas at Arlington, 2019

Supervising Professor: Guojun Liao

In differential geometry, computational diffeomorphism (smooth and invertible mapping) has become a fast-growing field in developing the theoretical frameworks and computational toolboxes for the tasks such as computer vision, movie production, gaming industry, medical imaging, etc. Mesh generation is one of components in computational diffeomorphism. In this dissertation, the deformation and variational methods (developed by Dr. Guojun Liao and his co-workers) for mesh generation are discussed, modified and generalized to 3D scenario. The former is based on the control of Jacobian determinant and the latter is based on the controls of both Jacobian determinant and curl vector of a diffeomorphism. In Brain Morphometry, image registration (identify a pixel-wise correspondent relationship of two images based on a dissimilarity measure) is a challenging problem, which demands a diffeomorphism to describe such pixel-wise correspondent relationship. The optimal control approach for image registration (developed by Dr. Guojun Liao and his co-workers) is revised and improved for cheaper computational costs and capability of 3D registration. A novel approach to averaging images is formulated based on averaging a given set

of diffeomorphisms. This approach to averaging images is implemented by an algorithm which includes the variational method and the optimal control image registration.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Introduction

The increasing demands in Precision Medicine Initiatives have been motivating computational scientists to build more robust, accurate and powerful computational capacities for the tasks of medical data managements, diseases diagnoses, health care analysis, conducting treatments and even laser surgeries, etc [11, 24, 51]. In the field of medical imaging, the major challenges includes image construction, which recovers an image from scanned data (e.g. Magnetic Resonant Imaging (MRI)); image registration, which maps an moving image to fixed image through a deformation transformation; image atlas construction, which builds a standard image to represent a certain class of brain images; so and so forth [24]. These computational tools provide the availabilities of digitizing the medical studies such as Brain Morphometry, which investigates variabilities in positions, orientations, shapes and sizes of brain structure from image data. In Brain Morphometry, a key mathematical task is how to characterize diffeomorphisms [17, 18, 43]. The prevailing methods in this field are to focus on the Jacobian determinant $\det\nabla(\boldsymbol{T})$ of a transformation $\boldsymbol{T}$ (ideally diffeomorphism), which models local size changes. The most popular employed approach is Voxel-Based Morphometry (VBM), introduced by Ashburner in [3, 4, 5]. The approach has met considerable controversies [6]. In fact, Fred L. Bookstein, the first recipient of the Morphometry Prize, claimed that "Voxel-Based Morphometry Should Not Be Used with Imperfectly Registered Images" [9]. The controversies and inconsistent results by different research groups [40, 52] indicate that there is a need for innovative

research in the methodology and computational tools for robust image registration and for construction of unbiased template[28, 29]. In this dissertation, it is shown that $\det\nabla(\boldsymbol{T})$ alone cannot completely determine a transformation and, instead, it must includes both Jacobian determinant of $\boldsymbol{T}$, $\det\nabla(\boldsymbol{T})$, and the curl vector of $\boldsymbol{T}$, $\nabla \times (\boldsymbol{T})$, which models local rotations, in all steps of Brain Morphometry studies. To elaborate this point, relevant prior studies from our group are reviewed and a series of computational algorithms for morphometry studies are built and numerically tested.

In chapter 2, the adaptive, moving mesh generation method — the deformation method [39, 35, 38] is reviewed which is achieved based on a prescribed Jacobian determinant. In 1990s, J. Moser and B. Dacorogna studied the existence and construction of diffeomorphisms under a positive Jacobian determinant constraint on a domain in $\mathbb{R}^n$ [16]. The deformation approach in their study has been extended to the adaptive mesh generation problem by our group and collaborators [31, 35]. Due to the demands for higher order element meshes [22, 54], a novel higher order meshing method based on the deformation method is developed. The proposed algorithm is based on the Least-Squares Finite Element Method [7, 27]. Numerical examples are shown to demonstrate the generalization, effectiveness and efficiency of this new meshing method.

In chapter 3, we review variational method for mesh generation, which is based on the prescriptions of both Jacobian determinant and the curl vector[13]. The idea of characterizing diffeomorphisms with the information of curl vector can be traced back to [36, 37]. We realized and generalized the 3D implementations of the method in this dissertation. Beyond that, an approach to the uniqueness conjecture of the variational method on a simple case is given; and based on the uniqueness conjecture, a direct method is formulated with approximated solution. Both numerical examples

of iterative and direct approaches suggest that to uniquely determine a transformation, both Jacobian determinant and curl vector are required.

In chapter 4, a revision of an optimal control approach to image registration method [25] is given. The method is formulated by optimizing control functions that are formed by Jacobian determinant and curl vector. Here, our modification simplifies the control functions with *Poisson* equations, which made the derivation of the first variational derivative cleaner and the computational costs cheaper. This optimal control image registration method has its advantage in producing fixed boundary registration deformations. In theory, image registration deformations encode variabilities of brain image, thus are good surrogate for morphometry study [47]. Based on our implementations, results suggest that the revised version satisfies the inverse consistency and transitivity properties, which are important features of a robust and accurate registration method[12, 47].

In chapter 5, a novel approach to averaging images is formulated based on our variational method and optimal control image registration. The proposed approach is designed through our method of averaging diffeomorphisms[14]. To average images in a meaningful way is the key component in constructing Image Atlas [50, 49, 20]. Different brain image atlases could impact the results of brain analysis [23, 42, 44, 45]. Examples on brain images are tested to validate our mathematical setting of a new way to include curl vector in Tensor Based Morphometry.

In chapter 6, a summary is given. Some possibilities and difficulties on future developments are discussed and highlighted.

CHAPTER 2

Higher Order Meshes by Deformation Method

2.1 Introduction

The deformation method can be traced back to [16], in the 1990s. Computational solution of the deformation method using Least-Squares Finite Element Method (LSFEM) was firstly formulated by [10, 19]. In their works, a **divergence − curl** system is formulated, whose right hand side is a prescribed positive Jacobian and enables the deformation method feasible for generating non-folding meshes on variable domains. In this chapter, we briefly review the deformation method through LSFEM and propose an algorithm based on modifications of the deformation method to generate a higher order mesh on domain Ω. The idea of our proposed approach is summarized in the following steps:

(1) Form a coarse (linear) mesh for a desired domain by the deformation method;

(2) Subdivide on each cell of the coarse mesh into an intermediate (linear) mesh;

(3) Deform the intermediate mesh by moving the new boundary nodes to satisfy the boundary condition, thus a new boundary conforming mesh is generated on the desired domain;

(4) Interpolate nodes based on the new locations to form a higher order (curved) mesh

In section 2.2, the deformation method is reviewed. In section 2.3, details for step (1) to (4) are given and furthered with numerical examples for $p = 3$ (p is the number of degree of the interpolated polynomials, which requires $p + 1$ many number of nodes for interpolation on each a dimension).

2.2 The Deformation Method for Fixed and Moving Domains

In [32], three versions of the deformation method are discussed and analytically introduced. These 3 versions reflect 3 different aspects of a similar motion of a conforming mapping on a simply connected and bounded domain $\Omega \subset \mathbb{R}^n$. However, in this chapter, the proposed approach is constructed through the 3rd version thanks to its generalization of computational applications. It goes as follows.

Let $\Omega_t \subset \mathbb{R}^n$ be a moving (includes fixed) domain with $n = 2$ or 3 and $0 \leq t \leq 1$. And let $\boldsymbol{v}(\boldsymbol{x}, t)$ be the velocity field of nodes on $\partial\Omega_t$ which $\boldsymbol{v}(\boldsymbol{x}, t) \cdot \mathbf{n} = 0$ on $\partial\Omega_t$ in the sense of slippery-wall boundary conditions, which $\mathbf{n}$ is the outward normal vector of $\partial\Omega_t$. Given scalar function $f(\boldsymbol{x}, t) > 0 \in C^1(\boldsymbol{x}, t)$ on the domain $\Omega_t \times [0, 1]$, such that

$$\begin{aligned} &f(\boldsymbol{x}, 0) = 1, \\ &\int_{\Omega_t} \frac{1}{f(\boldsymbol{x}, t)} d\boldsymbol{x} = |\Omega_0|. \end{aligned} \tag{2.2.1}$$

We look for a diffeomorphism $\boldsymbol{\phi}(\boldsymbol{\xi}, t) : \Omega_0 \to \Omega_t$

$$\det \nabla \boldsymbol{\phi}(\boldsymbol{\xi}, t) = f(\boldsymbol{\phi}(\boldsymbol{\xi}, t), t), \ \forall t \in [0, 1] \tag{2.2.2}$$

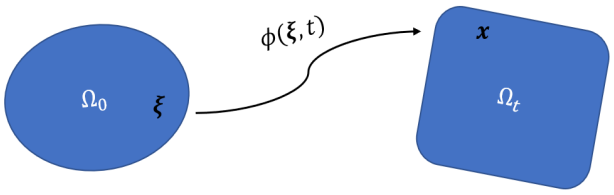


Figure 2.1: Illustration: $\boldsymbol{\phi}$ maps $\boldsymbol{\xi}$ to $\boldsymbol{x}$

In [10, 19], it is shown that such diffeomorphism $\boldsymbol{\phi}$ can be constructed firstly by solving a **divergence − curl** system with using LSFEM, then secondly followed by solving an ODE:

- determine $\boldsymbol{u}(\boldsymbol{x},t)$ on $\mathbb{R}^n$ by solving

$$\begin{cases} \nabla\cdot\boldsymbol{u}(\boldsymbol{x},t) = -\dfrac{\partial}{\partial t}(\dfrac{1}{f(\boldsymbol{x},t)}) \\ \nabla\times\boldsymbol{u}(\boldsymbol{x},t) = \boldsymbol{0} \\ \boldsymbol{u}(\boldsymbol{x},t) = \dfrac{\boldsymbol{v}(\boldsymbol{x},t)}{f(\boldsymbol{x},t)}, \text{ on } \partial\Omega_t \end{cases} \tag{2.2.3}$$

- determine $\boldsymbol{\phi}(\boldsymbol{\xi},t)$ on Ω_0 by solving

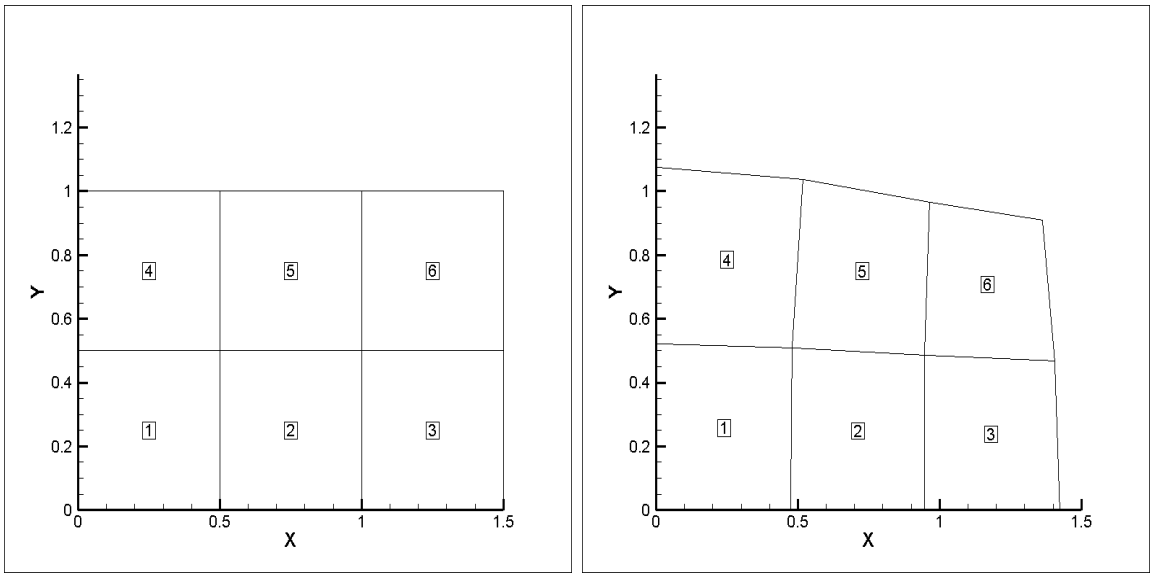
$$\begin{cases} \dfrac{\partial\boldsymbol{\phi}(\boldsymbol{\xi},t)}{\partial t} = f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t), \\ \boldsymbol{\phi}(\boldsymbol{\xi},0) = \boldsymbol{\xi}, \text{ on } \partial\Omega_t \end{cases} \tag{2.2.4}$$

Based on this two-step procedure we may construct a diffeomorphism to the problem of (2.2.1) to (2.2.2). An explicit and detailed theoretical framework of this version can found from [15]. The constraint (2.2.1) indicates that such diffeomorphism is deformed from $\boldsymbol{\phi}(\boldsymbol{\xi},0) = \boldsymbol{id}(\boldsymbol{\xi})$, which is a uniform mesh in terms of discretized computational domain. An effective numerical method for solving the $\mathbf{divergence-curl}$ system (2.2.3), in terms of accuracy, flexibility and compatibility, is using LSFEM. Here, we omitted the details of implementation of LSFEM due to its complexity and depth. Please see the its formulation of implementation in [10, 19] and its analytical convergence studies in chapter 6 of the book [7] and chapter 5 of the book [27]. As for (2.2.4), most standard numerical ODE methods can be implemented which can be chosen depending on the balance between computational accuracy and costs. For simplicity, in this chapter, the *Explicit-Euler* method is picked. The deformation method is a direct method itself, so once the whole package is built, the exact computational complexity is depended on the given number of time-step n_t and the desired number of elements in LSFEM. An algorithm for the $deformation\ method$ on variable domains is provided as follows.

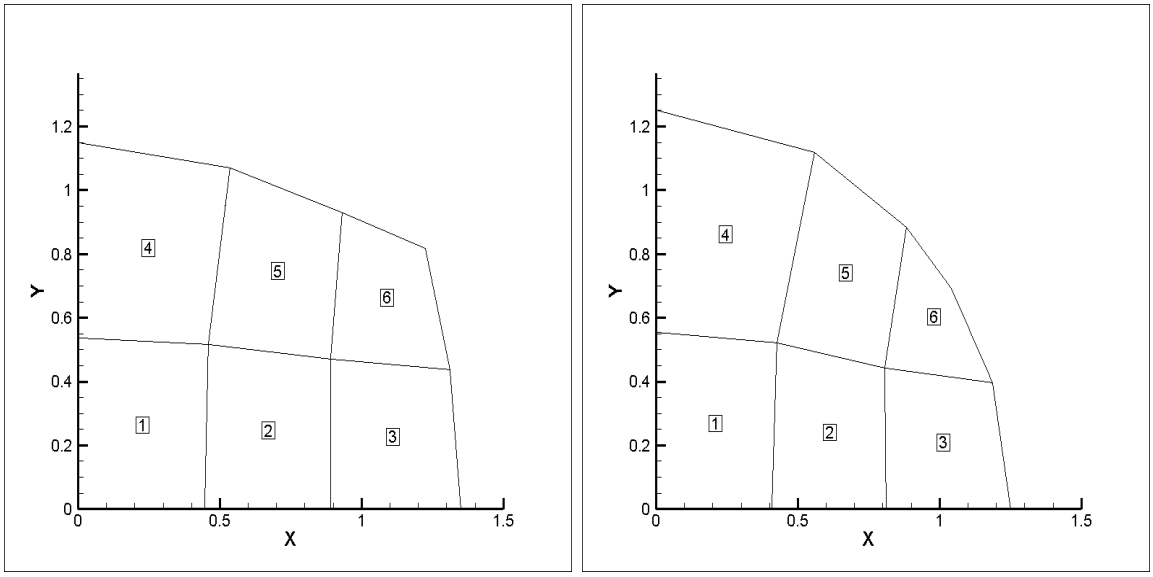
Algorithm 1 : Deformation Method

- 0: input the number of time step n_t & domain Ω_0 in Finite Element representation;
- 1: initialize $dt := \frac{1}{n_t}$, $t = 0$, $\boldsymbol{\phi}(\boldsymbol{\xi}, 0) = \boldsymbol{\xi}$, $f(\boldsymbol{x}, 0) = 1$, prescribe $f(\boldsymbol{x}, t)$;
- 2: for $t = t + dt$ till $t = 1$, do
  - 3: update $f(\boldsymbol{x}, t)$ on $t = t + dt$;
  - 4: normailize $f(\boldsymbol{x}, t) = f(\boldsymbol{x}, t) \int_{\Omega_t} \frac{1}{f(\boldsymbol{x}, t)} d\boldsymbol{x}$;
  - 5: compute $-\frac{\partial}{\partial t}(\frac{1}{f(\boldsymbol{x}, t)})$;
  - 6: solve (2.2.4) by LSFEM to get $\boldsymbol{u}(\boldsymbol{x}, t)$;
  - 7: apply appropriate boundary conditions to $\boldsymbol{u}(\boldsymbol{x}, t)$;
  - 8: update $\boldsymbol{\phi}(\boldsymbol{\xi}, t)$ by solving (2.2.4) using *Explicit-Euler* from Ω_t to Ω_{t+dt};
- 9: output $\Omega_t := \boldsymbol{\phi}(\Omega_0, 1)$.

As it shows below, the moving boundary Dirichlet conditions and slippery-wall Neumann condition can be dealt with at once by the deformation method. It demonstrates Algorithm 1 is compatible with various types of computational scenario.



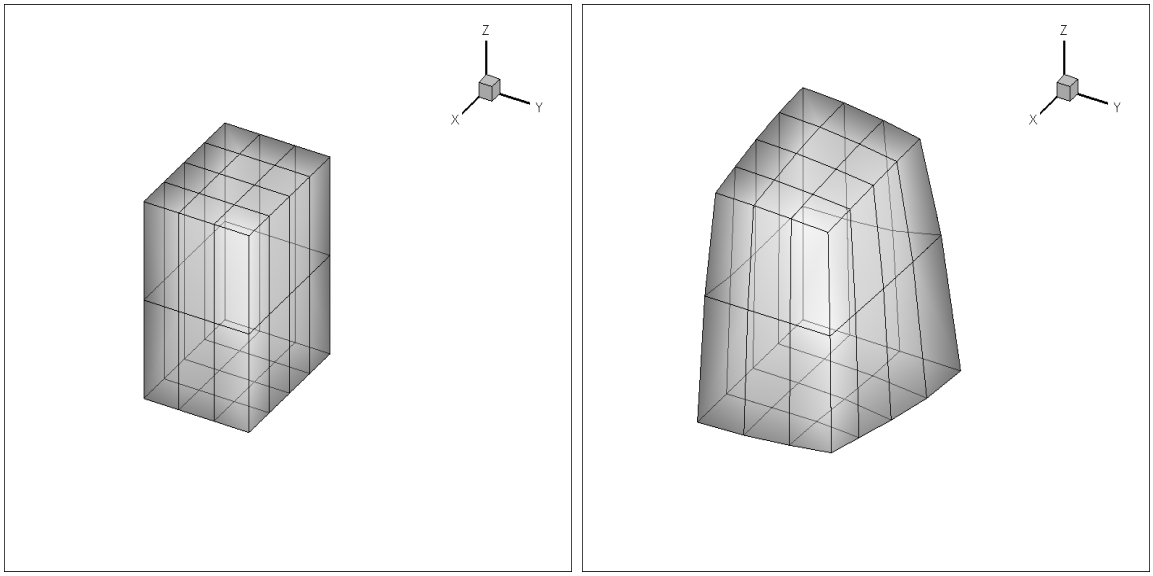
(a) t=0 (b) t=0.3



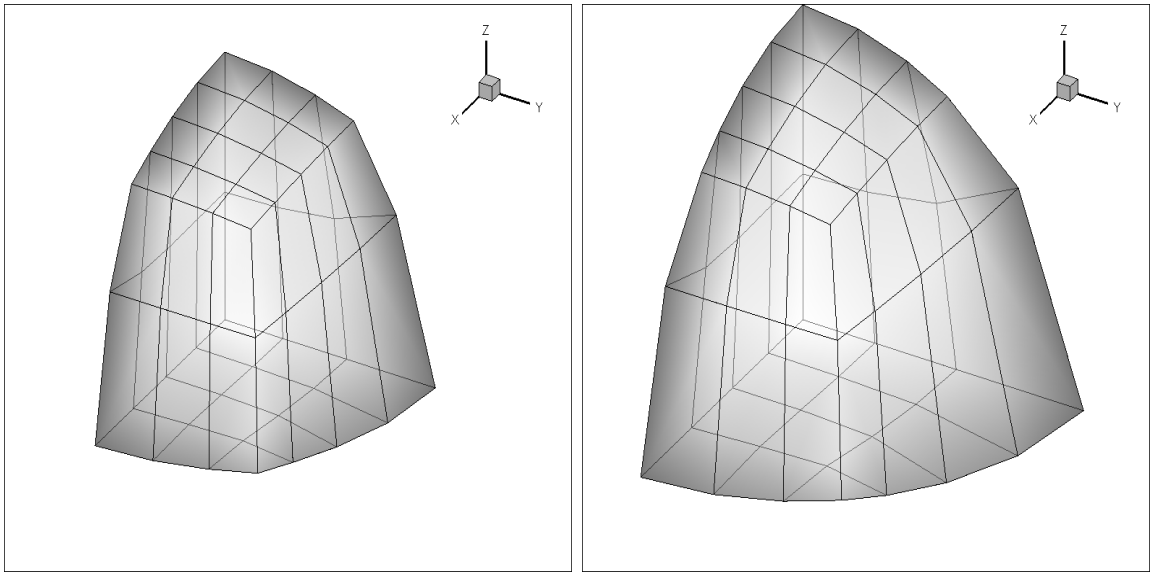
(c) t=0.6 (d) t=1

Figure 2.2: A rectangle to $\frac{1}{4}$ of a disk

Next figure shows a similar 3D example. A brick to is deformed into one-eighth of a ball that centers at the origin and stand on the first octant of $\mathbb{R}^3$. Those 3 faces on the back are slippery-wall condition.



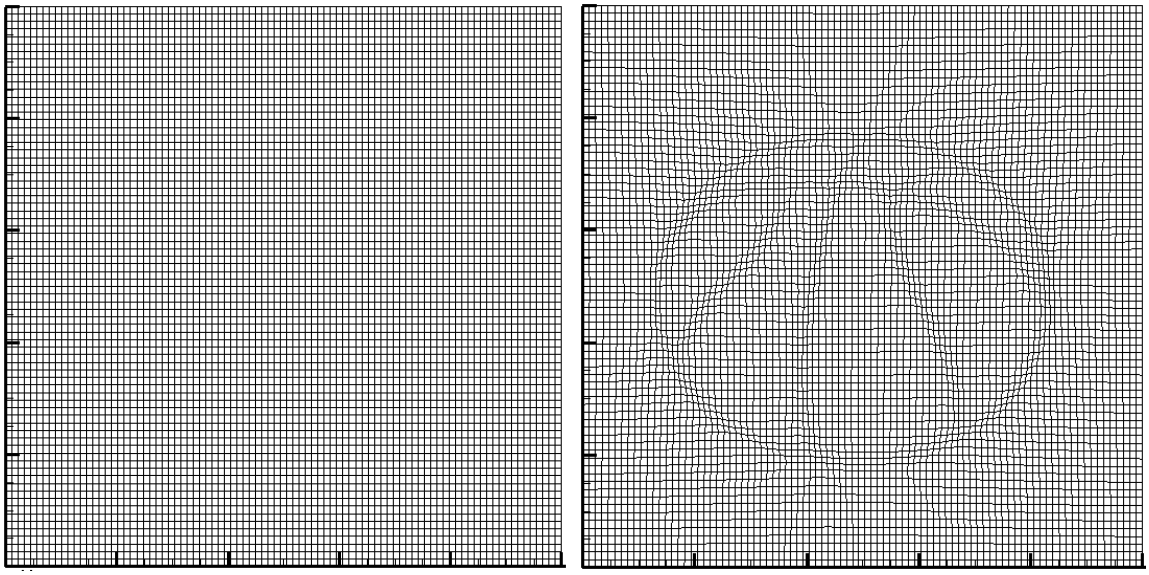
(a) t=0 (b) t=0.3



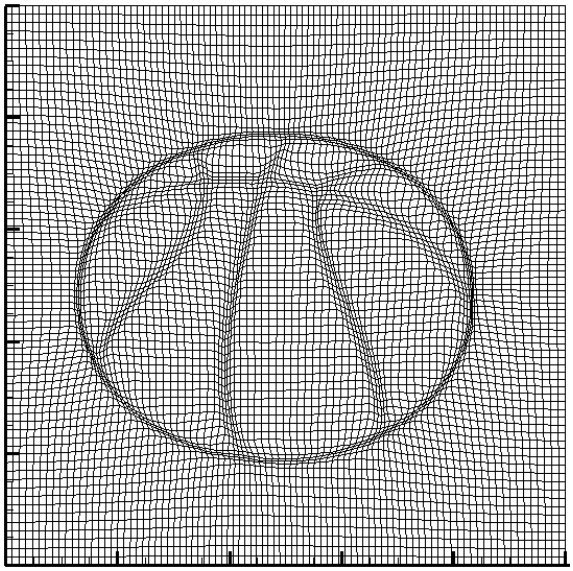
(c) t=0.6 (d) t=1

Figure 2.3: A brick to $\frac{1}{8}$ of a ball

Next, a few more examples demonstrate the flexibility of prescribing Jacobian determinant for desired meshes, in both 2D and 3D cases.

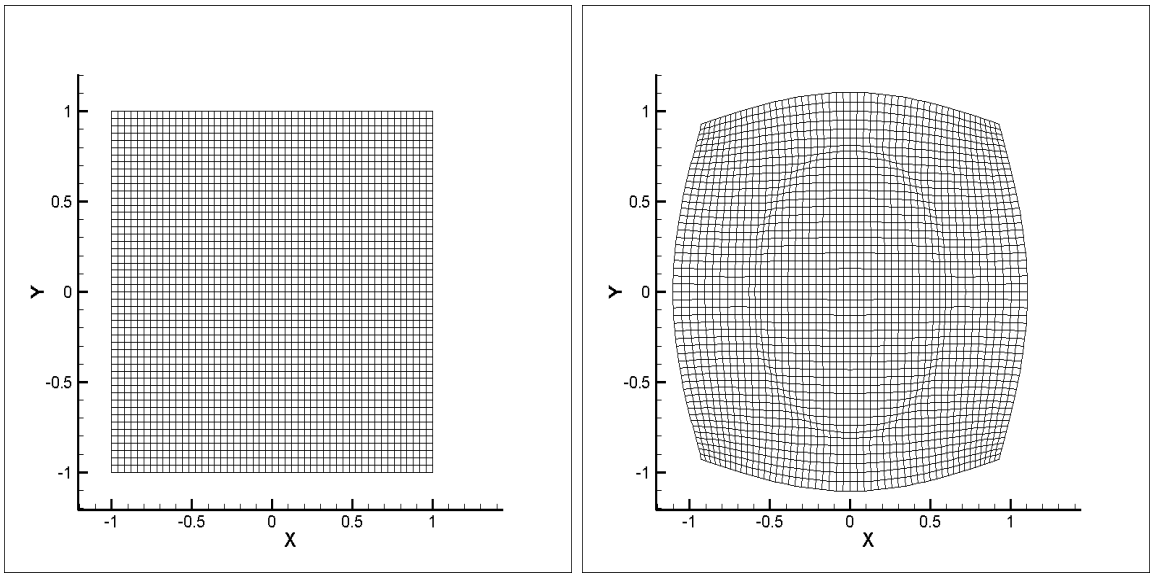


(a) t=0 (b) t=0.5



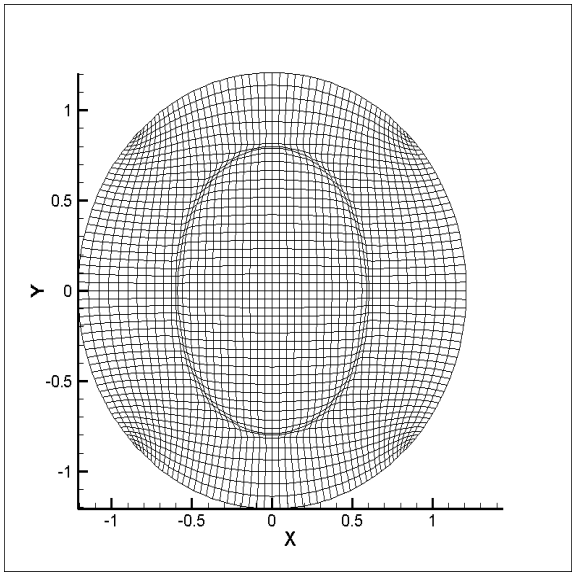
(c) t=1

Figure 2.4: Edge emphasized in a basketball picture



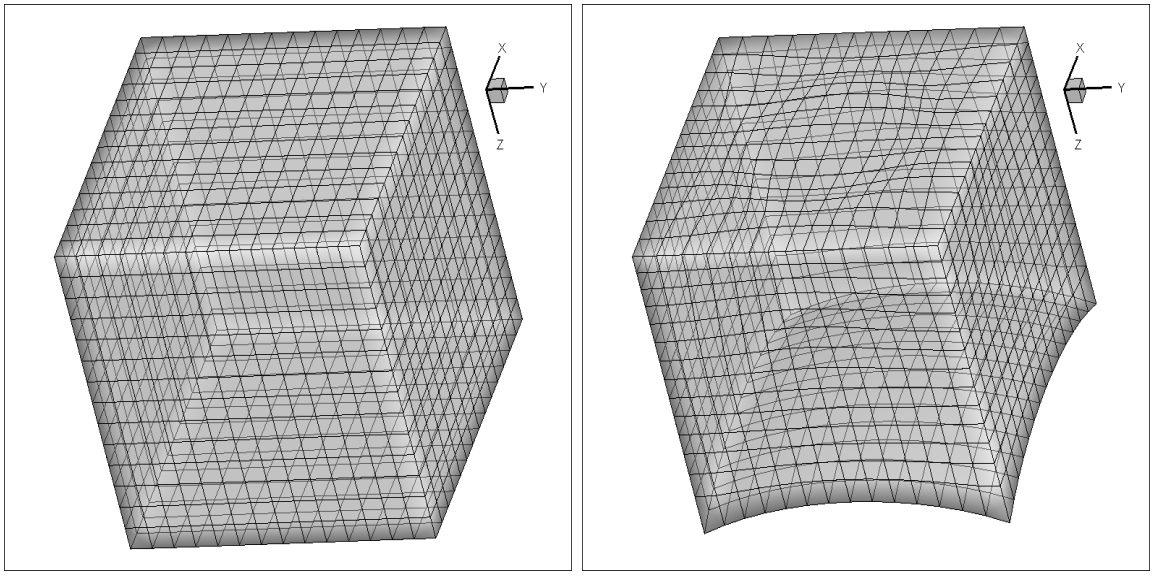
(a) t=0

(b) t=0.5

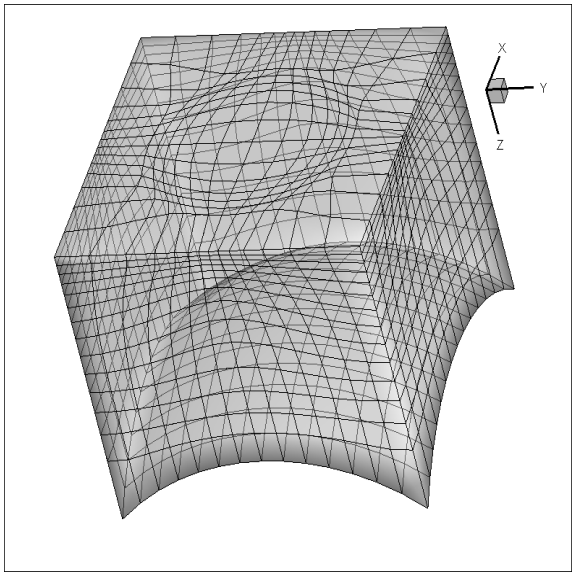


(c) t=1

Figure 2.5: Deform boundary to an ellipse while emphasizing shapes for interiors

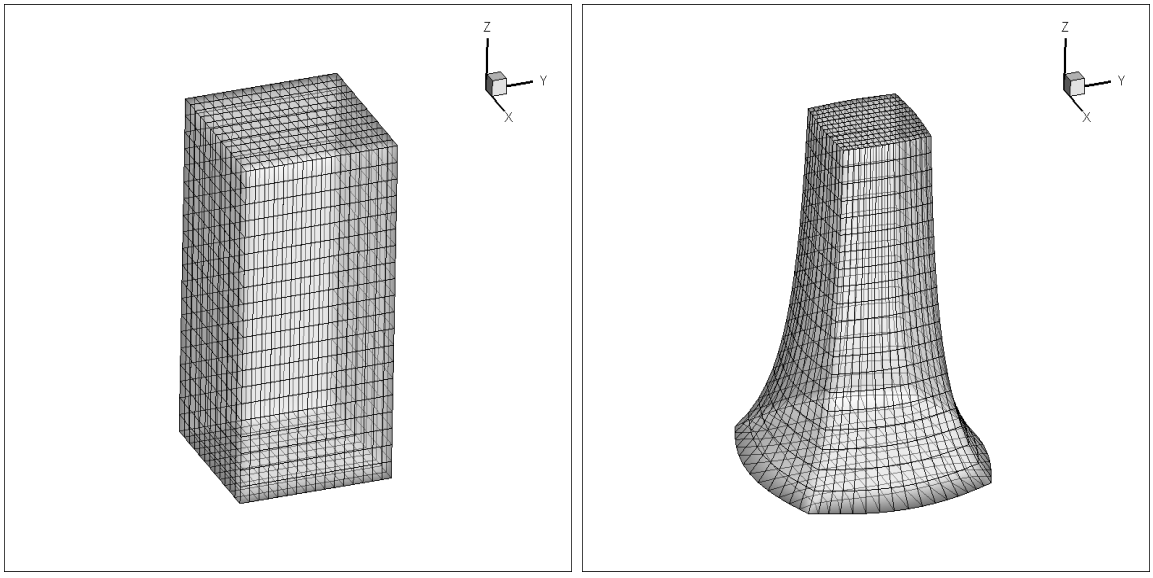


(a) t=0 (b) t=0.5

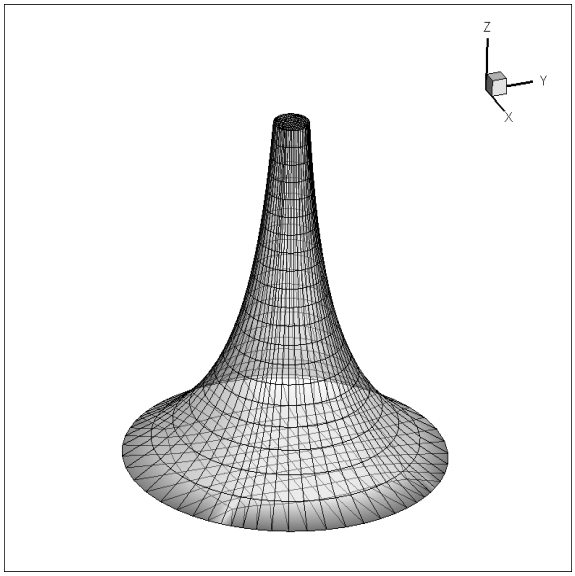


(c) t=1

Figure 2.6: Sitting a cube on a ball while the top is deformed

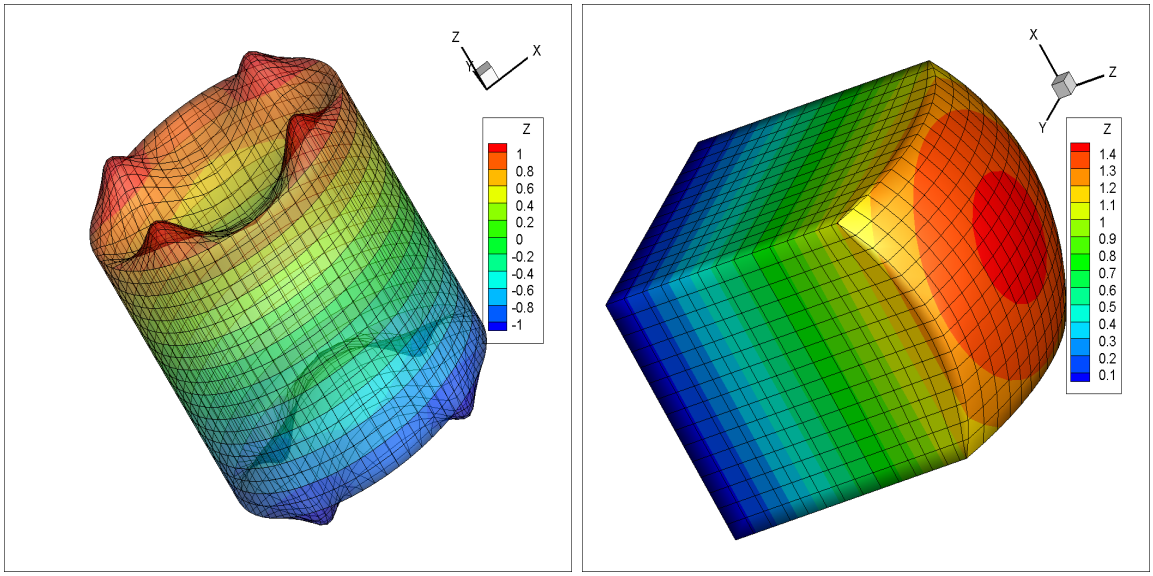


(a) t=0 (b) t=0.5



(c) t=1

Figure 2.7: Deform a tall brick to a horn



(a) A Cylinder like object

(b) Top face deformed to sphere

Figure 2.8: Two Colored 3D Examples

2.3 A New Algorithm for Higher Order Element Meshing

In the previous section, the deformation method was tested to generate computational diffeomorphism, namely, non-folding meshes, from a uniform mesh such

that $\boldsymbol{\phi}(\boldsymbol{\xi}, 0) = \boldsymbol{id}(\boldsymbol{\xi})$. One may ask, what if the prescribed Jacobian determinant $0 < f(\boldsymbol{x}, 0) \neq 1$, can we still construct such diffeomorphsim $\boldsymbol{\phi}(\boldsymbol{\xi}, t)$ by the fashion of deformation method, i. e., can $\boldsymbol{\phi}(\boldsymbol{\xi}, t)$ be found when $\boldsymbol{\phi}(\boldsymbol{\xi}, t) \neq \boldsymbol{id}(\boldsymbol{\xi})$ whose visualization is not represented by an uniform mesh? A quick answer in terms of computational capability is No. However, if there exists a diffeomorphism $\boldsymbol{\varphi}_0 : \Omega \to \Omega_0$, such that $\boldsymbol{\varphi}_0(\boldsymbol{\xi}') = \boldsymbol{\xi}$ and $\det\nabla\boldsymbol{\varphi}_0(\boldsymbol{\xi}') > 0$ describes the given domain Ω_0 and satisfies 2.2.2, then the answer is YES, which was suggested by **Remark 10** of [15]. This means that on a given non-folding mesh Ω_0 with $\det\nabla\boldsymbol{\varphi}_0(\boldsymbol{\xi}') > 0$, it is feasible to refine the mesh $\boldsymbol{\varphi}_0(\boldsymbol{\xi}')$ by the deformation method. In order to theoretically describe the YES scenario, a more general problem formulation for the deformation method is needed. It goes as follows:

Let Ω and $\Omega_t \subset \mathbb{R}^n$, $n = 2$, or 3 and $0 \leq t \leq 1$, be a moving domain and $\boldsymbol{v}(\boldsymbol{x}, t)$ be the velocity field on $\partial\Omega_t$, where $\boldsymbol{v}(\boldsymbol{x}, t) \cdot \mathbf{n} = 0$ on any part of $\partial\Omega_t$ with slippery-wall boundary conditions and $\mathbf{n}$ is the outward normal vector of $\partial\Omega_t$. Given diffeomorphism $\boldsymbol{\varphi}_0 : \Omega \to \Omega_0$ and scalar function $f(\boldsymbol{x}, t) > 0 \in C^1(\boldsymbol{x}, t)$ on the domain $\Omega_t \times [0, 1]$, such that

$$\begin{aligned} f(\boldsymbol{x}, 0) &= \det\nabla(\boldsymbol{\varphi}_0) \\ \int_{\Omega_t} \frac{1}{f(\boldsymbol{x}, t)} d\boldsymbol{x} &= |\Omega_0|. \end{aligned} \tag{2.3.1}$$

A new diffeomorphism $\boldsymbol{\phi}(\boldsymbol{\xi}, t) : \Omega_0 \to \Omega_t$, such that

$$\det\nabla(\boldsymbol{\phi}(\boldsymbol{\xi}, t)) = f(\boldsymbol{\phi}(\boldsymbol{\xi}, t), t), \ \forall t \in [0, 1] \tag{2.3.2}$$

Figure 2.9: $\boldsymbol{\varphi}_0$ maps $\boldsymbol{\xi}'$ to $\boldsymbol{\xi}$ then $\boldsymbol{\phi}$ maps $\boldsymbol{\xi}$ to $\boldsymbol{x}$

can be constructed by solving the following differential equations:

- First, determine $\boldsymbol{u}(\boldsymbol{x},t)$ on $\mathbb{R}^n$ by solving

$$\begin{cases} \nabla\cdot\boldsymbol{u}(\boldsymbol{x},t) = -\dfrac{\partial}{\partial t}(\dfrac{1}{f(\boldsymbol{x},t)}) \\ \nabla\times\boldsymbol{u}(\boldsymbol{x},t) = 0 \\ \boldsymbol{u}(\boldsymbol{x},t) = \dfrac{\boldsymbol{v}(\boldsymbol{x},t)}{f(\boldsymbol{x},t)}, \text{ on } \partial\Omega_t \end{cases} \tag{2.3.3}$$

- Second, determine $\boldsymbol{\phi}(\boldsymbol{\xi},t)$ on Ω_0 by solving

$$\begin{cases} \dfrac{\partial\boldsymbol{\phi}(\boldsymbol{\xi},t)}{\partial t} = f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t), \\ \boldsymbol{\phi}(\boldsymbol{\xi},0) = \boldsymbol{\xi} = \boldsymbol{\varphi}_0(\boldsymbol{\xi}') \end{cases} \tag{2.3.4}$$

Similar to the previous case in section 2, we use this two-step procedure with appropriate modification to construct diffeomorphism $\boldsymbol{\phi}(\boldsymbol{\xi},t)$. This generalization of the deformation method shares a similar theoretical derivation with a slight change of condition from $f(\boldsymbol{x},0)=1$ into $f(\boldsymbol{x},0)=\det\nabla(\boldsymbol{\varphi}_0)$ which was mentioned in [15]. For completion, we argue that the condition of (2.3.4) is valid in the deformation method, i. e., the computed Jacobian determinant of $\boldsymbol{\phi}$ in each t is always equal the prescribed $f(\boldsymbol{x},t)>0$. The proof is provided below.

Before proceeding to the proof, let's recall the **Liouville's formula** in differential equations: Let $A\in\mathbf{M}^{n\times n}$ be a matrix with continuous elements on an interval $I: a\le t\le b$ for some $a\le b\in\mathbb{R}$. Suppose $F(t)$ is a matrix of functions on I satisfying $\frac{d}{dt}(F(t)) = A(t)F(t)$. Then, $\frac{d}{dt}(\det F(t)) = \mathrm{Tr}(A(t))\det F(t)$.

Claim: If $\boldsymbol{\phi}$ is the solution of (2.3.1) to (2.3.2). Then

$$H(\boldsymbol{\xi},t) := \frac{\det\nabla(\boldsymbol{\phi}(\boldsymbol{\xi},t))}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)}$$

satisfies $\frac{\partial H}{\partial t}=0$, $\forall t\in[0,1]$.

Proof:

$$\begin{aligned}\frac{\partial H(\boldsymbol{\xi},t)}{\partial t} &= \frac{\partial}{\partial t}(\det\nabla(\boldsymbol{\phi}(\boldsymbol{\xi},t)))\frac{1}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)} + \det\nabla(\boldsymbol{\phi}(\boldsymbol{\xi},t))\frac{\partial}{\partial t}(\frac{1}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)}) \\ &= P(t) + Q(t)\end{aligned}$$

So, we have

$$\begin{aligned}P(t) =&\frac{\partial}{\partial t}(\det\nabla(\boldsymbol{\phi}(\boldsymbol{\xi},t)))\frac{1}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)} \\ =&\det\nabla(\frac{\partial}{\partial t}\boldsymbol{\phi}(\boldsymbol{\xi},t))\frac{1}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)} \\ =&\det\nabla(f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t))\frac{1}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)},\ \text{by (2.3.4)} \\ =&\det(\nabla_{\boldsymbol{\phi}}[f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)]\nabla\boldsymbol{\phi}(\boldsymbol{\xi},t))\frac{1}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)},\ \text{by chain rule}\end{aligned}$$

Taking $F(t) = \nabla\boldsymbol{\phi}(\boldsymbol{\xi},t)$, $A(t) = \nabla_{\boldsymbol{\phi}}[f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)]$ and $0 = a \le t \le b = 1$ in **Liouville′s formula**, then we get

$$\begin{aligned}P(t) =&\mathrm{Tr}(\nabla_{\boldsymbol{\phi}}[f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)])\det\nabla(\boldsymbol{\phi}(\boldsymbol{\xi},t))\frac{1}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)} \\ =&\mathrm{Tr}([\nabla_{\boldsymbol{\phi}}f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t) + f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\nabla_{\boldsymbol{\phi}}\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)])\frac{\det\nabla(\boldsymbol{\phi}(\boldsymbol{\xi},t))}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)} \\ =&[\nabla_{\boldsymbol{\phi}}f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t) + f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\nabla_{\boldsymbol{\phi}}\cdot\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)]\frac{\det\nabla(\boldsymbol{\phi}(\boldsymbol{\xi},t))}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)} \\ =&[\frac{\nabla_{\boldsymbol{\phi}}f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)} + \nabla_{\boldsymbol{\phi}}\cdot\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)]\det\nabla(\boldsymbol{\phi}(\boldsymbol{\xi},t))\ \text{by (2.3.4)} \\ =&[\frac{\nabla_{\boldsymbol{\phi}}f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)} + \frac{\partial}{\partial t}(\frac{-1}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)})]\det\nabla(\boldsymbol{\phi}(\boldsymbol{\xi},t))\ \text{by (2.3.3)} \\ =&[\frac{\nabla_{\boldsymbol{\phi}}f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)} + \frac{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)_t}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)^2}]\det\nabla(\boldsymbol{\phi}(\boldsymbol{\xi},t));\end{aligned}$$

and one the second term, we have

$$\begin{aligned}Q(t) =&\det\nabla(\boldsymbol{\phi}(\boldsymbol{\xi},t))\frac{\partial}{\partial t}(\frac{1}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)}) \\ =&-\frac{\nabla_{\boldsymbol{\phi}}f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\cdot\boldsymbol{\phi}(\boldsymbol{\xi},t)_t + f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)_t}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)^2}\det\nabla(\boldsymbol{\phi}(\boldsymbol{\xi},t)) \\ =&-[\frac{\nabla_{\boldsymbol{\phi}}f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)} + \frac{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)_t}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)^2}]\det\nabla(\boldsymbol{\phi}(\boldsymbol{\xi},t))\ \text{by (2.3.4)}\end{aligned}$$

Therefore,

$$\frac{\partial H(\boldsymbol{\xi},t)}{\partial t} = P(t) + Q(t) = 0 \text{ as desired}$$

and this completes the argument of our claim.

Next, it can be computed

$$H(t) = \int \frac{dH(t)}{dt}dt = \int \frac{\partial H(\boldsymbol{\xi},t)}{\partial t}dt = \int 0dt = C \in \mathbb{R}$$
$$H(t) = C = H(0) = \frac{\det\nabla(\boldsymbol{\phi}(\boldsymbol{\xi},0))}{f(\boldsymbol{\phi}(\boldsymbol{\xi},0),0)} = \frac{\det\nabla(\boldsymbol{\varphi}_0)}{f(\boldsymbol{x},0)} = 1 \text{ as desired.}$$

Therefore, $H(t) = 1$ is independent of $t \in [0,1]$ which implies the condition (2.3.2) is guaranteed.

Now, we modify Algorithm 1 with a local refinement technique. That is firstly to subdivide a given non-folding mesh (a given $\boldsymbol{\varphi}_0$ s.t. $\det\nabla(\boldsymbol{\varphi}_0(\boldsymbol{\xi'})) > 0$) or simply a resulting coarse mesh formed by Algorithm 1 to get an intermediate mesh; Second, use the deformation method to deform the newly-added boundary nodes of the intermediate mesh into a boundary conforming mesh; then, resulting mesh is refined and the non-folding property is preserved. The construction of a new diffeomorphism $\boldsymbol{\phi}(\boldsymbol{\xi},t)$ is needed as the problem from (2.3.1) to (2.3.2) indicates. That leads to the following algorithm of local refinement.

Algorithm 2 Deformation method with local refinements

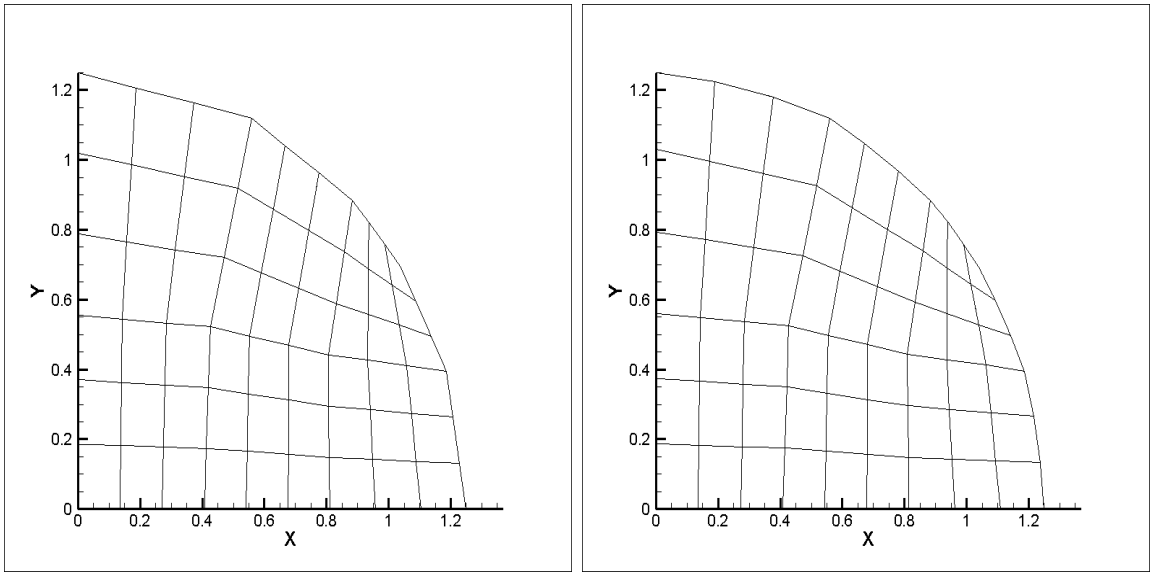
- 1: given a non-folding coarse mesh, Ω;
- 2: subdivide the coarse mesh up to desired order p, denoted Ω_0;
- 3: assign corresponding boundary conditions to the new nodes;
- 4: apply **Algorithm 1**;

In step 4 of Algorithm 2, the input n_t can be chosen relatively smaller, since the refinement happens only locally and the deformation is small, the larger number of n_t

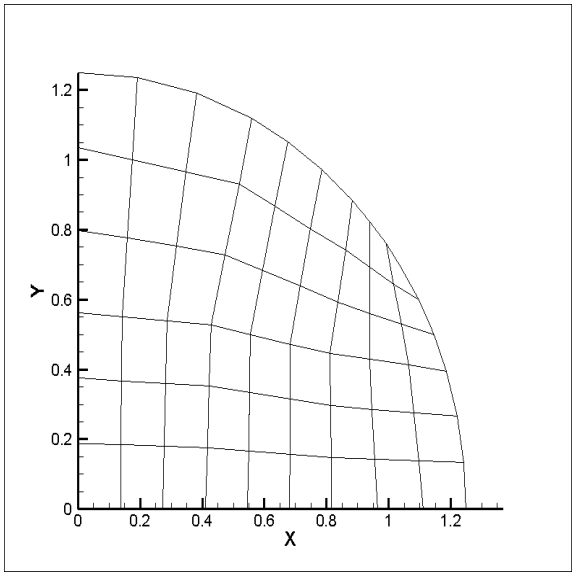
contributes lesser influence to the accuracy. This is confirmed with a simple example in later context.

In the next two Figures, in order to keep the coherence of our idea, we implemented the Algorithm 2 on resulting coarse meshes from Figure 2.2 and 2.3. In step 2 of Algorithm 2, as mentions in the beginning, p is the number of degree for interpolating polynomials and $p+1$ nodes are needed for interpolation on each dimension. So, we need to add $p-1$ new nodes in subdivision procedure. In these examples, $p=3$ is considered.

Once the new nodes had been added, the intermediate mesh is formed. Before the construction of boundary conforming mesh, we need to assign Dirichlet moving boundary conditions to the new nodes of the moving part and fixed-boundary conditions to old nodes of the moving part, because the old boundary nodes have already stand on the desired geometry. With appropriate boundary conditions, apply Algorithm 2 which is based on the deformation method to deform the intermediate mesh for a finer boundary conforming mesh Ω_t.

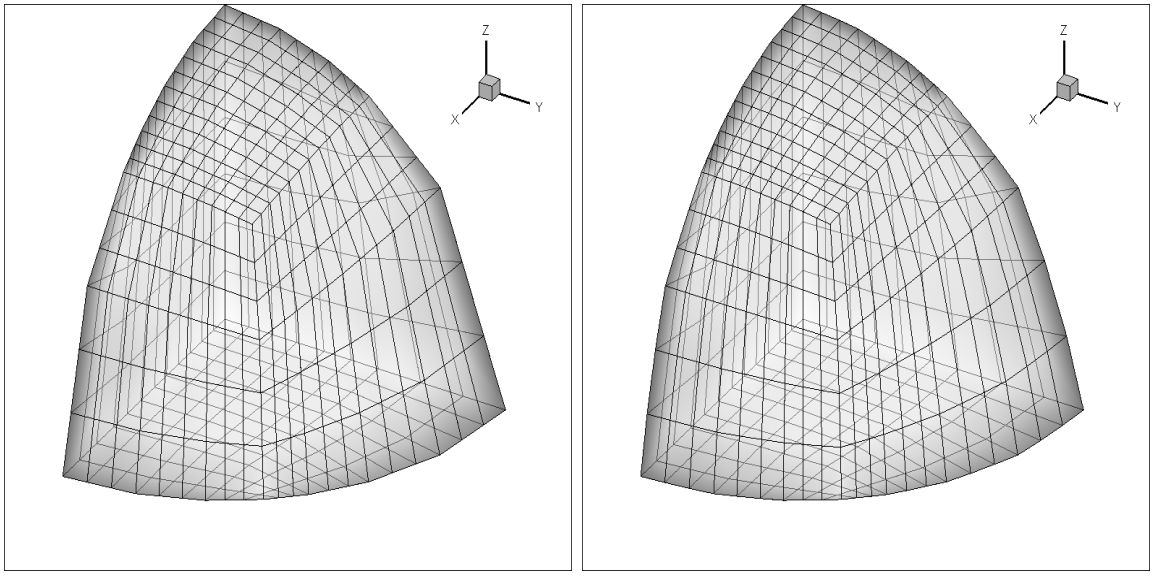


(a) t=0 (b) t=0.6

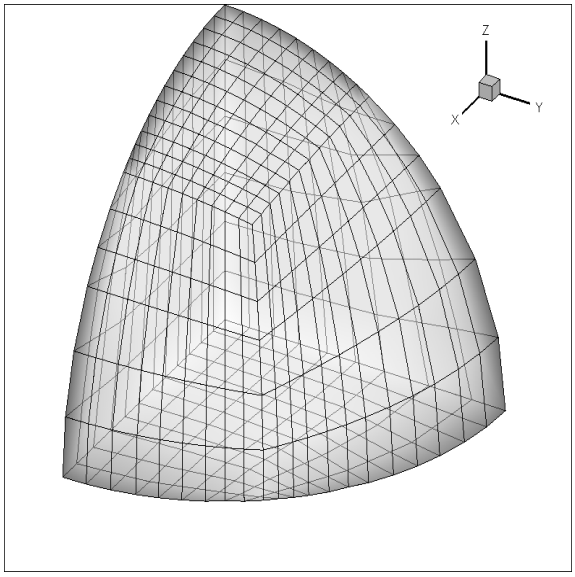


(c) t=1

Figure 2.10: Continued from Figure 2.2



(a) t=0 (b) t=0.6



(c) t=1

Figure 2.11: Continued from Figure 2.3

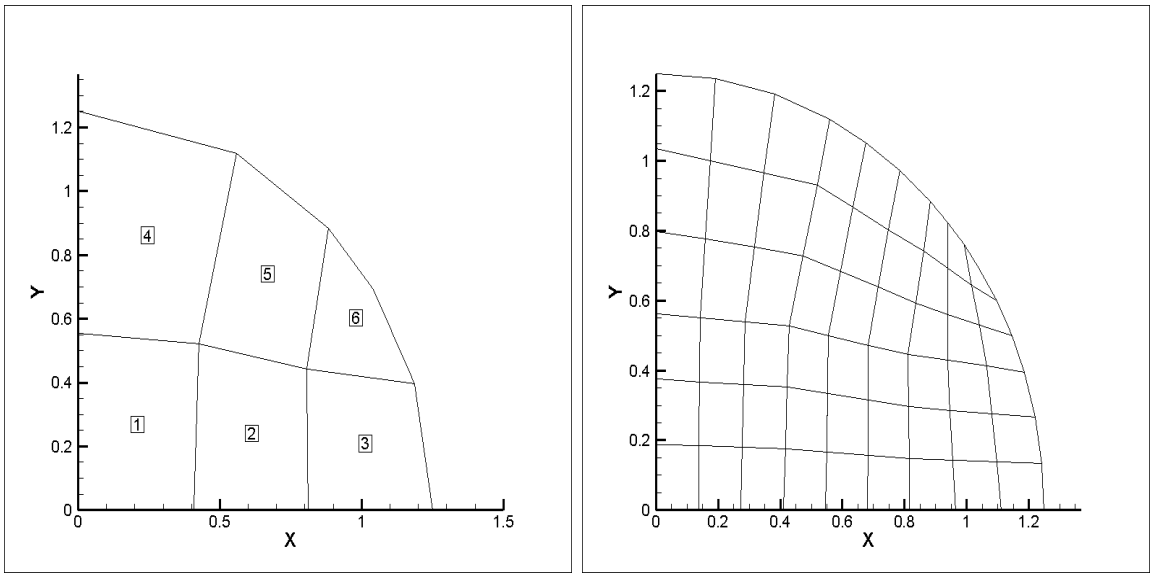
As the examples shown above, Algorithm 2 is able to construct a finer mesh representing domain. So, a boundary conforming finer mesh has been created. That is what motivates us to generate higher order element (HOE) meshes. Consider the following algorithm.

Algorithm 3 Mesh Interpolation for HOE

- 1: given a non-folding coarse mesh or simply apply **Algorithm 1** to get Ω;
- 2: apply **Algorithm 2** to Ω for a finer mesh Ω_t;
- 3: interpolate the finer mesh, Ω_t, with respect to the coarse mesh, Ω, up to desired order p;
- 4: plot the interpolated data based on the coarse mesh;

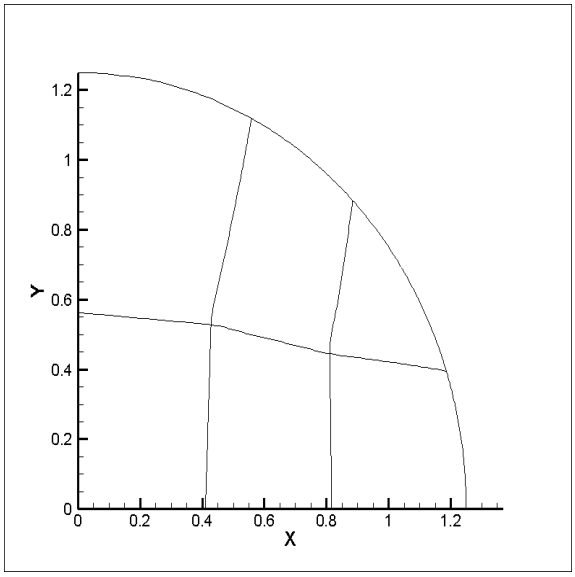
We combined with appropriate interpolation techniques, for instance, polynomials with degree $p = 3$, to define curves in 2D cases or surfaces in 3D case to represent the desired geometries. The simulated mesh may be different from the desired ge-

ometries, but with HOE representation, the higher order mesh is better than the original coarse (linear) ones. In the following HOE mesh examples shown in Figures, the MatLab interpolation package is used which computes with cubic-splines.



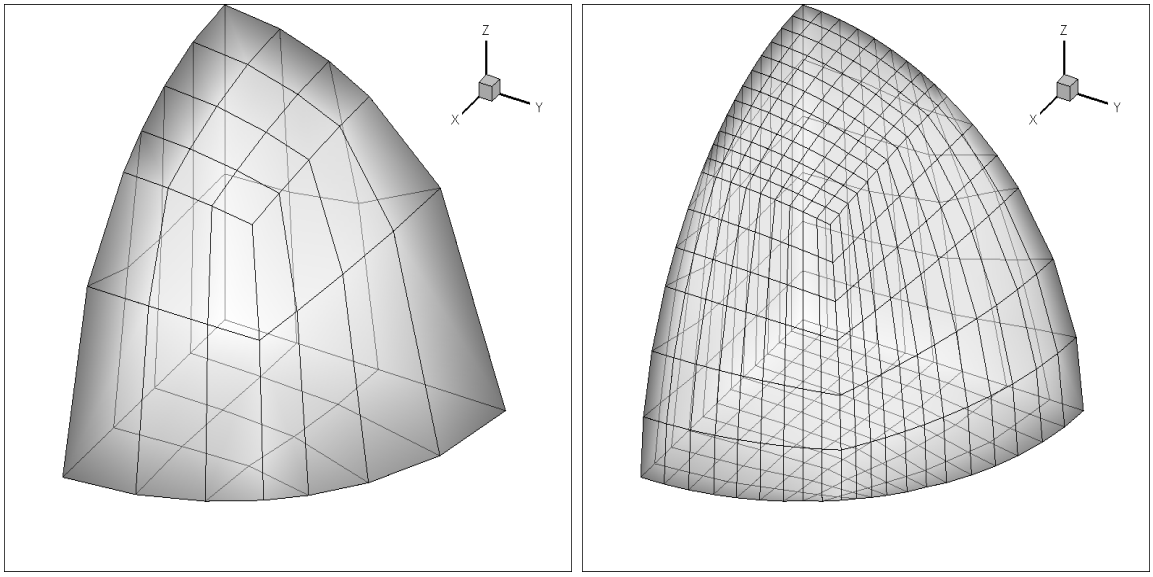
(a) coarse by Algorithm 1

(b) refined by Algorithm 2



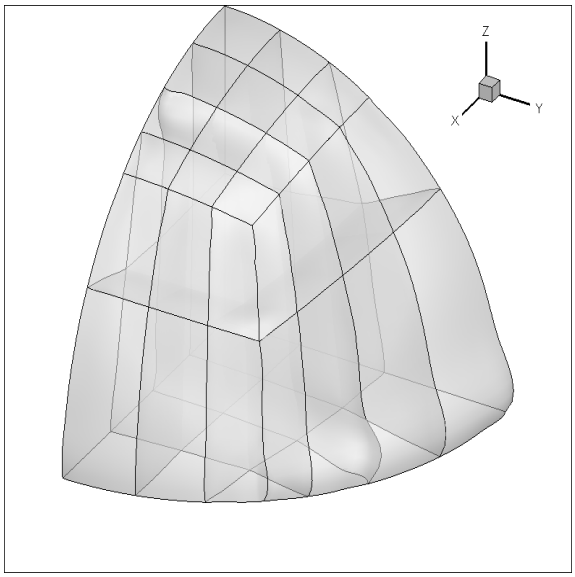
(c) HOE by Algorithm 3

Figure 2.12: Mesh evolution in 2D



(a) coarse by Algorithm 1

(b) refined by Algorithm 2

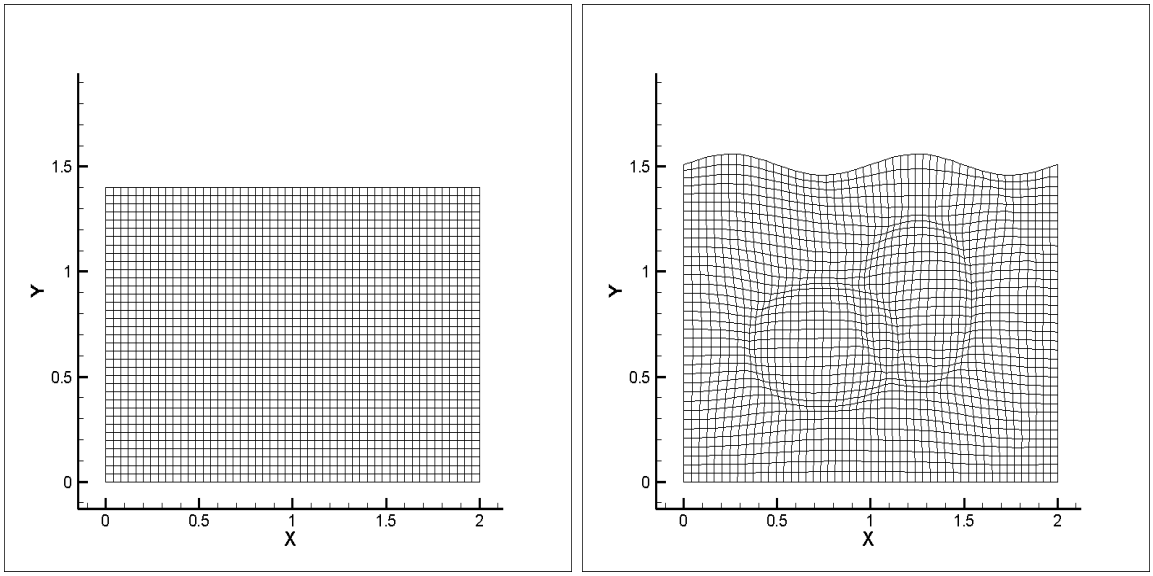


(c) HOE by Algorithm 3

Figure 2.13: Mesh evolution in 3D

In fact, the Algorithm 3 is an extension of Algorithm 1 and 2. It combines higher order polynomial interpolation and deformation method with LSFEM. The LSFEM is used for solving the **divergence** $-$ **curl** system, which is not a trivial task.

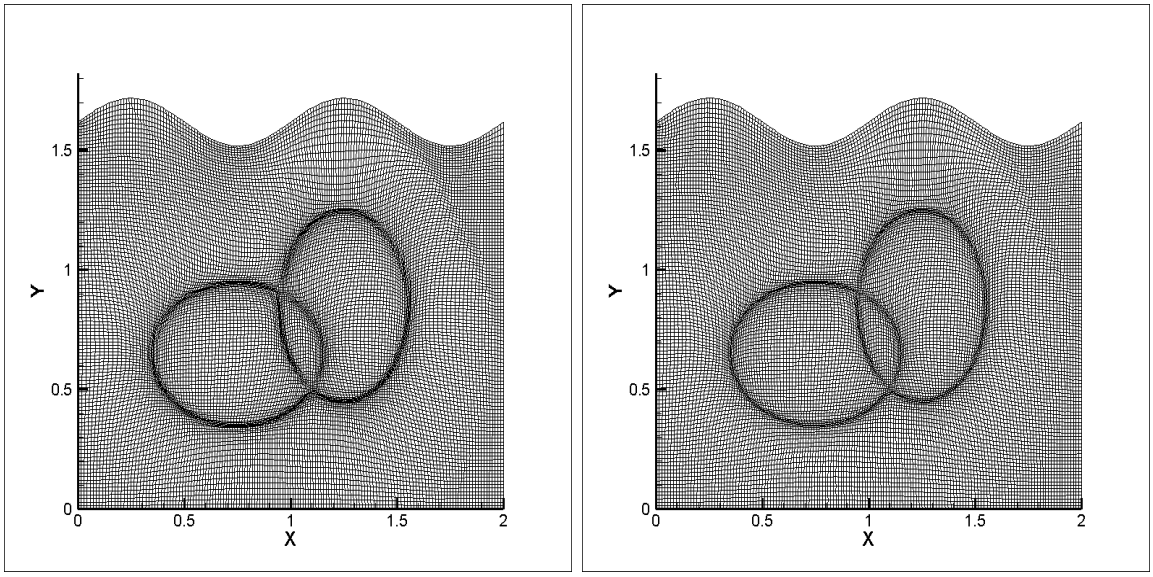
Furthermore, the implementation of LSFEM is the core asset of this approach to HOE meshes generation. One important and practical reason of choosing LSFEM is its flexibility in realizing of different boundary scenarios. In the next few examples, more sophisticated and complicated situations are handled by Algorithm 3. And they show the effectiveness of our approach in generating HOE meshes for most common geometries.



(a) t=0 (b) t=0.5

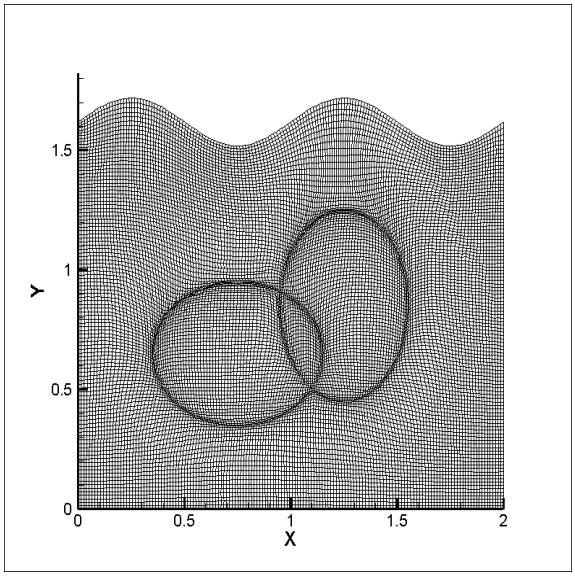
(c) t=0.8 (d) t=1

Figure 2.14: A wavy top rectangle with two emphasized ellipses by Algorithm 1



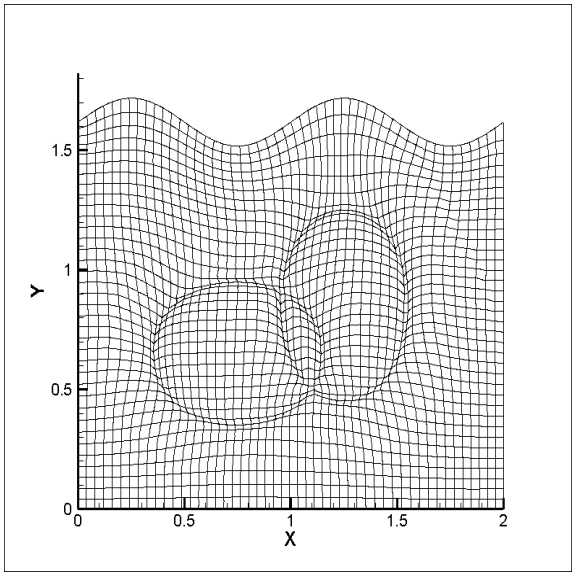
(a) t=0

(b) t=0.8



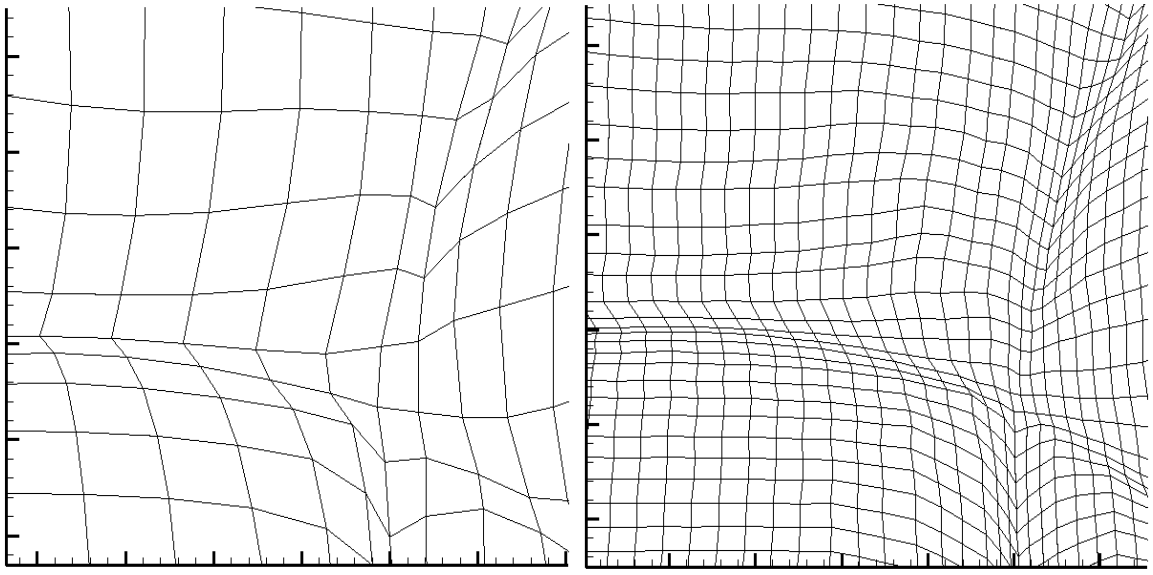
(c) t=1

Figure 2.15: Refinement by Algorithm 2



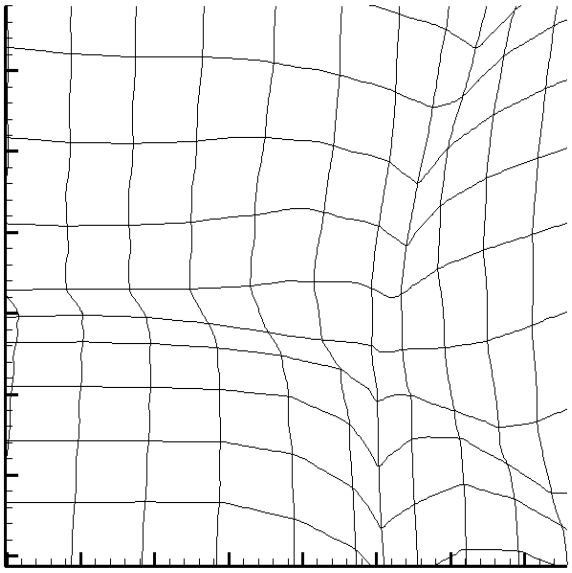
(a) HOE by Algorithm 3

Figure 2.16: HOE mesh representation by Algorithm 3



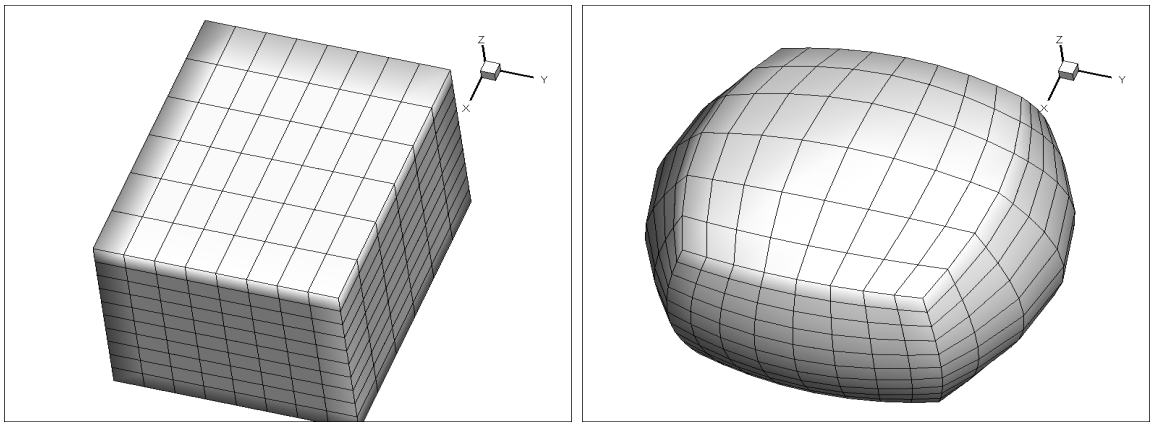
(a) coarse by Algorithm 1

(b) refined by Algorithm 2



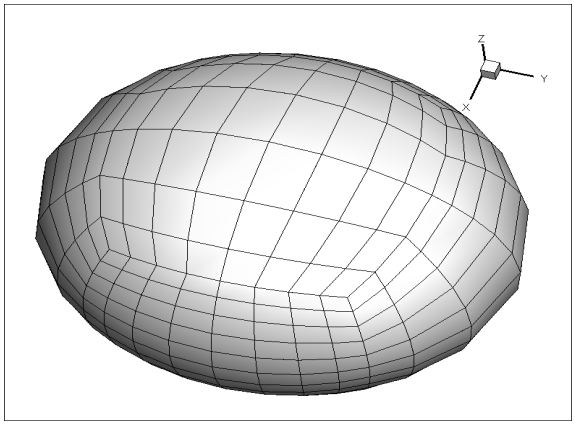
(c) HOE by Algorithm 3

Figure 2.17: Magnified views over intersection of the ellipses



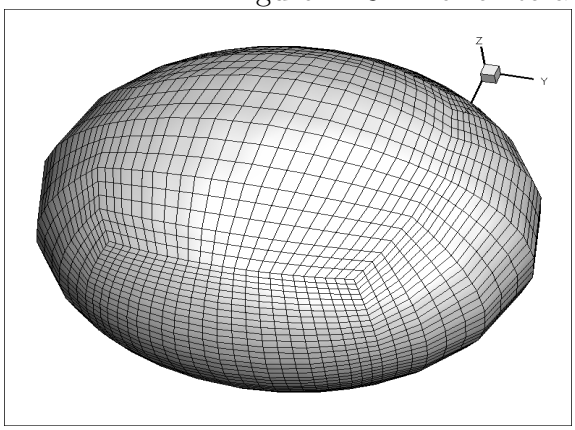
(a) t=0

(b) t=0.8

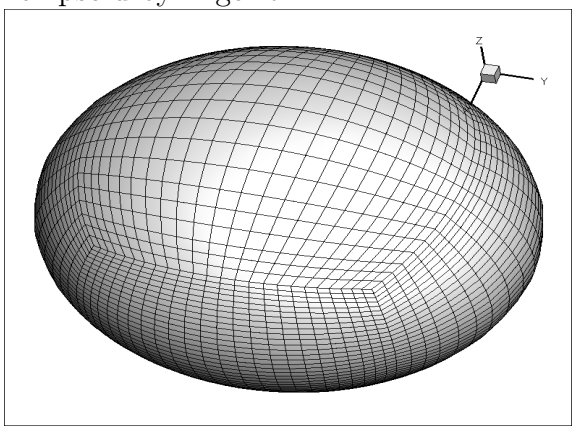


(c) t=1

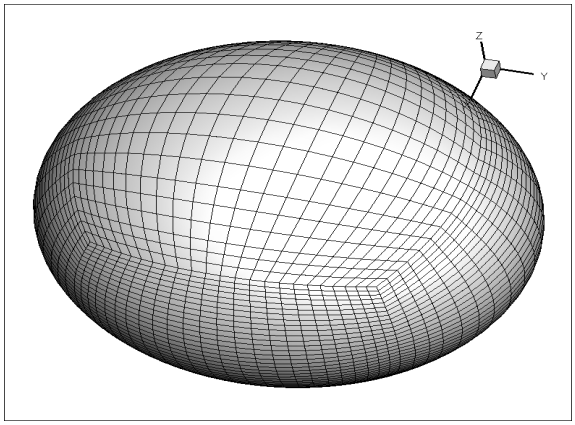
Figure 2.18: A brick to an ellipsoid by Algorithm 1



(a) t=0

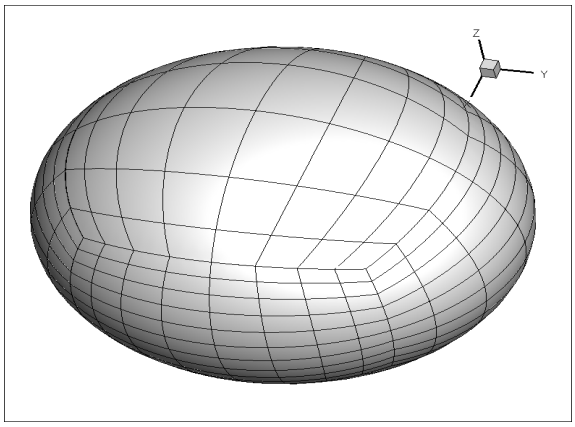


(b) t=0.8



(c) t=1

Figure 2.19: Refinement by Algorithm 2

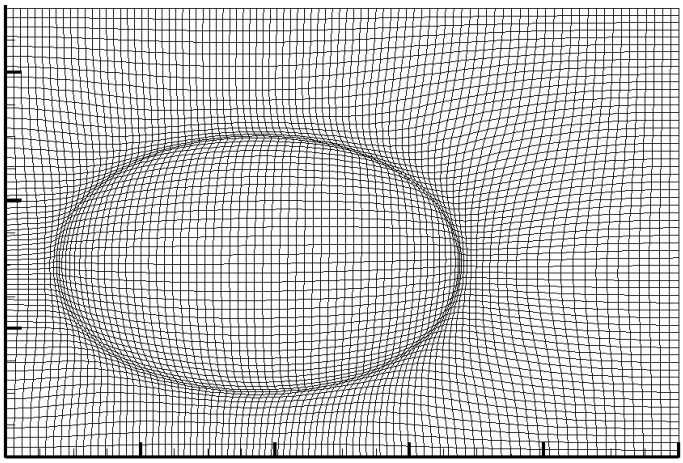


(a) HOE by Algorithm 3

Figure 2.20: HOE mesh representation by Algorithm 3

2.3.1 Comparison on 2D

This is an example of comparison with and without using the proposed Algorithm 3. The degree $p = 3$ is still considered. The following 2D example has number of finite elements $Nx * Ny = 91 * 61 = 5551$ on $\Omega_t = [0, 1] \times [0, 0.7]$ with $n_t = 10$.

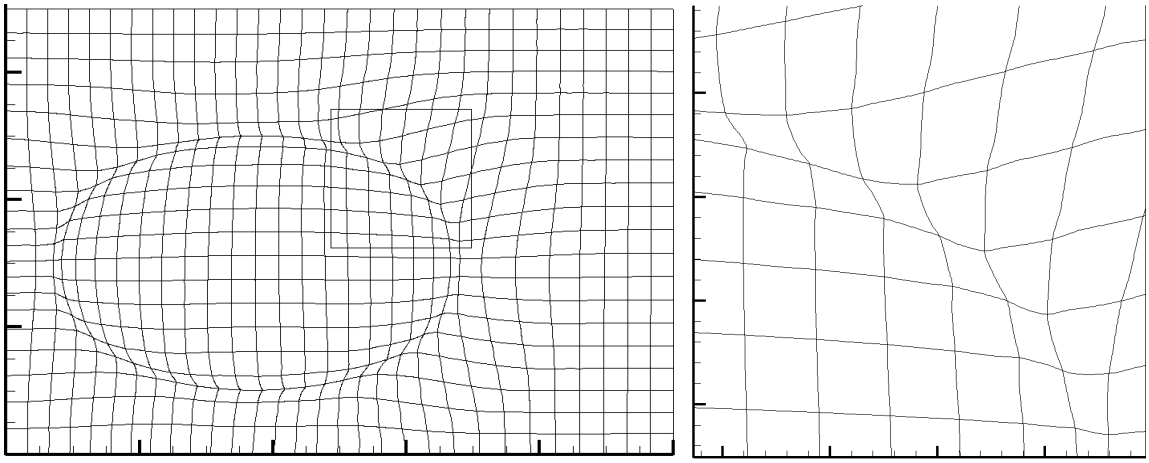


(a) HOE by Algorithm 3

Figure 2.21: A mesh by Algorithm 1

In the implementation of LSFEM, we need to solve a linear system $KE \in \mathbb{R}^{5551 \times 5551}$ for $n_t = 10$ times, which part occupies the most in computational costs. This simulation took about 45 seconds to complete without using our proposed HOE mesh approach.

To achieve the same level of fineness on HOE mesh, we apply the proposed Algorithm 3. The number of finite elements is $Nx * Ny = 31 * 21 = 1302$ over domain $\Omega_t = [0, 1] \times [0, 0.7]$ with $n_t = 10$ in step 1 and $Nx * Ny = [31 + 2(31 - 1)] * [21 + 2(21 - 1)] = 5551$ on $\Omega_t = [0, 1] \times [0, 0.7]$ with $n_t = 5$ in step 2 of Algorithm 3.



(a) HOE mesh by Algorithm 3 (b) Enlarged view

Figure 2.22: HOE mesh representation by Algorithm 3

Similarly, by LSFEM, there is a linear system $KE \in \mathbb{R}^{1302 \times 1302}$ needed to be solved for $n_t = 10$ times, then another system $KE \in \mathbb{R}^{5551 \times 5551}$ needed to be solved for $n_t = 5$ times in Algorithm 3. The total elapsed time of this simulation took about only 22 seconds. As the comparison demonstrates, not only the proposed Algorithm 3 generates a higher order element mesh but also generates it very efficiently compared to the original deformation method in linear mesh generation.

2.4 Conclusions and Future work

In this work, we formulated a novel approach to higher order element mesh generation. The main idea is to combine a local refinement technique with the deformation method on variable domains then using interpolation technique of higher degree polynomials. The new algorithm has a solid theoretical foundation. Moreover,

its numerical implementation of the Lease-Squares Finite Element Method is effective and efficient. The shown examples in this work are based on simple geometric models. Our goal is to develop a software package for real world problems in science and engineering.

CHAPTER 3

Determination of Diffeomorphisms based on Jacobian Determinant and Curl Vector

3.1 Introduction

The deformation method for mesh generation is discussed in the previous chapter, which constructs a diffeomorphism whose Jacobian determinant is monitored by a prescribed function $0 < f(\boldsymbol{x}) \in C^1(\boldsymbol{\Omega})$. The key component of the deformation method is to find the solution of **divergence − curl** system. However, the curl vector of the intermediate vector field $\boldsymbol{u}$, has been assigned to zero vector $\mathbf{0}$ (scalar 0 in case of 2D) due to its difficulty in implementation. So, a diffeomorphism constructed by the deformation method is not generally unique. To uniquely generate a mesh or determine a transformation, as studied in [36, 37], the information of the curl vector should not be ignored. Motivated by the limitation, a novel variational mesh generation method was developed in [15, 13]. The goal is to find a $\boldsymbol{\phi}$ on $\boldsymbol{\Omega} \subset \mathbb{R}^{2,3}$ which satisfies the conditions:

$$\begin{cases} \det\nabla(\boldsymbol{\phi}) = f_0 > 0 \\ \nabla \times (\boldsymbol{\phi}) = \boldsymbol{g}_0, \text{ where } \nabla \cdot \boldsymbol{g}_0 = 0 \end{cases}$$

This meshing approach is designed to control both Jacobian determinant and the curl vector of a transformation in an optimization scheme. The studies turned out quite successful in case of 2D earlier. In this chapter, the case of 3D is implemented. Based on its excellent performance on 2D scenario, it considered and that the Jacobian determinant cannot uniquely determine a transformation. In [15], a special case has for the uniqueness problem was analytically discussed in 2D. In this chapter, we will generalize the study to 3D scenario. The method will be reviewed in section 3.2; some

numerical examples are shown to explain how the uniqueness problem was formed in section 3.3; the special case of the uniqueness problem in 3D case is proved in section 3.4; lastly, in section 3.5, motivated by the uniqueness problem, a direct strategy is formed and numerically tested.

3.2 The Variational Method for Mesh Generation

Let bounded $\Omega \subset \mathbb{R}^3$ (It's similar for $\mathbb{R}^2$) be the domain and a scalar function on $f_0(\boldsymbol{x}) > 0$ and a vector-valued $\boldsymbol{g}_0(\boldsymbol{x})$ on the domain Ω with

$$\int_\Omega f_0(\boldsymbol{x})d\boldsymbol{x} = |\Omega| \text{ and } \nabla \cdot \boldsymbol{g}_0(\boldsymbol{x}) = 0. \tag{3.2.1}$$

We look for a diffeomorphism $\boldsymbol{\phi} : \Omega \to \Omega$

$$\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{\phi}_0(\boldsymbol{x}) + \boldsymbol{u}(\boldsymbol{x}) \tag{3.2.2}$$

that minimizes the cost functional — sum of squared difference:

$$SSD(\boldsymbol{\phi}(\boldsymbol{x})) = \frac{1}{2}\int_\Omega [(\det\nabla(\boldsymbol{\phi}(\boldsymbol{x})) - f_0(\boldsymbol{x}))^2 + |\nabla \times (\boldsymbol{\phi}(\boldsymbol{x})) - \boldsymbol{g}_0(\boldsymbol{x})|^2]d\boldsymbol{x} \tag{3.2.3}$$

subject to the constraints of control functions $f(\boldsymbol{x})$ and $\boldsymbol{g}(\boldsymbol{x})$

$$\begin{cases} \nabla \cdot \boldsymbol{u}(\boldsymbol{x}) = f(\boldsymbol{x}) - 1 \\ \nabla \times \boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{x}) \\ \boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{0} \qquad \text{on } \partial\Omega \end{cases} \text{in } \Omega \Rightarrow \begin{cases} \Delta u_1 = f(\boldsymbol{x})_{x_1} - g_{3x_2}(\boldsymbol{x}) + g_{2x_2}(\boldsymbol{x}) = F_1(\boldsymbol{x}) \\ \Delta u_2 = f(\boldsymbol{x})_{x_2} + g_{3x_1}(\boldsymbol{x}) - g_{1x_3}(\boldsymbol{x}) = F_2(\boldsymbol{x}) \\ \Delta u_3 = f(\boldsymbol{x})_{x_3} - g_{2x_1}(\boldsymbol{x}) + g_{1x_2}(\boldsymbol{x}) = F_3(\boldsymbol{x}) \end{cases}$$

$$\begin{cases} \Delta\boldsymbol{u}(\boldsymbol{x}) = \nabla f(\boldsymbol{x}) - \nabla \times \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{F}(\boldsymbol{x}) \qquad \text{in } \Omega \\ \boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{0} \qquad \text{on } \partial\Omega. \end{cases} \tag{3.2.4}$$

To find a solution to (3.2.3), the first variational gradient is needed. In order to implement a gradient descent numerical scheme, since the (3.2.3) subjects to the

Poisson equation (3.2.4), so the desired first variational gradient should be with respect to $\boldsymbol{F}(\boldsymbol{x})$ and was originally formulated in [15]. For completion, its derivation in case of 3D is summarized here.

Claim: Let $P = \det\nabla(\boldsymbol{\phi}) - f_0$ and $\boldsymbol{Q} = \nabla \times (\boldsymbol{\phi}) - \boldsymbol{g}_0$. Consider b_i's that satisfy $-\nabla b_i = \boldsymbol{a}_i$ where

$$\begin{cases} \boldsymbol{a}_1 = P \begin{pmatrix} \phi_{2x_2}\phi_{3x_3} - \phi_{3x_2}\phi_{2x_3} \\ \phi_{3x_1}\phi_{2x_3} - \phi_{2x_1}\phi_{3x_3} \\ \phi_{2x_1}\phi_{3x_2} - \phi_{2x_2}\phi_{3x_1} \end{pmatrix} + \begin{pmatrix} 0 \\ -Q_3 \\ Q_2 \end{pmatrix} \\ \boldsymbol{a}_2 = P \begin{pmatrix} \phi_{3x_2}\phi_{1x_3} - \phi_{1x_2}\phi_{3x_3} \\ \phi_{1x_1}\phi_{3x_3} - \phi_{1x_3}\phi_{3x_1} \\ \phi_{3x_1}\phi_{1x_2} - \phi_{1x_1}\phi_{3x_2} \end{pmatrix} + \begin{pmatrix} Q_3 \\ 0 \\ -Q_1 \end{pmatrix} \\ \boldsymbol{a}_3 = P \begin{pmatrix} \phi_{1x_2}\phi_{2x_3} - \phi_{2x_2}\phi_{1x_3} \\ \phi_{2x_1}\phi_{1x_3} - \phi_{1x_1}\phi_{2x_3} \\ \phi_{1x_1}\phi_{2x_2} - \phi_{2x_1}\phi_{1x_2} \end{pmatrix} + \begin{pmatrix} -Q_2 \\ Q_1 \\ 0 \end{pmatrix} \end{cases}.$$

Then, the first variational gradient of $SSD(\boldsymbol{\phi})$ with respect to $\boldsymbol{F}$ is

$$\frac{\partial SSD(\boldsymbol{\phi})}{\partial \boldsymbol{F}} = \boldsymbol{b} = (b_1, b_2, b_3).$$

Proof: From (3.2.3), we have

$$\begin{aligned}
\delta SSD(\boldsymbol{\phi}) &= \delta\frac{1}{2}\int_{\Omega}[(\det\nabla(\boldsymbol{\phi}) - f_0)^2 + |\nabla\times(\boldsymbol{\phi}) - \boldsymbol{g}_0|^2] \\
&= \int_{\Omega}[(\det\nabla(\boldsymbol{\phi}) - f_0)\delta(\det\nabla\boldsymbol{\phi}) + (\nabla\times(\boldsymbol{\phi}) - \boldsymbol{g}_0)\cdot\delta(\nabla\times\boldsymbol{\phi})] \\
&= \int_{\Omega}[(\det\nabla(\boldsymbol{\phi}) - f_0)\delta\det\begin{pmatrix}\phi_{1x_1} & \phi_{1x_2} & \phi_{1x_3}\\ \phi_{2x_1} & \phi_{2x_2} & \phi_{2x_3}\\ \phi_{3x_1} & \phi_{3x_2} & \phi_{3x_3}\end{pmatrix} + (\nabla\times(\boldsymbol{\phi}) - \boldsymbol{g}_0)\cdot\delta\begin{pmatrix}\phi_{3x_2} - \phi_{2x_3}\\ \phi_{1x_3} - \phi_{3x_1}\\ \phi_{2x_1} - \phi_{1x_2}\end{pmatrix}] \\
&= \int_{\Omega}[P\delta\det\begin{pmatrix}\phi_{1x_1} & \phi_{1x_2} & \phi_{1x_3}\\ \phi_{2x_1} & \phi_{2x_2} & \phi_{2x_3}\\ \phi_{3x_1} & \phi_{3x_2} & \phi_{3x_3}\end{pmatrix} + \boldsymbol{Q}\cdot\delta\begin{pmatrix}\phi_{3x_2} - \phi_{2x_3}\\ \phi_{1x_3} - \phi_{3x_1}\\ \phi_{2x_1} - \phi_{1x_2}\end{pmatrix}].
\end{aligned}$$

Here, $P = \det\nabla(\boldsymbol{\phi}) - f_0$ and $\boldsymbol{Q} = \nabla\times(\boldsymbol{\phi}) - \boldsymbol{g}_0$ are denoted, then we have

$$\begin{aligned}
\delta SSD(\boldsymbol{\phi}) = \int_{\Omega}[&P\delta(\phi_{1x_1}(\phi_{2x_2}\phi_{3x_3} - \phi_{2x_3}\phi_{3x_2}) - \phi_{1x_2}(\phi_{3x_3}\phi_{2x_1} - \phi_{3x_1}\phi_{2x_3}) \\
&+ \phi_{1x_3}(\phi_{2x_1}\phi_{3x_2} - \phi_{2x_2}\phi_{3x_1})) \\
&+ (Q_1, Q_2, Q_3)\cdot\delta(\phi_{3x_2} - \phi_{2x_3}, \phi_{1x_3} - \phi_{3x_1}, \phi_{2x_1} - \phi_{1x_2})] \\
= \int_{\Omega}[&P(\delta(\phi_{1x_1}\phi_{2x_2}\phi_{3x_3} - \phi_{1x_1}\phi_{2x_3}\phi_{3x_2}) - \delta(\phi_{1x_2}\phi_{3x_3}\phi_{2x_1} - \phi_{1x_2}\phi_{3x_1}\phi_{2x_3}) \\
&+ \delta(\phi_{1x_3}\phi_{2x_1}\phi_{3x_2} - \phi_{1x_3}\phi_{2x_2}\phi_{3x_1})) \\
&+ (Q_1, Q_2, Q_3)\cdot(\delta\phi_{3x_2} - \delta\phi_{2x_3}, \delta\phi_{1x_3} - \delta\phi_{3x_1}, \delta\phi_{2x_1} - \delta\phi_{1x_2})] \\
= \int_{\Omega}[&P(\delta\phi_{1x_1}\phi_{2x_2}\phi_{3x_3} + \phi_{1x_1}\delta\phi_{2x_2}\phi_{3x_3} + \phi_{1x_1}\phi_{2x_2}\delta\phi_{3x_3} \\
&- \delta\phi_{1x_1}\phi_{2x_3}\phi_{3x_2} - \phi_{1x_1}\delta\phi_{2x_3}\phi_{3x_2} - \phi_{1x_1}\phi_{2x_3}\delta\phi_{3x_2} \\
&- \delta\phi_{1x_2}\phi_{3x_3}\phi_{2x_1} - \phi_{1x_2}\delta\phi_{3x_3}\phi_{2x_1} - \phi_{1x_2}\phi_{3x_3}\delta\phi_{2x_1} \\
&+ \delta\phi_{1x_2}\phi_{3x_1}\phi_{2x_3} + \phi_{1x_2}\delta\phi_{3x_1}\phi_{2x_3} + \phi_{1x_2}\phi_{3x_1}\delta\phi_{2x_3}) \\
&+ \delta\phi_{1x_3}\phi_{2x_1}\phi_{3x_2} + \phi_{1x_3}\delta\phi_{2x_1}\phi_{3x_2} + \phi_{1x_3}\phi_{2x_1}\delta\phi_{3x_2} \\
&- \delta\phi_{1x_3}\phi_{2x_2}\phi_{3x_1} - \phi_{1x_3}\delta\phi_{2x_2}\phi_{3x_1} - \phi_{1x_3}\phi_{2x_2}\delta\phi_{3x_1}) \\
&+ (Q_1, Q_2, Q_3)\cdot(\delta\phi_{3x_2} - \delta\phi_{2x_3}, \delta\phi_{1x_3} - \delta\phi_{3x_1}, \delta\phi_{2x_1} - \delta\phi_{1x_2})].
\end{aligned}$$

Since (3.2.2), we have $\delta\boldsymbol{\phi} = \delta\boldsymbol{u}$. Then, we have

$$\begin{aligned}
\delta SSD(\boldsymbol{\phi}) = \int_\Omega [P(&\delta u_{1x_1}\phi_{2x_2}\phi_{3x_3} + \phi_{1x_1}\delta u_{2x_2}\phi_{3x_3} + \phi_{1x_1}\phi_{2x_2}\delta u_{3x_3} \\
&- \delta u_{1x_1}\phi_{2x_3}\phi_{3x_2} - \phi_{1x_1}\delta u_{2x_3}\phi_{3x_2} - \phi_{1x_1}\phi_{2x_3}\delta u_{3x_2} \\
&- \delta u_{1x_2}\phi_{3x_3}\phi_{2x_1} - \phi_{1x_2}\delta u_{3x_3}\phi_{2x_1} - \phi_{1x_2}\phi_{3x_3}\delta u_{2x_1} \\
&+ \delta u_{1x_2}\phi_{3x_1}\phi_{2x_3} + \phi_{1x_2}\delta u_{3x_1}\phi_{2x_3} + \phi_{1x_2}\phi_{3x_1}\delta u_{2x_3} \\
&+ \delta u_{1x_3}\phi_{2x_1}\phi_{3x_2} + \phi_{1x_3}\delta u_{2x_1}\phi_{3x_2} + \phi_{1x_3}\phi_{2x_1}\delta u_{3x_2} \\
&- \delta u_{1x_3}\phi_{2x_2}\phi_{3x_1} - \phi_{1x_3}\delta u_{2x_2}\phi_{3x_1} - \phi_{1x_3}\phi_{2x_2}\delta u_{3x_1}) \\
&+ (Q_1, Q_2, Q_3)\cdot(\delta u_{3x_2} - \delta u_{2x_3}, \delta u_{1x_3} - \delta u_{3x_1}, \delta u_{2x_1} - \delta u_{1x_2})] \\
= \int_\Omega ([P &\begin{pmatrix} \phi_{2x_2}\phi_{3x_3} - \phi_{3x_2}\phi_{2x_3} \\ \phi_{3x_1}\phi_{2x_3} - \phi_{2x_1}\phi_{3x_3} \\ \phi_{2x_1}\phi_{3x_2} - \phi_{2x_2}\phi_{3x_1} \end{pmatrix} + \begin{pmatrix} 0 \\ -Q_3 \\ Q_2 \end{pmatrix}] \cdot \nabla\delta u_1 \\
&+ [P \begin{pmatrix} \phi_{3x_2}\phi_{1x_3} - \phi_{1x_2}\phi_{3x_3} \\ \phi_{1x_1}\phi_{3x_3} - \phi_{1x_3}\phi_{3x_1} \\ \phi_{3x_1}\phi_{1x_2} - \phi_{1x_1}\phi_{3x_2} \end{pmatrix} + \begin{pmatrix} Q_3 \\ 0 \\ -Q_1 \end{pmatrix}] \cdot \nabla\delta u_2 \\
&+ [P \begin{pmatrix} \phi_{1x_2}\phi_{2x_3} - \phi_{2x_2}\phi_{1x_3} \\ \phi_{2x_1}\phi_{1x_3} - \phi_{1x_1}\phi_{2x_3} \\ \phi_{1x_1}\phi_{2x_2} - \phi_{2x_1}\phi_{1x_2} \end{pmatrix} + \begin{pmatrix} -Q_2 \\ Q_1 \\ 0 \end{pmatrix}] \cdot \nabla\delta u_3).
\end{aligned}$$

Here, let's recall the *Green's* formulas in vector calculus: Let f, $g \in C^2(\Omega)$ for some compact $\Omega \subset \mathbb{R}^3$. Then,

(i) $\int_\Omega (g\Delta f + \nabla g \cdot \nabla f) d\boldsymbol{\omega} = \int_{\partial\Omega} \frac{\partial f}{\partial g} f d\boldsymbol{s}$;

(ii) $\int_\Omega (g\Delta f - \Delta g f) d\boldsymbol{\omega} = \int_{\partial\Omega} (\frac{\partial g}{\partial f} g - \frac{\partial f}{\partial g} f) d\boldsymbol{s}$.

The "tall vector"s from previous step are denoted as $\boldsymbol{a}_i$. Let's consider some functions b_i's satisfy $-\nabla b_i = \boldsymbol{a}_i$, then it can be continued by the *Green*'s formulas (in fixed boundary case, contour integrals of (i-ii) are 0) as follows,

$$\begin{aligned}
\delta SSD(\boldsymbol{\phi}) &= \int_\Omega [\boldsymbol{a}_1 \cdot \nabla \delta u_1 + \boldsymbol{a}_2 \cdot \nabla \delta u_2 + \boldsymbol{a}_3 \cdot \nabla \delta u_3] \\
&= \int_\Omega (-\nabla b_1 \cdot \nabla \delta u_1 - \nabla b_2 \cdot \nabla \delta u_2 - \nabla b_3 \cdot \nabla \delta u_3) \\
&= \int_\Omega (\nabla \cdot \nabla b_1 \delta u_1 + \nabla \cdot \nabla b_2 \delta u_2 + \nabla \cdot \nabla b_3 \delta u_3) \\
&= \int_\Omega (\Delta b_1 \delta u_1 + \Delta b_2 \delta u_2 + \Delta b_3 \delta u_3) \\
&= \int_\Omega (b_1 \delta \Delta u_1 + b_2 \delta \Delta u_2 + b_3 \delta \Delta u_3) \\
&= \int_\Omega (b_1 \delta F_1 + b_2 F_2 + b_3 F_3) \\
&= \int_\Omega \boldsymbol{b} \cdot \delta \boldsymbol{F}
\end{aligned}$$

Therefore, the first variational gradient with respect to the control function $\boldsymbol{F}$ is of the form:

$$\frac{\partial SSD(\boldsymbol{\phi})}{\partial \boldsymbol{F}} = \boldsymbol{b} = (b_1, b_2, b_3)$$

where b_i's satisfy $\Delta b_i = -\nabla \cdot \boldsymbol{a}_i$.

According to the first variational gradient (3.2) above, a gradient descent numerical scheme can be implemented by the following algorithm where $Ratio := \frac{SSD_{new}}{SSD_{init}}$(usually in practices, $0 < Ratio \leq 1$) and FFT is the Fast Fourier Transform based *Poisson* solver.

Algorithm 4 Variational Method

- 0: input f_0, $\boldsymbol{g}_0$;
- 1: initialize:

  $\boldsymbol{F}_{k-1} = \boldsymbol{0}$, $\boldsymbol{b}_{k-1} = \boldsymbol{0}$, $\boldsymbol{\phi} = \boldsymbol{x}$, set Δt, t_{tol}, $ratio$, $ratio_{tol}$, $iter_{max}$;
- 2: while $\Delta t > t_{tol}$ and $ratio > ratio_{tol}$ and $iter > iter_{max}$;
  - 3: if $better = true$
    - 4: compute $\boldsymbol{b}_k$ by $\Delta b_i = -\nabla \cdot \boldsymbol{a}_i$ (solve $Poisson$ by FFT);
  - 5: update $\boldsymbol{F}_k = \boldsymbol{F}_{k-1} - \Delta t * \boldsymbol{b}_k$;
  - 6: solve $\Delta \boldsymbol{u}_k = \boldsymbol{F}_k$ (solve $Poisson$ by FFT);
  - 7: update $\boldsymbol{\phi} = \boldsymbol{x} + \boldsymbol{u}_k$;
  - 8: check SSD and compute $ratio$;
  - 9: if SSD decrease,
    - 10:$better = true$;
    - 11:$\Delta t = \Delta t * t_{up}$;
    - 12:$\boldsymbol{F}_{k-1} = \boldsymbol{F}_k$;

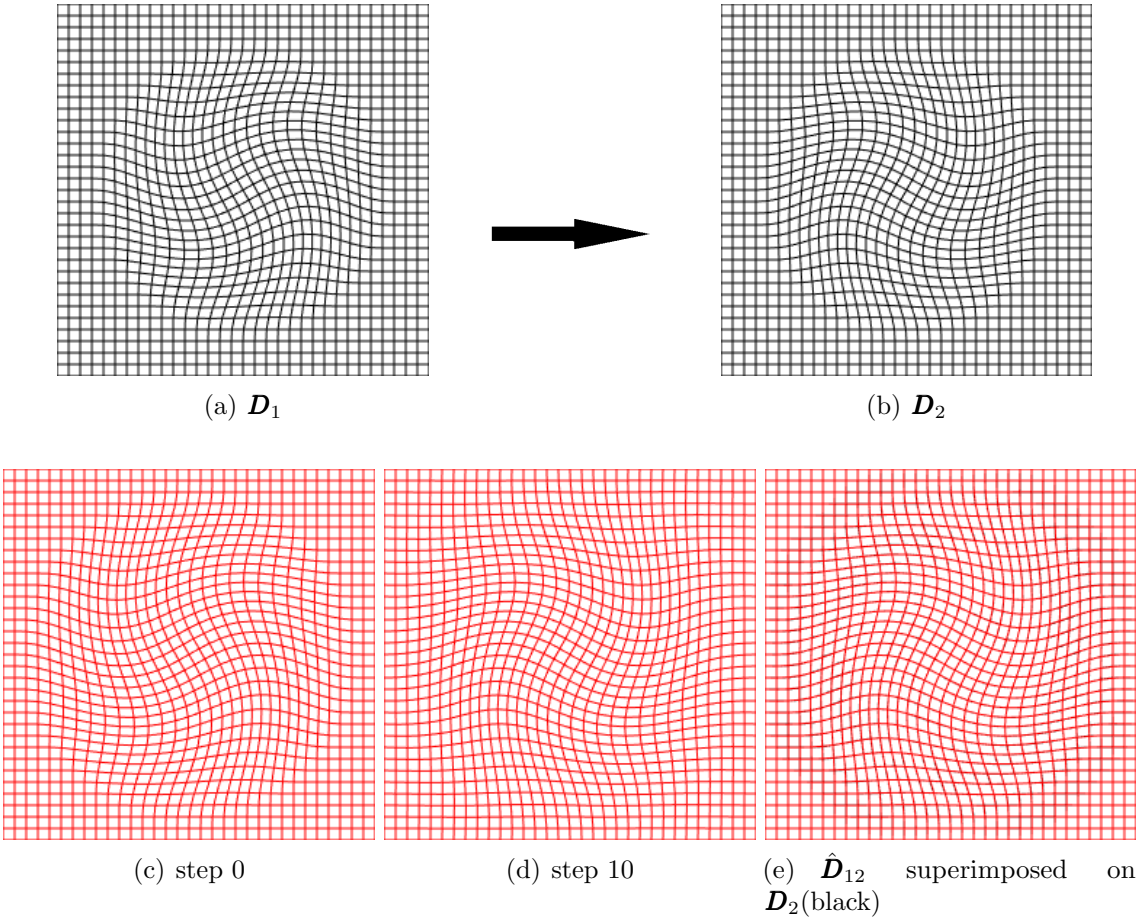
    else
    - 13:$better = false$;
    - 14:$\Delta t = \Delta t * t_{down}$;

    go to 2;
- 15: output $\boldsymbol{\phi}$.

Here, we include a simple example to show the effectiveness of the variational method in 2D.

Example 1: Variational Method in 2D

Deform $\boldsymbol{D}_1$ to $\boldsymbol{D}_2$ by the Variational Method. Taking $f_0 = \det\nabla(\boldsymbol{D}_2)$ and $g_0 = \nabla \times (\boldsymbol{D}_2)$, $\boldsymbol{D}_1$ is deformed to a transformation $\hat{\boldsymbol{D}}_{12}$, as shown in (e) of next Figure that is almost identical to $\boldsymbol{D}_2$ by prescribing its Jacobian as $\det\nabla(\boldsymbol{D}_2)$ and its curl as $\nabla \times (\boldsymbol{D}_2)$. Total elapsed time for 25 iteration steps is 0.955899 seconds.



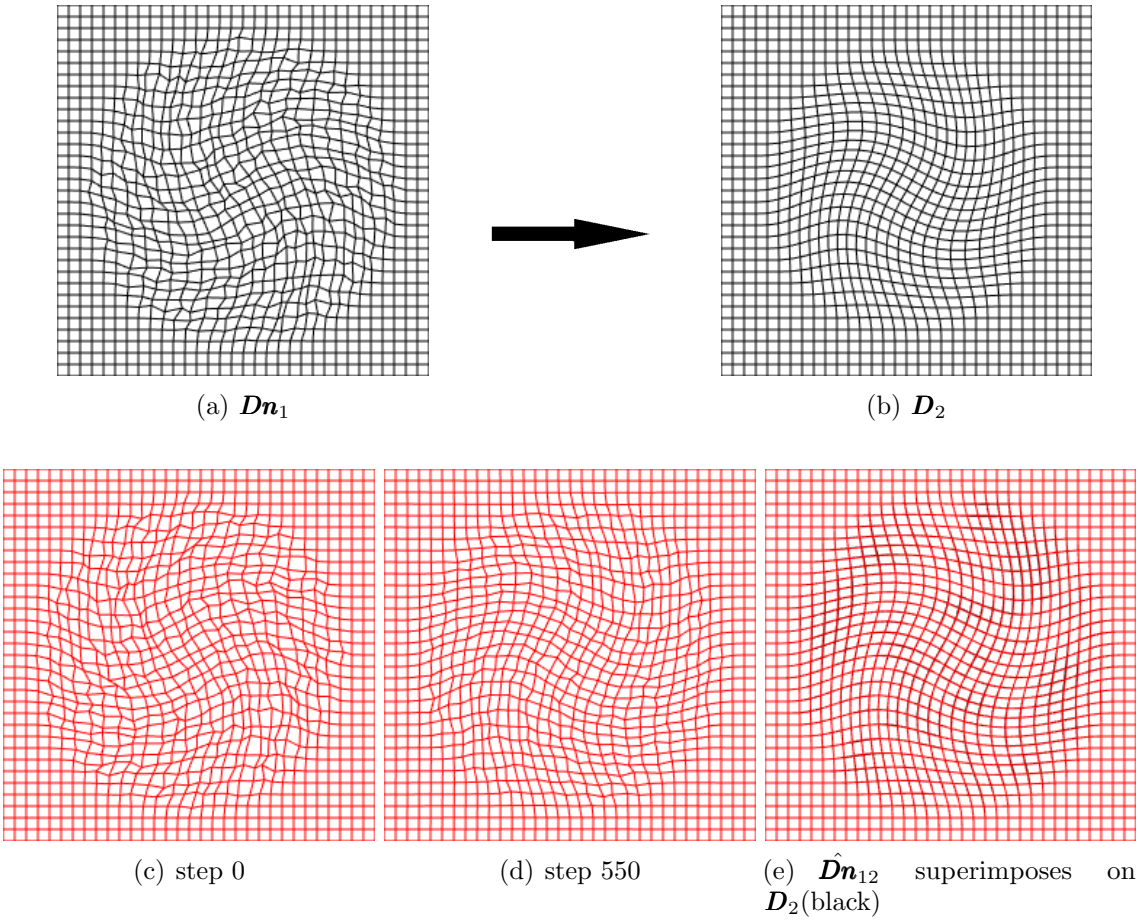
(a) $\boldsymbol{D}_1$ (b) $\boldsymbol{D}_2$

(c) step 0 (d) step 10 (e) $\hat{\boldsymbol{D}}_{12}$ superimposed on $\boldsymbol{D}_2$(black)

Figure 3.1: $\boldsymbol{D}_1$ to $\boldsymbol{D}_2$

We next add noise to $\boldsymbol{D}_1$ and deform the distorted $\boldsymbol{D}_1$, denoted as $\boldsymbol{Dn}_1$, to $\boldsymbol{D}_2$ as before. The calculated transformation $\hat{\boldsymbol{Dn}}_{12}$ (in red lines) by is superimposed on

$\boldsymbol{D}_2$ as it shows in (e) of next Figure. The red lines are almost identical to the black lines. This indicates that the results of the Variational method are very accurate.



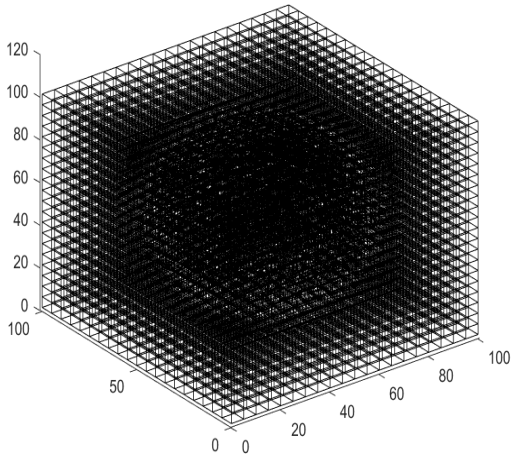
(a) $\boldsymbol{Dn}_1$ (b) $\boldsymbol{D}_2$

(c) step 0 (d) step 550 (e) $\hat{\boldsymbol{Dn}}_{12}$ superimposes on $\boldsymbol{D}_2$(black)

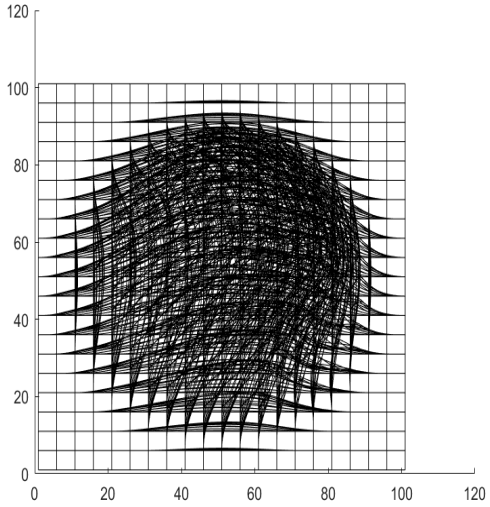
Figure 3.2: $\boldsymbol{Dn}_1$ to $\boldsymbol{D}_2$

Example 2: Variational Method in 3D

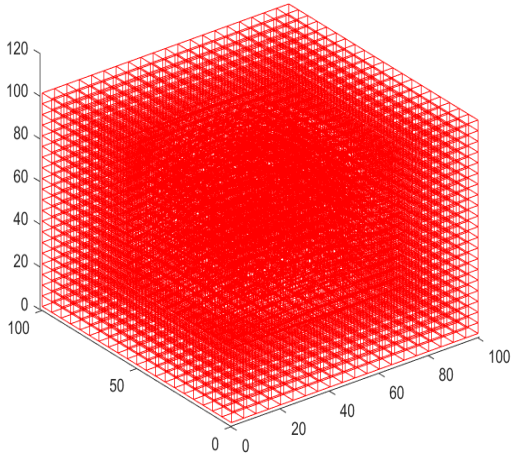
We purposely define a transformation using cutoff 3D rotation, 3D translation in a complicated way so that we may test the algorithm in a more general situation. This example took about 31 seconds and 16 iteration to reach $ratio = 0.0097$.



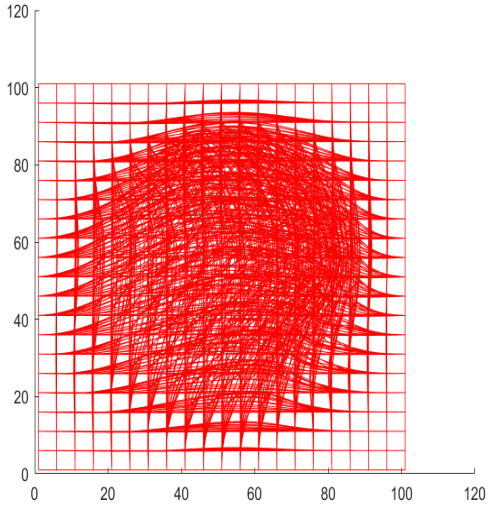
(a) $\boldsymbol{\Phi}$



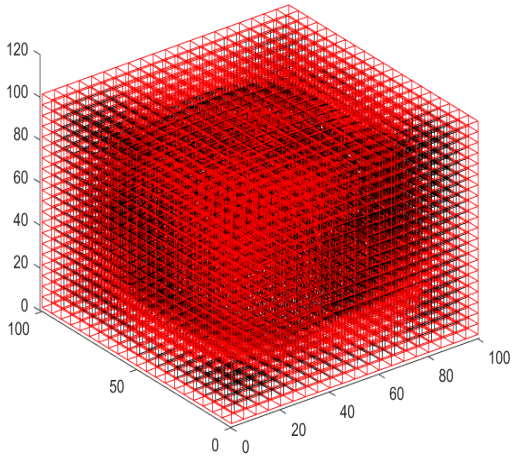
(b) $\boldsymbol{\Phi}$ bird view



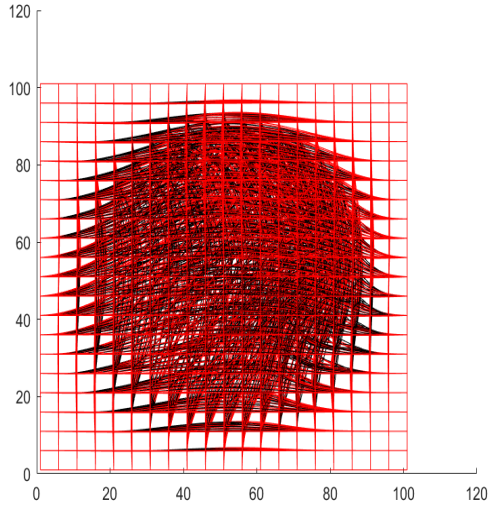
(c) $\widehat{\boldsymbol{\Phi}}$-red



(d) $\widehat{\boldsymbol{\Phi}}$-red bird view



(e) $\widehat{\boldsymbol{\Phi}}$ superimposes $\boldsymbol{\Phi}$



(f) $\widehat{\boldsymbol{\Phi}}$ superimposes $\boldsymbol{\Phi}$ bird view

Figure 3.3: $\widehat{\boldsymbol{\Phi}}$ recovered with $f_0 = \nabla \cdot (\boldsymbol{\Phi})$ & $\boldsymbol{g}_0 = \nabla \times (\boldsymbol{\Phi})$

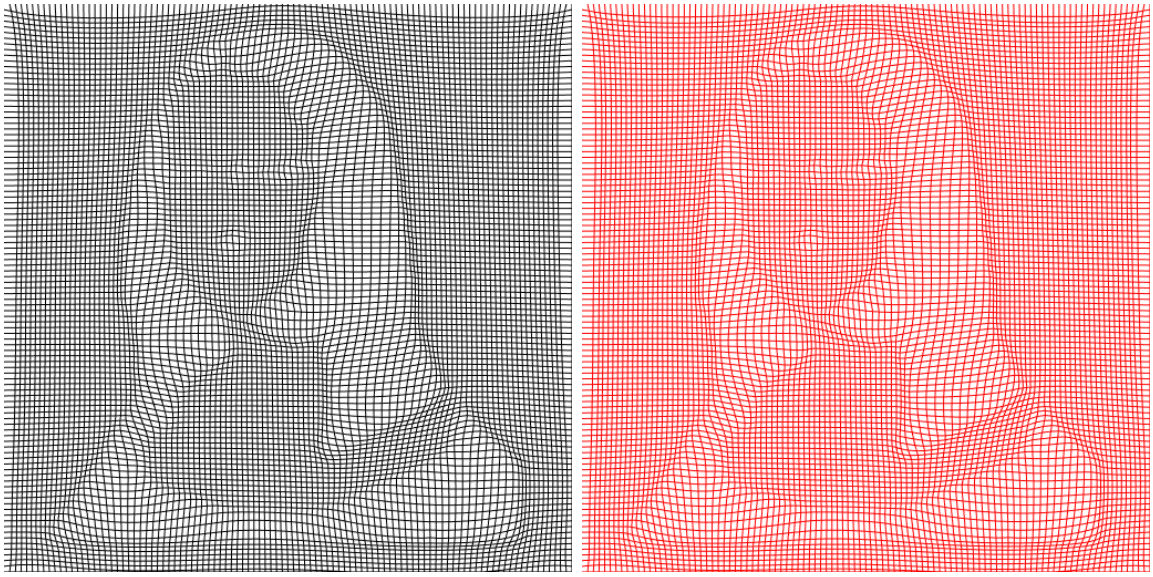
Next, we proceed to see how the uniqueness problem is formed.

3.3 Uniqueness suggested by Numerical Examples

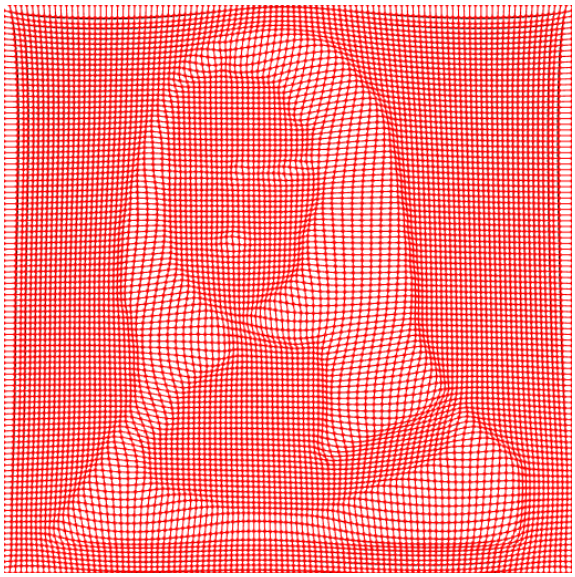
To understand the uniqueness problem, firstly, let's see an example how a transformation is recovered by prescribing both Jacobian determinant and curl.

Example 3: Curls in Meshes

Given a diffeomorphism $\boldsymbol{\Phi}$, in (a) of next Figure, we compute $f_0 = \det\nabla(\boldsymbol{\Phi})$ and $\boldsymbol{g}_0 = \nabla \times (\boldsymbol{\Phi})$ to be the prescription of Jacobian determinant and curl for (3.2.3). Then, by implementing a gradient descent optimization scheme, a mesh $\hat{\boldsymbol{\Phi}}$ in red is recovered, which almost overlaps $\boldsymbol{\Phi}$ in black, as shown in the following Figure (b).



(a) $\boldsymbol{\Phi}$ (b) $\hat{\boldsymbol{\Phi}}$-red

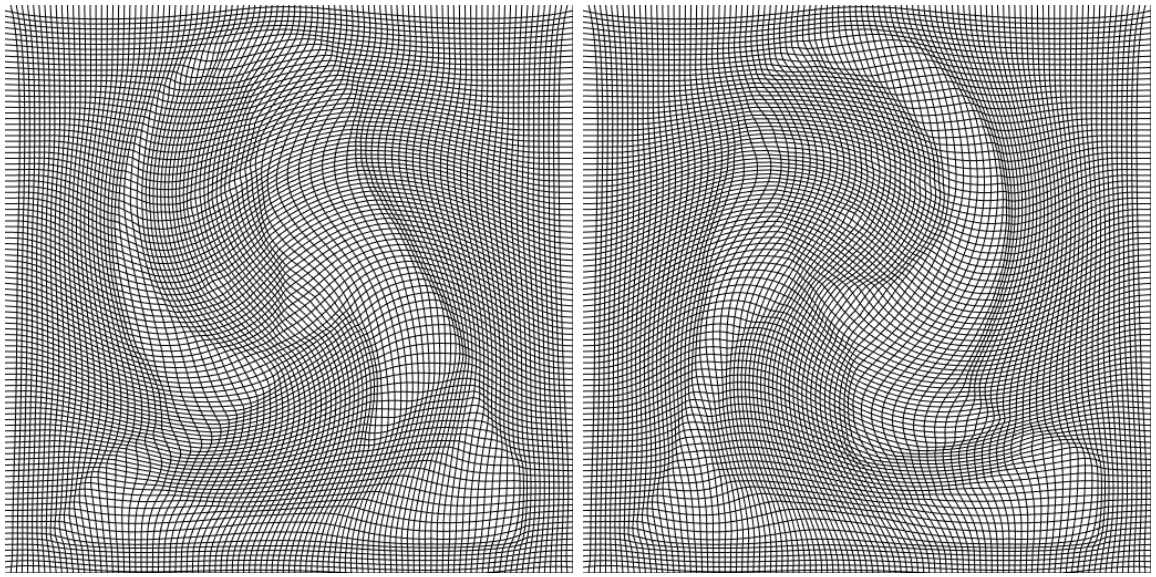


(c) $\hat{\boldsymbol{\Phi}}$ superimposes $\boldsymbol{\Phi}$

Figure 3.4: $\hat{\boldsymbol{\Phi}}$ recovered with $f_0 = \nabla \cdot (\boldsymbol{\Phi})$ & $\boldsymbol{g}_0 = \nabla \times (\boldsymbol{\Phi})$

Next, to see how different curls changes transformations from one to another even when the Jacobian determinants are the same: we, firstly, differed $\boldsymbol{g}_0 = \nabla \times (\boldsymbol{\Phi})$ into $\boldsymbol{g_1}_0$ and $\boldsymbol{g_2}_0$ such that $\frac{1}{2}(\boldsymbol{g_1}_0 + \boldsymbol{g_2}_0) = \boldsymbol{g}_0$ holds; then, we couple the prescribed

Jacobian determinant $f_0 = \det\nabla(\boldsymbol{\Phi})$ with $\boldsymbol{g_{1}}_0$ and $\boldsymbol{g_{2}}_0$, respectively, to recover $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$, as shown in the next Figure.



(a) $\boldsymbol{\Phi}_1$ (b) $\boldsymbol{\Phi}_2$

Figure 3.5: Effects of curl vector

In this example, we may concluded that the Jacobian determinant can not uniquely determine a transformation without the curl information and different curls will lead to different transformations even if the Jacobian determinants are the same.

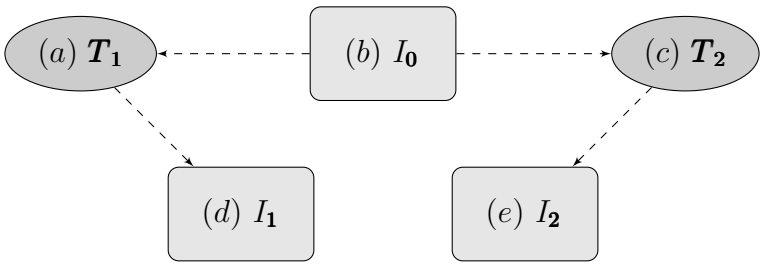
Example 4: Curls in Images

Image Re-sampling is a technique widely used in many tasks of image processing. A 1-to-1 and onto transformation is needed to determine the correspondent pairwise relationship from the original image to the re-sampled image. In this example, we show that to differ the curl of a re-sampling transformation —$\nabla \times (\boldsymbol{\phi})$ can differ the re-sampled result.

Step 1: Start with a Ground Truth image $I_{\mathbf{0}}$, to re-sample $I_{\mathbf{0}}$ by the identity map $\boldsymbol{id}$, we should get back $I_{\mathbf{0}}$ itself. Define $\boldsymbol{T_1}$ and $\boldsymbol{T_2}$ by a cut-off rotation transformation of

$\boldsymbol{id}$ with $\frac{\pi}{4}$ and $-\frac{\pi}{4}$, respectively, such that the $1 = \det\nabla(\boldsymbol{id}) \approx \frac{\det\nabla(\boldsymbol{T}_1)+\det\nabla(\boldsymbol{T}_2)}{2}$ and $\mathbf{0} = \mathrm{curl}(\boldsymbol{id}) \approx \frac{\mathrm{curl}(\boldsymbol{T}_1)+\mathrm{curl}(\boldsymbol{T}_2)}{2}$ hold. This means $\boldsymbol{T_1}$ and $\boldsymbol{T_2}$ have Jacobian determinants close to $1 = \det\nabla\boldsymbol{id}$ (in fact, they are between 0.996 and 1.003), but each of their curls are different.

Step 2: We re-sample $I_{\mathbf{0}}$ by $\boldsymbol{T_1}$ and $\boldsymbol{T_2}$, respectively, to get $I_{\mathbf{1}} = I_{\mathbf{0}}(\boldsymbol{T_1})$ and $I_{\mathbf{2}} = I_{\mathbf{0}}(\boldsymbol{T_2})$. The results are shown in next Figure the belows diagram.

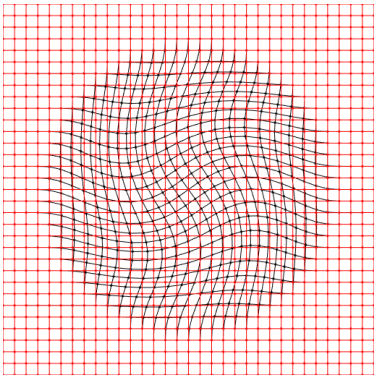


(a) $\boldsymbol{T_1}$ (b) I_0 (c) $\boldsymbol{T_2}$

(d) I_1 (e) I_2

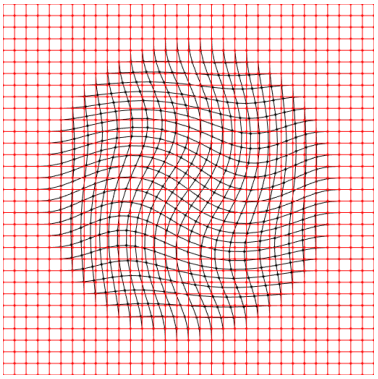
Figure 3.6: Re-sampled I_0 on $\boldsymbol{T_1}$ and $\boldsymbol{T_2}$

As it can be seen, different curls will result in different transformations, therefore, different curls will affect further on other studies that depend on these transformations. Step 3: To confirm these settings are of our intention, we compute the Jacobian Determinants $\boldsymbol{T_1}$ and $\boldsymbol{T_2}$. Then, recover two transformations, $\hat{\boldsymbol{T}}_{\boldsymbol{1}}$ and $\hat{\boldsymbol{T}}_{\boldsymbol{2}}$ by (3.2.3) without the second term of the functional. These recovered transformations should be very closed to $\boldsymbol{id}$, as shown in the following Figure 3.4



(a) $\hat{\boldsymbol{T}}_{\mathbf{1}}$-red vs $\boldsymbol{T}_{\mathbf{1}}$-black

(b) $I_{\mathbf{0}}(\hat{\boldsymbol{T}}_{\mathbf{1}})$



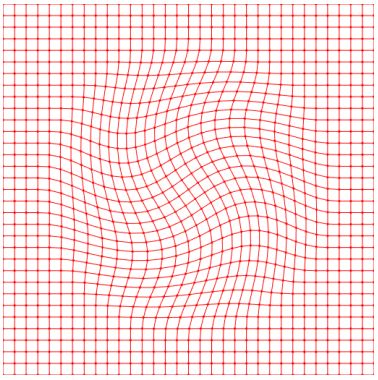
(c) $\hat{\boldsymbol{T}}_{\mathbf{2}}$-red vs $\boldsymbol{T}_{\mathbf{2}}$-black

(d) $I_{\mathbf{0}}(\hat{\boldsymbol{T}}_{\mathbf{2}})$

Figure 3.7: Reconstructed $\boldsymbol{T}_{\mathbf{1}}$ and $\boldsymbol{T}_{\mathbf{2}}$ without curls and their re-sampled images

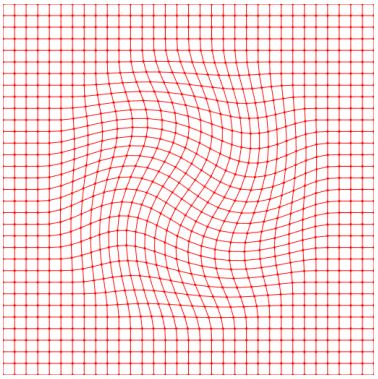
The re-sampled images on (b) and (d) of above Figure look just the same as $I_{\mathbf{0}}$, therefore, $\hat{\boldsymbol{T}}_{\mathbf{1}}$ and $\hat{\boldsymbol{T}}_{\mathbf{2}}$ are almost identical to the $\boldsymbol{id}$.

Step 4: This time, we include the curl of $\boldsymbol{T}_{\mathbf{1}}$ and $\boldsymbol{T}_{\mathbf{02}}$ with their Jacobian determinants in (3.2.3) for the recovering. So, the re-sampled images should be different according to different curls.



(a) $\check{\boldsymbol{T}}_{\mathbf{1}}$-red vs $\boldsymbol{T}_{\mathbf{1}}$-black

(b) $I_{\mathbf{0}}(\check{\boldsymbol{T}}_{\mathbf{1}})$



(c) $\check{\boldsymbol{T}}_{\mathbf{2}}$-red vs $\boldsymbol{T}_{\mathbf{2}}$-black

(d) $I_{\mathbf{0}}(\check{\boldsymbol{T}}_{\mathbf{2}})$

Figure 3.8: Reconstructed $\boldsymbol{T}_{\mathbf{1}}$, $\boldsymbol{T}_{\mathbf{2}}$ and their re-sampled images

This example shows that indistinguishable Jacobian determinant of a diffeomorphism with different curl vectors could lead to different diffeomorphism, and then it could lead to different re-sampled images. So, it was considereddd that Jacobian determinant alone cannot uniquely determine a diffeomorphism without combining with the curl vector.

3.4 The Uniqueness of the Variational Method

In this section, we consider only a simple case as follows. The argument in general is left to future study. Suppose two smooth transformations $\boldsymbol{\phi}$, $\boldsymbol{\psi} : \Omega \to \Omega$, $\Omega \subset \mathbb{R}^3$ have the same Jacobian determinant and curl-vector, namely,

$$\det\nabla(\boldsymbol{\phi}) = \det\nabla(\boldsymbol{\psi}) \tag{3.4.1}$$

$$\nabla \times (\boldsymbol{\phi}) = \nabla \times (\boldsymbol{\psi}) \tag{3.4.2}$$

$$\boldsymbol{\phi} = \boldsymbol{\psi}, \text{ on } \partial\Omega \tag{3.4.3}$$

We would like to ask: can the above conditions guarantee that $\boldsymbol{\phi} \equiv \boldsymbol{\psi}$ on Ω?

First,Let $\boldsymbol{\phi}$ and $\boldsymbol{\psi} : \Omega \to \Omega$ be two smooth transformations by

$$\boldsymbol{\phi} = \boldsymbol{id} + \boldsymbol{u} \tag{3.4.4}$$

$$\boldsymbol{\psi} = \boldsymbol{id} \tag{3.4.5}$$

where $\boldsymbol{u}$ is sufficiently small transformation in the Sobolev's space $H_0^2(\Omega)$. One can suppose that $\boldsymbol{u}$ satisfies

$$\|\boldsymbol{u}\|_{H_0^2(\Omega)} < \epsilon \tag{3.4.6}$$

Hence, we have

$$\begin{cases} \|\boldsymbol{u}\|_{L^2} < \epsilon \\ \|\nabla\boldsymbol{u}\|_{L^2} < \epsilon \\ \|\Delta\boldsymbol{u}\|_{L^2} < \epsilon \end{cases} \tag{3.4.7}$$

Second, from $\boldsymbol{\phi}(x_1, x_2, x_3) = (x_1+u_1(x_1, x_2, x_3), x_2+u_2(x_1, x_2, x_3), x_3+u_3(x_1, x_2, x_3))$ and $\boldsymbol{\psi}(x_1, x_2, x_3) = (x_1, x_2, x_3)$, we may derive

$$\det\nabla(\boldsymbol{\phi}) = \begin{vmatrix} 1 + u_{1x_1} & u_{2x_1} & u_{3x_1} \\ u_{1x_2} & 1 + u_{2x_2} & u_{3x_2} \\ u_{1x_3} & u_{2x_3} & 1 + u_{3x_3} \end{vmatrix}$$

$$= 1 + u_{1x_1} + u_{2x_2} + u_{3x_3}$$

$$+u_{1x_1}u_{2x_2}u_{3x_3} + u_{1x_3}u_{2x_1}u_{3x_2} + u_{1x_2}u_{2x_3}u_{3x_1}$$

$$-u_{1x_1}u_{2x_3}u_{3x_2} - u_{1x_2}u_{2x_1}u_{3x_3} - u_{1x_3}u_{2x_2}u_{3x_1}$$

$$+u_{1x_1}u_{2x_2} + u_{1x_1}u_{3x_3} + u_{2x_2}u_{3x_3}$$

$$-u_{1x_2}u_{2x_1} - u_{1x_3}u_{3x_1} - u_{2x_3}u_{3x_2}$$

$$= 1 + \nabla \cdot (\boldsymbol{u}) + \det\nabla(\boldsymbol{u})$$

$$+u_{1x_1}u_{2x_2} + u_{1x_1}u_{3x_3} + u_{2x_2}u_{3x_3}$$

$$-u_{1x_2}u_{2x_1} - u_{1x_3}u_{3x_1} - u_{2x_3}u_{3x_2}$$

$$= 1 + \nabla \cdot (\boldsymbol{u}) - \mathcal{F}(\boldsymbol{u})$$

where we denote

$$\mathcal{F}(\boldsymbol{u}) = -[\det\nabla(\boldsymbol{u}) + Tail(\boldsymbol{u})]$$

and

$$Tail(\boldsymbol{u}) = u_{1x_1}u_{2x_2} + u_{1x_1}u_{3x_3} + u_{2x_2}u_{3x_3} - u_{1x_2}u_{2x_1} - u_{1x_3}u_{3x_1} - u_{2x_3}u_{3x_2} \tag{3.4.8}$$

According to (3.4.1) and (3.4.2) we have,

$$0 = \det\nabla(\boldsymbol{\phi}) - \det\nabla(\boldsymbol{\psi}) = 1 + \nabla \cdot (\boldsymbol{u}) - \mathcal{F}(\boldsymbol{u}) - 1 = \nabla \cdot (\boldsymbol{u}) - \mathcal{F}(\boldsymbol{u})$$

$$\mathbf{0} = \nabla \times (\boldsymbol{\phi}) - \nabla \times (\boldsymbol{\psi}) = \begin{pmatrix} u_{3x_2} - u_{2x_3} \\ u_{1x_3} - u_{3x_1} \\ u_{1x_2} - u_{2x_1} \end{pmatrix} - \mathbf{0} = \nabla \times (\boldsymbol{u})$$

i.e.,

$$\begin{cases} \nabla \cdot (\boldsymbol{u}) = \mathcal{F}(\boldsymbol{u}) \\ \nabla \times (\boldsymbol{u}) = \mathbf{0}. \end{cases} \tag{3.4.9}$$

It follows from (3.4.9) that $\boldsymbol{u}$ satisfies the *Poisson* equations:

$$\begin{cases} \Delta u_1 = \nabla \cdot (\boldsymbol{u})_{x_1} - [\nabla \times (\boldsymbol{u})_3]_{x_2} + [\nabla \times (\boldsymbol{u})_2]_{x_3} = \mathcal{F}(\boldsymbol{u})_{x_1} \\ \Delta u_2 = \nabla \cdot (\boldsymbol{u})_{x_2} + [\nabla \times (\boldsymbol{u})_3]_{x_1} - [\nabla \times (\boldsymbol{u})_1]_{x_3} = \mathcal{F}(\boldsymbol{u})_{x_2} \\ \Delta u_3 = \nabla \cdot (\boldsymbol{u})_{x_3} - [\nabla \times (\boldsymbol{u})_2]_{x_1} + [\nabla \times (\boldsymbol{u})_1]_{x_2} = \mathcal{F}(\boldsymbol{u})_{x_3} \end{cases}$$

$$\Rightarrow \Delta \boldsymbol{u} = (\mathcal{F}(\boldsymbol{u})_{x_1}, \mathcal{F}(\boldsymbol{u})_{x_2}, \mathcal{F}(\boldsymbol{u})_{x_3}) = \nabla_{\boldsymbol{x}} \mathcal{F}(\boldsymbol{u})$$

Note that the dominating terms of $\mathcal{F}(\boldsymbol{u})$ at (3.4.8) are products of only the first partial derivatives of $\boldsymbol{u}$, so the dominating terms of $\nabla_{\boldsymbol{x}} \mathcal{F}(\boldsymbol{u})$ at (3.4.10) are the terms in products of the first and second partial derivatives of $\boldsymbol{u}$. This means, by (3.4.7), we get

$$\|\Delta \boldsymbol{u}\|_{L^2} = (\int_\Omega |\Delta \boldsymbol{u}|^2)^{\frac{1}{2}} = (\int_\Omega |\nabla_{\boldsymbol{x}} \mathcal{F}(\boldsymbol{u})|^2)^{\frac{1}{2}} = \|\nabla_{\boldsymbol{x}} \mathcal{F}(\boldsymbol{u})\|_{L^2} \tag{3.4.10}$$

$$\Rightarrow \|\Delta \boldsymbol{u}\|_{L^2} < \epsilon \cdot \epsilon = \epsilon^2 \tag{3.4.11}$$

Next, we will establish an inequality for $\|\boldsymbol{u}\|_{L^2}$. Since $\boldsymbol{u} = 0$ on $\partial\Omega$, by *Green's* formula, we can derive,

$$\int_\Omega |\nabla \boldsymbol{u}|^2 = |\int_\Omega \boldsymbol{u} \cdot \Delta \boldsymbol{u}| \tag{3.4.12}$$

Applying the $Cauchy - Schwarz$ inequality and properties of integration to the RHS of (3.4.12) to get

$$\int_\Omega |\nabla \boldsymbol{u}|^2 = |\int_\Omega \boldsymbol{u} \cdot \Delta \boldsymbol{u}| \leq \int_\Omega |\boldsymbol{u}||\Delta \boldsymbol{u}| \leq \|\boldsymbol{u}\|_{L^2} \|\Delta \boldsymbol{u}\|_{L^2}$$

$$\Rightarrow \|\nabla \boldsymbol{u}\|_{L^2}^2 \leq \|\boldsymbol{u}\|_{L^2} \|\Delta \boldsymbol{u}\|_{L^2} \tag{3.4.13}$$

Applying the *Poincare's* inequality to the LHS of (3.4.12), $\exists\ 0 < C \in \mathbb{R}$, such that

$$\|\boldsymbol{u}\|_{L^2}^2 \leq C \|\nabla \boldsymbol{u}\|_{L^2}^2 = C \int_\Omega |\nabla \boldsymbol{u}|^2 \tag{3.4.14}$$

Next, we combine (3.4.13) and (3.4.14) to have

$$\|\boldsymbol{u}\|_{L^2}^2 \leq C \|\nabla \boldsymbol{u}\|_{L^2}^2 \leq C \|\boldsymbol{u}\|_{L^2} \|\Delta \boldsymbol{u}\|_{L^2} \tag{3.4.15}$$

$$\Rightarrow \|\boldsymbol{u}\|_{L^2} \leq C \|\Delta \boldsymbol{u}\|_{L^2} < C\epsilon^2 \tag{3.4.16}$$

And, as for $\|\Delta \boldsymbol{u}\|_{L^2}$, by (3.4.11), (3.4.13) and (3.4.16), we may bound $\|\nabla \boldsymbol{u}\|_{L^2}$ as

$$\|\nabla \boldsymbol{u}\|_{L^2} \leq (\|\boldsymbol{u}\|_{L^2} \|\Delta \boldsymbol{u}\|_{L^2})^{\frac{1}{2}} \tag{3.4.17}$$

$$\Rightarrow \|\nabla \boldsymbol{u}\|_{L^2} < (C\epsilon^2 \cdot \epsilon^2)^{\frac{1}{2}} = C^{\frac{1}{2}}\epsilon^2 \tag{3.4.18}$$

Now, $\boldsymbol{u}$ satisfies

$$\begin{cases} \|\boldsymbol{u}\|_{L^2} < C\epsilon^2 \\ \|\nabla \boldsymbol{u}\|_{L^2} < C^{\frac{1}{2}}\epsilon^2 \\ \|\Delta \boldsymbol{u}\|_{L^2} < \epsilon^2 \end{cases} \tag{3.4.19}$$

Third, let's treat the procedure of (3.4.7) to (3.4.19) as Step-0, and repeat it again with replacing (3.4.7) by the result (3.4.19). So, plug (3.4.18) into the first derivative of $\boldsymbol{u}$ in (3.4.10), we get

$$\|\Delta \boldsymbol{u}\|_{L^2} = \|\nabla_{\boldsymbol{x}} \mathcal{F}(\boldsymbol{u})\|_{L^2} < \epsilon \cdot C^{\frac{1}{2}}\epsilon^2 = C^{\frac{1}{2}}\epsilon^3 \tag{3.4.20}$$

Then, by (3.4.15), we get

$$\|\boldsymbol{u}\|_{L^2} \leq C \|\Delta \boldsymbol{u}\|_{L^2} < C \cdot C^{\frac{1}{2}}\epsilon^3 = C^{(1+\frac{1}{2})}\epsilon^3 \tag{3.4.21}$$

and by (3.4.17), we get

$$\|\nabla \boldsymbol{u}\|_{L^2} \leq (\|\boldsymbol{u}\|_{L^2} \|\Delta \boldsymbol{u}\|_{L^2})^{\frac{1}{2}} < (C^{(1+\frac{1}{2})}\epsilon^3 \cdot C^{\frac{1}{2}}\epsilon^3)^{\frac{1}{2}} = C\epsilon^3 \tag{3.4.22}$$

which above can be combined from (3.4.19), (3.4.20) and (3.4.21) into

$$\begin{cases} \|\boldsymbol{u}\|_{L^2} < C^{(1+\frac{1}{2})}\epsilon^3 \\ \|\nabla \boldsymbol{u}\|_{L^2} < C\epsilon^3 \\ \|\Delta \boldsymbol{u}\|_{L^2} < C^{\frac{1}{2}}\epsilon^3 \end{cases} \tag{3.4.23}$$

Fourth, in order to complete the argument of the simple case, an iterative process based on the procedure (3.4.19) to (3.4.23) is constructed as follows:

- Step-0: From (3.4.7) to (3.4.19), and denote (3.4.19) as $(3.4.19)_k$, set $k = 0$;
- Step-1: Start iteration (Step-2 to 5) on $k = k + 1$;
- Step-2: Apply $(3.4.19)_k$ on (3.4.10) to get $\|\Delta \boldsymbol{u}\|_{L^2} < C^{(0+\frac{k}{2})}\epsilon^{(2+k)}$, and denote it as $(3.4.10)_k$;
- Step-3: Apply $(3.4.20)_k$ on (3.4.15) to get $\|\boldsymbol{u}\|_{L^2} < C^{(1+\frac{k}{2})}\epsilon^{(2+k)}$, and denote it as $(3.4.15)_k$;
- Step-4: Apply $(3.4.15)_k$ on (3.4.17) to get $\|\nabla \boldsymbol{u}\|_{L^2} < C^{(\frac{1}{2}+\frac{k}{2})}\epsilon^{(2+k)}$, and denote it as $(3.4.17)_k$;
- Step-5: Combine $(3.4.10)_k$, $(3.4.15)_k$, $(3.4.17)_k$ to form $(3.4.19)_{k+1}$ (For example: $(3.4.19)_1$ is (3.4.23)), then back to Step-1.

Hence, on the k-th iteration, (3.4.19) can be improved to $(3.4.19)_k$, i.e.,

$$\begin{cases} \|\boldsymbol{u}\|_{L^2} < C^{(1+\frac{k}{2})}\epsilon^{(2+k)} \\ \|\nabla \boldsymbol{u}\|_{L^2} < C^{(\frac{1}{2}+\frac{k}{2})}\epsilon^{(2+k)} \\ \|\Delta \boldsymbol{u}\|_{L^2} < C^{(0+\frac{k}{2})}\epsilon^{(2+k)} \end{cases}$$

It is natural to take a step forward at $k + 1$. So, plug $(3.4.19)_k$ into (3.4.10), we get

$$\|\Delta \boldsymbol{u}\|_{L^2} = \|\nabla_{\boldsymbol{x}} \mathcal{F}(\boldsymbol{u})\|_{L^2} < \epsilon \cdot C^{(\frac{1}{2}+\frac{k}{2})}\epsilon^{(2+k)} = C^{(0+\frac{k+1}{2})}\epsilon^{(2+k+1)}$$

and denote it as $(3.4.10)_{k+1}$. Then, by (3.4.15) we get

$$\|\boldsymbol{u}\|_{L^2} \leq C \|\Delta \boldsymbol{u}\|_{L^2} < C \cdot C^{(0+\frac{k+1}{2})}\epsilon^{(2+k+1)} = C^{(1+\frac{k+1}{2})}\epsilon^{(2+k+1)}$$

and denote it as $(3.4.15)_{k+1}$. Then, by (3.4.17) we get

$$\|\nabla \boldsymbol{u}\|_{L^2} < (C^{(1+\frac{k+1}{2})}\epsilon^{(2+k+1)} \cdot C^{(0+\frac{k+1}{2})}\epsilon^{(2+k+1)})^{\frac{1}{2}} = C^{(\frac{1}{2}+\frac{k+1}{2})}\epsilon^{(2+k+1)}$$

and denote it as $(3.4.17)_{k+1}$. This leads to the immediate inductive step $(3.4.19)_{k+1}$

$$\begin{cases} \|\boldsymbol{u}\|_{L^2} < C^{(1+\frac{k+1}{2})}\epsilon^{(2+k+1)} \\ \|\nabla \boldsymbol{u}\|_{L^2} < C^{(\frac{1}{2}+\frac{k+1}{2})}\epsilon^{(2+k+1)} \\ \|\Delta \boldsymbol{u}\|_{L^2} < C^{(0+\frac{k+1}{2})}\epsilon^{(2+k+1)} \end{cases}$$

Therefore, the system of inequalities $(3.4.17)_k$ is iterated to reduce the bound of $\|\boldsymbol{u}\|_{L^2}$ for every increment of $k = k+1$. Since $\boldsymbol{u}$ satisfies (3.4.7), then we may conclude such $\boldsymbol{u}$ on Ω is also satisfying $\|\boldsymbol{u}\|_{L^2} < C^{(1+\frac{k}{2})}\epsilon^{(2+k)}$, where $C^{(1+\frac{k}{2})}\epsilon^{(2+k)}$ converges to 0 by the choice of $0 < \epsilon < \min\{1, 1/\sqrt{C}\}$, as $k \longrightarrow \infty$. So it can also be concluded that $\|\boldsymbol{\phi} - \boldsymbol{\psi}\|_{L^2} = \|\boldsymbol{u}\|_{L^2} \longrightarrow 0$ as $k \longrightarrow \infty$, therefore $\boldsymbol{\phi} \equiv \boldsymbol{\psi} = \boldsymbol{id}$ on Ω.

In this section, we described an approach to the uniqueness problem based on the simple case which the two smooth transformations are close to each other and one of them is the ***identity*** map. The general uniqueness problem from (3.4.1) to (3.4.3) remains open. An interesting intermediate step is to show that, for any two sufficiently close $\boldsymbol{\phi}$, $\boldsymbol{\psi}$, a similar argument can be applied.

3.5 Direct Method v.s. Iterative Method

Motived by the uniqueness problem, we curiously attempt to argue that if the global minimizer exists for (3.2.3), then it should also satisfies the *Poisson* equation with the right-hand-side formed by the given monitor functions $f_0(\boldsymbol{x}) > 0$ and $\boldsymbol{g}_0(\boldsymbol{x})$. To see this, let $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{id}(\boldsymbol{x}) + \boldsymbol{u}(\boldsymbol{x})$, $\det\nabla(\boldsymbol{\phi}) = f_0(\boldsymbol{x})$ and $\nabla \times (\boldsymbol{\phi}) = \boldsymbol{g}_0(\boldsymbol{x})$, then one may derive

$$\begin{cases} \det\nabla(\boldsymbol{\phi}) = 1 + \nabla \cdot (\boldsymbol{u}) - \mathcal{F}(\boldsymbol{u}) & \\ \nabla \times (\boldsymbol{\phi}) = \nabla \times (\boldsymbol{u}) & \text{in } \Omega \\ \boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{x} \qquad \text{on } \partial\Omega & \end{cases}$$

where $\mathcal{F}(\boldsymbol{u}) = -[\det\nabla(\boldsymbol{u}) + Tail(\boldsymbol{u})]$ and

$$Tail(\boldsymbol{u}) = u_{1x_1}u_{2x_2} + u_{1x_1}u_{3x_3} + u_{2x_2}u_{3x_3} - u_{1x_2}u_{2x_1} - u_{1x_3}u_{3x_1} - u_{2x_3}u_{3x_2}$$

so,

$$\begin{cases} \nabla \cdot (\boldsymbol{u}) = \det\nabla(\boldsymbol{\phi}) - 1 + \mathcal{F}(\boldsymbol{u}) \\ \nabla \times (\boldsymbol{u}) = \nabla \times (\boldsymbol{\phi}) \\ \quad \boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{0} \qquad \text{on } \partial\Omega \end{cases} \text{in } \Omega$$

then we have

$$\begin{cases} \Delta\boldsymbol{u}(\boldsymbol{x}) = \nabla(\det\nabla(\boldsymbol{\phi})) - \nabla \times (\det\nabla(\boldsymbol{\phi})) + \nabla_{\boldsymbol{x}}\mathcal{F}(\boldsymbol{u}) \qquad \text{in } \Omega \\ \quad \boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{0} \qquad \text{on } \partial\Omega \end{cases}$$

$$\Rightarrow \begin{cases} \Delta\boldsymbol{u}(\boldsymbol{x}) = \nabla f_0(\boldsymbol{x}) - \nabla \times \boldsymbol{g}_0(\boldsymbol{x}) + \nabla_{\boldsymbol{x}}\mathcal{F}(\boldsymbol{u}) \qquad \text{in } \Omega \\ \quad \boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{0} \qquad \text{on } \partial\Omega \end{cases}$$

As suggested by the uniqueness of simple case discussed in the previous section, the term $\nabla_{\boldsymbol{x}}\mathcal{F}(\boldsymbol{u})$ contains terms of second partial derivatives of $\boldsymbol{u}$ which is not significantly large. So, we take an approximation of the right-hand-side by chopping $\nabla_{\boldsymbol{x}}\mathcal{F}(\boldsymbol{u})$ off as follows

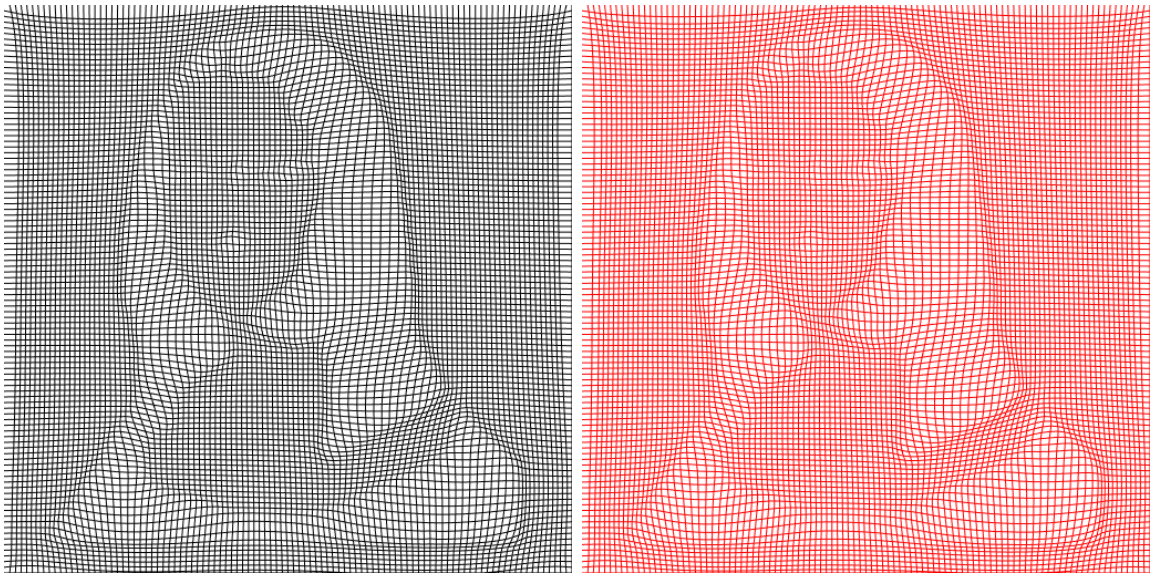
$$\begin{cases} \Delta\tilde{\boldsymbol{u}}(\boldsymbol{x}) = \nabla f_0(\boldsymbol{x}) - \nabla \times \boldsymbol{g}_0(\boldsymbol{x}) = \boldsymbol{F}_0 \qquad \text{in } \Omega \\ \quad \tilde{\boldsymbol{u}}(\boldsymbol{x}) = \boldsymbol{0} \qquad \text{on } \partial\Omega \end{cases} \tag{3.5.1}$$

Once $\Delta\tilde{\boldsymbol{u}}(\boldsymbol{x}) = \boldsymbol{F}_0$ is solved, then we define $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{x} + \tilde{\boldsymbol{u}}(\boldsymbol{x})$ to be an answer to the direct strategy. We tested the next two example demonstrate the effectiveness of this direct strategy on both 2D and 3D cases and compare it with the iterative scheme provide before.

Example 5: Direct V.S. Iterative in 2D

In this example, the direct computational strategy is tested on the problem in Example 2. We compute the Jacobian determinant and curl vector of $\boldsymbol{\Phi}$ then define

$f_0(\boldsymbol{x}) = \det\nabla(\boldsymbol{\Phi})$ and $\boldsymbol{g}_0(\boldsymbol{x}) = \nabla \times (\boldsymbol{\Phi})$. So, the right-hand-side of *Poisson* equation (3.5.1) the can be formed. The next figure shows the result with the direct strategy.

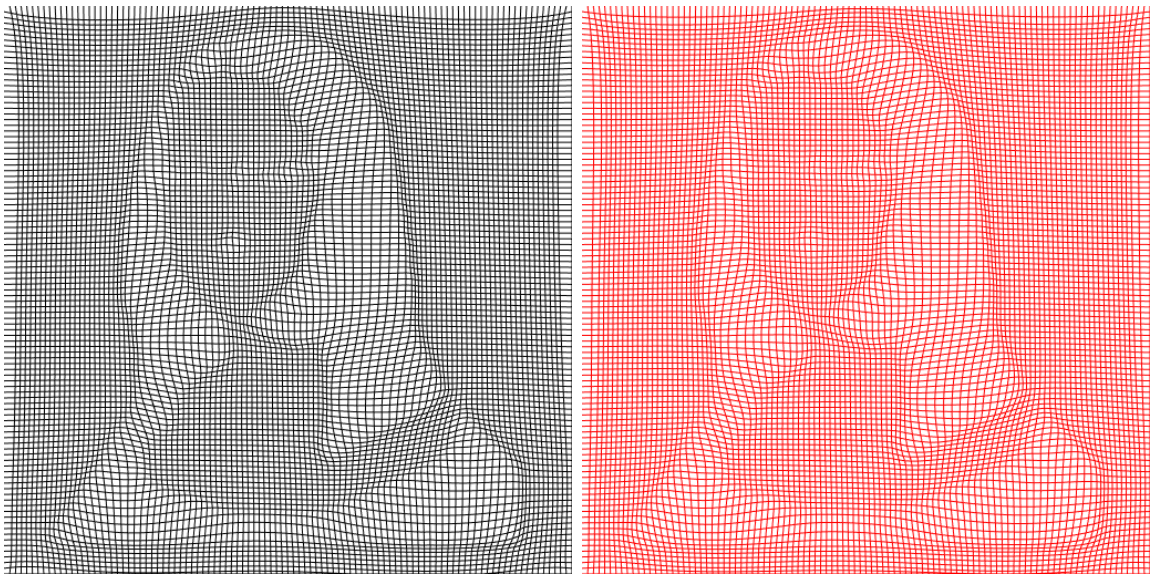


(a) $\boldsymbol{\Phi}$ (b) $\boldsymbol{\Phi_d}$-red

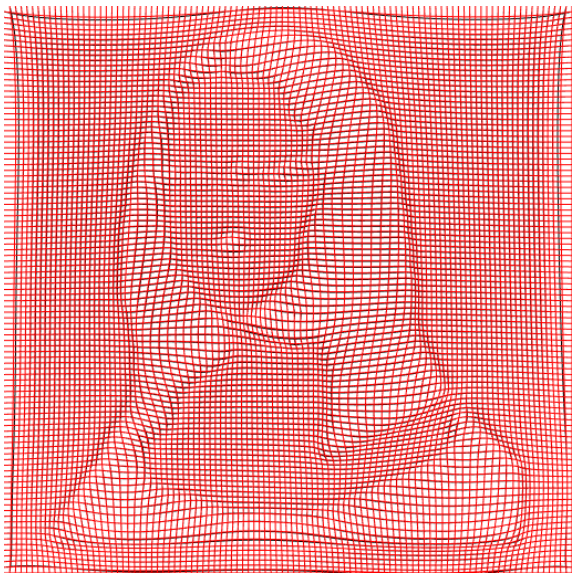
(c) $\boldsymbol{\Phi_d}$ superimposes $\boldsymbol{\Phi}$

Figure 3.9: $\boldsymbol{\Phi_d}$ found directly with $f_0 = \nabla \cdot (\boldsymbol{\Phi})$ & $\boldsymbol{g}_0 = \nabla \times (\boldsymbol{\Phi})$

To understand how well or bad the direct strategy has achieved compare to the original iterative scheme, we set the SSD deduction of (3.2.3) realized by the direct strategy to be the stopping criteria in the iterative scheme. And the result is shown in the next Figure.



(a) $\boldsymbol{\Phi}$ (b) $\boldsymbol{\Phi_i}$-red



(c) $\boldsymbol{\Phi_i}$ superimposes $\boldsymbol{\Phi}$

Figure 3.10: $\boldsymbol{\Phi_i}$ recovered with $f_0 = \nabla \cdot (\boldsymbol{\Phi})$ & $\boldsymbol{g}_0 = \nabla \times (\boldsymbol{\Phi})$

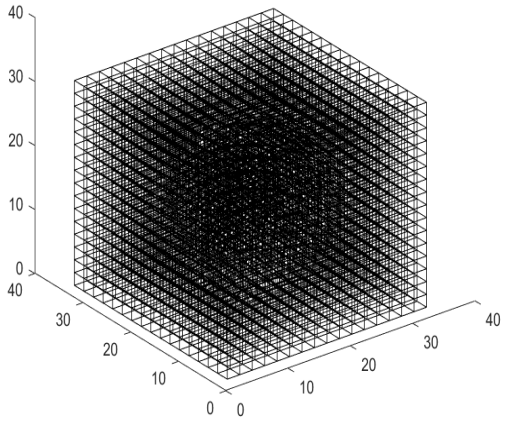
Some of the key measures are brought together in the following Table.

Table 3.1: Direct V.S. Iterative

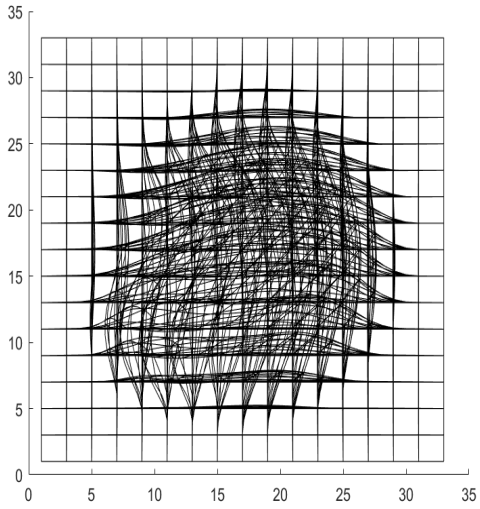
	Direct	Iterative
Mesh Ω	100^2	100^2
Elapsed seconds	0.005815	2.7152
SSD_{init}	812.0032	812.0032
SSD_{new}	17.1791	17.1309
$Ratio$	0.0212	0.0211
Iteration(s)	1	616
max det∇ difference	0.4350	0.5324
max $\nabla\times$ differece	0.0768	0.1406
max $\|\|\boldsymbol{\Phi_i} - \boldsymbol{\Phi}\|\|$	0.4350	0.5325

Example 6: Direct V.S. Iterative in 3D

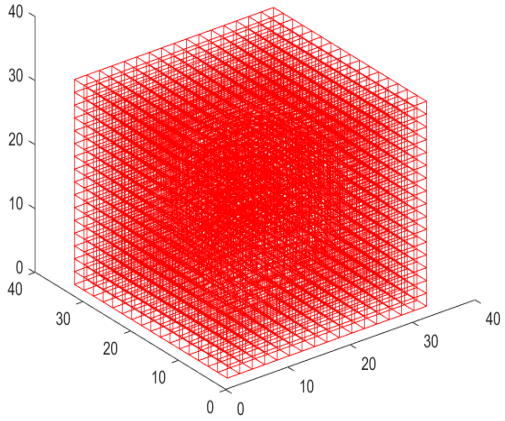
Similarly to the 2D case, given $\boldsymbol{\Phi}$ then it is defined $f_0(\boldsymbol{x}) = \det\nabla(\boldsymbol{\Phi})$ and $\boldsymbol{g}_0(\boldsymbol{x}) = \nabla(\boldsymbol{\Phi})$. Form the right-hand-side as in (3.2.4). The following Figures shown our computational results.



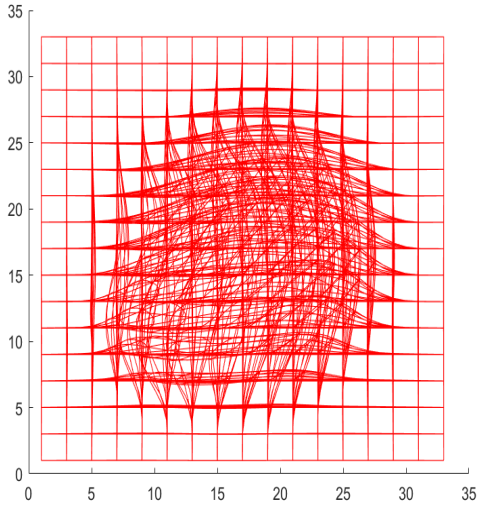
(a) $\boldsymbol{\Phi}$



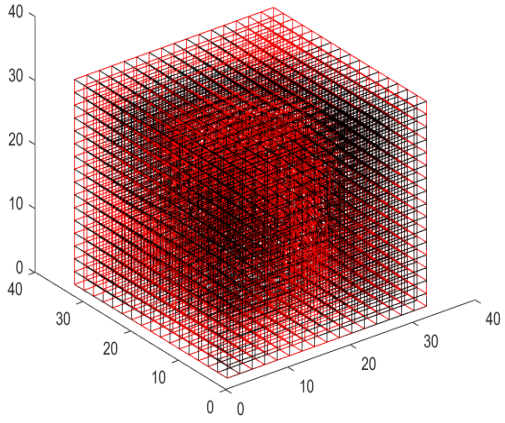
(b) $\boldsymbol{\Phi}$ bird view



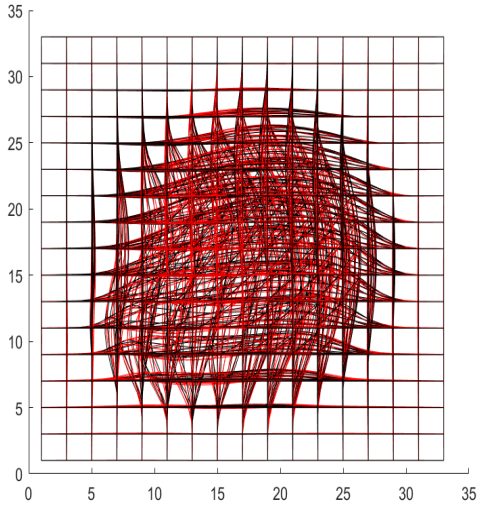
(c) $\boldsymbol{\Phi_d}$-red



(d) $\boldsymbol{\Phi_d}$-red bird view



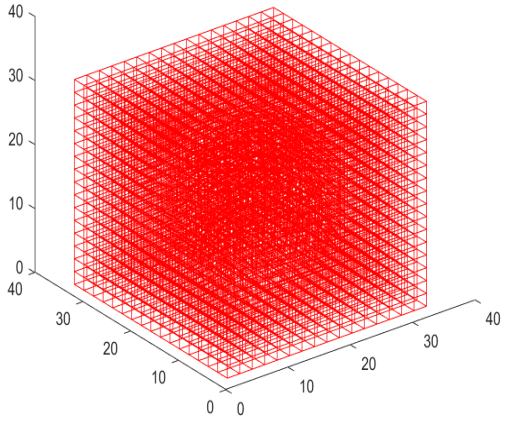
(e) $\boldsymbol{\Phi_d}$ superimposes $\boldsymbol{\Phi}$



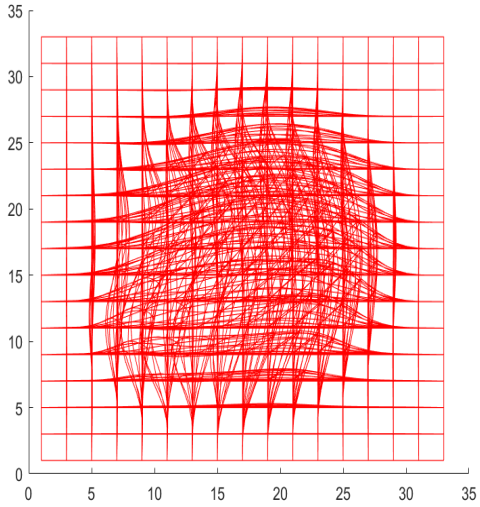
(f) $\boldsymbol{\Phi_d}$ superimposes $\boldsymbol{\Phi}$ bird view

Figure 3.11: $\boldsymbol{\Phi_d}$ found directly with $f_0 = \nabla \cdot (\boldsymbol{\Phi})$ & $\boldsymbol{g}_0 = \nabla \times (\boldsymbol{\Phi})$

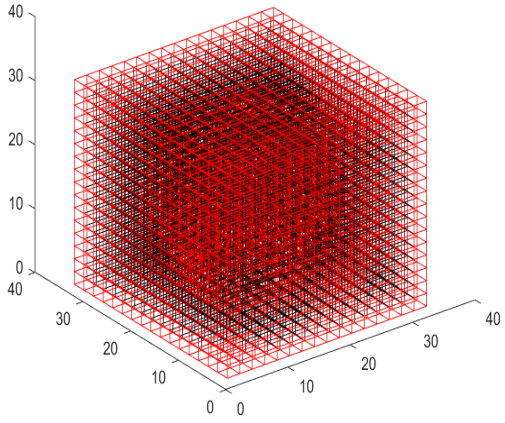
The transformation $\boldsymbol{\phi}$ seems pretty well recovered according to the picture, however, more details reveal that the direct strategy was not that successful on this problem as it looks. The relevant measures shown in the Table below the next Figures confirms this.



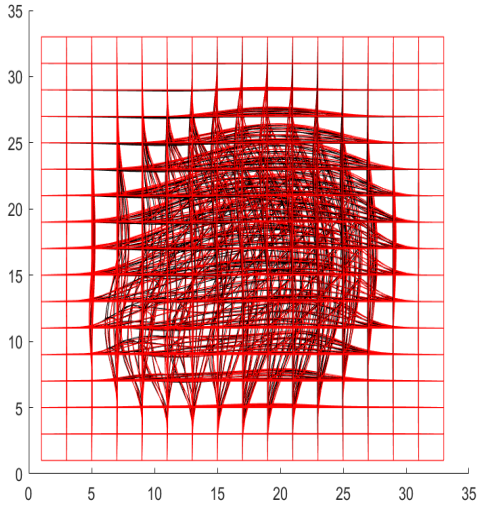
(a) $\boldsymbol{\Phi_i}$-red



(b) $\boldsymbol{\Phi_i}$-red bird view



(c) $\boldsymbol{\Phi_i}$-red superimposes $\boldsymbol{\Phi}$-black



(d) $\boldsymbol{\Phi_i}$-red superimposes $\boldsymbol{\Phi}$-black bird view

Figure 3.12: $\boldsymbol{\Phi_i}$ found directly with $f_0 = \nabla \cdot (\boldsymbol{\Phi})$ & $\boldsymbol{g}_0 = \nabla \times (\boldsymbol{\Phi})$

Based on the most important measure, *ratio*, from the Table, the direct strategy does not seem converged at all in this problem. The iterative scheme may took a bit extra time, yet, it does finds its way to get the better solution. So, the direct strategy could be a fast result, but not necessarily good results in general. To the least we

Table 3.2: Direct V.S. Iterative in 3D

	Direct	Iterative
Grid Ω	32^3	32^3
Elapsed seconds	0.042273	2.1211
SSD_{init}	3825.6001	3825.6001
SSD_{new}	2159.9317	37.4521
$Ratio$	0.5646	0.0211
Iteration(s)	1	40
max det∇ difference	0.4350	0.5638
max $\nabla\times$ differece	1.9361	0.1396
max $\|\|\boldsymbol{\Phi_i} - \boldsymbol{\Phi}\|\|$	0.7323	5.8725

may say, in some scenarios, the direct strategy can be used to generate meaningful initial meshes. Compared to the iterative scheme, the direct strategy is incapable to find a better solution to decrease *ratio*. But, both two approaches can be made useful depend on the exact practical situations.

3.6 Conlusion

In this chapter, we extensively study the variational method and realized the 3D scenario. The unique determination of transformations based on Jacobian determinant and curl vector is analytically discussed in a special simple case in 3D, while The general case of the problem remains open. To give meaningful prescriptions of Jacobian determinant and curl vector can still be a challenge. Because the variational method successfully realized in 3D, then we may characterize diffeomorphisms with both Jacobian determinant and curl vector at the same time. A novel approach to averaging given diffeomorphisms is formulated. This technique of averaging diffeomorphisms is the key component of our approach to averaging images, which is introduced detailedly discussed in chapter 5.

CHAPTER 4

Optimal Control Approach to Image Registration

4.1 Introduction

In [25], we proposed an optimal control approach that has no explicit penalty terms (and hence no parameters). The registration transformation $\boldsymbol{T}$ is iteratively determined. In the version described in [25] has a mechanism that monitors the positivity of Jacobian determinant, which in turn assures that $\boldsymbol{T}$ is a diffeomorphism (invertible and smooth). And in each iteration, the solution is found under the control functions formed by the Jacobian determinant and curl vector. The control functions are simplified to satisfy a *Poisson* equation with fixed boundary condition, so each iterative solution always strictly limited on the desired fixed boundary. This is essential and beneficial to image analysis. It enables us understanding the variabilities of image data by studying registration transformations. In this chapter, we revised the original version and show its advantages by an 3D numerical example. Furthermore, inverse consistency and transitivity of our registration method are checked numerically.

4.2 Old method Image Registration based on Deformation method

The general image registration problem can be stated in the following setting: Let $I_{\boldsymbol{m}}$ be a ***moving*** image is to be registered to a ***fixed*** image $I_{\boldsymbol{f}}$ on the fixed and bounded domain $\Omega \subset \mathbb{R}^3$ (It's similar for $\mathbb{R}^2$). Define the cost functional - sum of squared difference:

$$SSD(\boldsymbol{\phi}(\boldsymbol{x})) = \frac{1}{2}\int_{\Omega}[I_{\boldsymbol{m}}(\boldsymbol{\phi}(\boldsymbol{x})) - I_{\boldsymbol{f}}(\boldsymbol{x})]^2 d\boldsymbol{x} \tag{4.2.1}$$

To look for a diffeomorphism $\boldsymbol{\phi} : I_{\boldsymbol{f}} \rightarrow I_{\boldsymbol{m}}$ minimizes $SSD(\boldsymbol{\phi}(\boldsymbol{x}))$:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{\phi}_0(\boldsymbol{x}) + \frac{\boldsymbol{u}(\boldsymbol{x})}{1 + \nabla \cdot \boldsymbol{u}(\boldsymbol{x})} \triangle t, \qquad \triangle t \text{ is an artificial time-step} \tag{4.2.2}$$

subjects to the constraints $f(\boldsymbol{x}) = J(\boldsymbol{x}) > 0$ and $\boldsymbol{g}(\boldsymbol{x})$ with

$$\begin{cases} \nabla \cdot \boldsymbol{u}(\boldsymbol{x}) = f(\boldsymbol{x}) - 1 \\ \nabla \times \boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{x}) \\ \boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{0} \qquad \text{on } \partial\Omega \end{cases} \text{in } \Omega \Rightarrow \begin{cases} \Delta u_1 = f(\boldsymbol{x})_{x_1} - g_{3x_2}(\boldsymbol{x}) + g_{2x_2}(\boldsymbol{x}) = F_1(\boldsymbol{x}) \\ \Delta u_2 = f(\boldsymbol{x})_{x_2} + g_{3x_1}(\boldsymbol{x}) - g_{1x_3}(\boldsymbol{x}) = F_2(\boldsymbol{x}) \\ \Delta u_3 = f(\boldsymbol{x})_{x_3} - g_{2x_1}(\boldsymbol{x}) + g_{1x_2}(\boldsymbol{x}) = F_3(\boldsymbol{x}) \end{cases}$$

$$\Rightarrow \begin{cases} \Delta \boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{F}(\boldsymbol{x}) \qquad \text{in } \Omega \\ \boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{0} \qquad \text{on } \partial\Omega \end{cases} \tag{4.2.3}$$

The original proof for its first variational gradient in 2-D can be found from [25]. For convenience and comparison, we revise the proof in a more rigorous manner and extend it to 3-D, as follows.

Claim: Let a vector-valued function $\boldsymbol{A}(\boldsymbol{x})$ and a scalar function $B(\boldsymbol{x})$ satisfying

$$\Delta \boldsymbol{A}(\boldsymbol{x}) = \frac{\boldsymbol{G}(\boldsymbol{x})}{1 + \nabla \cdot \boldsymbol{u}(\boldsymbol{x})} \quad \text{and} \quad \Delta B(\boldsymbol{x}) = \frac{\boldsymbol{G}(\boldsymbol{x}) \cdot \boldsymbol{u}(\boldsymbol{x})}{(1 + \nabla \cdot \boldsymbol{u}(\boldsymbol{x}))^2}.$$

Then, first variational gradient of $SSD(\boldsymbol{\phi}(\boldsymbol{x}))$ with respect to $\boldsymbol{F}(\boldsymbol{x})$ is

$$\frac{\partial SSD(\boldsymbol{\phi}(\boldsymbol{x}))}{\partial \boldsymbol{F}(\boldsymbol{x})} = [\boldsymbol{A}(\boldsymbol{x}) + \nabla B(\boldsymbol{x})] \triangle t$$

Proof: Firstly, we need to see that

$$\begin{aligned} &\delta \boldsymbol{\phi}(\boldsymbol{x}) = (\frac{\delta \boldsymbol{u}(\boldsymbol{x})}{1 + \nabla \cdot \boldsymbol{u}(\boldsymbol{x})} + \frac{\boldsymbol{u}(\boldsymbol{x}) \nabla \cdot \delta \boldsymbol{u}(\boldsymbol{x})}{(1 + \nabla \cdot \boldsymbol{u}(\boldsymbol{x}))^2}) \triangle t \\ &\text{and } \delta \Delta \boldsymbol{u}(\boldsymbol{x}) = \Delta \delta \boldsymbol{u}(\boldsymbol{x}) = \delta \boldsymbol{F}(\boldsymbol{x}). \end{aligned} \tag{4.2.4}$$

Then, it can be derived from (4.2.2),

$$\begin{aligned}\delta SSD(\boldsymbol{\phi}(\boldsymbol{x})) &= \int_\Omega [I_{\boldsymbol{m}}(\boldsymbol{\phi}(\boldsymbol{x})) - I_{\boldsymbol{f}}(\boldsymbol{x})]\nabla I_{\boldsymbol{m}}(\boldsymbol{\phi}(\boldsymbol{x})) \cdot \delta\boldsymbol{\phi}(\boldsymbol{x}) d\boldsymbol{x} \\ &\quad (\text{ denote } \boldsymbol{G}(\boldsymbol{x}) = [I_{\boldsymbol{m}}(\boldsymbol{\phi}(\boldsymbol{x})) - I_{\boldsymbol{f}}(\boldsymbol{x})]\nabla I_{\boldsymbol{m}}(\boldsymbol{\phi}(\boldsymbol{x})) \text{ and by (2.2.3), we get }) \\ &= \int_\Omega \boldsymbol{G}(\boldsymbol{x}) \cdot (\frac{\delta\boldsymbol{u}(\boldsymbol{x})}{1+\nabla\cdot\boldsymbol{u}(\boldsymbol{x})} + \frac{\boldsymbol{u}(\boldsymbol{x})\nabla\cdot\delta\boldsymbol{u}(\boldsymbol{x})}{(1+\nabla\cdot\boldsymbol{u}(\boldsymbol{x}))^2})\triangle t d\boldsymbol{x} \\ &= \triangle t \int_\Omega \frac{\boldsymbol{G}(\boldsymbol{x})}{1+\nabla\cdot\boldsymbol{u}(\boldsymbol{x})} \cdot \delta\boldsymbol{u}(\boldsymbol{x}) + \frac{\boldsymbol{G}(\boldsymbol{x})\cdot\boldsymbol{u}(\boldsymbol{x})}{(1+\nabla\cdot\boldsymbol{u}(\boldsymbol{x}))^2}\nabla\cdot\delta\boldsymbol{u}(\boldsymbol{x}) d\boldsymbol{x}\end{aligned}$$

consider a vector-valued function $\boldsymbol{A}(\boldsymbol{x})$ and a scalar function $B(\boldsymbol{x})$ satisfying

$$\Delta\boldsymbol{A}(\boldsymbol{x}) = \frac{\boldsymbol{G}(\boldsymbol{x})}{1+\nabla\cdot\boldsymbol{u}(\boldsymbol{x})} \quad \text{and} \quad \Delta B(\boldsymbol{x}) = \frac{\boldsymbol{G}(\boldsymbol{x})\cdot\boldsymbol{u}(\boldsymbol{x})}{(1+\nabla\cdot\boldsymbol{u}(\boldsymbol{x}))^2} \tag{4.2.5}$$

Further,

$$\begin{aligned}\delta SSD(\boldsymbol{\phi}(\boldsymbol{x})) &= \triangle t \int_\Omega \Delta\boldsymbol{A}(\boldsymbol{x})\cdot\delta\boldsymbol{u}(\boldsymbol{x}) + \Delta B(\boldsymbol{x})\nabla\cdot\delta\boldsymbol{u}(\boldsymbol{x}) d\boldsymbol{x} \\ &\quad (\text{ by } Green's \text{ formulas with } \boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{0} \text{ on } \partial\Omega \,) \\ &= \triangle t \int_\Omega \boldsymbol{A}(\boldsymbol{x})\cdot\Delta\delta\boldsymbol{u}(\boldsymbol{x}) + B(\boldsymbol{x})\nabla\cdot\Delta\delta\boldsymbol{u}(\boldsymbol{x}) d\boldsymbol{x} \\ &\quad (\text{ by } Green's \text{ formulas with } \boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{0} \text{ on } \partial\Omega \,) \\ &= \triangle t \int_\Omega \boldsymbol{A}(\boldsymbol{x})\cdot\delta\Delta\boldsymbol{u}(\boldsymbol{x}) + \nabla B(\boldsymbol{x})\cdot\delta\Delta\boldsymbol{u}(\boldsymbol{x}) d\boldsymbol{x} \\ &= \triangle t \int_\Omega [\boldsymbol{A}(\boldsymbol{x}) + \nabla B(\boldsymbol{x})]\cdot\delta\boldsymbol{F}(\boldsymbol{x}) d\boldsymbol{x}\end{aligned}$$

Therefore, the first variational gradient with respect to the control function is

$$\frac{\partial SSD(\boldsymbol{\phi}(\boldsymbol{x}))}{\partial\boldsymbol{F}(\boldsymbol{x})} = [\boldsymbol{A}(\boldsymbol{x}) + \nabla B(\boldsymbol{x})]\triangle t \tag{4.2.6}$$

The above setting from (4.2.3) to (4.2.6) in defining $\boldsymbol{\phi}(\boldsymbol{x})$ was done with respect to the case 1 of our deformation method in [32]. A gradient descent based numerical scheme can be applied for implementation. Howeverm, without using the trick of a multi-resolution scheme, this method is not capable of 3D registration due to slow convergent rate.

4.2.1 New Developments of Our Image Registration

Start with a similar problem formulation as above. Let $I_{\boldsymbol{m}}$ be registered to $I_{\boldsymbol{f}}$ on the fixed and bounded domain $\Omega \subset \mathbb{R}^3$. Define the cost functional:

$$SSD(\boldsymbol{\phi}^k(\boldsymbol{x})) = \frac{1}{2}\int_\Omega [I_{\boldsymbol{m}}(\boldsymbol{\phi}^k(\boldsymbol{x})) - I_{\boldsymbol{f}}(\boldsymbol{x})]^2 d\boldsymbol{x} \tag{4.2.7}$$

To look for a diffeomorphism $\boldsymbol{\phi}^k : I_{\boldsymbol{f}} \to I_{\boldsymbol{m}}$ that minimizes $SSD(\boldsymbol{\phi}^k(\boldsymbol{x}))$:

$$\boldsymbol{\phi}^k(\boldsymbol{x}) = \boldsymbol{\phi}^{k-1}(\boldsymbol{x} + \boldsymbol{u}^k(\boldsymbol{x})) \tag{4.2.8}$$

subjects to the constraints $f^k(\boldsymbol{x}) > 0$ and $\boldsymbol{g}^k(\boldsymbol{x})$ with

$$\begin{cases} \nabla \cdot \boldsymbol{u}^k(\boldsymbol{x}) = f^k(\boldsymbol{x}) - 1 \\ \nabla \times \boldsymbol{u}^k(\boldsymbol{x}) = \boldsymbol{g}^k(\boldsymbol{x}) \\ \boldsymbol{u}^k(\boldsymbol{x}) = \boldsymbol{0} \qquad \text{on } \partial\Omega \end{cases} \text{in } \Omega \Rightarrow \begin{cases} \Delta u_1^k = f^k(\boldsymbol{x})_{x_1} - g^k_{3x_2}(\boldsymbol{x}) + g^k_{2x_3}(\boldsymbol{x}) = F_1^k(\boldsymbol{x}) \\ \Delta u_2^k = f^k(\boldsymbol{x})_{x_2} + g^k_{3x_1}(\boldsymbol{x}) - g^k_{1x_3}(\boldsymbol{x}) = F_2^k(\boldsymbol{x}) \\ \Delta u_3^k = f^k(\boldsymbol{x})_{x_3} - g^k_{2x_1}(\boldsymbol{x}) + g^k_{1x_2}(\boldsymbol{x}) = F_3^k(\boldsymbol{x}) \end{cases}$$

$$\Rightarrow \begin{cases} \Delta \boldsymbol{u}^k(\boldsymbol{x}) = \nabla f^k(\boldsymbol{x}) - \nabla \times \boldsymbol{g}^k(\boldsymbol{x}) = \boldsymbol{F}^k(\boldsymbol{x}) \qquad \text{in } \Omega \\ \boldsymbol{u}^k(\boldsymbol{x}) = \boldsymbol{0} \qquad \text{on } \partial\Omega \end{cases} \tag{4.2.9}$$

The derivation in case 3D (similar for 2D) goes as follows.

Claim: Consider $\Delta a_j^k(\boldsymbol{x}) = \sum_{i=1}^3 b_i^k(\boldsymbol{x})\phi_{iy_j}^{k-1}(\boldsymbol{x})\delta u_1^k(\boldsymbol{x})$ for $j = 1, 2, 3$, where

$$\begin{aligned} &\boldsymbol{b}^k(\boldsymbol{x}) = [I_{\boldsymbol{m}}(\boldsymbol{\phi}^k(\boldsymbol{x})) - I_{\boldsymbol{f}}(\boldsymbol{x})]\nabla I_{\boldsymbol{m}}(\boldsymbol{\phi}^k(\boldsymbol{x})), \\ &\delta\boldsymbol{\phi}^k(\boldsymbol{x}) = \delta(\boldsymbol{\phi}^{k-1}(\boldsymbol{x} + \boldsymbol{u}^k(\boldsymbol{x}))) = [\nabla\boldsymbol{\phi}^{k-1}(\boldsymbol{x} + \boldsymbol{u}^k(\boldsymbol{x}))] \cdot \delta\boldsymbol{u}^k(\boldsymbol{x}) \\ &\text{and } \Delta\delta\boldsymbol{u}^k(\boldsymbol{x}) = \delta\Delta\boldsymbol{u}^k(\boldsymbol{x}) = \delta\boldsymbol{F}^k(\boldsymbol{x}). \end{aligned} \tag{4.2.10}$$

Proof: It can be derived from (4.2.7)

$$\delta SSD(\boldsymbol{\phi}^k(\boldsymbol{x})) = \int_\Omega [I_{\boldsymbol{m}}(\boldsymbol{\phi}^k(\boldsymbol{x})) - I_{\boldsymbol{f}}(\boldsymbol{x})]\nabla I_{\boldsymbol{m}}(\boldsymbol{\phi}^k(\boldsymbol{x})) \cdot \delta\boldsymbol{\phi}^k(\boldsymbol{x})d\boldsymbol{x}$$

(Denote $\boldsymbol{b}^k(\boldsymbol{x}) = [I_{\boldsymbol{m}}(\boldsymbol{\phi}^k(\boldsymbol{x})) - I_{\boldsymbol{f}}(\boldsymbol{x})]\nabla I_{\boldsymbol{m}}(\boldsymbol{\phi}^k(\boldsymbol{x}))$ and by (4.2.10), we get)

$$= \int_\Omega \boldsymbol{b}^k(\boldsymbol{x})^\top([\nabla\boldsymbol{\phi}^{k-1}(\boldsymbol{x}+\boldsymbol{u}^k(\boldsymbol{x}))] \cdot \delta\boldsymbol{u}^k(\boldsymbol{x}))d\boldsymbol{x}$$

$$= \int_\Omega \begin{bmatrix} b_1^k(\boldsymbol{x}) \\ b_2^k(\boldsymbol{x}) \\ b_3^k(\boldsymbol{x}) \end{bmatrix}^\top \begin{bmatrix} \phi_{1y_1}^{k-1}(\boldsymbol{x}) & \phi_{1y_2}^{k-1}(\boldsymbol{x}) & \phi_{1y_3}^{k-1}(\boldsymbol{x}) \\ \phi_{2y_1}^{k-1}(\boldsymbol{x}) & \phi_{2y_2}^{k-1}(\boldsymbol{x}) & \phi_{2y_3}^{k-1}(\boldsymbol{x}) \\ \phi_{3y_1}^{k-1}(\boldsymbol{x}) & \phi_{3y_2}^{k-1}(\boldsymbol{x}) & \phi_{3y_3}^{k-1}(\boldsymbol{x}) \end{bmatrix} \begin{bmatrix} \delta u_1^k(\boldsymbol{x}) \\ \delta u_2^k(\boldsymbol{x}) \\ \delta u_3^k(\boldsymbol{x}) \end{bmatrix} d\boldsymbol{x}, \text{ where } (\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{u}^k(\boldsymbol{x}))$$

$$= \int_\Omega \sum_{i=1}^3 b_i^k(\boldsymbol{x})\phi_{iy_1}^{k-1}(\boldsymbol{x})\delta u_1^k(\boldsymbol{x}) + \sum_{i=1}^3 b_i^k(\boldsymbol{x})\phi_{iy_2}^{k-1}(\boldsymbol{x})\delta u_1^k(\boldsymbol{x}) + \sum_{i=1}^3 b_i^k(\boldsymbol{x})\phi_{iy_3}^{k-1}(\boldsymbol{x})\delta u_1^k(\boldsymbol{x})d\boldsymbol{x},$$

(consider $\Delta a_j^k(\boldsymbol{x}) = \sum_{i=1}^3 b_i(\boldsymbol{x})\phi_{iy_j}^{k-1}(\boldsymbol{x})\delta u_1^k(\boldsymbol{x})$ for $j = 1, 2, 3$, we get)

$$= \int_\Omega \Delta a_1^k(\boldsymbol{x})\delta u_1^k(\boldsymbol{x}) + \Delta a_2^k(\boldsymbol{x})\delta u_2^k(\boldsymbol{x}) + \Delta a_3^k(\boldsymbol{x})\delta u_3^k(\boldsymbol{x})d\boldsymbol{x}$$

$$= \int_\Omega \Delta\boldsymbol{a}^k(\boldsymbol{x}) \cdot \delta\boldsymbol{u}^k(\boldsymbol{x})d\boldsymbol{x}$$

(by $Green's$ formulas with $\boldsymbol{u}^k(\boldsymbol{x}) = \boldsymbol{0}$ on $\partial\Omega$ and by (4.2.10), we get)

$$= \int_\Omega \boldsymbol{a}^k(\boldsymbol{x}) \cdot \Delta\delta\boldsymbol{u}^k(\boldsymbol{x})d\boldsymbol{x}$$

$$= \int_\Omega \boldsymbol{a}^k(\boldsymbol{x}) \cdot \delta\Delta\boldsymbol{u}^k(\boldsymbol{x})d\boldsymbol{x}$$

$$= \int_\Omega \boldsymbol{a}^k(\boldsymbol{x}) \cdot \Delta\delta\boldsymbol{F}^k(\boldsymbol{x})d\boldsymbol{x}$$

Therefore,

$$\frac{\partial SSD(\boldsymbol{\phi}^k(\boldsymbol{x}))}{\partial \boldsymbol{F}^k(\boldsymbol{x})} = \boldsymbol{a}^k(\boldsymbol{x}) \qquad \text{or} \qquad \frac{\partial SSD(\phi_j^k(\boldsymbol{x}))}{\partial F_j^k(\boldsymbol{x})} = a_j^k(\boldsymbol{x}) \text{ for } j = 1, 2, 3. \quad (4.2.11)$$

4.3 A Numerical Algorithm based on Gradient Descent

As the variational gradient (4.2.11) is derived, a gradient-descent numerical scheme can be implemented by the following algorithm.

Algorithm 5 Optimal Control Approach to Image Registration

- 0: input $I_{\boldsymbol{m}}$, $I_{\boldsymbol{f}}$;
- 1: set $\boldsymbol{F}_0 = \boldsymbol{0}$, $\boldsymbol{a}_0 = \boldsymbol{0}$, $\boldsymbol{\phi}_0(\boldsymbol{x}) = \boldsymbol{id}(\boldsymbol{x})$, $iter_{max}$, $tstep$, $t_{up} > 1$, $t_{down} \in (0, 1)$, t_{tol};
- 2 while $iter < iter_{max}$ & $tstep < t_{tol}$ & $ratio > ratio_{tol}$;
  - 3: if *better*
    - 4: solve for $\boldsymbol{a}_k$ by FFT, where $\Delta \boldsymbol{a}_k(\boldsymbol{x}) = \boldsymbol{b}_k(\boldsymbol{x})^{\mathsf{T}}[\nabla \boldsymbol{\phi}_{k-1}(\boldsymbol{x} + \boldsymbol{u}_k(\boldsymbol{x}))]$;
  - 5: update $\boldsymbol{F}_k = \boldsymbol{F}_{k-1} - tstep * \boldsymbol{a}_k$;
  - 6: solve $\Delta \boldsymbol{u}_k(\boldsymbol{x}) = \boldsymbol{F}_k$ to form $\boldsymbol{\phi}_k(\boldsymbol{x}) \approx \boldsymbol{x} + \boldsymbol{u}_k(\boldsymbol{x})$;
  - 7: re-sample $I_{\boldsymbol{m}}(\boldsymbol{\phi}_k(\boldsymbol{x}))$ by interpolation and check SSD_k;
  - 8: if SSD_k decrease,
    - 9: $better = true$;
    - 10: $tstep = tstep * t_{up}$;
    - 11: $\boldsymbol{F}_{k-1} = \boldsymbol{F}_k$;
    - 12: $\boldsymbol{\phi}_{k-1} = \boldsymbol{\phi}_k$;

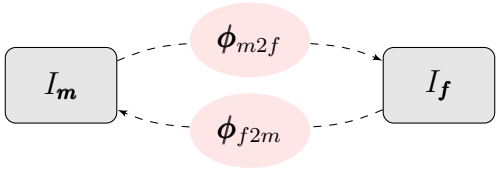
    else,
    - 13: $better = false$;
    - 14: $tstep = tstep * t_{down}$;

    go to 2;
- 15: output $\boldsymbol{\phi}_k$, $\boldsymbol{u}_k$, $I_{\boldsymbol{m}}(\boldsymbol{\phi}_k)$.

4.3.1 Inverse Consistency with 2D Numerical Demonstration

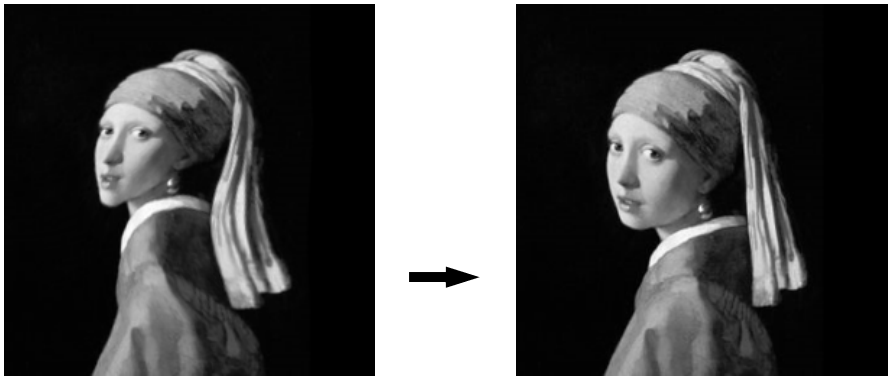
The inverse consistency is one of many standards of a good image registration method. In [12], the inverse consistency for registration method is studied in theory and tested numerically. Here, we numerical test directly if the two registration transformations are symmetric between $I_{\boldsymbol{m}}$ and $I_{\boldsymbol{f}}$, i.e., if the compositions of these

two result transformations, namely, $\boldsymbol{\phi}_{m2f} \circ \boldsymbol{\phi}_{f2m}$ and $\boldsymbol{\phi}_{f2m} \circ \boldsymbol{\phi}_{m2f}$, are close to $\boldsymbol{id}$. The illustration can be viewed from the diagram.

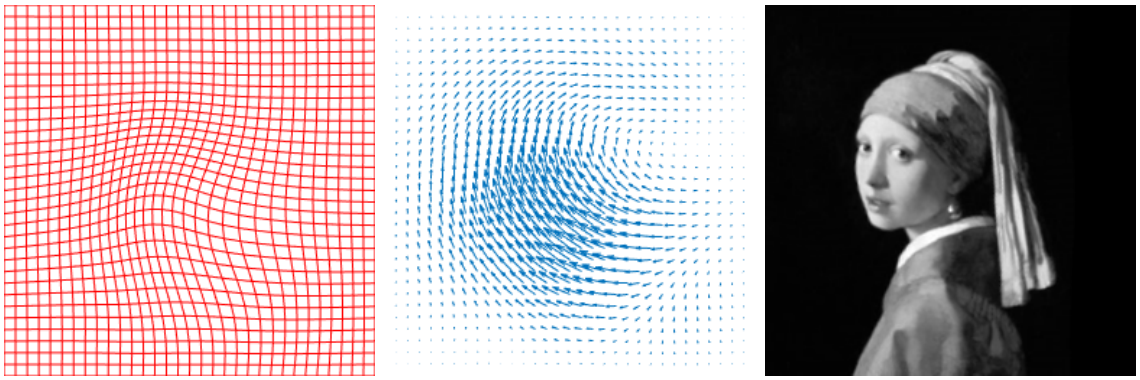


Example 1: The Girl with a Pearl Earring

The next Figure register a distorted image $I_{\boldsymbol{m}}$ to the original image $I_{\boldsymbol{f}}$. It shows the result of using the proposed algorithm is effective.



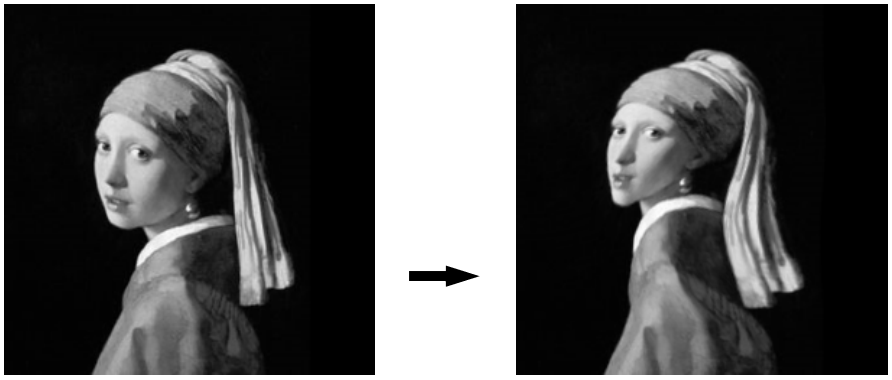
(a) $I_{\boldsymbol{m}}$ (b) $I_{\boldsymbol{f}}$



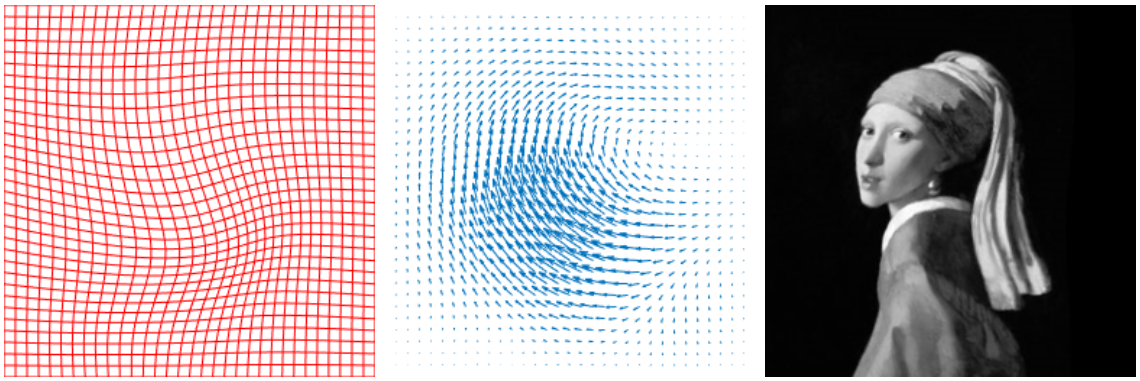
(c) $\boldsymbol{\phi}_{m2f} = \boldsymbol{\phi}_k$ (d) $\boldsymbol{u}_k$ (e) $I_{\boldsymbol{m}}(\boldsymbol{\phi}_{m2f})$

Figure 4.1: Register $I_{\boldsymbol{m}}$ to $I_{\boldsymbol{f}}$

The *ratio* is reduced to 1.88% in 400 iterations and the elapsed time took about 15.24 seconds. Next Figure recorded the opposite direction of above. It is registered from $I_{\boldsymbol{f}}$ to $I_{\boldsymbol{m}}$. The *ratio* is reduced to 1.22% in 400 iterations and the elapsed time took about 15.39 seconds.



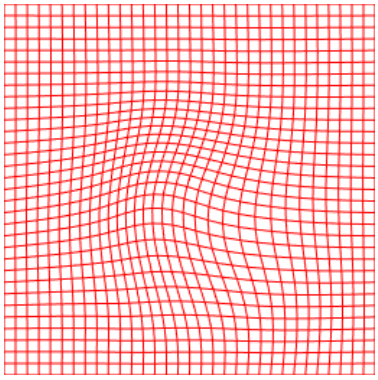
(a) $I_{\boldsymbol{f}}$ (b) $I_{\boldsymbol{m}}$



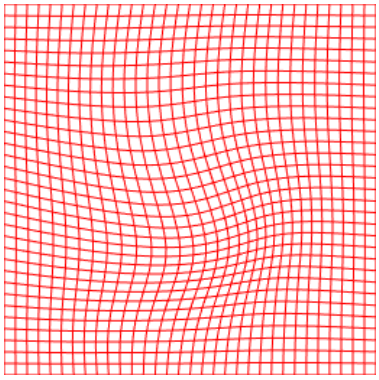
(c) $\boldsymbol{\phi}_{f2m}$ (d) $\boldsymbol{v}_k$ (e) $I_{\boldsymbol{f}}(\boldsymbol{\phi}_{f2m})$

Figure 4.2: Register $I_{\boldsymbol{f}}$ to $I_{\boldsymbol{m}}$

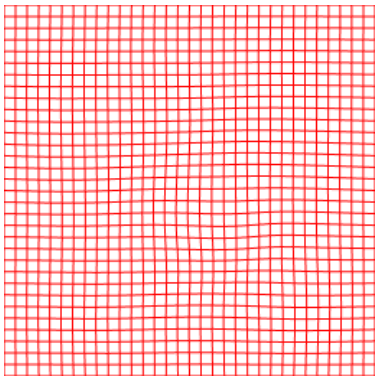
The numerical test gives positive answers as Fig.7 shows.



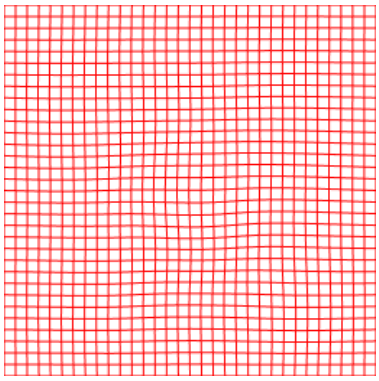
(a) $\boldsymbol{\phi}_{m2f}$



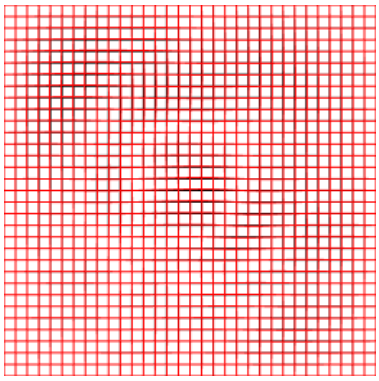
(b) $\boldsymbol{\phi}_{f2m}$



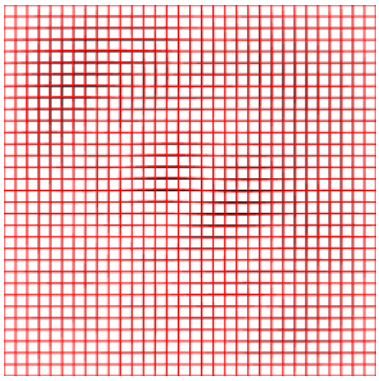
(c) $\boldsymbol{\phi}_{m2f} \circ \boldsymbol{\phi}_{f2m} \approx \boldsymbol{id}$



(d) $\boldsymbol{\phi}_{f2m} \circ \boldsymbol{\phi}_{m2f} \approx \boldsymbol{id}$



(e) $\boldsymbol{\phi}_{m2f} \circ \boldsymbol{\phi}_{f2m}$-red superimposes $\boldsymbol{id}$-black



(f) $\boldsymbol{\phi}_{f2m} \circ \boldsymbol{\phi}_{m2f}$-red superimposes $\boldsymbol{id}$-black

Figure 4.3: Compositions are close to $\boldsymbol{id}$

We may visually see those compositions are almost identical to the $\boldsymbol{id}$ as desired. And table below record 2 important measures of the resulting compositions. It demonstrate how close these compositions are to $\boldsymbol{id}$. This result suggest our registration method satisfies the inverse consistency property [47].

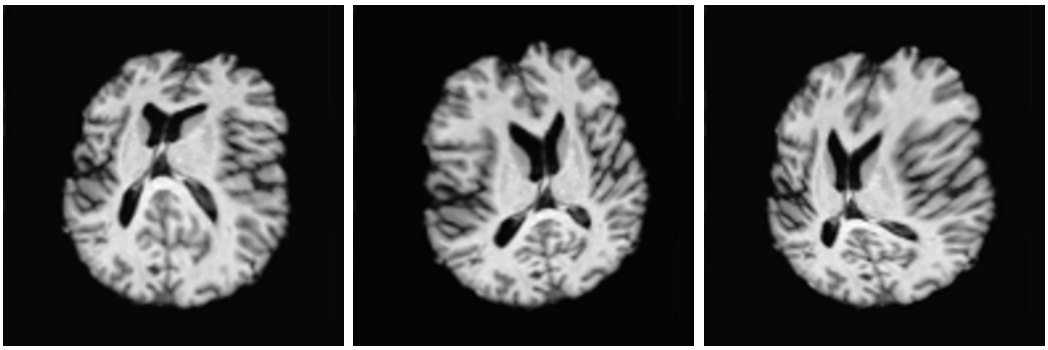
Table 4.1: Inverse Consistency

	$\max\|\|\boldsymbol{\Phi} - \boldsymbol{id}\|\|$	$\max\|\|\boldsymbol{\Phi} - \boldsymbol{id}\|\|/N$ with domain $N^2 = 256^2$
$\boldsymbol{\Phi} = \boldsymbol{\phi}_{m2f} \circ \boldsymbol{\phi}_{f2m}$	1.97	$7.70{*}10^{-3}$
$\boldsymbol{\Phi} = \boldsymbol{\phi}_{f2m} \circ \boldsymbol{\phi}_{m2f}$	2.09	$8.16{*}10^{-3}$

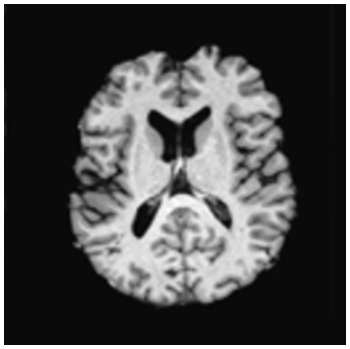
4.3.2 Transitivity through Unbiased Template

Furthermore, we check if our optimal control image registration method satisfies the transitivity property by following a similar work flow as we did for inverse consistency property. Let's consider a new image pool $\{I_1, I_2, I_3\}$ and an unbiased template I_R (Construction of an unbiased template of given image is introduced in chapter 5) of $\{I_1, I_2, I_3\}$ are given.

Example 2: Transitivity of 3 distorted 2D images



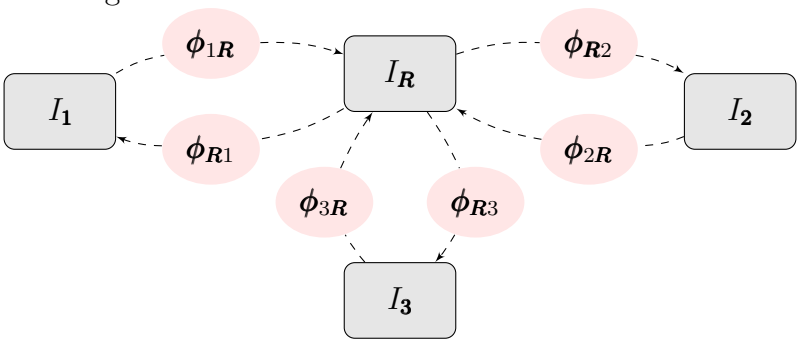
(a) I_1 (b) I_2 (c) I_3

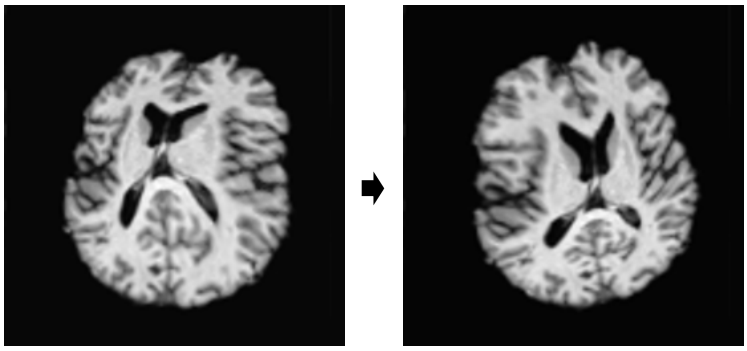


(d) $I_{\boldsymbol{R}}$

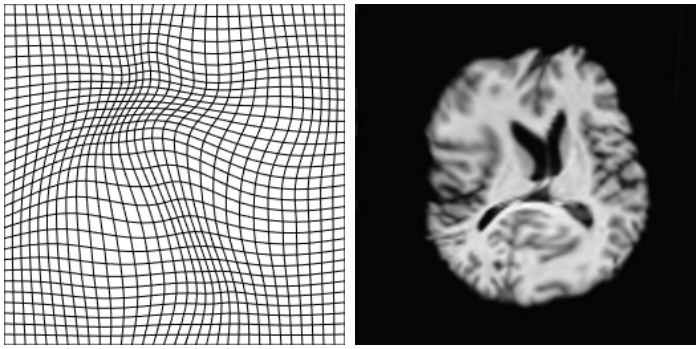
Figure 4.4: Image pool and its unbiased template

We wil numerically test that whether the compositions of two output transformations $\boldsymbol{\phi}_{12}$ and $\boldsymbol{\phi}_{21}$, namely $\boldsymbol{\phi}_{23} \circ \boldsymbol{\phi}_{12}$, is close to the and $\boldsymbol{\phi}_{12}$, which $\boldsymbol{\phi}_{12}$, $\boldsymbol{\phi}_{23}$ and $\boldsymbol{\phi}_{13}$ are acquired through using an unbiased template. The next 3 Figures demonstrated the construction of $\boldsymbol{\phi}_{12} = \boldsymbol{\phi}_{\boldsymbol{R}2} \circ \boldsymbol{\phi}_{1\boldsymbol{R}}$, $\boldsymbol{\phi}_{23} = \boldsymbol{\phi}_{\boldsymbol{R}3} \circ \boldsymbol{\phi}_{2\boldsymbol{R}}$ and $\boldsymbol{\phi}_{13} = \boldsymbol{\phi}_{\boldsymbol{R}3} \circ \boldsymbol{\phi}_{1\boldsymbol{R}}$; and each of back-and-forth transformations are found using our registration method, which is illustrated in the diagram.



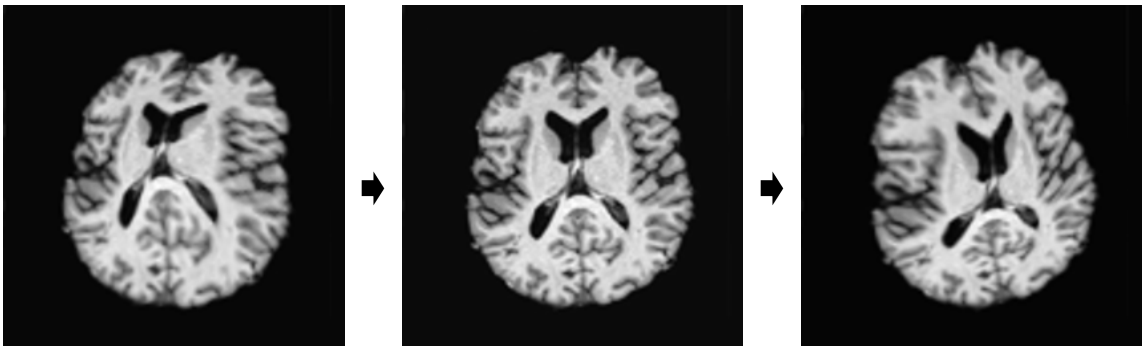


(a) I_1 (b) I_2

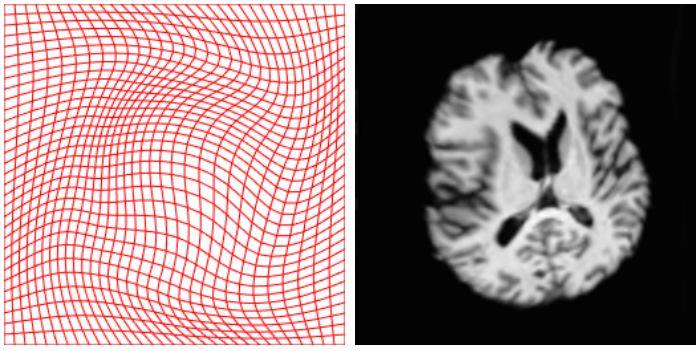


(c) $\hat{\boldsymbol{\phi}}_{12}$ not converged (d) $I_1(\hat{\boldsymbol{\phi}}_{12})$

Figure 4.5: Direct Register not Converged: I_1 to I_2



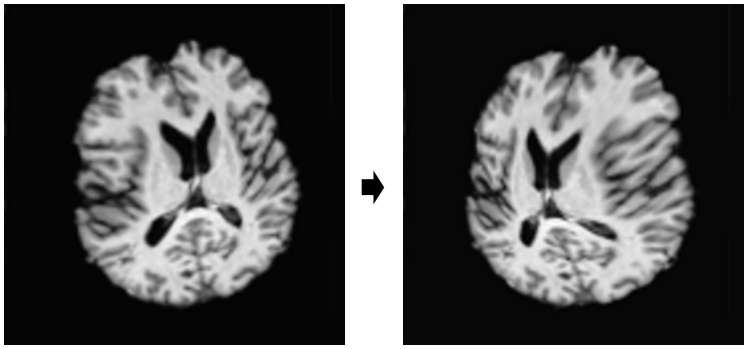
(a) I_1 (b) $I_{\boldsymbol{R}}$ (c) I_2



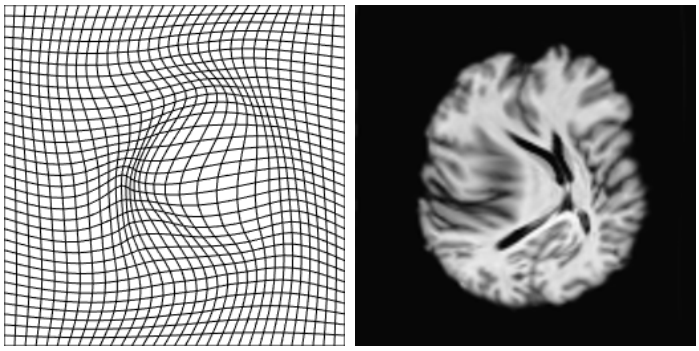
(d) $\boldsymbol{\phi}_{12} = \boldsymbol{\phi}_{\boldsymbol{R}2} \circ \boldsymbol{\phi}_{1\boldsymbol{R}}$ (e) $I_1(\boldsymbol{\phi}_{12})$

Figure 4.6: Register with Unbiased Template from I_1 to I_2

The *ratio* is reduced to 2.40% with using $I_{\boldsymbol{R}}$ as the bridge image.

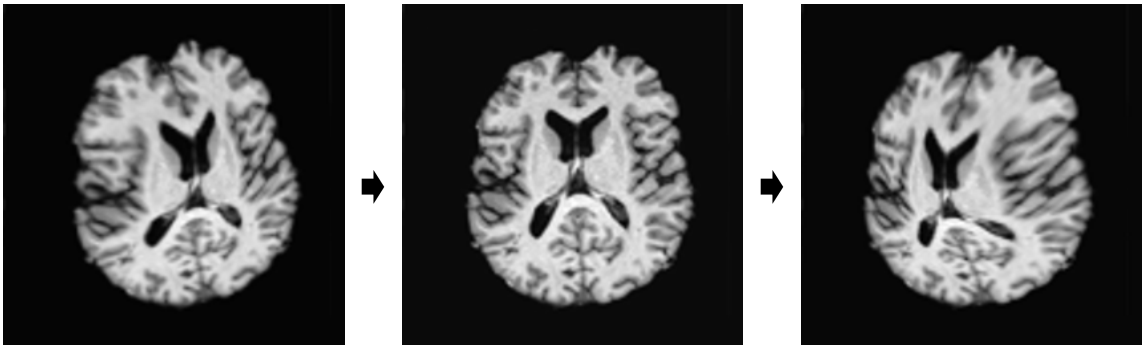


(a) I_2 (b) I_3

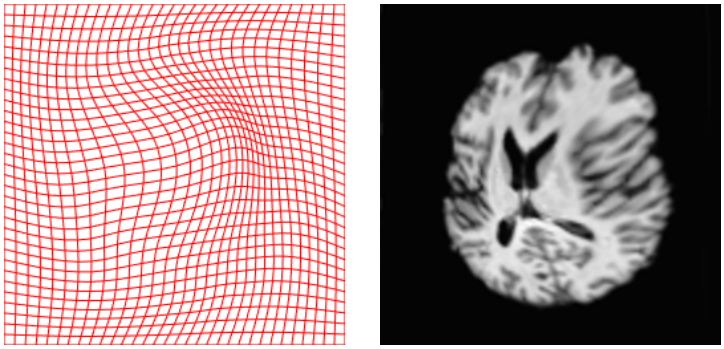


(c) $\hat{\boldsymbol{\phi}}_{23}$ not converged (d) $I_2(\hat{\boldsymbol{\phi}}_{23})$

Figure 4.7: Direct Register not Converged: I_1 to I_2



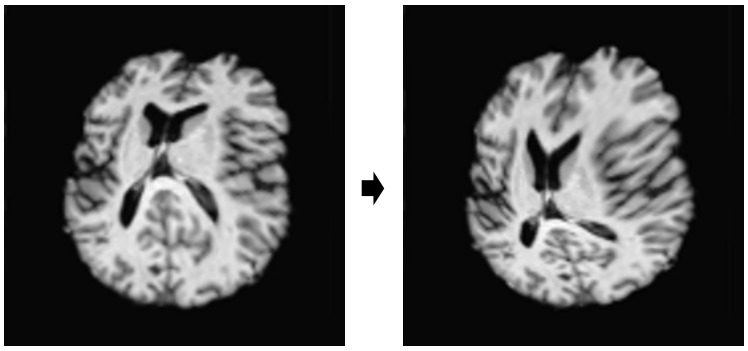
(a) I_2 (b) I_R (c) I_3



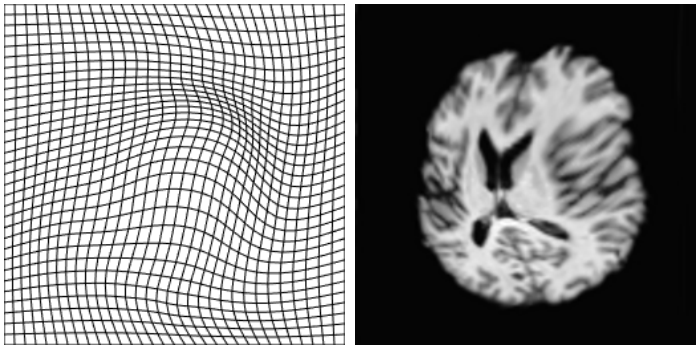
(d) $\boldsymbol{\phi}_{23} = \boldsymbol{\phi}_{\boldsymbol{R}3} \circ \boldsymbol{\phi}_{2\boldsymbol{R}}$ (e) $I_2(\boldsymbol{\phi}_{23})$

Figure 4.8: Register with Unbiased Template from I_2 to I_3

The *ratio* is reduced to 1.29% with using $I_{\boldsymbol{R}}$ as the bridge image.

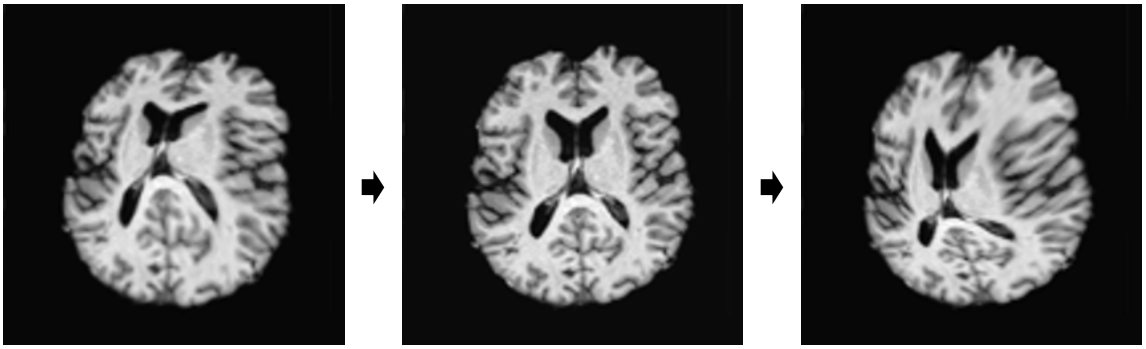


(a) I_1 (b) I_3

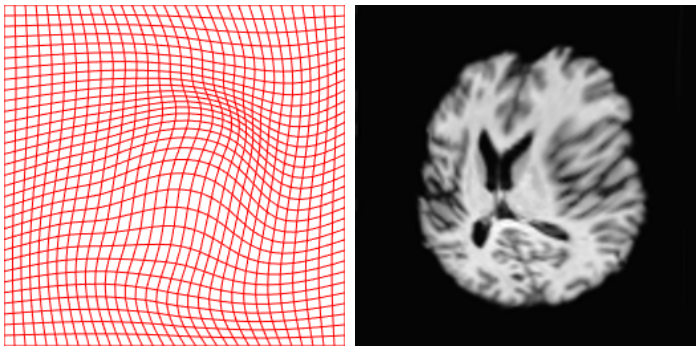


(c) $\hat{\boldsymbol{\phi}}_{13}$ converged (d) $I_1(\hat{\boldsymbol{\phi}_{13}})$

Figure 4.9: Direct Register Converged: I_1 to I_3



(a) I_1 (b) $I_{\boldsymbol{R}}$ (c) I_3



(d) $\boldsymbol{\phi}_{13} = \boldsymbol{\phi}_{\boldsymbol{R}3} \circ \boldsymbol{\phi}_{1\boldsymbol{R}}$ (e) $I_1(\boldsymbol{\phi}_{13})$

Figure 4.10: Register with Unbiased Template from I_2 to I_3

The *ratio* is reduced to 1.32% with using $I_{\boldsymbol{R}}$ as the bridge image.

In the case that a direct registration converged, a composed registration transformation coincides with the registration transformation found by direct transformation. The next few figures of this section demonstrate the transitivity property of our computational results.

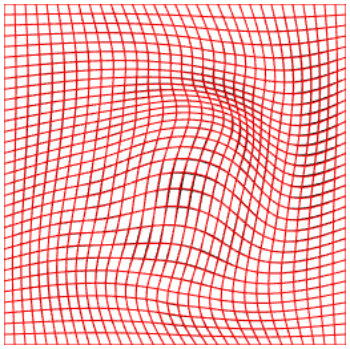
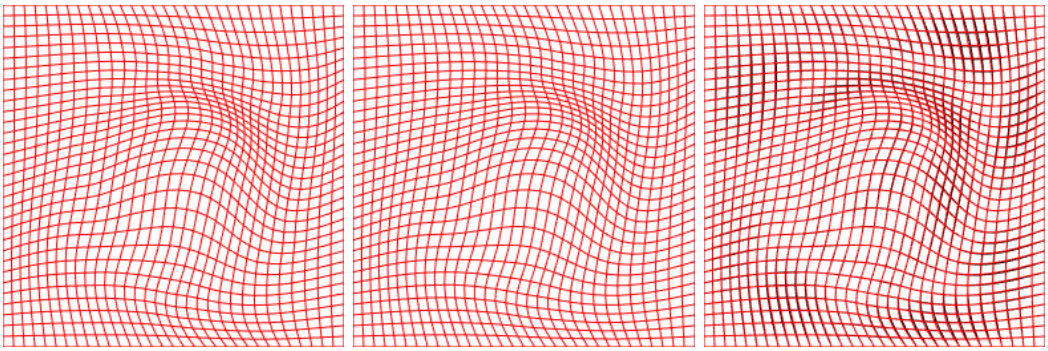


Figure 4.11: $\hat{\boldsymbol{\phi}}_{13}$-red superimposed on $\boldsymbol{\phi}_{13}$-balck

Table 4.2: Transitivity 1

	$\max\|\|\boldsymbol{\Phi} - \boldsymbol{\phi}_{13}\|\|$	$\max\|\|\hat{\boldsymbol{\phi}}_{13} - \boldsymbol{\phi}_{13}\|\|/N$, with domain $N^2 = 128^2$
$\boldsymbol{\Phi} = \boldsymbol{\phi}_{\boldsymbol{R}3} \circ \boldsymbol{\phi}_{1\boldsymbol{R}}$	0.47	$1.83{*}10^{-3}$



(a) $\boldsymbol{\phi}_{23} \circ \boldsymbol{\phi}_{12}$ (b) $\boldsymbol{\phi}_{13} = \boldsymbol{\phi}_{\boldsymbol{R}3} \circ \boldsymbol{\phi}_{1\boldsymbol{R}}$ (c) $\boldsymbol{\phi}_{13}$-red superimposed on $\boldsymbol{\phi}_{23} \circ \boldsymbol{\phi}_{12}$-black

Figure 4.12: $\boldsymbol{\phi}_{23} \circ \boldsymbol{\phi}_{12} \approx \boldsymbol{\phi}_{13}$

Table 4.3: Transitivity 2

	$\max\|\|\boldsymbol{\Phi} - \boldsymbol{\phi}_{13}\|\|$	$\max\|\|\boldsymbol{\Phi} - \boldsymbol{\phi}_{13}\|\|/N$, with domain $N^2 = 128^2$
$\boldsymbol{\Phi} = \boldsymbol{\phi}_{23} \circ \boldsymbol{\phi}_{12}$	0.61	$2.38{*}10^{-3}$

This example confirms that our registration method is transitive. It also indicates when a direct registration doesn't converge, with the help of using an unbiased template, a composed registration transformation can still be constructed.

Example 4: Registration of 3D Teapot Images by Optimal Control Method

The result shows that the revised version can reach a slightly better result in a slightly shorter elapsed time than the original one on 2D cases by implementing the proposed Algorithm. We should expect a much more significant improvement on a 3D example test. We now describe the volume image I_0 and the twisted image I_t in Step-1. In Step-2, we register I_t to I_0, and out put the registration transformation and the deformed image I_t. The results show that the twisted image I_t is deformed back to the I_0 with high accuracy.

Step-1: A teapot is rotated about its vertical axis passing through the tip of its lid. Snapshots are taken from a fixed camera at 5° intervals. A total of 72 photos are taken as the teapot completes 360° rotation and return to the initial position. Each of these photos is re-sampled as a 72×72 grayscale image I_{0i}. These 72 images I_{0i} for $i = 1, \ldots, 72$ are used to form a three dimensional volume image of size $72 \times 72 \times 72$. We refer to this volume image as the Rotating Teapot Image, denoted as I_0.

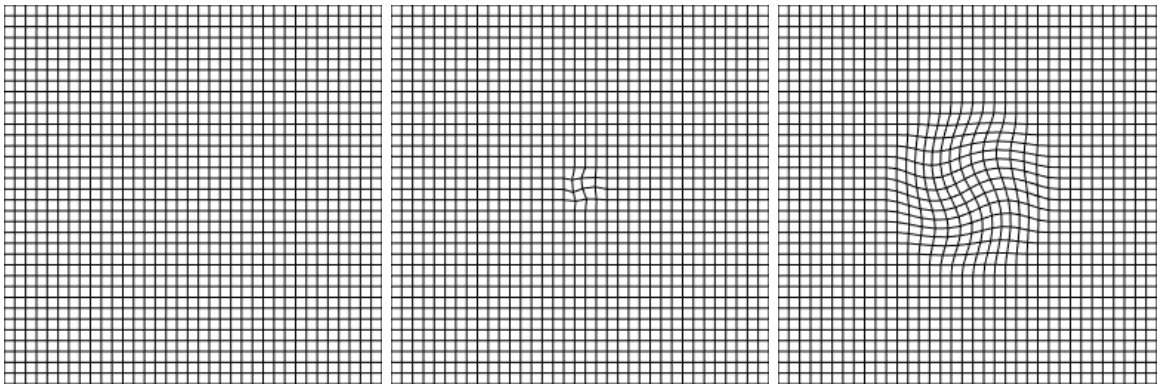
(a) 0° (b) 45° (c) 90°

(d) 135° (e) 180° (f) 225°

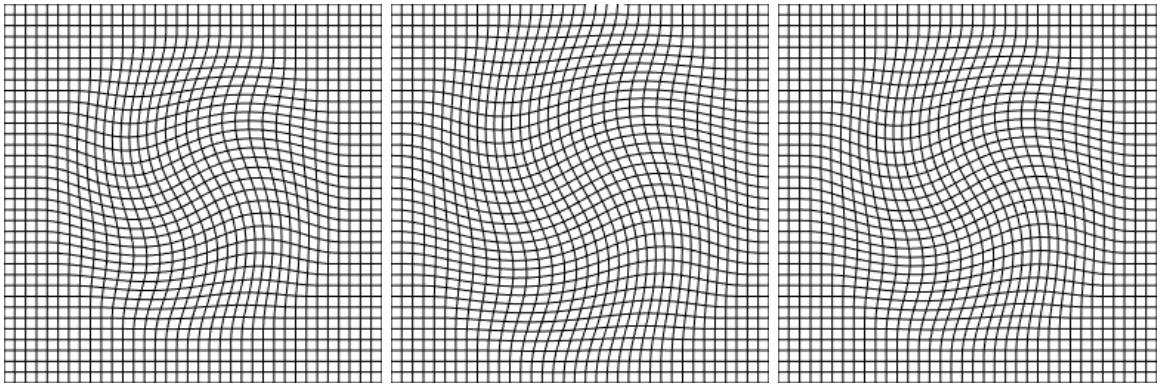
(g) 270° (h) 315° (i) 360°

Figure 4.13: Teapot—I_0

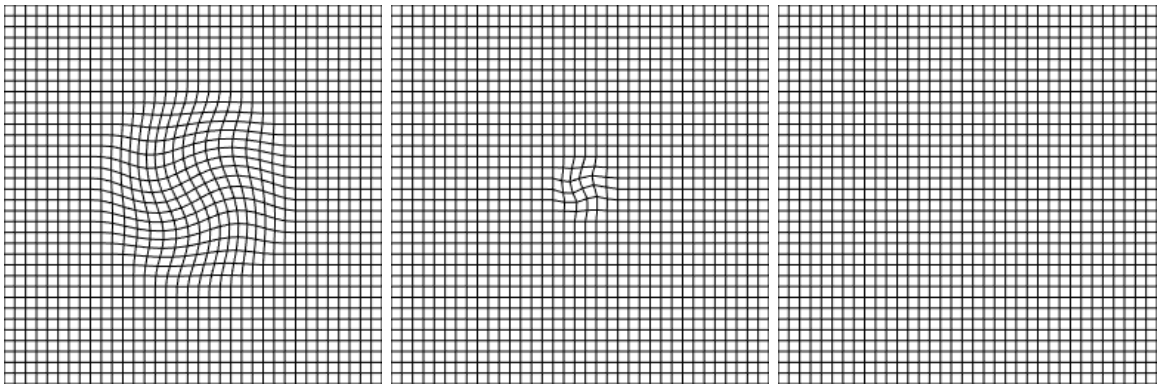
As the teapot rotates, we deform each slice I_{0i} (of I_0) to I_{ti} by a rotation $\boldsymbol{T}_i$ as show in next Figure that is cut-off near the boundary for $i = 1, \ldots, 72$.



(a) $\boldsymbol{T}_1$ (b) $\boldsymbol{T}_9$ (c) $\boldsymbol{T}_{18}$



(d) $\boldsymbol{T}_{27}$ (e) $\boldsymbol{T}_{36}$ (f) $\boldsymbol{T}_{45}$



(g) $\boldsymbol{T}_{54}$ (h) $\boldsymbol{T}_{63}$ (i) $\boldsymbol{T}_{72}$

Figure 4.14: $\boldsymbol{T}_i$'s

These 72 twisted 72×72 images form a $72 \times 72 \times 72$ volume image referred to as the Twisted Teapot, denoted as I_t as shown in the next Figure.

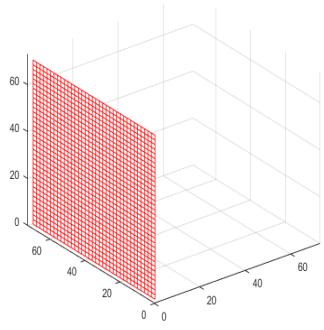
(a) 0° (b) 45° (c) 90°

(d) 135° (e) 180° (f) 225°

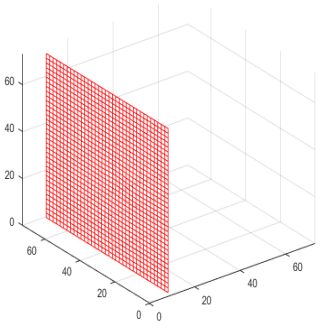
(g) 270° (h) 315° (i) 360°

Figure 4.15: Twisted Teapot—$I_t = \boldsymbol{T}(I_0)$

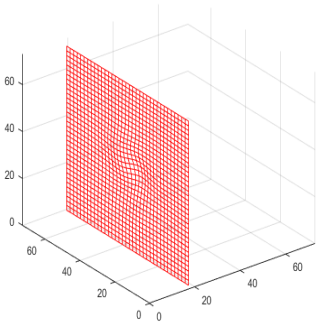
The deformations $\boldsymbol{T}_i$ for $i = 1, \ldots, 72$ and the Twisted Teapot image I_t can be seen from Twisted Teapot (CLICK HERE). The resulting registration deformation $\boldsymbol{\phi}$, restricted to each ith slice, is expected to be very close to $\boldsymbol{T}_i$ as shown in the next Figure.



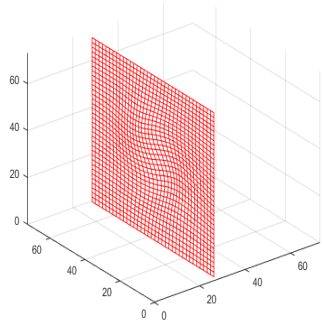
(a) $\boldsymbol{\phi}_1$



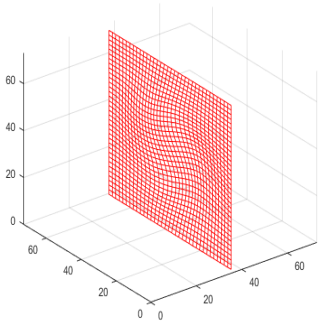
(b) $\boldsymbol{\phi}_9$



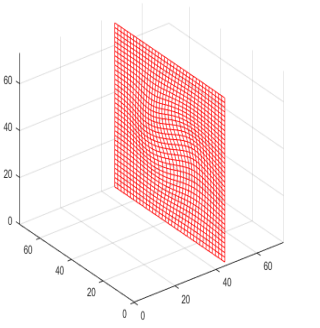
(c) $\boldsymbol{\phi}_{18}$



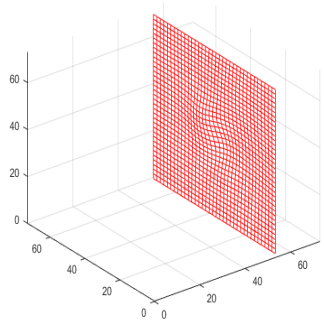
(d) $\boldsymbol{\phi}_{27}$



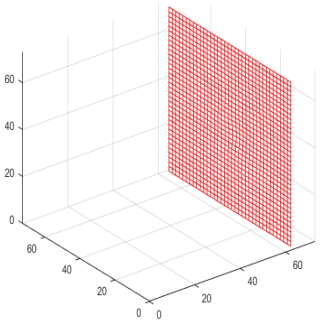
(e) $\boldsymbol{\phi}_{36}$



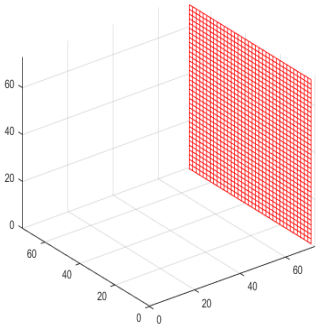
(f) $\boldsymbol{\phi}_{45}$



(g) $\boldsymbol{\phi}_{54}$



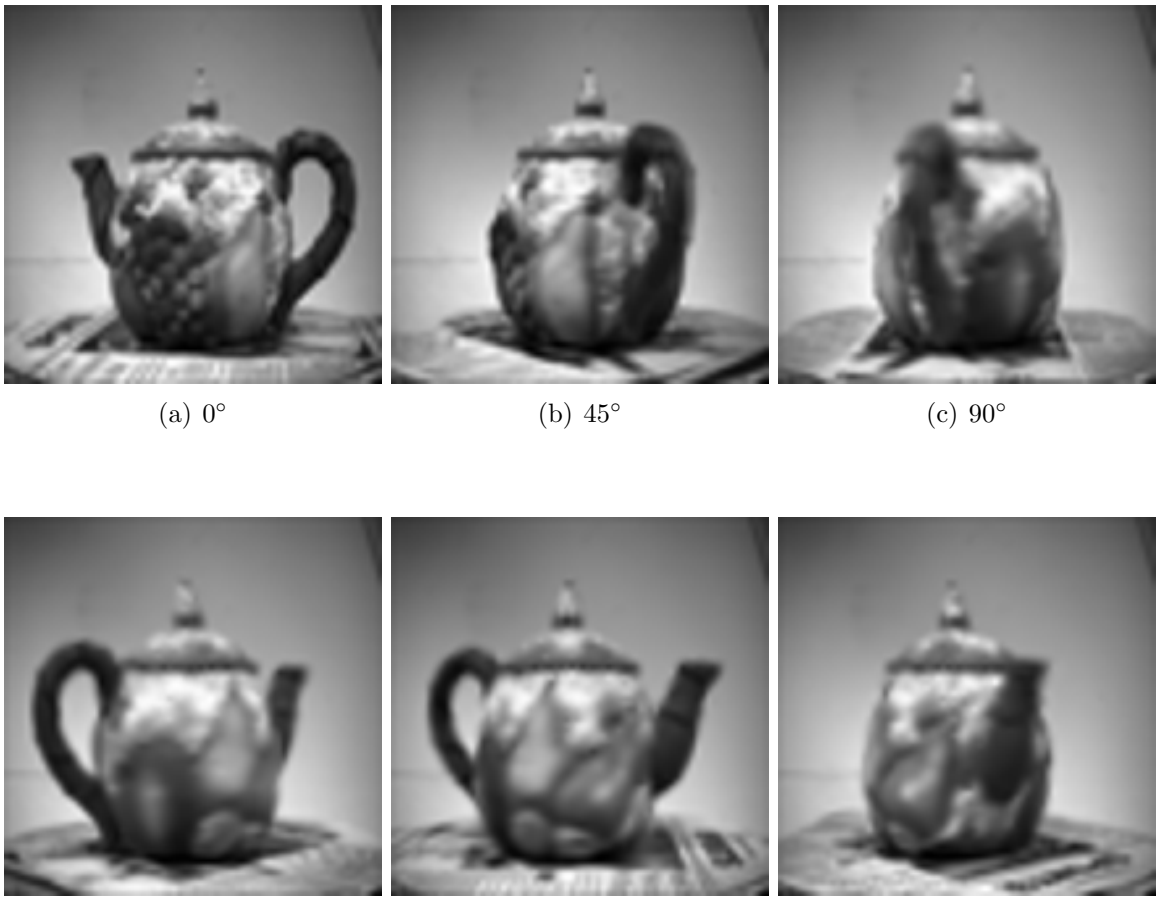
(h) $\boldsymbol{\phi}_{63}$



(i) $\boldsymbol{\phi}_{72}$

Figure 4.16: $\boldsymbol{\phi}_i$'s

and the deformed images of the twisted teapot — $I_t(\boldsymbol{\phi})$ is expected to be close to I_0 as it shows on next Figure. The total elapsed time for Step-2 is 927.601032 seconds.



(a) 0° (b) 45° (c) 90°

(d) 135° (e) 180° (f) 225°

(g) 270° (h) 315° (i) 360°

Figure 4.17: Reversed Twisted Teapot—$I_t(\boldsymbol{\phi})$

The registration deformation $\boldsymbol{\phi}$ from Step-2 and the corresponding reversed twisted teapot image $I_t = \boldsymbol{T}(I_0)$ are shown from: Reversed Twisted Teapot (CLICK HERE). From these visualization files, we conclude that our optimal control method correctly recovered the ground truth deformations $\boldsymbol{T}_i$ for $i = 1, \ldots, 72$ and the ground truth images I_{0i} for $i = 1, \ldots, 72$, as expected.

(a) 0° (b) 45° (c) 90°

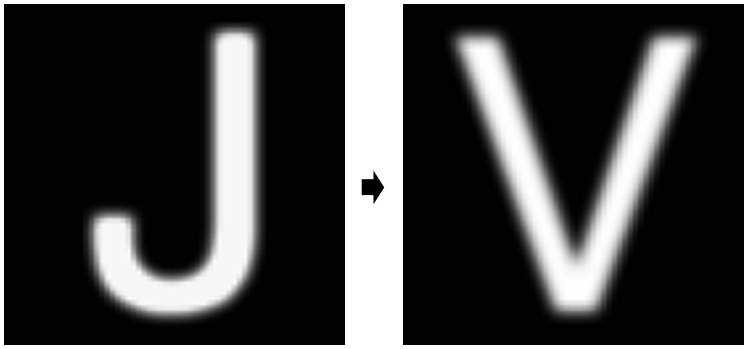
(d) 135° (e) 180° (f) 225°

(g) 270° (h) 315° (i) 360°

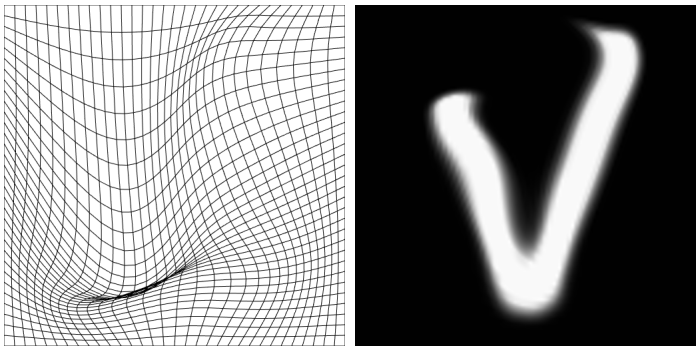
Figure 4.18: Reversed Twisted Teapot—$I_t(\boldsymbol{\phi})$

4.4 Jacobian Determinant Monitored Algorithm for Image Registration

From the discovery of transitivity of our optimal control approach to image registration, a modified version of Algorithm 5 is formulated, which produces better transformations in terms of the distribution of Jacobian determinant. Let us take a look on a very challenging registration problem, which attempts to register an image of letter "J" to an image of letter "V". It is implemented with Algorithm 5.



(a) I_J (b) I_V



(c) $\hat{\boldsymbol{\phi}}_{J2V}$ not converged (d) $I_J(\hat{\boldsymbol{\phi}}_{J2V})$

Figure 4.19: Direct Register: I_J to I_V

In fact, the implementation overflowed as $tstep < t_{tol}$ where it stopped at a local minimum. The worse part is that $\det\nabla(\hat{\boldsymbol{\phi}}_{J2V}) < 0$ happened at locations, which destroyed the diffeomophism as a zoomed in view is shown in a latter Figure.

Table 4.4: Direct I_J to I_V

$tstep$	$ratio$	$iter$	$Elapsed\ Time$	JD_{max}	JD_{min}
$8.9526 * 10^{-17}$	0.13	270	1.0623 sec	2.5788	-0.1797

It is known that if $\boldsymbol{\Phi} = \boldsymbol{\phi}_1(\boldsymbol{\phi}_2) = \boldsymbol{\phi}_1 \circ \boldsymbol{\phi}_2$, then $\det\nabla(\boldsymbol{\Phi}) = \det\nabla(\boldsymbol{\phi}_1)\det\nabla(\boldsymbol{\phi}_2)$. By the transitivity of our computational results, we may monitor the positivity of $\det\nabla(\boldsymbol{\phi})$ in each iteration. Let $\boldsymbol{\Phi} = \boldsymbol{\phi}_{k-1}(\boldsymbol{\phi}_k)$ where $\det\nabla(\boldsymbol{\phi}_k) > JD_{minTol} \in (0, 1)$ for

all $k > 0$. Then, the value $\det\nabla(\boldsymbol{\Phi}) = \det\nabla(\boldsymbol{\phi}_{k-1})\det\nabla(\boldsymbol{\phi}_k) > 0$ is assured. So, to prevent $min(\det\nabla(\boldsymbol{\Phi}))$ become negative and destroy diffeomorphism, we formulated Algorithm 6 — a $\det\nabla$ Monitored version of Algorithm 5.

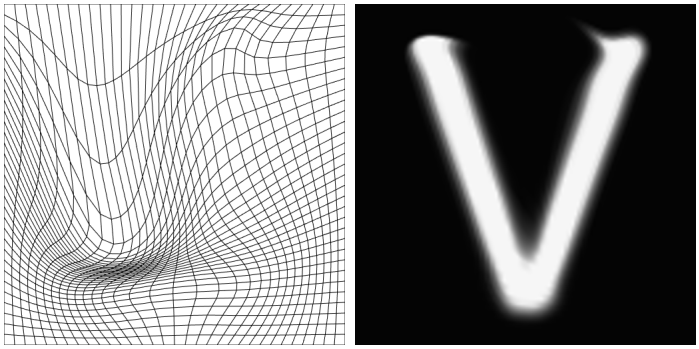
Algorithm 6 det∇ Monitored Image Registration

- 0: input $I_{\boldsymbol{m}}^0 = I_{\boldsymbol{m}}$, $I_{\boldsymbol{f}}$.
- 1: set ($I_{\boldsymbol{m}} = I_{\boldsymbol{m}}^k$ when $k > 0$), $\boldsymbol{F}_0 = \boldsymbol{0}$, $\boldsymbol{a}_0 = \boldsymbol{0}$, $\boldsymbol{\Phi}_k = \boldsymbol{\phi}_0(\boldsymbol{x}) = \boldsymbol{id}(\boldsymbol{x})$, $iter_{max}$, $tstep$, $t_{up} > 1$, $t_{down} \in (0, 1)$, t_{tol}; set $JD_{maxTol} \leq 2$ and $JD_{minTol} \geq 0.5$.
- 2 while $iter < iter_{max}$ & $tstep < t_{tol}$ & $ratio > ratio_{tol}$;
  - 3: if *better*
    - 4: solve for $\boldsymbol{a}_k$ by FFT, where $\Delta \boldsymbol{a}_k(\boldsymbol{x}) = \boldsymbol{b}_k(\boldsymbol{x})^{\mathsf{T}}[\nabla \boldsymbol{\phi}_{k-1}(\boldsymbol{x} + \boldsymbol{u}_k(\boldsymbol{x}))]$;
  - 5: update $\boldsymbol{F}_k = \boldsymbol{F}_{k-1} - tstep * \boldsymbol{a}_k$;
  - 6: solve $\Delta \boldsymbol{u}_k(\boldsymbol{x}) = \boldsymbol{F}_k$ to form $\boldsymbol{\phi}_k(\boldsymbol{x}) \approx \boldsymbol{x} + \boldsymbol{u}_k(\boldsymbol{x})$;
  - 7: compute max(det$\nabla(\boldsymbol{\phi}_k)$) and min(det$\nabla(\boldsymbol{\phi}_k)$);
  - 8: if $max(\det\nabla(\boldsymbol{\phi}_k)) > JD_{maxTol}$ or $min(\det\nabla(\boldsymbol{\phi}_k)) < JD_{minTol}$;
    - 9: compute $\boldsymbol{\Phi}_k = \boldsymbol{\phi}_{k-1} \circ \boldsymbol{\phi}_k$ by interpolation;
    - 10: re-sample $I_{\boldsymbol{m}}^k = I_{\boldsymbol{m}}^0(\boldsymbol{\Phi}_k)$ and go to 1;

    else
    - 11: re-sample $I_{\boldsymbol{m}}(\boldsymbol{\Phi}_k)$ by interpolation and check SSD_k;
    - 12: if SSD_k decrease
      - 13: $better = true$;
      - 14:$tstep = tstep * t_{up}$;
      - 15:$\boldsymbol{F}_{k-1} = \boldsymbol{F}_k$;
      - 16: $\boldsymbol{\phi}_{k-1} = \boldsymbol{\phi}_k$;

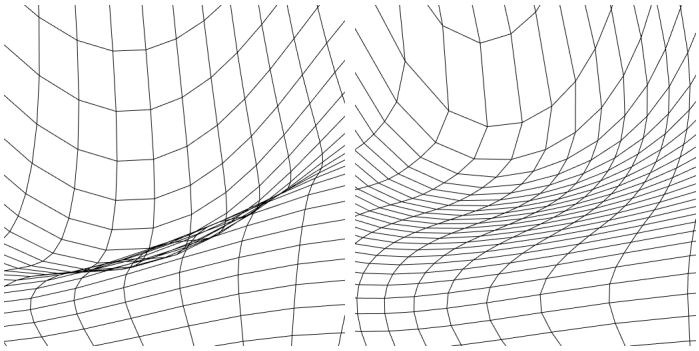
      else
      - 17:$better = false$;
      - 18:$tstep = tstep * t_{down}$;

      go to 2
- 19: compute $\boldsymbol{\Phi}_k = \boldsymbol{\phi}_{k-1} \circ \boldsymbol{\phi}_k$ by interpolation;
- 20: output $\boldsymbol{\Phi}_k$, $\boldsymbol{u}_k$, $I_{\boldsymbol{m}}(\boldsymbol{\Phi}_k)$, k.



(a) $\boldsymbol{\phi}_{J2V}$ converged (b) $I_J(\boldsymbol{\phi}_{J2V})$

Figure 4.20: det∇ Monitored Register: I_J to I_V



(a) $\hat{\boldsymbol{\phi}}_{J2V}$ (b) $\boldsymbol{\phi}_{J2V}$

Figure 4.21: Zoomed In Comparison

Table 4.5: det∇ Monitored I_J to I_V

tstep	*ratio*	*Elapsed Time*	JD_{max}	JD_{min}	det∇ Monitored times
$8.2689 * 10^{-9}$	0.0099	9.6614 *sec*	7.4832	0.1288	$k = 4$

As it shows, with the det∇ being monitored, **Algorithm 6** prevents bad local convergence and restricts det$\nabla > 0$ of the transformation it produces.

4.5 Conclusions

In this chapter, the control approach to nonrigid image registration is revised, numerically tested, demonstrated with both 2D and 3D registrations. In particular, we effectively tested the inverse consistency and transitivity properties of this method. In the future study, we will modify the mathematical setting to restrict diffeomorphisms.

CHAPTER 5

Averaging Images by Averaging Diffeomorphisms

5.1 Introduction

In brain science, an brain atlas is an anatomical and standard norm in representing a certain group of brain images [48]. Scientists has been working on different approaches to construct atlases. One of the ideas of building brain atlases is to average a group of given brain images. Our group proposed an innovative concept of averaging a set of diffeomorphisms based on averaging the Jacobian determinants and curl vectors of the diffeomorphisms. Before the process of averaging images, we need to discuss the process of averaging diffeomorphisms. In [14], the 2D case was realized. With the developments done in chapter 3, now, we may average diffeomorphisms in 3D case as well. This is the key component to average images in our method. Furthermore, for differences in a group of images, we suggest to use Jacobian determinants and curl vectors as features in Tensor Based Morphometry studies.

5.2 Averaging Diffoemorphisms by Variational Method

The average of diffeomorphisms, which coincides with concept of the uniqueness conjecture based on Jacobian determinant and curl vector. As long as we may determine what are the averaged Jacobian determinant and curl vector, we may use the variatotional method to reconstruct such diffeomorphism. Then, we may define it to be the averaged diffeomorphism. Let $\{\boldsymbol{T}_i\}_{i\leq n}$ be the given n many diffeomorphisms.

So $\{\det\nabla(\boldsymbol{T}_i)\}_{i\leq n}$ are the Jacobian determinants and $\{\nabla\times(\boldsymbol{T}_i)\}_{i\leq n}$ are the curl vectors for each $\boldsymbol{T}_i$. Then, compute

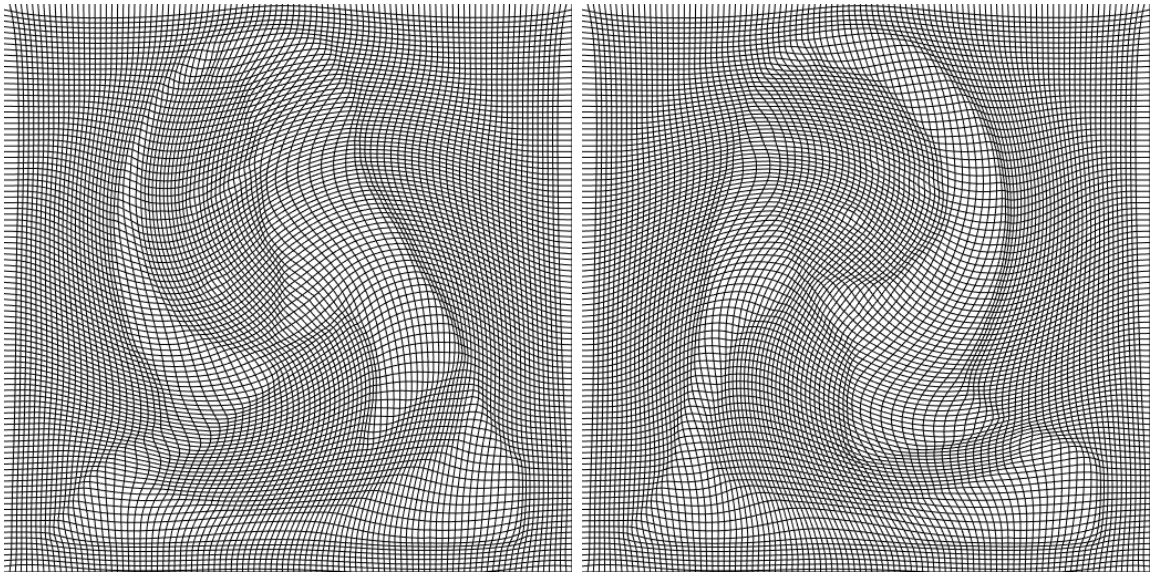
$$\begin{cases} f_0 = \sum_{i=1}^{n} \omega_i \det\nabla(\boldsymbol{T}_i) \\ \boldsymbol{g}_0 = \sum_{i=1}^{n} \omega_i \nabla\times(\boldsymbol{T}_i) \end{cases} \quad \text{where } 0<\omega_i<1 \text{ are weight-coefficients.} \tag{5.2.1}$$

and define the solution $\boldsymbol{\phi}$ reconstructed by our Variational Method to be the average diffeomorphisms. If the transformations are equally weighted, then we define $\omega_i = \frac{1}{n}$.

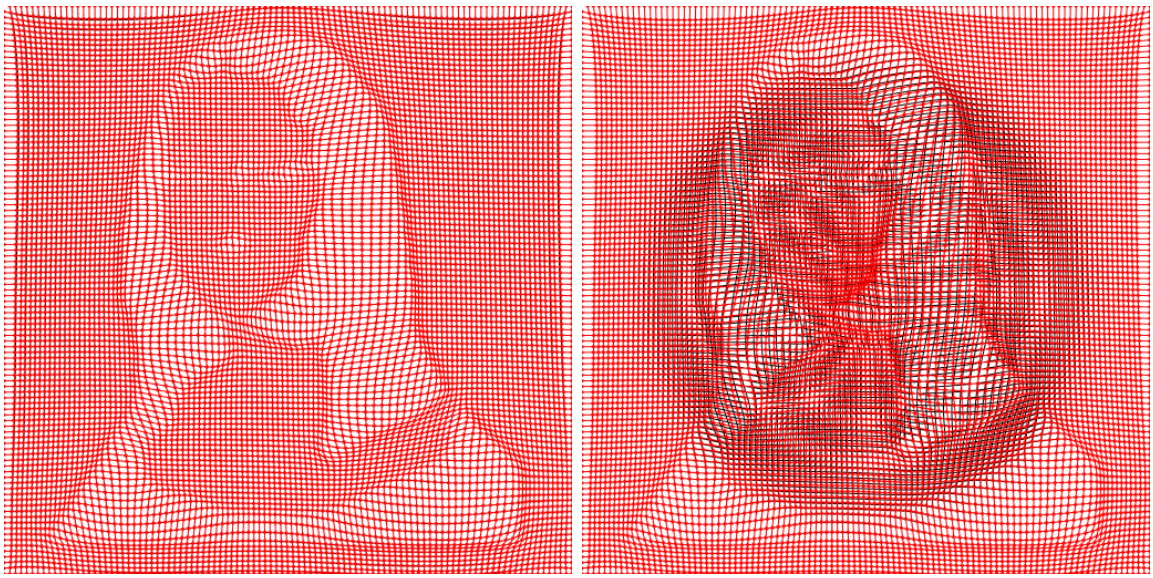
5.2.1 A Numerical Demonstration

We acquired $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$ by using a cut-off rotation transformation manually on $\boldsymbol{\Phi}$ with $\frac{\pi}{4}$ and $-\frac{\pi}{4}$, respectively, so that $\det\nabla(\boldsymbol{\Phi}) = \frac{\det\nabla(\boldsymbol{\Phi}_1)+\det\nabla(\boldsymbol{\Phi}_2)}{2}$ and $\nabla\times(\boldsymbol{\Phi}) = \frac{\nabla\times(\boldsymbol{\Phi}_1)+\nabla\times(\boldsymbol{\Phi}_2)}{2}$ are satisfied. But, to show the effectiveness of the average process, we treat it there is no $\boldsymbol{\Phi}$ as the ground truth and suppose there are only these two given diffeomorphisms, $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$. Again, both of the grid sizes of the Figures and in the computations are of 101x101. The second red grid $Euclid(\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2)$, is the Euclidean average of the location coordinates, namely, $x = \frac{1}{2}\sum x_i$ and $y = \frac{1}{2}\sum y_i$, do not overlap on the black grid well.

Example 1: Average $\boldsymbol{\Phi}_1$ & $\boldsymbol{\Phi}_2$ from Example 3 of Chapter 3



(a) $\boldsymbol{\Phi}_1$ (b) $\boldsymbol{\Phi}_2$



(c) $avg(\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2)$-red vs $\boldsymbol{\Phi}$-black (d) $Euclid(\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2)$-red vs $\boldsymbol{\Phi}$-black

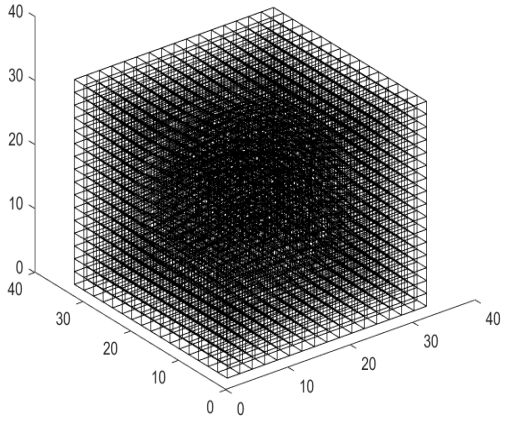
Figure 5.1: Average $\boldsymbol{\Phi}_1$ & $\boldsymbol{\Phi}_2$ with known $\boldsymbol{\Phi}$

The averaging process of $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$ took about 26 seconds, with 2000 iteration, and reached $ratio = 0.0006$.

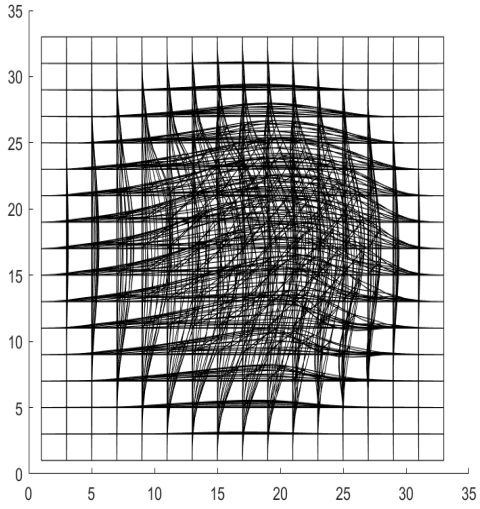
Example 2

For the clarity of visualization in case of 3D, in this example, we averaged diffeomorphisms $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ whose known averaged transformation is the identity map $\boldsymbol{id}$, with $\omega_i = \frac{1}{2}$. In fact, we have

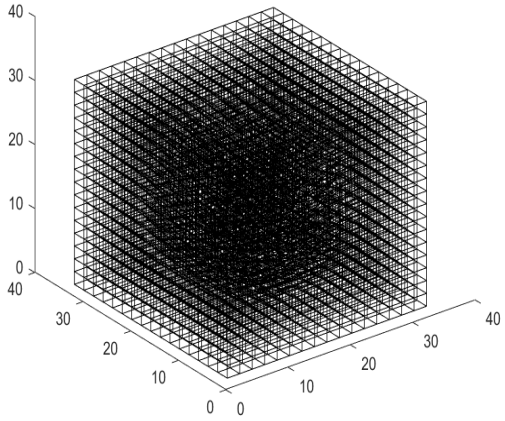
$$\frac{1}{2}\sum \det\nabla(\boldsymbol{T}_i) = 1 \qquad \text{and} \qquad 0 \leq ||\frac{1}{2}\sum \nabla \times (\boldsymbol{T}_i)||_2^2 \leq \ 6.1106 * 10^{-4}$$



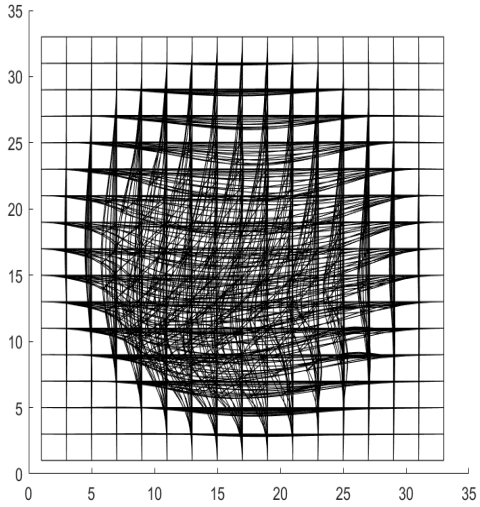
(a) $\boldsymbol{T}_1$



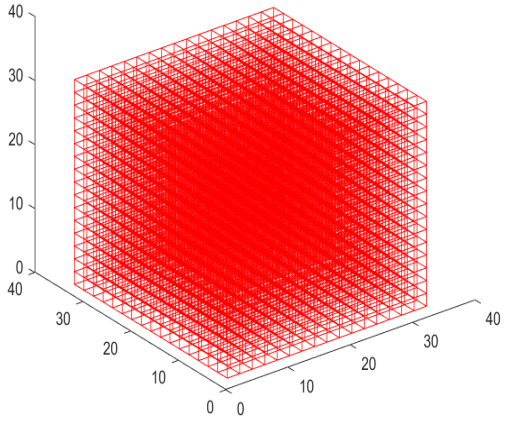
(b) $\boldsymbol{T}_1$ bird view



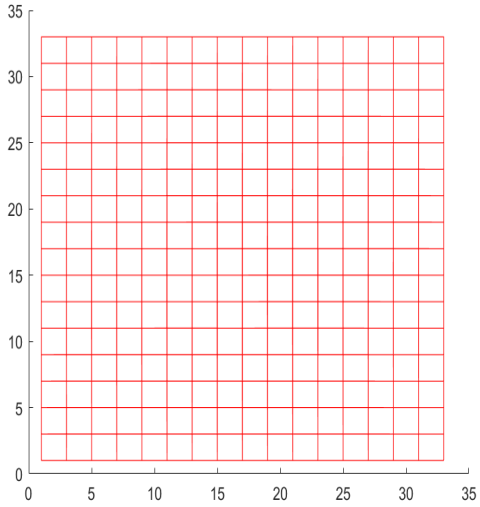
(c) $\boldsymbol{T}_2$



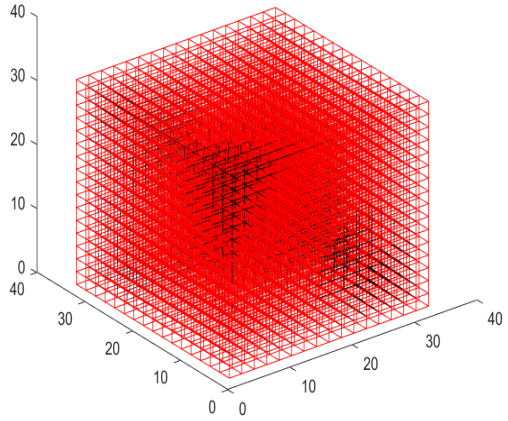
(d) $\boldsymbol{T}_2$ bird view



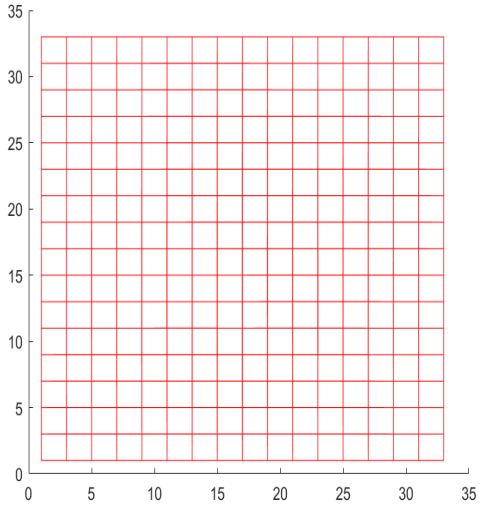
(e) $avg(\boldsymbol{T}_1, \boldsymbol{T}_2)$-red



(f) $avg(\boldsymbol{T}_1, \boldsymbol{T}_2)$-red bird view



(g) $avg(\boldsymbol{T}_1, \boldsymbol{T}_2)$-red superimposes $\boldsymbol{id}$-black



(h) $avg(\boldsymbol{T}_1, \boldsymbol{T}_2)$-red superimposes $\boldsymbol{id}$-black bird view

Figure 5.2: Average $\boldsymbol{T}_1$ & $\boldsymbol{T}_2$ with known $avg(\boldsymbol{T}_1, \boldsymbol{T}_2) = \boldsymbol{id}$

As it can be seem, the averaged transformation $avg(\boldsymbol{T}_1, \boldsymbol{T}_2)$ almost overlaps known $\boldsymbol{id}$. The averaging process of $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ took about 6.9 seconds, with 144 iteration, and reached $ratio = 0.4216$. The $ratio$ may seem a bit too high, but if we break it down to $SSD_{init} = 0.2505$ and $SSD_{new} = 0.1056$, it can be said the initial $SSD_{initial}$ is already very small. It means the starting mesh is very close to the $\boldsymbol{id}$ already which would not have much room to improve in terms of SSD. Anyway, this example still shows the whole process of our averaging diffeomorphisms method is effective.

5.3 Proposed Approach to Averaging Images

we propose the following algorithm which is formed by combining our methods for image registration and for averaging diffeomorphisms through the Variational method. For simplicity, let's take n, the sample number of images, to be some small integer, say within 10, i.e., $n \leq 10$. For the strategy of large sample number, we leave it to the future study.

Algorithm 7 Averaging Images by Average-Diffoemorphism

- 1: given $n \geq 2$ many images $\{I_i\}$;
- 2: choose $I_{\boldsymbol{m}} = I_k$ for a fixed $1 \leq k \leq n$;
- 3: apply **Algorithm 5** from $I_{\boldsymbol{m}} = I_k$ to $I_{\boldsymbol{f}} = I_{i \neq k}$ and acquire $\{\boldsymbol{T}_i\}$ where $i = 1, \ldots, k-1, k+1, \ldots, n$ and when $i = k$, $\boldsymbol{T}_k = \boldsymbol{id}$;
- 4: determine ω_i, then compute

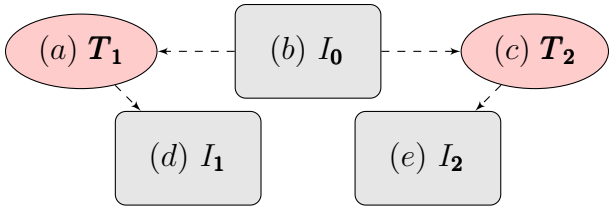
$$f_0(\boldsymbol{x}) = \sum_{i=1}^{n} \omega_i \det \nabla(\boldsymbol{T}_i(\boldsymbol{x})) \text{ and } \boldsymbol{g}_0(\boldsymbol{x}) = \sum_{i=1}^{n} \omega_i \nabla \times (\boldsymbol{T}_i(\boldsymbol{x}));$$

- 5: construct $\boldsymbol{T}_{\boldsymbol{avg}}$ by **Algorithm 4** with f_0 and $\boldsymbol{g}_0$;
- 6: re-sample $I_{\boldsymbol{m}}$ by $\boldsymbol{T}_{\boldsymbol{avg}}$ to get $I_{\boldsymbol{avg}} = I_{\boldsymbol{m}}(\boldsymbol{T}_{\boldsymbol{avg}}(\boldsymbol{x}))$;

Here, we need to make an observation without proving: in order to connect from step 3 to 4 in theoretical structure, it is needed to identify the solution space of the simplified version of image registration is a subset of the admissible set of the Variational method, namely, the solution $\boldsymbol{\Phi}$ from simplified version of image registration satisfies the conditions of Variational method.

Example 3: 2D example

To understand and test the proposed Algorithm 6, let's recall the Example 4 of Chapter 3, two images $I_{\boldsymbol{1}}$ and $I_{\boldsymbol{2}}$ are created with a known ground truth $I_{\boldsymbol{0}}$. The purpose for the setup is to eventually find a image $I_{\boldsymbol{avg}}$ which is expected be close to $I_{\boldsymbol{0}}$. See following diagram and next Figure for illustration. This reflects step 1 of Algorithm 6.



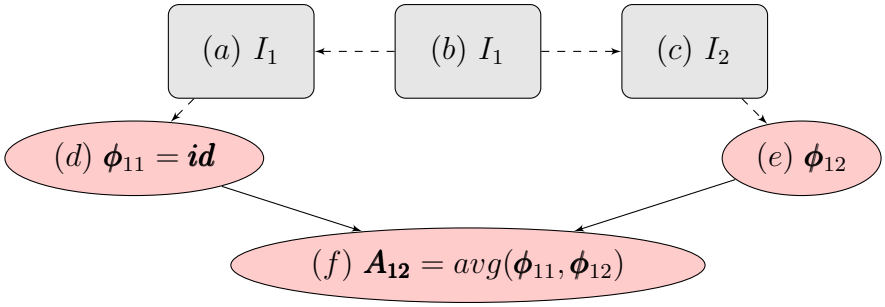
(a) $\boldsymbol{T_1}$ (b) I_0 (c) $\boldsymbol{T_2}$

(d) I_1 (e) I_2

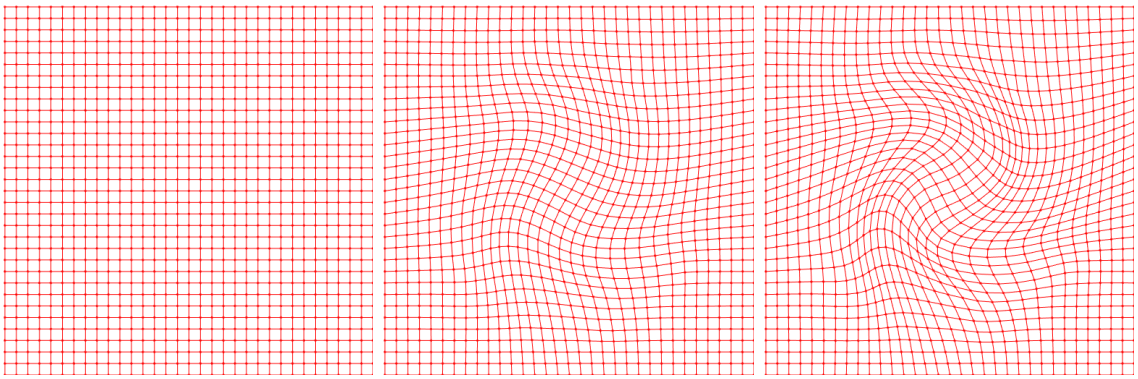
Figure 5.3: Re-sampled I_0 on $\boldsymbol{T_1}$ and $\boldsymbol{T_2}$

For step 2, we choose $I_m = I_1$; on step 3, we need to do the registrations from I_1 to I_1 and I_2, respectively, to get two diffeomorphisms $\boldsymbol{\phi_{11}}$ and $\boldsymbol{\phi_{12}}$; on step 4, we compute $f_0(\boldsymbol{x}) = \frac{\det\nabla\boldsymbol{\phi_{11}}+\det\nabla\boldsymbol{\phi_{12}}}{2}$ and $\boldsymbol{g}_0 = \frac{\nabla\times\boldsymbol{\phi_{11}}+\nabla\times\boldsymbol{\phi_{12}}}{2}$; on step 5, we reconstruct a transformation $\boldsymbol{\phi_{avg}}$ by forwarding f_0 and $\boldsymbol{g}_0$ to the variational method and this $\boldsymbol{\phi_{avg}}$

is defined to be the averaged transformation; lastly, on step 6, we re-sample $I_{\boldsymbol{m}}$ by $\boldsymbol{\phi_{avg}}$ and defined the averaged image $I_{\boldsymbol{avg}} := I_{\boldsymbol{m}}(\boldsymbol{\phi_{avg}}(\boldsymbol{x}))$.

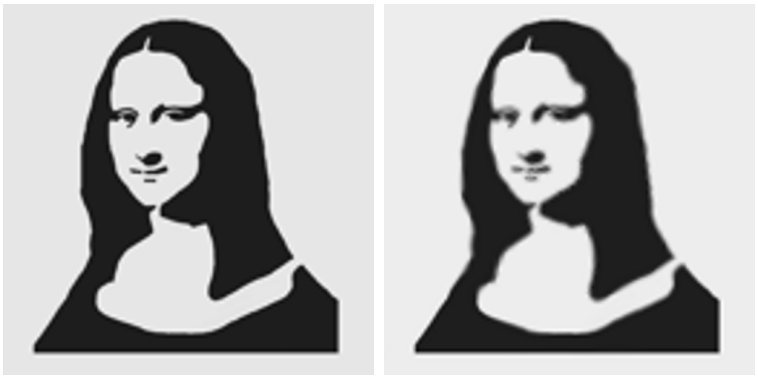


(a) $I_{\mathbf{1}}$ (b) $I_{\mathbf{1}}$ (c) $I_{\mathbf{2}}$



(d) $\boldsymbol{\phi_{11}} = \boldsymbol{id}$ (e) $\boldsymbol{A_{12}} = avg(\boldsymbol{\phi_{11}}, \boldsymbol{\phi_{12}})$ (f) $\boldsymbol{\phi_{12}}$

Figure 5.4: Register $I_{\mathbf{1}}$ to $I_{\mathbf{1}}$ and $I_{\mathbf{1}}$ to $I_{\mathbf{2}}$ acquired $\boldsymbol{\phi_{11}} = \boldsymbol{id}$ and $\boldsymbol{\phi_{12}}$



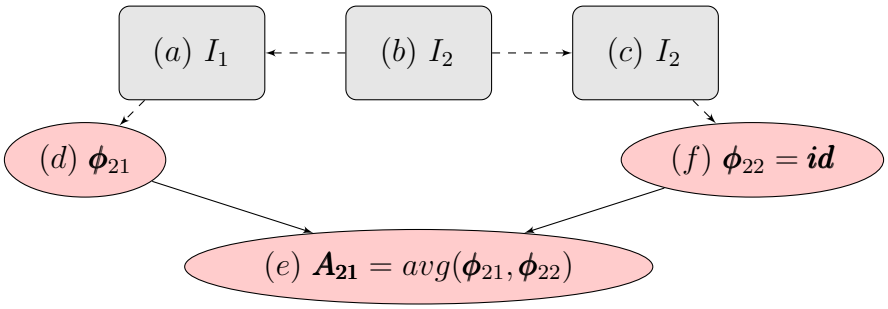
(a) $I_{\mathbf{0}}$ (b) $I_{\boldsymbol{avg}(1)}$

Figure 5.5: Resample $I_{\mathbf{1}}$ by $A_{\mathbf{12}}$ to get $I_{\boldsymbol{avg}(1)}$

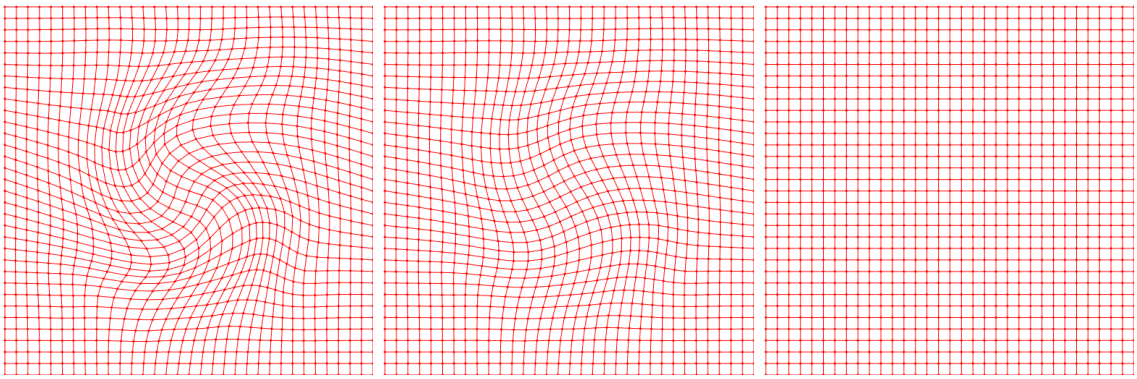
As the table shows, the computation performance of averaging $I_{\mathbf{1}}$ and $I_{\mathbf{2}}$ with choose $I_{\boldsymbol{m}} = I_{\mathbf{1}}$ turned out very good.

SSD_{old}	SSD_{new}	*Ratio*	*Elapsed time*
6.5771x10^7	3.3914x10^6	0.0516	$\approx$ 45 *secs*

The following diagram and Figures demonstrate a symmetric application of Algorithm 6 with taking $I_{\boldsymbol{m}} = I_{\mathbf{2}}$.

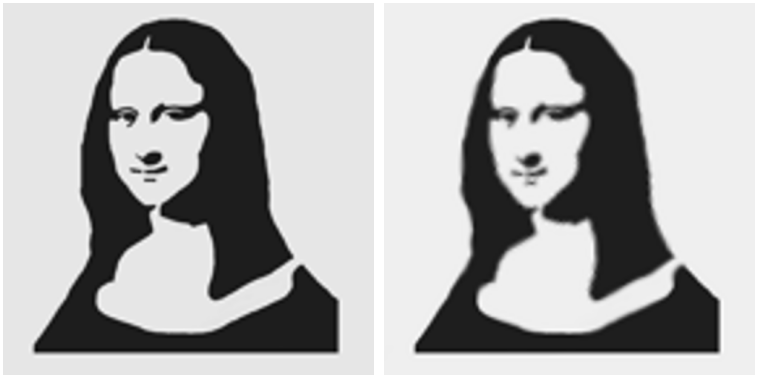


(a) I_1 (b) I_2 (c) I_2



(d) ϕ_{21} (e) $A_{21} = avg(\phi_{21}, \phi_{22})$ (f) $\phi_{22} = id$

Figure 5.6: Register I_1 to I_1 and I_1 to I_2, we get $\phi_{11} = id$ and ϕ_{12}



(a) I_0 (b) $I_{avg(2)}$

Figure 5.7: Resample I_2 by A_{21} to get $I_{avg(2)}$

As the table shows, computation performance of averaging I_1 and I_2 with choosing $I_m = I_2$ also turned out successful.

SSD_{old}	SSD_{new}	$Ratio$	$Elapsed\ time$
6.5920x10^7	3.4485x10^6	0.0523	$\approx 43\ secs$

The computational results with choosing $I_m = I_1$ or $I_m = I_2$ are very similar. Both of the averaged images look very close to the ground truth image. With this observation, we claim that our image averaging process produces unbiased result.

5.4 Construction of Unbiased Template

We further tested Algorithm 7 in construction of unbiased templates. Until this stage, we may connect the prior works on characterizing diffeomorphisms to the studies of Brain Morphometry. The results of this section confirm that our early mentioned point of view about the curl vector $\nabla\times$ should have been included in all process of the studies of Brain Morphometry together with the Jacobian determinant $\det\nabla$. Let's proceed to the following examples to see this point.

Example 4: Unbiased Template in 2D

To test the effectiveness of the proposed Algorithm 6, firstly, we obtain six brain images by re-sampling a ground truth brain image on six intentionally designed transformations $\boldsymbol{D}_i$, where $i = 1, \ldots, 6$.

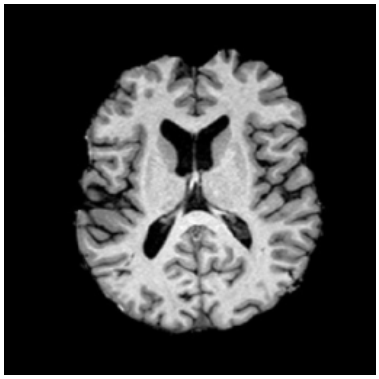
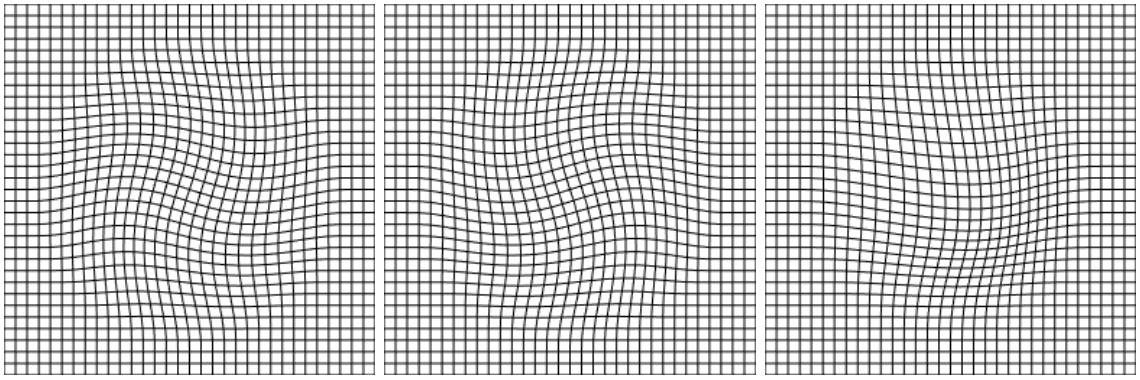
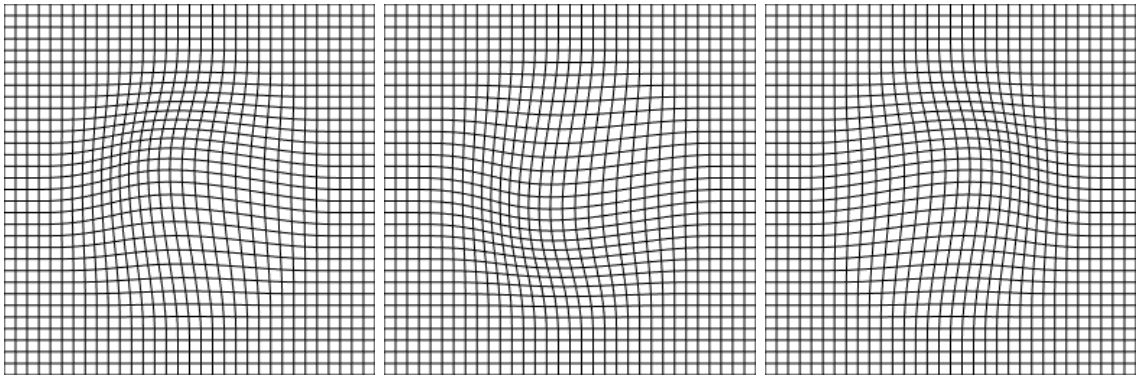


Figure 5.8: I_0 Ground Truth (GT)

we constructed six transformations $\boldsymbol{D}_i$ where $i = 1, \ldots, 6$, which are shown below:



(a) $\boldsymbol{D}_1$ (b) $\boldsymbol{D}_2$ (c) $\boldsymbol{D}_3$



(d) $\boldsymbol{D}_4$ (e) $\boldsymbol{D}_5$ (f) $\boldsymbol{D}_6$

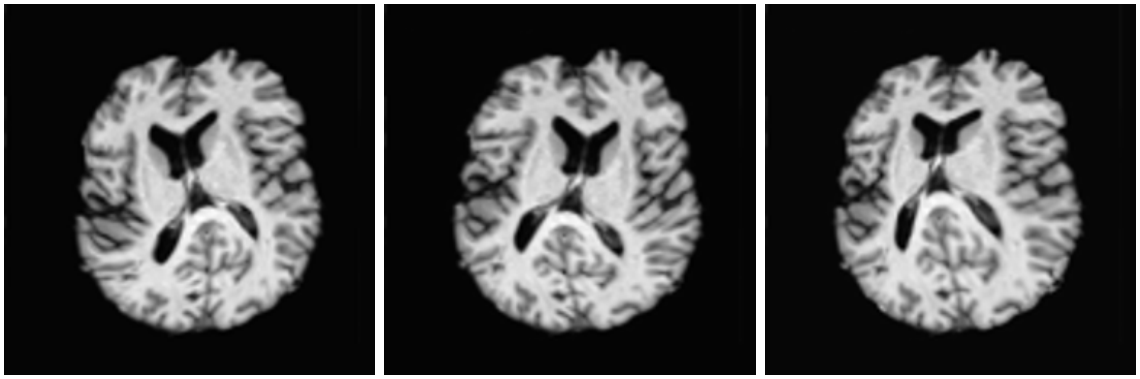
Figure 5.9: Transformations $\boldsymbol{D}_{1-6}$

These transformations are constructed in such a way that their Jacobian determinants have average equal to 1, and their curls have average 0. In fact, we have

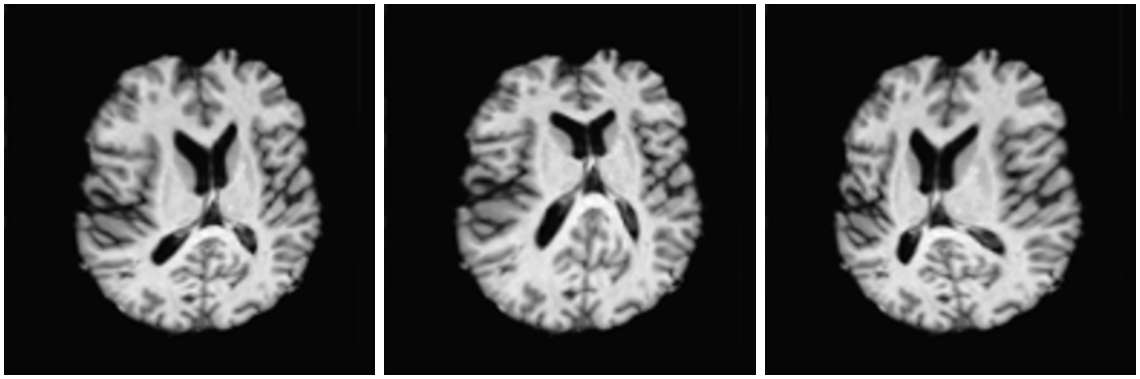
$$0.999986099590103 \leq \frac{1}{6}\sum_{i=1}^{6} \det\nabla(\boldsymbol{D}_i) \leq 1.000016719949570;$$

$$-6.4029 * (10^{-6}) \leq \frac{1}{6}\sum_{i=1}^{6} \nabla \times (\boldsymbol{D}_i) \leq \ 6.4029 * (10^{-6}).$$

Six modeled brain images I_i, where $i = 1, \ldots, 6$ are generated by re-sampling GT on each of the six transformations $\boldsymbol{D}_i$, where $i = 1, \ldots, 6$. These images are shown following Figure.



(a) I_1 (b) I_2 (c) I_3



(d) I_4 (e) I_5 (f) I_6

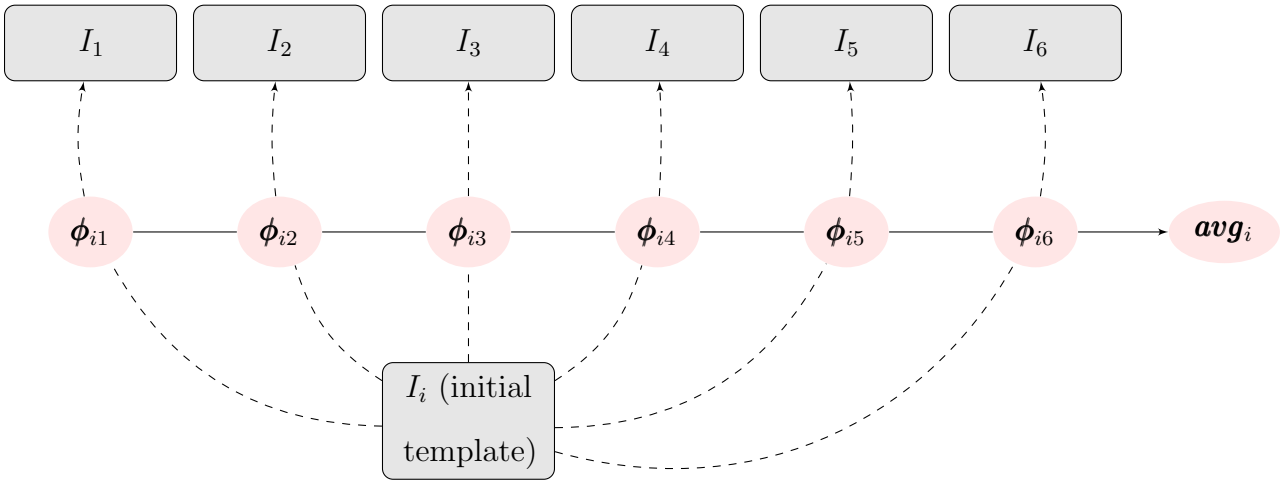
Figure 5.10: Image I_{1-6}

Now when we take $f_0 = \frac{1}{6}\sum_i^6 \det\nabla(\boldsymbol{D}_i) \approx 1$ and $g_0 = \frac{1}{6}\ \sum_i^6 \nabla \times (\boldsymbol{D}_i) \ \approx 0$, by the Variational Principle, the average of these images is expected to be a good approximation to GT. We will verify this in Example 3. The Sum of Squared Differences (SSD) between I_i and GT — $SSD(I_i, I_0)$ (they can be seen as the initial difference from I_i to I_0) are shown in the following table:

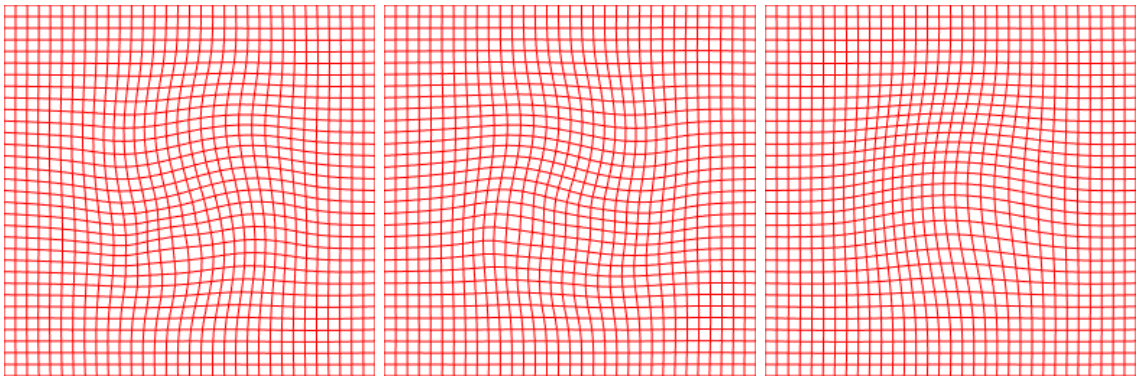
i	1	2	3	4	5	6
$SSD(I_i, I_0) = (10^6)*$	1.8923	1.8906	1.8986	1.9610	1.8309	1.8353

Example 3: General Approach

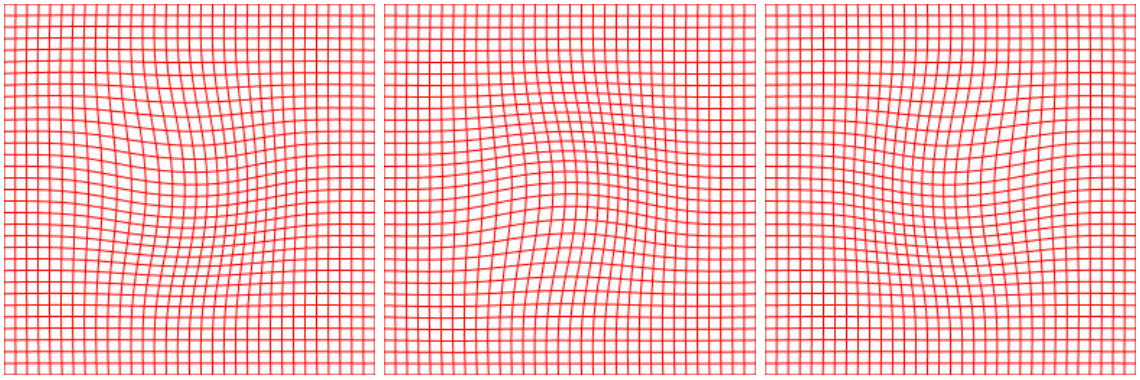
In this example, we will walk through our method for the construction of unbiased template in a general sense. Step 1: take one image I_i out of the six images as the initial template, then register I_i to all six images I_j for $j = 1, 2, \ldots, 6$ (arrowed dash-lines) to find six registration transformations — $\boldsymbol{\phi}_{ij}$ for $j = 1, 2, \ldots, 6$; Step 2: find the average transformation $\boldsymbol{avg}_i = avg(\boldsymbol{\phi}_{ij=1,\ldots,6})$ of the six registration deformations by the Variational method (arrowed solid-line), as are demonstrated in the following diagram.



Step 3, re-sample I_i on the average transformation $\boldsymbol{avg}_i$, indicated as the next diagram and shown in Figure 7, to get the biased temporary templates $\hat{T}emplate_i = I_i(\boldsymbol{avg}_i)$ for each $i = 1, 2, \ldots, 6$ which are shown in Figure 8.

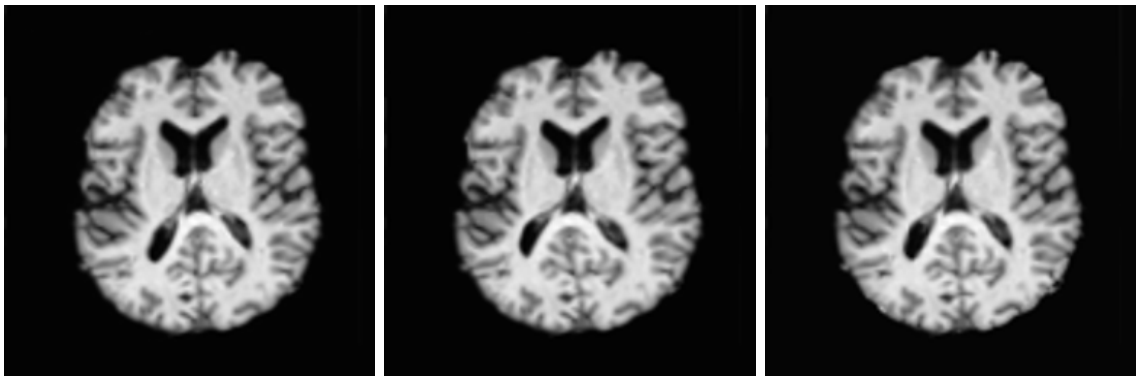


(a) $\boldsymbol{avg}_1$ (b) $\boldsymbol{avg}_2$ (c) $\boldsymbol{avg}_3$

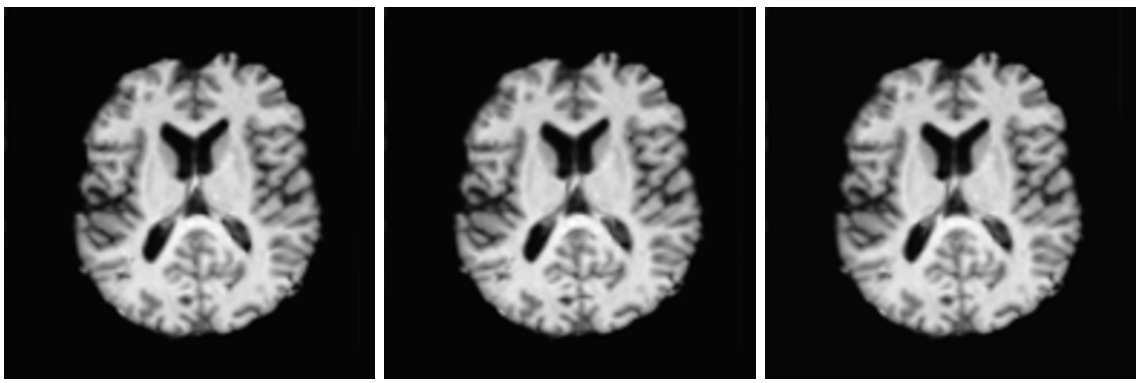


(d) $\boldsymbol{avg}_4$ (e) $\boldsymbol{avg}_5$ (f) $\boldsymbol{avg}_6$

Figure 5.11: Average transformation — $\boldsymbol{avg}_i = avg(\boldsymbol{\phi}_{ij=1,\ldots,6})$



(a) $\hat{T}emplate_1$ (b) $\hat{T}emplate_2$ (c) $\hat{T}emplate_3$



(d) $\hat{T}emplate_4$ (e) $\hat{T}emplate_5$ (f) $\hat{T}emplate_6$

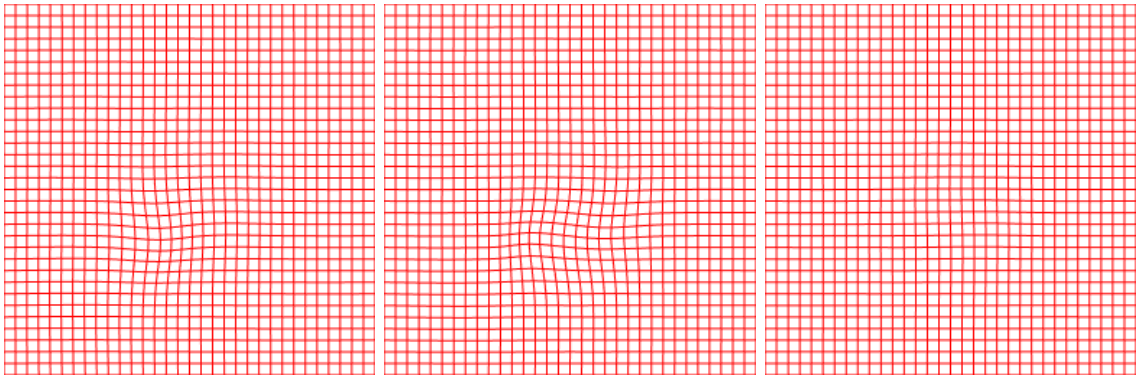
Figure 5.12: Biased Temporary Templates — $\hat{T}emplate_i = I_i(\boldsymbol{avg}_i)$

The SSD between $\hat{I}_i$ with the GT — $SSD(\hat{T}emplate_i, I_0)$ and $Error_i = \frac{SSD(\hat{T}emplate_i, I_0)}{SSD(I_i, I_0)}$ are shown in the following table. $Error_i$ measures the error reduction in temporary template $\hat{T}emplate_i$ compared to the image I_i.

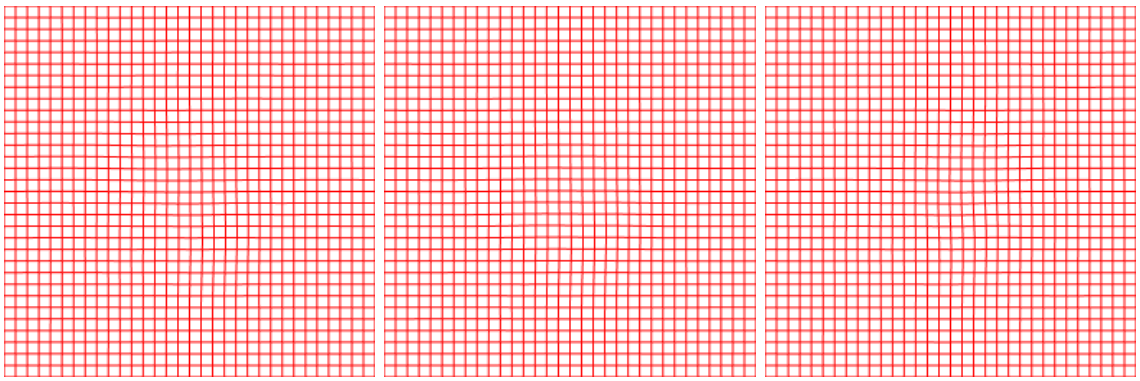
i	1	2	3	4	5	6
$SSD(\hat{T}emplate_i, I_0) = (10^6)*$	2.0504	3.4073	0.4963	0.4905	0.4835	0.5116
$Error_i$	0.1084	0.1802	0.0261	0.0250	0.0264	0.0279

As it can be seen, $\hat{T}emplate_2$ has the highest error = 0.1802. At this stage, $\hat{T}emplate_4$ is the best with an error 0.0250 (this means 97.5% of the initial SSD between image I_4 and GT is reduced by our optimal control registration method). The large difference between $Error_2 = 0.1802$ and $Error_4 = 0.0250$ indicates the existence of bias towards the image that is used as an initial template. In order to quantify the bias, we calculate the sample mean and standard deviation of the six $SSD(\hat{T}emplate_i, I_0)$'s: Sample Mean $= 1.2400 * (10^6)$, Sample Standard Deviation $= 1.2306 * (10^6)$. So, $SSD(\hat{T}emplate_6, I_0)$ is the closest to the Sample Mean. But the large standard deviation of this sample disqualifies any one of these templates as unbiased.

This leads to Step-4: repeat Step-1 to Step-3 on biased temporary templates — $\hat{T}emplate_i$ to reduce their bias. So we get a new group of average transformation $\boldsymbol{Avg}_i$ as shown in Figure 9 and the re-sampled images are the Unbiased Templates — $Template_i = \hat{T}emplate_i(\boldsymbol{Avg}_i)$ as shown in Figure 10.

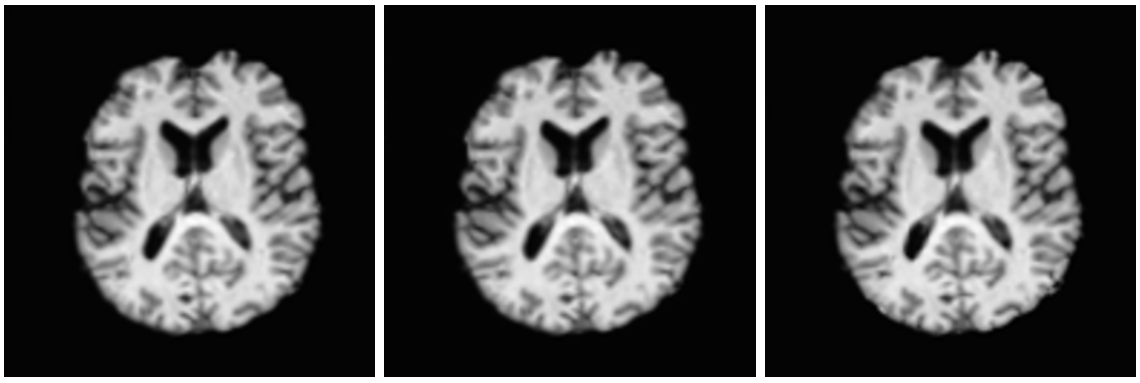


(a) $\boldsymbol{Avg}_1$ (b) $\boldsymbol{Avg}_2$ (c) $\boldsymbol{Avg}_3$

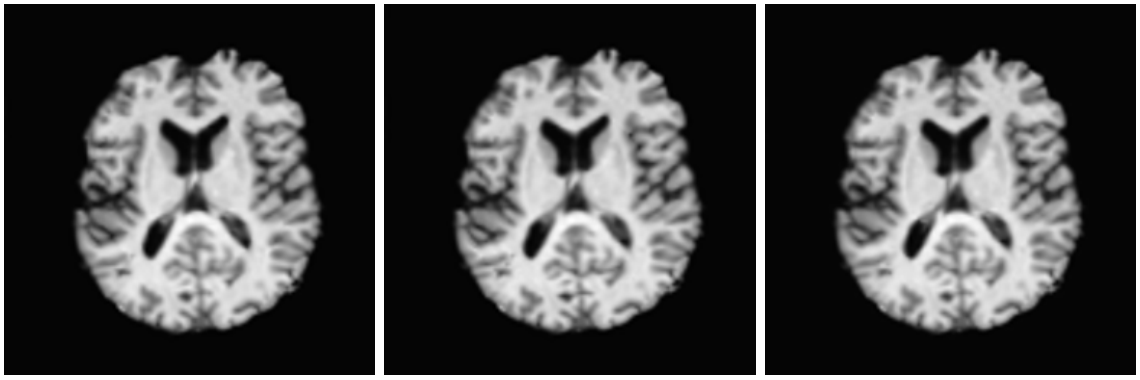


(d) $\boldsymbol{Avg}_4$ (e) $\boldsymbol{Avg}_5$ (f) $\boldsymbol{Avg}_6$

Figure 5.13: Average transformation — $\boldsymbol{Avg}_i$



(a) $Template_1$ (b) $Template_2$ (c) $Template_3$



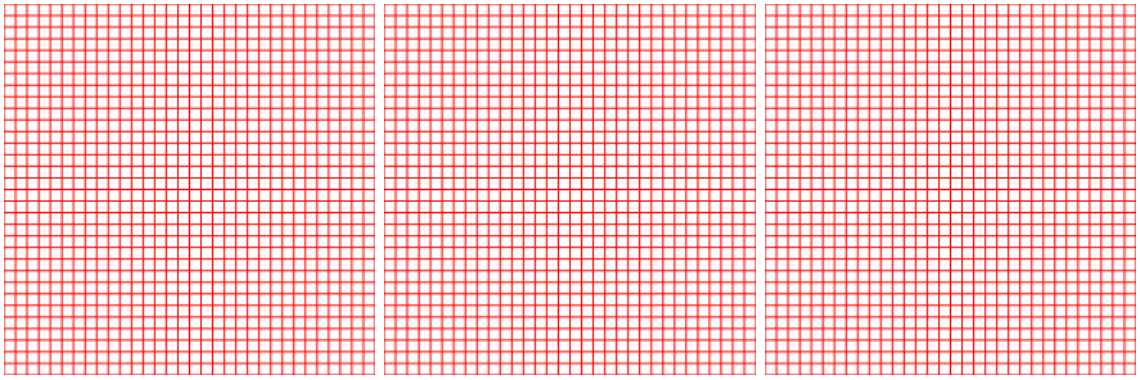
(d) $Template_4$ (e) $Template_5$ (f) $Template_6$

Figure 5.14: Unbiased Templates — $Template_i = \hat{T}emplate_i(\boldsymbol{Avg}_i)$

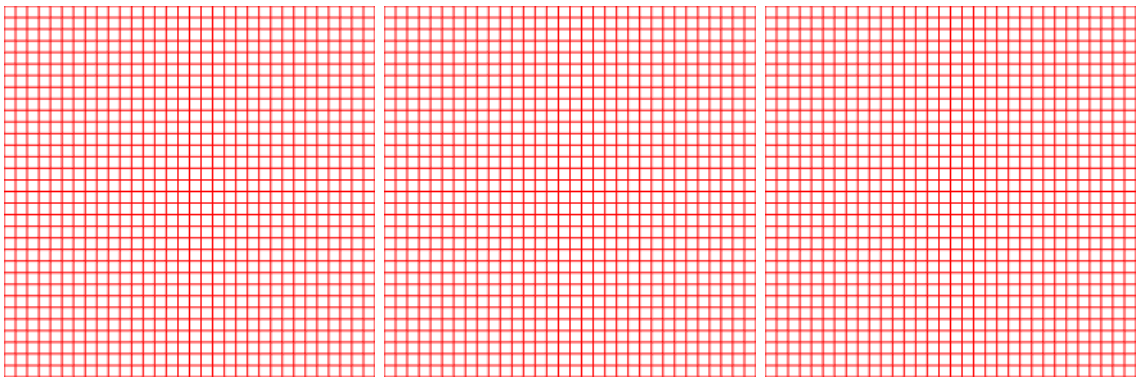
The SSD between $Template_i$ with the GT — $SSD(Template_i, I_0)$ and $Error_i = \frac{SSD(Template_i, I_0)}{SSD(I_i, I_0)}$ are shown in the following table which measures the error reduction in temporary template $Template_i$ compared to the image I_i.

i	1	2	3	4	5	6
$SSD(Template_i, I_0) = (10^5)*$	6.7786	6.6041	5.5583	5.6649	5.7469	5.8088
$Error_i$	0.0358	0.0349	0.0292	0.0289	0.0314	0.0317

The statistics are significantly improved: New Sample Mean $= 6.0269 * (10^5)$; New Sample Standard Deviation $= 5.2436 * (10^4)$. $Template_6$ is closest to the New Sample Mean. The New Sample Standard Deviation $= 5.2436 * (10^4)$ is now only 4.3566% of the previous Sample Standard Deviation $= 1.2306 * (10^6)$. This means, the effectiveness of repeating Step-1 to Step-3 on the biased temporary templates has greatly reduced the bias of biased tmporary template $\{\hat{T}emplate_i\}$. Hence, we can take any of the new templates $Template_i$ as an unbiased template. To check that, we may register each of $Template_i$ to I_0 to get six register transformations and all of them are expected to be close to the identity map $\boldsymbol{Id}$ as the results are shown in Figure 11.



(a) $\hat{\boldsymbol{Id}}_1$ (b) $\hat{\boldsymbol{Id}}_2$ (c) $\hat{\boldsymbol{Id}}_3$



(d) $\hat{\boldsymbol{Id}}_4$ (e) $\hat{\boldsymbol{Id}}_5$ (f) $\hat{\boldsymbol{Id}}_6$

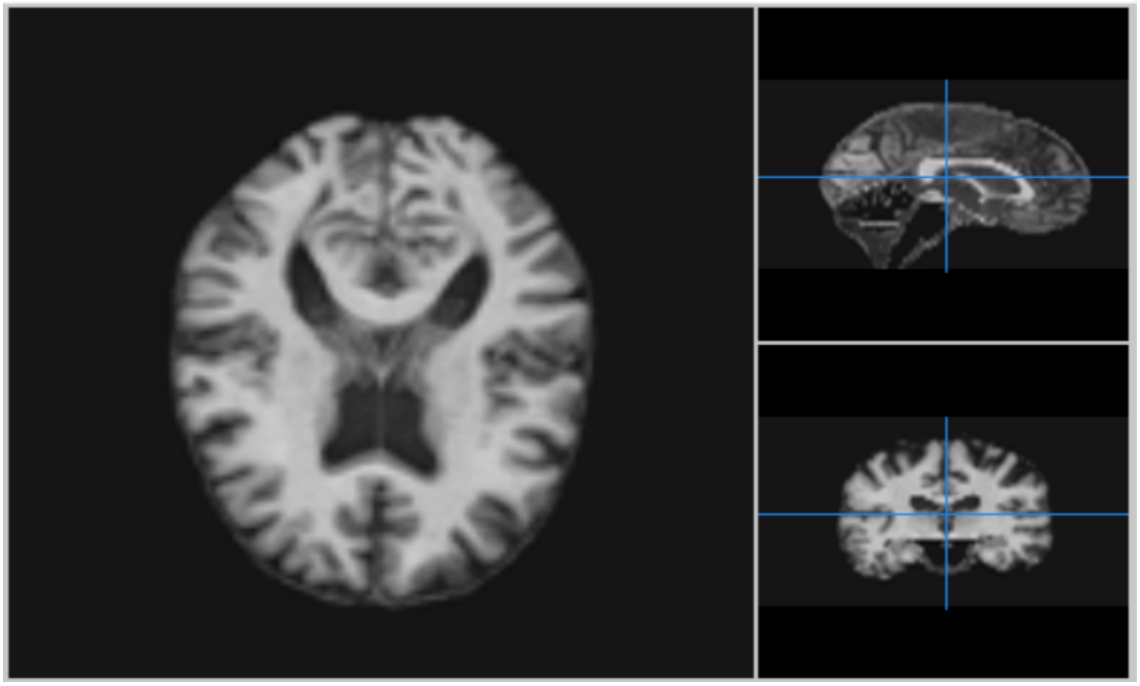
Figure 5.15: Average transformation — $\hat{\boldsymbol{Id}}_i$

The behaviors of their Jacobian determinant and curl are shown in the following table. The more Jacobian determinant is close to 1 and the curl is more close to 0, the more $\hat{\boldsymbol{Id}}_i$ is close to the identity map $\boldsymbol{id}$, which means the Algorithm 6 is generating templates more unbiased.

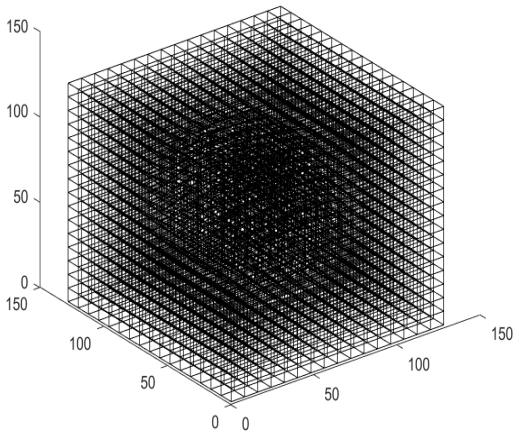
i	1	2	3	4	5	6
max $\det\nabla(\hat{\boldsymbol{Id}}_i)$	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
min $\det\nabla(\hat{\boldsymbol{Id}}_i)$	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
max $\nabla \times (\hat{\boldsymbol{Id}}_i) = 10^{-14} *$	0.3553	0.3553	0.1776	0.1776	0.3553	0.3553
min $\nabla \times (\hat{\boldsymbol{Id}}_i) = 10^{-14} *$	-0.3553	-0.3553	-0.1776	-0.1776	-0.3553	-0.3553

Example 4: A 3D Brain MRI Image

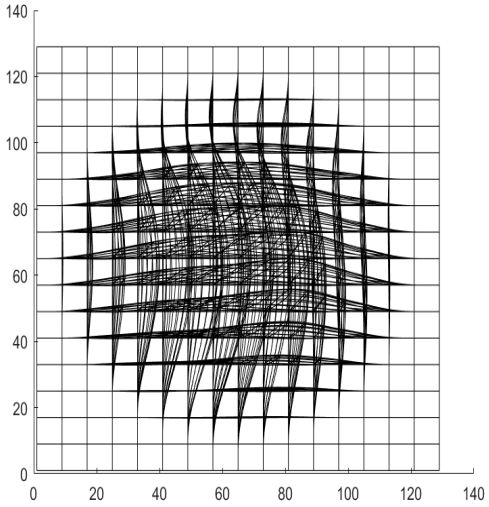
In this example, a 3D MRI brain image is tested to demonstrate the potential capability in handling real brain image data, which usually are in 3D. It is followed with a similar work flow as in 2D, i.e., (d) and (g) are deformed symmetrically from (a) by (b-c) and (e-f), shown in the next Figure. Then, (d) and (e) are the averaged image based on (b) and (c), respectively, which solved by Algorithm 7.



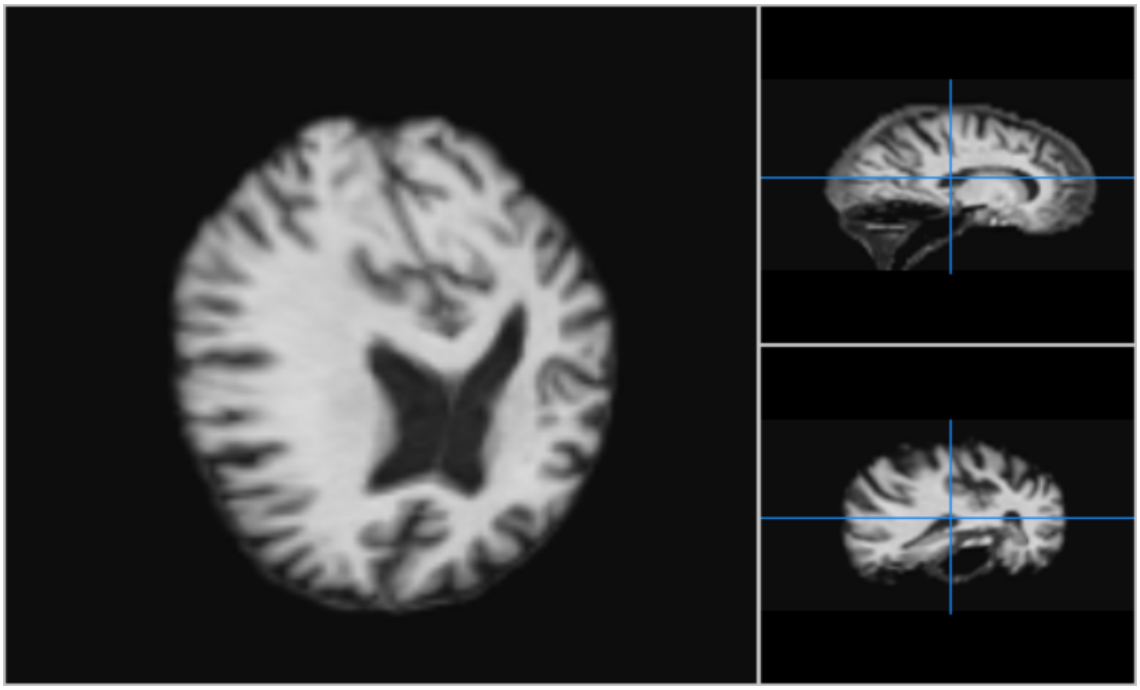
(a) I_0



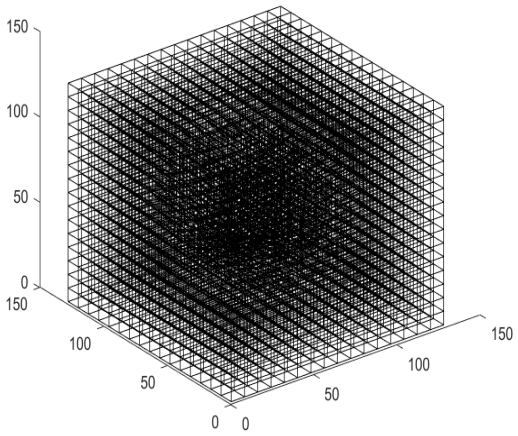
(b) $\boldsymbol{\Phi}_1$



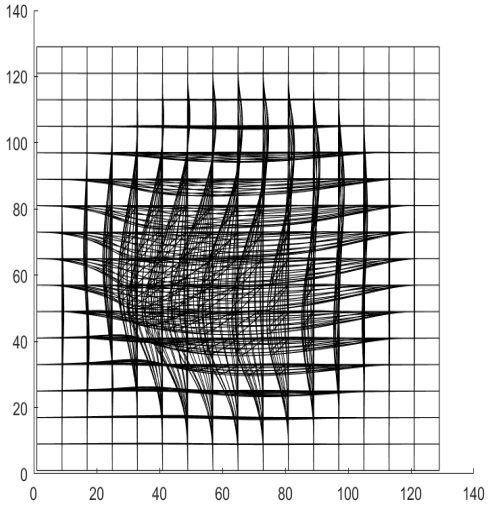
(c) $\boldsymbol{\Phi}_1$ bird view



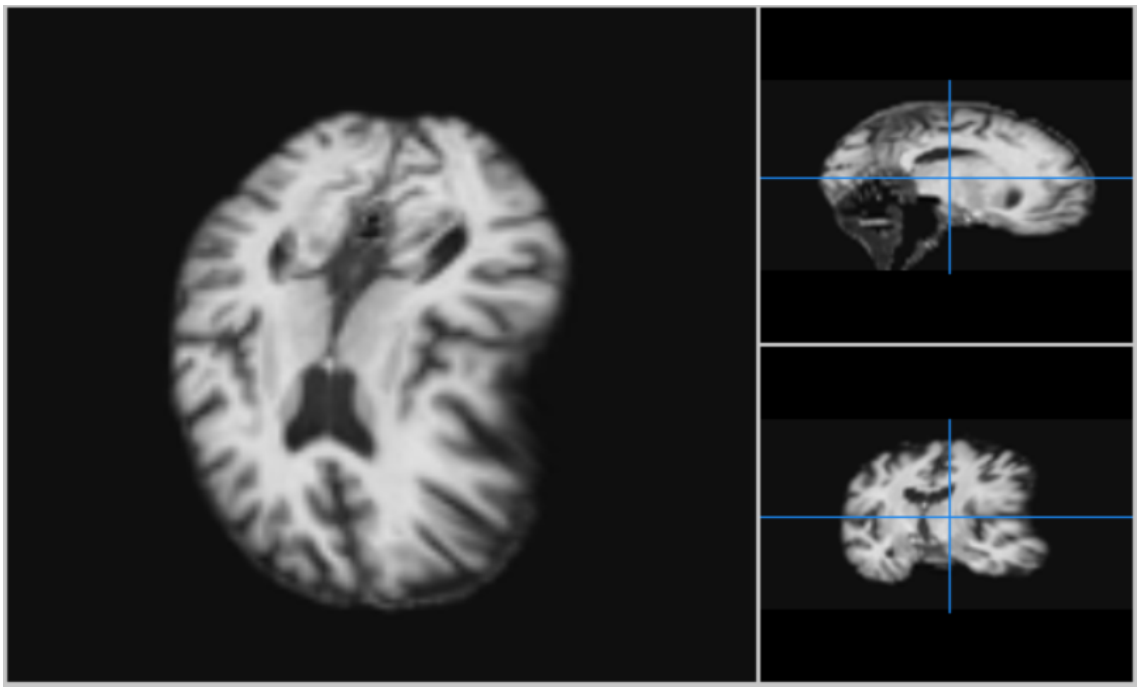
(d) I_1



(e) $\boldsymbol{\Phi}_2$



(f) $\boldsymbol{\Phi}_2$ bird view

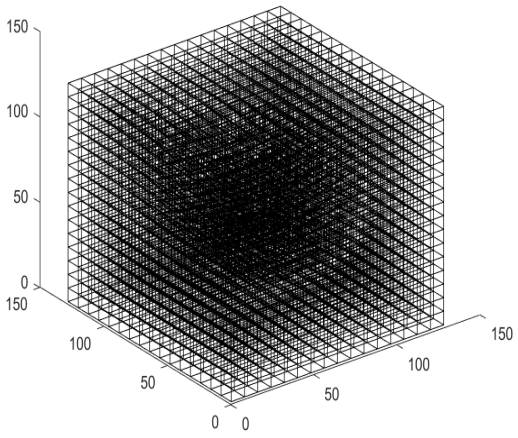


(g) I_1

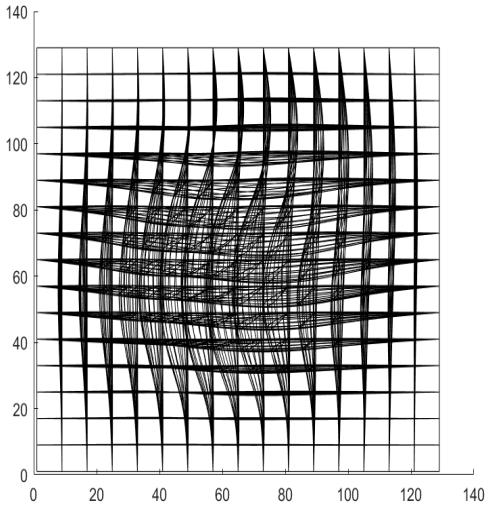
Figure 5.16: I_1, I_2 deformed symmetrically from I_0 by $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$

Continued with Algorithm 6, we may find two averaged transformations $\boldsymbol{Avg}_{1to2}$ and $\boldsymbol{Avg}_{2to1}$. The followed Figures display the re-sampled images $I_1(\boldsymbol{Avg}_{1to2})$ and

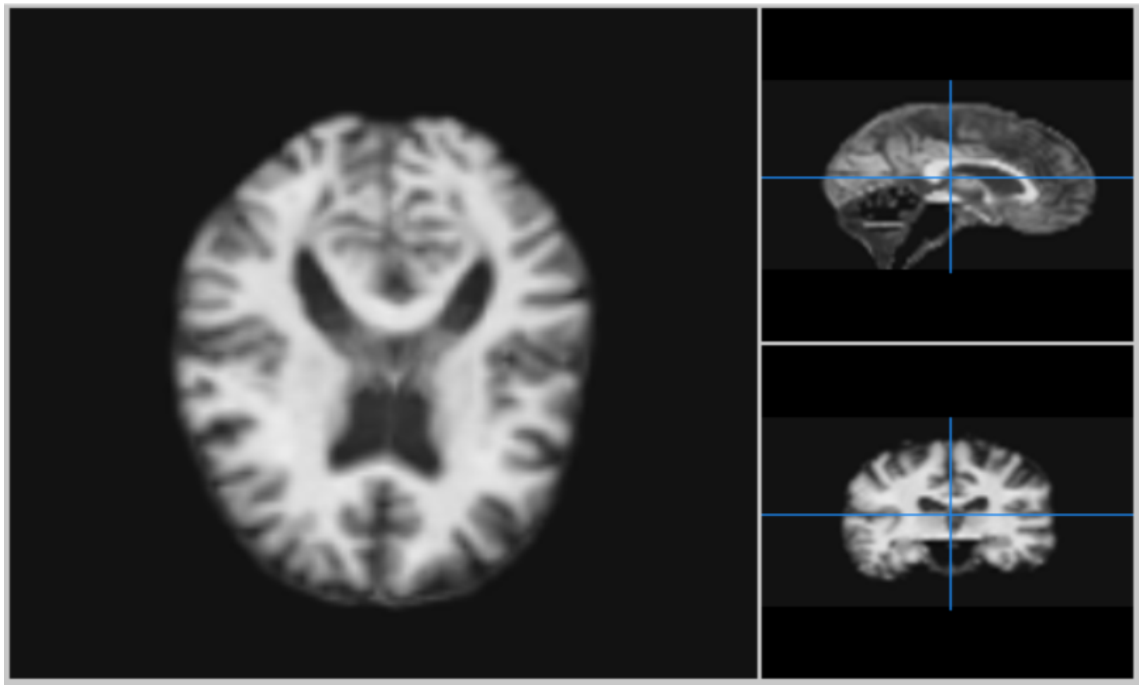
$I_2(\boldsymbol{Avg}_{2to1})$. They look almost identical to ground truth I_0. This observation is confirmed in table below the Figures.



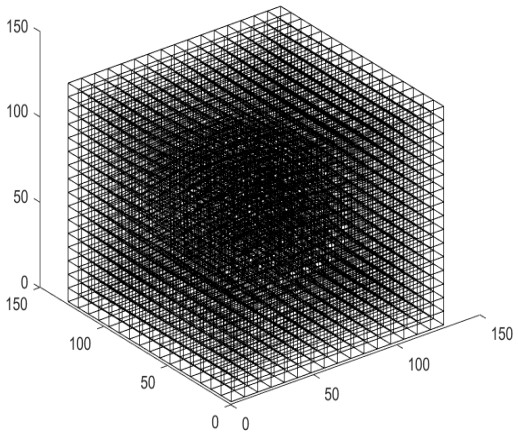
(a) $\boldsymbol{Avg}_{1to2}$



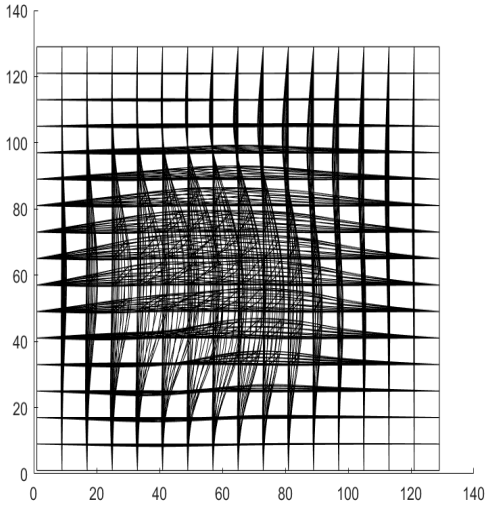
(b) $\boldsymbol{Avg}_{1to2}$ bird view



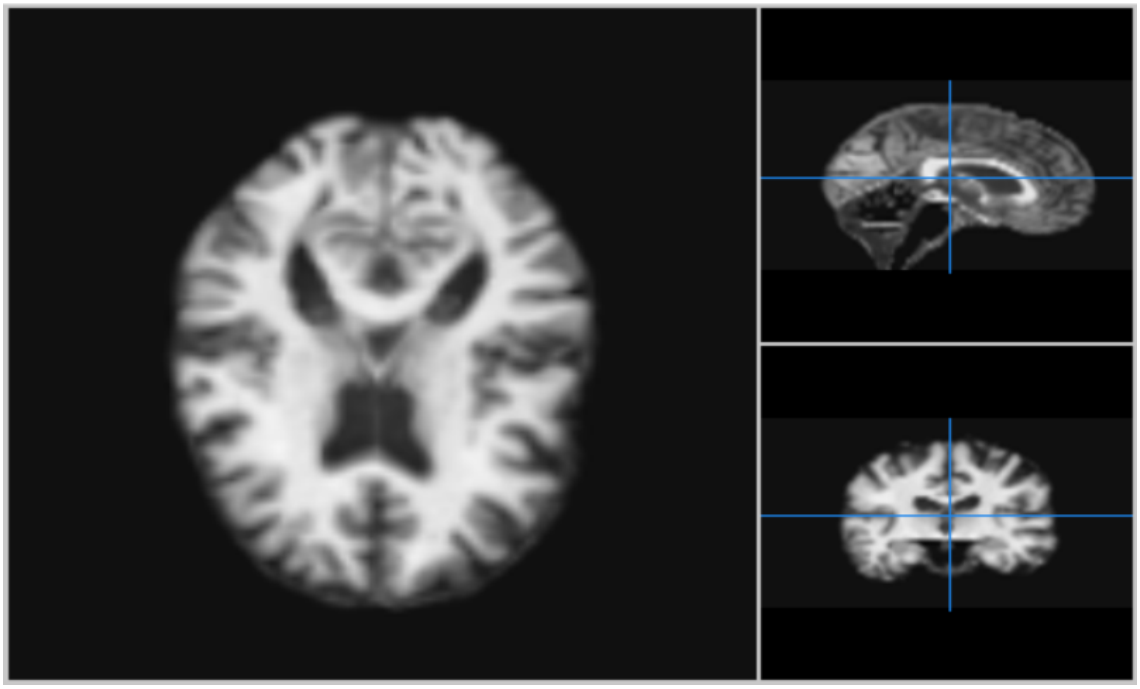
(c) $I_{avg(1)} = I_1(\boldsymbol{Avg}_{1to2})$



(d) $\boldsymbol{Avg}_{2to1}$



(e) $\boldsymbol{Avg}_{2to1}$ bird view



(f) $I_{avg(2)} = I_2(\boldsymbol{Avg}_{2to1})$

Figure 5.17: $I_{avg(1)}$ and $I_{avg(2)}$

As the table shows, the averaged images are close to the ground truth image as well as close to each other.

	SSD_{init}	SSD_{new}	*Ratio*	*Elapsed time*
$I_{\boldsymbol{m}} = I_1$	1.3639x10^9	1.0171x10^8	0.0746	$\approx$ 9667 *secs*
$I_{\boldsymbol{m}} = I_2$	1.4224x10^9	1.1349x10^8	0.0798	$\approx$ 9495 *secs*

$SSD_{init}(I_1, I_2)$	$SSD_{new}(I_1, I_2)$	*Ratio*
1.9963x10^9	1.3882x10^8	$\approx$ 0.0695

5.5 Summary

In the tested examples, we ran small data sets to confirm the effectiveness of our method with known ground truth. As we noticed, there is no known ground truth to compare with in most real world problems. However, the results confirms our suggestion in including the curl vector $\nabla\times$ together with Jacobian determinant $\det\nabla$ in characterization of diffeomorphisms for the studies of Brain Morphometry. With our works, a new way to utilize curl vector's information is made possible and effective. This could make the Tensor-Based Morphometry available for future Brain Mophometry studies.

CHAPTER 6

Conclusions and Future works

In chapter two, the deformation method for mesh generation is reviewed and revised to develop higher order elements mesh, in both 2D and 3D scenarios, through a local interpolation technique. In future, a user determined package can be built based on this higher order element deformation method with LSFEM.

In chapter three, to understand the idea of determination of diffeomorphisms with given Jacobian determinant and curl vector, the variational method was reviewed. its correspondent algorithm was revised so that the numerical computations can be realized in case 3D; a uniqueness problem is analytically discussed in a special simple case in 3D; and a direct strategy motivated by the uniqueness problem is theoretically discussed and numerically tested compare to the iterative variational method in case of both 2D and 3D.

In chapter four, the optimal control approach to image registration is revised so a cheaper computational scheme is achieved, also it is made capable for 3D registration. In future, we will combine convolutional neural network in deep learning with this method to build a trained algorithm which has the potential to perform the registration task in very short amount of time even if the data is large.

In chapter five, a novel method to average images is built, by applying a technique in averaging diffeomorphisms by the variational method, so that an unbiased template of given images can be built. Next goal for this image averaging method is to apply to large data set of MRI images to build an Brain Atlas. We suggest our approach for

atlas construction will give more standard exemplar and will check it by testing on real world problems in future.

We mentioned earlier, the proposed image averaging method has its potential to build image atlases. So, in the future, a new goal is to run examples on real data and build image atlases by our method, collaborate with brain scientists and see whether our atlases provide better researches. In the last 3D example, we see the computational time is quite long. That is mainly due to the step of Image Registration, which requires to solve 6 *Poisson* equations on each iteration. To extend our method to large data sets, we will need to enhance our algorithms to be more computationally efficient. A mixed method combines our approaches with deep learning methods is under construction. Based on the outperformed results of deep learning methods in many other problems, we are optimistic with this mixed approach.

In general, the studies carried out by our group have connected image analysis to the studies in geometrical features of meshes (ideally, diffeomorphisms) under differential operators — Jacobian determinant, curl vector and Laplacian. These suggest a new perspective on analyzing images. Our works are of course not limited to brain image. But it is a practical next step for us to focus on the study of brain images.

BIOGRAPHICAL STATEMENT

Zicong Zhou was born in Nanning, Guangxi, China, in 1988. He received his B.A. degree in Business Administration and B.S. degree in Mathematics from University of Maine at Presque Isle, USA, in 2012, his M.S. and Ph.D. degrees from the University of Texas at Arlington in 2015 and 2019, respectively, in Computational and Applied Mathematics. During the years of his doctoral studies, he worked as a Graduate teaching assistant in the department of mathematics and taught a number of undergraduate courses as instructor. He received Outstanding Graduate Research Award in 2018. His current research interests include nonrigid image registration, image atlas, adaptive moving mesh and deep leaning methods on brain image problem.

REFERENCES

[1] Anderson, D. and Liao, G.: An New Approach to Grid Generation. Applicable Analysis: An international Journal, vol. 44, no. 3-4, pp. 285-298, 1992.

[2] Akinlar, M.: A NEW METHOD FOR NONRIGID REGISTRATION OF 3D IMAGES, Dissertation, University of Texas at Arlington, 2009.

[3] Ashburner, J., Hutton, C., Frackowiah, R., Johnsrude, I., Price, C. and Friston, K.: Identifying Global Anatomical Differences: Deformation Based Morphometry. Human Brain Mapp., vol. 6, no. 5-6, pp. 348-357, 1998.

[4] Ashburner, J.: A fast diffeomorphic image registration algorithm. NCBI Neuroimage. vol. 38, no. 1, pp. 95-113, 2007. DOI: 10.1016/j.neuroimage.2007.07.007

[5] Ashburner, J.: VBM Tutorial. http://www.fil.ion.ucl.ac.uk/ john/misc/VBM-class10.pdf, Mar 15, 2010.

[6] Ashburner, J and Friston, K. J.: COMMENTS AND CONTROVERSIES, Why Voxel-Based Morphometry Should Be Used. NeuroImage, vol. 14, pp. 1238-1243, 2001. doi:10.1006/nimg.2001.0961

[7] Bochev, P. B.and Gunzburger, M. D.: Least-Squares Finite Element Methods. Springer Press. ISBN 978-0-387-30888-3, 2000.

[8] Bochev, P. B., Liao, G. and dela Pena, G.: Analysis and Computation of Adaptive Moving Grids by Deformation Numerical Methods for PDEs. vol. 12, no. 4, pp. 489-506. 1996

[9] Bookstein, F. L.: Voxel-Based Morphometry Should Not Be Used with Imperfectly Registered Images. NeuroImage, vol. 14, Issue 6, pp. 1454-1462. 2001

[10] Cai, X. X., Fleitas, D., Jiang, B. and Liao, G.: Adaptive grid generation based on the least-squares finite elements method. Computers and Mathematics with Applications, vol. 48, no 7-8, pp. 1007-1085, 2004.

[11] Castleman, K. R.: Digital Image Processing. Prentice Hall. ISBN 7-302-02828-1, 1998.

[12] Chen, Y. and Ye, X.: INVERSE CONSISTENT DEFORMABLE IMAGE REGISTRATION. The Legacy of Alladi Ramakrishnan in the Mathematical Sciences, SpringerVerlag, pp. 419-440, 2010.

[13] Chen, X. and Liao, G.: New Variational Method of Grid Generation with Prescribed Jacobian determinant and Prescribed Curl. arxiv.org/pdf/1507.03715, 2015.

[14] Chen, X. and Liao, G.: New method of averaging diffeomorphisms based on Jacobian determinant and curl vector. https://arxiv.org/abs/1611.03946, 2016.

[15] Chen, X.: Numerical Construction of Diffeomorphism and the Applications to Grid Generation and Image Registration. Dissertation, University of Texas at Arlington, 2015.

[16] Dacoragna, B. and Moser, J.: On A Partial Differential Equation Involving the Jacobian determinant. Ann. Inst H Poincaré, vol. 7, no. 1, pp. 1-26, 1990.

[17] Dalca, A. Balakrishinan, G., Guttag, J and Sabuncu, M.: Unsupervised Learning for Fast Probabilistic Diffeomorphic Registration. https://arxiv.org/abs/1805.04605, 2018.

[18] Dupuis, P., Grenander, U. and Miller, M. I.: Variational Problems on Flows of Diffeomorphisms for Image Matching. http://citeseerx.ist.psu.edu/viewdoc/similar?doi=10.1.1.53.8813&type=cc, 1998.

[19] Fleitas, D.: The Least-Square Finite Elements Method for Grid Deformation and Meshfree Applications. Dissertation, University of Texas at Arlington, 2005.

[20] Geng, X., et al: Unbiased Group-wise Image Registration: Applications in Brain Fiber Tract Atlas Construction and Functional Connectivity Analysis. J Med Syst. vol. 35, no. 5, pp. 921-928, 2011.

[21] Grajewski, M., Koster, M. and Turek, S: Numerical Analysis and Implementational Aspects of a New Multilevel Grid Deformation Method, 2009.

[22] Fortunato, M. and Persson, P.: High-order Unstructured Curved Mesh Generation using the Winslow Equation. Journal of Computational Physics. vol. 307, pp. 1-14, 2016.

[23] Gholipour, A., et al: A Normative Spatiotemporal MRI Atlas of the Fetal Brain for Automatic Segmentation and Analysis of Early Brain Growth. Nature. Scientific Reports vol. 7, no. 476, 2017.

[24] Goshtasby, A. A.: 2-D and 3-D Image Registration for Medical, Remote Sensing, and Industrial Applications. Wiley. ISBN 978-0-471-64954-0, 2005.

[25] Hsiao, H-Y., Hsieh, C-Y., Chen, X., Gong, Y., Luo, X. and Liao, G.: New Development of Nonrigid Registration. ANZIAM Journal, vol. 55, pp. 289-297, 2014. doi:10.1017/S1446181114000091

[26] Hsiao, H-Y: Helmholtz's Theorem Based Parametric Non-rigid Image Registration. Dissertation, University of Texas at Arlington, 2008

[27] Jiang, B.: The Least-Squares Finite Element Method: Theory and Applications on Computational Fluid Dynamics and Electromagnetic. Springer Press. ISBN 3-540-63934-9, 1998.

[28] Joshi, S., Davis, B., Jomier, M. and Gerig, G.: Unbiased Diffeomorphic Atlas Construction for Computational Anatomy NeuroImage, vol. 23, pp. 151-160, 2004. doi:10.1016/j.neuroimage.2004.07.068

[29] Joshi, S. and Modin, K.: Diffeomorphic Density Matching by Optimal Information Transport. SIAM Journal on Imaging Sciences, Jan 2015. DOI: 10.1137/151006238

[30] Joshi, S. and Modin, K.: On The Geometry and Shape of Brain Sub-Manifolds. International Journal of Pattern Recognition and Artificial Intelligence, vol. 11, no. 8, pp 1317-1343, 1997.

[31] Liao, G.,Pan, T.-W. and Su, J.: Numerical Grid Generator Based on Moser's Deformation Method. Numerical Methods for Partial Differential Equations, vol. 10, no. 1, pp. 21-31, 1994.

[32] Liu, J.: New Development of the Deformation Method. Dissertation, University of Texas at Arlington, 2006.

[33] Liu, F., Ji, S. and Liao, G.: An Adaptive Grid method and its Application to Steady Euler Flow calculations. SIAM J. Sci. Comput, vol. 20, no. 3 pp. 811-825, 1998.

[34] Liseikin, V.: Grid Generation Method. Springer Press. ISBN 3-54065686-3, 1999.

[35] Liu, F., Ji, S. and Liao, G.: An Adaptive Grid Method and its Application to Steady Euler Flow Calculations, SIAM J. Sci. Comput. 20, pp. 811–825, 1999. doi:10.1137/S1064827596305738

[36] Liao, G, Cai, X. X., Liu, J, Luo, X, Wang, J and Xue, J: Construction of Differentiable Transformations. Applied Mathematics Letters, Vol. 22, pp. 1543-1548, 2009.

[37] Liao, G., Cai, X., Fleitas, D., Luo, L., Wang, J. and Xue, J.: Optimal Control Approach to Data Set Alignment Applied Mathematics Letters, Vol 21, pp. 898–905, 2008.

[38] Liao, G., Lei, Z. and de la Pena, G.: Adaptive Grids for Resolution Enhancement. Shock Waves 12, pp. 153–156, 2002. doi:10.1007/s00193-002-0149-y

[39] Liao, G., Liu, F., de la Pena, G. C., Peng, D. and Osher, S.: Level-set-based Deformation Methods for Adaptive Grids, J. Comput. Phys. 159, pp. 103–122, 2000. doi:10.1006/jcph.2000.6432

[40] Narayana, P. A., Datta, S., Tao, G., Steinberg, J. L. and Moeller, F. G.: Effect of Cocaine on Structural Changes in Brain: MRI Volumetry using Tensor-Based Morphometry. Drug Alcohol Depend, vol. 111, no. 3, pp. 191-199, 2010.

[41] Novak, G. and Einstein, S. G.: Structural Magnetic Resonance Imaging as a Biomarker for the Diagnosis, Progression, and Treatment of Alzheimer Disease. Janssen Research and Development, 1125 Trenton-Harbourton Road, Titusville, NJ 08560, USA. Translational, Tools for CNS Drug Discovery, Development and Treatment. 2013.

[42] Ota, K., Oishi, N., Fukuyama, H. and SEAD-J Study Group: A Comparison of Three Brain Atlases for MCI Prediction. Journal of Neuroscience Methods. vol. 221, pp. 139-150, 2014.

[43] Pennec, X., Cachier, P. and Ayache, N.: Understanding the "Demon's Algorithm": 3D Non-Rigid Registration by Gradient Descent. MICCA. LNCS, vol. 1679, pp. 597-606, 1999.

[44] Sabuncu, A. et al: Construction of a 3D Probabilistic Atlas of Human Cortical structures. Neuroimage. vol. 39, no.3 pp. 1064–1080, 2008.

[45] Shattuck, A., Balci, S. and Golland, P.:: Discovering Modes of an Image Population through Mixture Modeling MICCA. Part II, LNCS 5242, pp. 381-389, 2008.

[46] Solin, P., Segeth, K. and Dolezel, I.: Higher-Order Finite Element Methods. CRC Press, ISBN 1-58488-438-X, 2004.

[47] Sotiras, A., Davatzikos, C. and Paragios, N.: Deformable Medical Image Registration: A Survey. IEEE Transactions on Medical Imaging, vol. 32, no. 7, 2013.

[48] Sunkin, S., et al: Deformable Medical Image Registration: A Survey. Nucleic Acids Research, vol. 41, no.7, 2013.

[49] Tang, Y., et al: The Construction of a Chinese MRI Brain Atlas: A Morphometric Comparison Study Between Chinese and Caucasian Cohorts. Neuroimage. vol. 51, no.1 pp. 33–41, 2010.

[50] Thompson, P. and Toga, Arthur.: Detection, Visualization and Animation of Abnormal Anatomic Structure with a Deformable Probabilistic Brain Atlas Based on Random Vector Field Transformations. Medical Image Analysis, Vol. 1, no. 4, pp 271-294, 1996.

[51] Tekalp, A. M.: Digital Video Processing. Pentice Hall, ISBN 7-302-02927-X, 1998.

[52] Thacker, N. A.: Tutorial: A Critical Analysis of Voxel Based Morphometry (VBM). http://www.tina-vision.net/docs/memos/2003-011.pdf, May, 2008.

[53] Tikhonov, A.N. and Arsenin, V.Y.: Solutions of Ill-Posed Problems. Winston and Sons, Washington DC, 1977.

[54] Xie, Z., Sevilla, R., Hassan, O. and Morgan, K.: The Generation of Arbitrary Order Curved Meshes for 3D Finite Element Analysis. Computational Mechanics, vol. 51. no.3, pp. 361-374, 2013.

[55] Zhou, Z., Chen, X and Liao, G.: A Novel Deformation Method for Higher Order Mesh Generation. arxiv.org/abs/1710.00291, 2017.

[56] Zhou, Z., Hildebrandt, B., Chen, X. and Liao, G.: Computational Technologies for Brain Morphometry. https://arxiv.org/abs/1810.04833, 2018.

[57] Zhou, Z., Chen, X., Cai. X. X., and Liao, G.: Uniqueness of Transformations Based on Jacobian Determinant and Curl Vector. https://arxiv.org/abs/1712.03443, 2017.