

MUTUAL INFORMATION BASED NON-RIGID IMAGE REGISTRATION  
USING ADAPTIVE GRID GENERATION: GPU  
IMPLEMENTATION AND APPLICATION  
TO BREAST MRI

by

MEI YI CHU

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 2008

Copyright © by Mei Yi Chu 2008

All Rights Reserved

## ACKNOWLEDGEMENTS

I would like to thank every person who helped me throughout my doctoral work. I gratefully acknowledge advisor Dr. Hua-mei Chen for his advice and guidance. He patiently led me back to the correct tracks when I was struggling on the wrong aspects of the research problems and he also gave me freedom to work on flexible schedule.

I wish to thank Dr. Guojun Liao, Dr. Jean Gao, Dr. Heng Huang and Dr. Roger Walker for being my committee members, who provided inspirations, support and time commitment. Dr. Liao contributed an intelligent mathematics framework as the backbone. His expertise and willingness to help in the mathematical methods is one of the crucial elements in this work.

Dr. Qi Peng from the University of Texas Health Science Center at San Antonio generously provided me numerous breast MRI datasets for my experiment. He gave me valuable comments and suggestions on working in the real-world clinical data.

I owe my gratitude to Hsi-Yue Hsiao, Chih-Yao Hsieh and Ting-Hung Lin who thoughtfully shared ideas and findings in our group. They assisted me to learn the new concepts and do the experiments.

My studies would not be possible without the administration and financial support of the Computer Science and Engineering Department and the Office of Graduate Studies. Endless thanks are dedicated to the graduate advisors Dr. Ramesh Yerraballi and Dr. Bahram Khalili for their step-by-step and practical instructions.

Finally, I would like to thank my family in Hong Kong for their remote care and high expectations are the major momentum for me to continue my studies. I am especially indebted to my loving husband who is constantly standing by my side, nourishing the household and providing unlimited encouragement.

July 22, 2008

## ABSTRACT

# MUTUAL INFORMATION BASED NON-RIGID IMAGE REGISTRATION USING ADAPTIVE GRID GENERATION: GPU IMPLEMENTATION AND APPLICATION TO BREAST MRI

Mei Yi Chu, PhD.

The University of Texas at Arlington, 2008

Supervising Professor: Hua-mei Chen

In this dissertation a new approach for non-rigid image registration using mutual information is introduced. A fast method for non-rigid registration is developed by adjusting divergence and curl of an intermediate vector field from which the deformation field is computed using finite difference method. The similarity measure mutual information is employed in the gradient-based cost minimization (or mutual information maximization) of the registration. The huge amount of data associated with MRI is handled by fully automated algorithm optimized with a multi-resolution topology preserving regriding scheme. The adaptive grid system naturally distributes more grids to deprived areas. The positive monitor function disallows grid folding and provides a mean to control the ratio of the areas between the original and transformed domain. The flexibility of the adaptive grid allocation could dramatically reduce processing time with quality preserved. Mutual information facilitates robust registration between different image modalities. Different types of joint histogram estimation are compared and integrated with the system. The whole system is also implemented on GPU which allows efficient parallel computation of vast

amount of 3D data in SIMT manner during different procedures. The GPU implementation offers up to 221 times speed up in the gradient normalization routine and around 40 times speed up in the overall calculation. This scheme is applied on 3D dynamic contrast-enhanced breast MRI, which requires the registration algorithm to be non-rigid, contrast-enhanced features preserving and within clinical visit time limit. Experiments show promising results and great potential for future extension.

## TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....	iii
ABSTRACT.....	iv
LIST OF ILLUSTRATIONS.....	ix
LIST OF TABLES.....	xii
Chapter	Page
1. INTRODUCTION .....	1
2. REVIEW OF RELATED WORK .....	3
2.1 Non-rigid Image Registration.....	3
2.2 Deformation Method .....	4
2.2.1 Deformation Based Grid Generation .....	4
2.3 Numerical Implementation.....	5
2.3.1 Finite Difference Div-Curl Solver.....	5
2.3.2 ODE Solver .....	6
2.4 Optimization.....	6
2.4.1 Gradient.....	7
2.4.2 Multi-resolution.....	8
2.5 Topology Preserving Regridding .....	8
2.6 Mutual Information .....	10
2.6.1 Joint Histogram Estimation .....	13
2.6.2 Interpolation Induced Artifacts. ....	14
2.6.3 Gradient Derivation .....	15
2.7 GPGPU and CUDA Programming.....	17
2.7.1 Image Processing using GPU.....	17

2.7.2 CUDA Programming.....	18
2.8 Breast MRI.....	19
3. PROPOSED WORK.....	21
3.1 Overview.....	21
3.2 MI Gradient w.r.t. $x, y, z$ Derivation .....	22
3.3 MI Gradient w.r.t. $f^i$ Derivation .....	24
3.4 Multi-resolution Topology Preserving Regridding.....	25
3.5 Compressibility-aware Monitoring.....	27
4. PROPOSED EFFICIENT IMPLEMENTATION USING CUDA .....	25
4.1 Method.....	29
4.1.1 Data Partition .....	30
4.1.2 Parallel Program .....	32
5. EXPERIMENTAL RESULTS VALIDATION.....	33
5.1 Setup.....	38
5.1.1 Contrast Enhanced Simulated 2D Image Data .....	38
5.1.2 Cropped 2D Image Data .....	38
5.1.3 Identical 2D Image Data .....	39
5.1.4 Simulated 3D Image Data.....	39
5.1.5 Cropped 3D Image Data.....	40
5.2 Error Measurement.....	40
5.3 Comparisons on Different Joint Histogram Estimation for Mutual Information.....	46
5.4 Comparisons on Different Similarity Measures .....	51
5.5 Comparisons on GPU and CPU Implementation.....	53
5.6 Compressibility-aware Monitoring .....	59
6. EXPERIMENTAL RESULTS USING CLINIC BREAST MRI.....	63
6.1 Overview.....	63

6.2 Setup.....	63
6.3 Results.....	64
7. CONCLUSIONS.....	73
8. FUTURE WORKS.....	74
REFERENCES.....	75
BIOGRAPHICAL INFORMATION.....	82



## LIST OF ILLUSTRATIONS

Figure	Page
2.1 Illustration of two-steps method in 2D.....	13
2.2 Illustration of one-step method in 2D .....	14
2.3 MI vs. horizontal displacement indicating interpolation artifact in PVI based rigid image registration.....	15
3.1 Algorithm Overview .....	21
4.1 Image size and Grid size of 3D Image .....	31
4.2 3D image data lined up as 1D array and form numerous 1D blocks.....	31
4.3 3D grid data lined up as 1D array and form numerous 1D blocks .....	31
4.4 Image data lined up as 3D arrays and form numerous 3D blocks .....	32
4.5 3D grid data lined up as 3D arrays and form numerous 3D blocks .....	32
4.6 2D redistribution filter.....	36
4.7 Two separable 1D redistribution filters .....	37
5.1 From left to right: The reference image $R$ , the template image $T$ with simulated distortion and contrast-enhanced cancerous regions and the subtracted image between $T$ and $R$ .....	39
5.2 From left to right: The reference image $R$ (time step 0), the template image $T$ (time step 1) and the subtracted image between $T$ and rigid-transformed $R$ .....	39
5.3 Changes in mutual information versus iterations.....	41
5.4 Average error versus iterations.....	42
5.5 Top: Deformation grid at different resolution level; Bottom: difference of the reconstructed reference image $R$ and the template image $T$ . Grid resolution improves from left to right.....	42
5.6 From left to right: The ground truth of the deformation grid and the reconstructed deformation grid.....	44
5.7 From left to right: The resultant divergence (top) and curl (bottom); the gradient of MI w.r.t. divergence (top) and curl (bottom); and the gradient of MI w.r.t. displacement in $x$ (top) and $y$ (bottom) directions.....	44

5.8	Average warping index values at different number of iterations for maximum allowed random displacements $d=2, 3, 4, 5$ . Markers indicate grid resolution refinement.....	46
5.9	MI values at different number of iterations for maximum allowed random displacements 2, 3, 4, 5. Markers indicate grid resolution refinement.....	46
5.10	Comparisons of various methods GPVE (top), PVI (middle) and Parzen Windows (bottom) in MI and gradient of MI with respect to displacement in $x$ . The top graph is MI, the middle graph is the analytical gradient and the bottom graph is the gradient estimated by finite difference.....	48
5.11	Difference image of template image and reconstructed reference image by GPVE (left), PVI (middle) and Parzen windows (right).....	50
5.12	Percentage increase in MI versus number of iterations for GPVE, PVI and Parzen windows.....	50
5.13	SSD and MI vs. number of iterations. Markers show the regriding points.....	52
5.14	Comparisons of average Warping Index of SSD and MI vs. number of iterations. Markers show the regriding points.....	53
5.15	Amount of M Grid/s of data processed by GPU vs. Total Grid Size of Runge-Kutta method and gradient normalization.....	56
5.16	Amount of M Pixel/s of data processed by GPU vs. Total Image Size.....	57
5.17	Observed Speed up of GPU to CPU vs. Total Grid Size.....	58
5.18	Jacobian vs. Count for registering reference image at time 1 to template Image at time 0.....	60
5.19	Jacobian vs Count for registering reference image at time 1 to template image at time 0. A number of extreme values are observed.....	60
5.20	Volume changes during different iterations in registering template image at time 0 with reference image at time 1.....	61
5.21	Comparison on the effect of with and without monitoring on MI vs. no. of iterations. Markers show grid refinement points .....	61
5.22	Comparison on the effect of with and without monitoring on average warping index vs. no. of iterations. Markers show grid refinement points.....	62
5.23	Deformed grids of near uniform sizes.....	62
6.1	MI vs. no. of iterations for registrations at time step 1 to 10.....	67

6.2	Differences of the central slice before and after registration at different time steps for different methods .....	68
6.3	Joint histograms of registration at different time steps of MI-Match.....	69
6.4	Joint histograms of registration at different time steps of MI-Norm (top) and SSD-Match (bottom).....	70
6.5	Top 10,000 intensity enhancement vs. time steps.....	70
6.6	Top 500 intensity enhancement vs. time steps.....	71
6.7	Intensity enhancement of possible malignant tissues after setting a threshold.....	71
6.8	Intensity enhancements of four sample points. Point A and B are possible lesion regions. Point C and D are normal tissues.....	72

## LIST OF TABLES

Table		Page
3.1	The complexity of the components in the framework .....	25
4.1	Nvidia GeForce 8800GTX Specification.....	30
5.1	A configuration of different resolution levels and corresponding resultant number of iterations, mutual information and average error.....	43
5.2	Time comparison of GPVE, PVI and Parzen windows on joint histogram estimation and gradient calculation.....	49
5.3	Time comparison of MI (GPVE) and SSD operations.....	53
5.4	Speed up of GPU to CPU at different resolution levels for grid-level operations.....	57
5.5	Speed up of GPU to CPU at different resolution levels for pixel-level operations.....	58
6.1	Actual Time for Different Time Steps.....	63

## CHAPTER 1

### INTRODUCTION

Medical image processing aims to relieve the overwhelming demand of manual analysis by radiologists. Image registration, which primarily aligns images from different modalities, time or viewing angle to integrate information, is among the most investigated image processing techniques. Information on the anatomical structure or pathological changes in organs and tissues becomes clearer for speedy examination by radiologists or physicians. Image registration is the indispensable to tasks include motion correction – reduction of respiratory motion in cardiac imaging; radiotherapy – offsetting the differences in registration of planning images acquired pre-treatment and during treatment in external beam radiotherapy; motion correction in aligning chest and abdomen images; calculation of the localized dose distributions which helps precisely deliver higher doses to be delivered to cancerous tissue without harming nearby normal tissue in radiotherapy; detection of osteoarthritis in joints imaging; spatial normalization in brain imaging; and dynamic contrast-enhanced MRI – motion correction before and after contrast injection and between scans acquired over time in breast imaging [13].

The adaptive grid generation algorithm in [8] has been proven to be a powerful non-rigid image registration tool in [9][22][23]. By adjusting the divergence and curl parameters, the grids move accordingly. The framework is built with gradient optimization which successfully accelerates the similarity measure minimization process. The compressibility-aware monitor function is directly related to the input divergence parameter instead of acting as constraints to the system. It is born with the ability to control the area or volume changes due to the applied geometrical transformation. Setting the ratio to unity resembles the incompressibility of human tissues. Yet a simple similarity measure, Sum of Square Difference (SSD) has been considered in [9][22][23], which may not be robust enough for some practical applications. To further

expand its capability in clinical applications, an all-purpose similarity measure is desirable. This motivates my study to include Mutual Information (MI) as similarity measure in the existing framework. MI is robust against noise and different illumination similarity measure [48]. Gradient of different MI variation are derived for the optimization scheme. This helps to investigate the artifacts brought by different joint histogram estimation methods. The introduction of MI in the registration process also increases the complexity of the system. Multi-resolution strategy with topology preserving regriding [10] cuts down the registration time which makes it feasible for clinical practice. With advanced GPU implementation, the performance of the system successfully breaks through all algorithm time bounds. Data intensity and computational intensity are instantly relieved. Our algorithm is capable to handle 3D real image data and capable to register between difference modalities. The algorithm is validated by synthetic motion to compensate the absence of ground truth. The framework is applied to dynamic contrast enhanced breast MRI which requires the algorithm to be able to handle local motion of soft tissues and contrast enhanced region. This is a positive match with the compressibility-awareness of adaptive grid generation and robustness of MI. The experiments show promising results for the 3D breast MRI. It is possible to include this algorithm in computerized breast cancer detection, which is an essential module in the next generation of clinical procedure.

## CHAPTER 2

### REVIEW OF RELATED WORK

#### 2.1 Non-rigid Image Registration

The goal of image registration is to find the spatial correspondence between two images of the same object taken at different time or angle, or from different modalities, or under different lighting condition. Rigid image registration corrects global alignment error by universal rotation, translation and scaling. Rigid image registration algorithm looks for the best global parameters set optimizing a similarity measure in the discrete form of search space. Since most human tissues do not undergo rigid transformation in the images, non-rigid image registration is commonly researched in medical image processing. Non-rigid image registration gives point-to-point precision correspondence between the input images. The motion associated with the correspondence could be specified by optical flow [16], free-form mode [36], elastic model [6] or viscous fluid [12]. Sometimes additional regularization term, penalty term or smoothness constraint [7] [34] [36] [37] is required to further control the motion.

In the literature, [36] is the pioneer researching on non-rigid image registration using mutual information, which proposed a multi-resolution scheme with a core B-spline free form deformation model focused on Breast MRI application. The missing gradient derivation in [36] was found in [42], where the analytical gradient of the Parzen windows method using B-splines was formulated in depth. This is a big advance in using mutual information since gradient is necessary for optimization in image registration. Numerous works has been proposed after that. To tailor-make the general non-rigid image registration for medial images taken in short time period, volume preserving condition should be imposed. Various approaches [34][37] are reported to include constraints on the Jacobian of the transformation in addition to the

deformation model. Normally, putting more counter-constraints leads to more experimental adjustment and complexity to the algorithm.

## 2.2 Deformation Method

Deformation model defines how one image can be warped to match another with a specific type and number of possible transformations. Affine transformation is used in rigid transformations. Rigid registration is fast since it involves only global rotation, translation and scaling of an image to align with another. Non-rigid registration is relatively slower since it requires local deformation thus more degrees of freedom [13].

In most existing work, non-rigid transformation is modeled as deformation of physical bodies driven by applied forces. As a result, the deformation is determined when the internal force is balanced by the external force. Alternatively, a new non-rigid image registration using the adaptive grid generation method developed in [8] is reported in [9]. It first constructs a vector field which satisfies a div-curl system. After that, local deformation field can be obtained from the intermediate vector field constructed earlier. This model is employed in this dissertation to solve the non-rigid medical image registration because the monitor function naturally restricts the deformation to enforce the incompressibility constraint [34][37].

### *2.2.1 Deformation Based Grid Generation*

The mathematic function of this work, the deformation based grid generation [8] is described as follows.

The task is to find a mapping function  $\phi_1(\xi)$ , given a monitor function  $f(\xi)$ , such that

$$J(\phi_1) \equiv \det \nabla \phi_1(\xi) = f(\xi) \quad (2.1)$$

To find such a transformation, the following steps were developed in [8].

First, find a vector field  $V(\xi)$  that satisfies

$$\text{div}V(\xi) = f(\xi) - 1 \quad (2.2)$$

Second, form



$$V_t = \frac{V}{t + (1-t)f} \quad (2.3)$$

then find  $\phi_t(\xi)$  by solving the ODE

$$\frac{d\phi_t(\xi)}{dt} = V_t(\phi_t) \quad t \in [0,1] \quad (2.4)$$

where  $\phi_t(\xi) = \phi(\xi, t)$  and  $\phi_1(\xi) = \phi(\xi, t=1)$ . A formal proof of this approach is presented in [28].

This profound adaptive grid generation method is actively researched. In [29], data set alignment problem is solved by optimal control of a similarity functional subject to a system of linear partial differential operators.

Notice that the solution  $\phi$  is not unique because only the divergence of the vector field  $\eta$  is specified. To obtain an unique  $\eta$ , both of its divergence as well as its curl need to be specified. This is known as the Helmholtz's theorem [20]. The task becomes finding a vector field  $\eta(x)$  that satisfies the following div-curl system:

$$\begin{cases} \nabla \eta(x) = f(x) - 1 \\ \nabla \times \eta(x) = g(x) \end{cases} \quad (2.5)$$

with null boundary condition  $\eta(x)=0$  and  $x \in \partial\Omega$ . In 2D,  $g(x)$  is a scalar function. In 3D,  $g(x)$  is a 3D vector function specifying the curl of the vector field.

### 2.3 Numerical Implementation

#### *2.3.1 Finite Difference Div-Curl Solver*

A div-curl system can be decoupled into 2 or 3 Poisson equations [24]. In 3D case, a div-curl system is given by

$$\begin{aligned} \text{div}U &= \frac{\partial U^x}{\partial x} + \frac{\partial U^y}{\partial y} + \frac{\partial U^z}{\partial z} = f^1 & \text{curl}_x U &= \frac{\partial U^z}{\partial y} - \frac{\partial U^y}{\partial z} = f^2 \\ \text{curl}_y U &= \frac{\partial U^x}{\partial z} - \frac{\partial U^z}{\partial x} = f^3 & \text{curl}_z U &= \frac{\partial U^y}{\partial x} - \frac{\partial U^x}{\partial y} = f^4 \end{aligned} \quad (2.6)$$

where  $f^1$  is the scalar monitor function  $f()$ , and  $f^2, f^3$  and  $f^4$  are the three components of the curl function  $g()$ . Assume that  $f^i, i = 1, 2, 3, 4$  are at least  $C^1$  continuous. Taking the derivative of

both sides of each equation with respect to (w.r.t.)  $x$ ,  $y$ , and  $z$  and combine the relevant terms, three Poisson equations are formed.

$$\begin{aligned}\Delta U^x &= f_x^1 + f_z^3 - f_y^4 \equiv F^1 \\ \Delta U^y &= f_y^1 + f_x^4 - f_z^2 \equiv F^2 \\ \Delta U^z &= f_z^1 + f_y^2 - f_x^3 \equiv F^3\end{aligned}\tag{2.7}$$

where  $f_k^i = \frac{\partial f^i}{\partial k}$ .

Fast Fourier Transform (FFT) based Poisson equation solver is chosen to solve the problem due to its algorithm simplicity and  $O(N \log N)$  serial time bound. A 3D version is derived from the 2D version described in [14].

### 2.3.2 ODE Solver

A multiple steps Euler's method is employed in Runge-Kutta method to iteratively approximate the displacement field from the intermediate vector field in Equation (2.4).

## 2.4 Optimization

Optimization process searches for better transformation parameters by exploring their impact on the computation of the image similarity metric [13] and greatly determines the efficiency of the procedure. Several optimization techniques incorporated in the framework include gradient descent method, multi-resolution reduction, pre-calculation and table lookup.

### 2.4.1 Gradient

Gradient descent method is used to help the optimization converge by finding local maxima along the MI versus  $f^i$  ( $i=1,2,3,4$ ) curves. To apply the gradient descent method for improvement on the iterative algorithm, gradient of the similarity measure SSD with respect to (w.r.t.) the control parameters  $f^i$  are required.

The gradient of SSD w.r.t. the input parameter  $f^i$  can be obtained by applying chain rule to the discretized form in 3D [22]:

$$\begin{aligned}
\frac{\partial ssd}{\partial f^J(I_i)} &= \sum_{k \in N(j)} \frac{\partial ssd}{\partial \phi^x(I_k)} \frac{\partial \phi^x(I_k)}{\partial \varphi^x(I_k)} \sum_{j \in N(i)} \frac{\partial \varphi^x(I_k)}{\partial F^1(I_j)} \frac{\partial F^1(I_j)}{\partial f^i(I_j)} \\
&+ \sum_{k \in N(j)} \frac{\partial ssd}{\partial \phi^y(I_k)} \frac{\partial \phi^y(I_k)}{\partial \varphi^y(I_k)} \sum_{j \in N(i)} \frac{\partial \varphi^y(I_k)}{\partial F^2(I_j)} \frac{\partial F^2(I_j)}{\partial f^i(I_j)} \\
&+ \sum_{k \in N(j)} \frac{\partial ssd}{\partial \phi^z(I_k)} \frac{\partial \phi^z(I_k)}{\partial \varphi^z(I_k)} \sum_{j \in N(i)} \frac{\partial \varphi^z(I_k)}{\partial F^3(I_j)} \frac{\partial F^3(I_j)}{\partial f^i(I_j)}
\end{aligned} \tag{2.8}$$

where  $J=1, 2, 3, 4$ ;  $\mathbf{f}$  represents the parametric vector;  $\phi$  denotes the deformation field;  $\varphi$  denotes the intermediate velocity field; and  $N(i)$  and  $N(j)$  denote some neighborhood of grid point  $I_i$  and  $I_j$  respectively. The effect of varying  $F^l$  value at grid  $I_j$  on  $\varphi^k$ ,  $l = 1,2,3$  and  $K =$

$x,y,z$ , determines the term  $\frac{\partial \varphi^k(I_k)}{\partial F^l(I_j)}$  and  $N(j)$  is its influence domain. This effect is established

through the inverse filter  $m^{-1}$  of the discrete Laplacian operator [22]. Similarly, the effect of

varying  $f^J$  value at grid  $I_i$  on  $F^l$ ,  $J = 1,2,3,4$  and  $l = 1,2,3$ , determines the term  $\frac{\partial F^l(I_j)}{\partial f^i(I_j)}$  and its

influence domain  $N(i)$ . This information can be attained from Equation (2.6). Using finite central difference method to approximate the involved derivatives, they can be represented as the following  $3 \times 3 \times 3$  filters.

$\frac{\partial F^1}{\partial f^1}, \frac{\partial F^2}{\partial f^1}, \frac{\partial F^3}{\partial f^1}$  are the  $3 \times 3 \times 3$  gradient filter in x, y, z-direction respectively.

$$\begin{aligned}
\frac{\partial F^2}{\partial f^2} &= -\frac{\partial F^3}{\partial f^1}, \frac{\partial F^3}{\partial f^2} = \frac{\partial F^2}{\partial f^1}, \frac{\partial F^1}{\partial f^3} = \frac{\partial F^3}{\partial f^1}, \frac{\partial F^3}{\partial f^3} = -\frac{\partial F^1}{\partial f^1}, \frac{\partial F^1}{\partial f^4} = -\frac{\partial F^2}{\partial f^1}, \frac{\partial F^2}{\partial f^4} = \frac{\partial F^1}{\partial f^1}, \\
\frac{\partial F^1}{\partial f^2} &= \frac{\partial F^2}{\partial f^3} = \frac{\partial F^3}{\partial f^4} = 0
\end{aligned} \tag{2.9}$$

Therefore, the gradient information can be computed through a series of convolutions as

$$\nabla_{f^J} ssd = \nabla_{\phi^x} ssd \otimes m^{-1} \otimes \nabla_{\frac{\partial F^1}{\partial f^J}} + \nabla_{\phi^y} ssd \otimes m^{-1} \otimes \nabla_{\frac{\partial F^2}{\partial f^J}} + \nabla_{\phi^z} ssd \otimes m^{-1} \otimes \nabla_{\frac{\partial F^3}{\partial f^J}} \tag{2.10}$$

### 2.4.2 Multi-resolution

Multi-resolution image registration is also known as progressive registration [26]. Registration is performed at a series of intermediate stages starting with lower complexity transforms and progressively using more complex ones. The process improves the results and shortens processing time. It is not unusual to use different types of transforms at each level of the multi-resolution pyramid. The coarse-to-fine improvement process increases the computational speed, accuracy and robustness. Local deformation of at one resolution level is used to initialize the displacement field at the next finer resolution level but the grids are reset to regular positions. In the literature, resolution refines in grid density, image resolution, B-spline levels and Gaussian pyramid in [18] [26] [31] [36] as non-rigid image registration is usually iterative upgrading to search for the best transform parameters.

### 2.5 Topology Preserving Regridding

The topology preserving regridding is proposed in [10]. A regular  $d$ -dimensional grid  $X$  is defined by

$$X \equiv \{\tilde{x}_{j_1 \dots j_d}, 1 \leq j_l \leq n_l, l = 1, \dots, d\} \quad (2.11)$$

where  $\tilde{x}_{j_1 \dots j_d} = (x_{j_1}, \dots, x_{j_d}) \in Z^d$  denotes a grid point (node) and  $n_l$  the size of the grid  $X$  along the  $l^{\text{th}}$  dimension. In the same manner, a general grid denoted by  $\mathfrak{X}$  is defined through grid points  $\tilde{x}_{j_1 \dots j_d} = (x_{j_1}, \dots, x_{j_d}) \in P^d$ . Let  $S : Z^d \rightarrow R$  and  $T : Z^d \rightarrow R$  be two discrete  $d$ -dimensional images,  $d=2$  or  $3$  and  $T^C : R^d \rightarrow R$  the continuous version of  $T$ . In accordance with [12],  $S$  is referred to as the data image / volume and  $T$  the template image / volume, which is a geometrically distorted version of an otherwise ideal template  $T^{\text{ideal}}$ .  $T^{\text{ideal}}$  is assumed to be perfectly aligned with the data  $S$ . The goal of image registration is to find a geometric transformation  $\phi : Z^d \rightarrow R^d$  such that the deformed template

$$T^C(X; \phi) \equiv T^C(\phi(X)) \equiv T^C(X + U(X)) \quad (2.12)$$

Is geometrically aligned with the data  $S(X)$ , i.e.  $T^C(\phi(X)) \equiv T^{ideal}(X)$ , where  $U(X) = \phi(X) - X$  denotes the deformation (displacement) field. Notice that the regular grid  $X$  is deformed into a general grid  $\aleph$  through transformation  $\phi$  by  $\aleph = \phi(X)$ . Regridding is defined in non-rigid image registration as the process of reinitializing a general (deformed) grid into regular at an intermediate stage of a non-rigid image registration process.

Let  ${}^n\varphi = {}^n\varphi \circ {}^{n-1}\varphi \circ \dots \circ {}^1\varphi$  denote the composite transformation using linear interpolation. Here each transformation  ${}^i\varphi$  is related to the corresponding incremental deformation field  ${}^iU$  through the following relation:

$${}^i\phi(X) = X + {}^iU(X) \quad (2.13)$$

where  $X$  is a regular grid and  ${}^i\phi(X)$  is a deformed grid  $\in \aleph$ . Assume  $J({}^n\varphi) \equiv \det \nabla^n \varphi \geq 0$  and  $J({}^{n+1}\varphi) < 0$ . That is, the topology of the deformed template  $T^{(n)}(X) = T^C({}^n\varphi(X))$  is preserved but not after concatenating on more incremental transformation  ${}^{n+1}\phi$ . When this

occurs, grids are repaired in the following steps such that  $J({}^{n+1}\hat{\varphi})$  is strictly positive, where

${}^{n+1}\hat{\varphi} = {}^{n+1}\hat{\varphi} \circ {}^n\varphi \circ \dots \circ {}^1\varphi$  and  ${}^{n+1}\hat{\varphi}$  denotes the incremental transformation after repairing from  ${}^{n+1}\phi$ .

GR\_step 1: Obtain the Jacobian determinant  $J({}^n\varphi)$  which is greater than 0.

GR\_step 2: Find the monitor function  $f$  and curl function  $g$  that result in  ${}^{n+1}\hat{\varphi}$ .

GR\_step 3: Modify the monitor function  $f$  by  $\hat{f} = \max(th / J({}^n\varphi), f)$

where  $th$  is the allowed minimum Jacobian determinant value and the operator  $\max$  is preformed in an element by element fashion.

GR\_step 4: Construct the repaired transformation  $\hat{\varphi}^{n+1}$  from  $\hat{f}$  and  $\hat{g}$  using the grid generation method in section 2.2.

## 2.6 Mutual Information

Mutual Information (MI) is a similarity measure which quantitatively measures how well a transformed reference image matches the template image. The metric is used by the optimization process to evaluate the quantitative criterion at various positions in the transform parameter search space. For gradient-based optimization schemes, the derivatives of the measure with respect to each transform parameter are also required.

MI is invariant to changes on lightings, robust to noise, has sharp maxima and computational simplicity [36]. Most existing similarity measures for intensity based image registration problems depend on a certain specific relationship between the intensities of the images to be registered. When images from different sources or sensors are involved in the registration, an information theoretic or statistical criterion such as MI is a preferable choice of similarity measure [45]. The computation of MI involves joint histogram estimation and the global maximization.

### *2.6.1 Joint Histogram Estimation*

Theoretically, MI of two random variables  $T$ ,  $R$  is defined by

$$MI(T, R) = \sum_{t,r} P_{T,R}(t, r) \log \frac{P_{T,R}(t, r)}{P_T(t)P_R(r)} \quad (2.14)$$

where  $P_T(t)$  and  $P_R(r)$  are the marginal probability mass functions and  $P_{T,R}(t,r)$  is the joint probability mass function of  $T$  and  $R$ :

It measures the magnitude of dependence of  $T$  and  $R$  by computing the distance between the joint probability  $P_{T,R}(t,r)$  and the probability associated with the case of complete independence  $P_T(t)P_R(r)$ , by means of relative entropy:

$$MI(T, R) = H(T) + H(R) - H(T, R) \quad (2.15)$$

where  $H(T)$  and  $H(R)$  are the entropies of  $T$  and  $R$  and  $H(T, R)$  is their joint entropy:

$$\begin{aligned}
H(T) &= \sum_t -P_T(t) \log P_T(t) \\
H(R) &= \sum_r -P_R(r) \log P_R(r) \\
H(T, R) &= \sum_{t,r} -P_{T,R}(t, r) \log P_{T,R}(t, r)
\end{aligned} \tag{2.16}$$

where  $P_T(t)$  and  $P_R(r)$  are the marginal probability mass functions and  $P_{T,R}(t, r)$  is the joint probability mass function of the two images  $T$  and  $R$ :

$$\begin{aligned}
P_{T,R}(t, r) &= \frac{h(t, r)}{\sum_{t,r} h(t, r)} \\
P_T(t) &= \sum_r P_{T,R}(t, r) \\
P_R(r) &= \sum_t P_{T,R}(t, r)
\end{aligned} \tag{2.17}$$

where  $h$  is the joint histogram of the image pair with each image have  $L$  intensity levels. It is a 2D matrix given by

$$\begin{bmatrix}
h(0,0) & h(0,1) & \dots & h(0, L-1) \\
h(1,0) & h(1,1) & & h(1, L-1) \\
\dots & & & \dots \\
h(L-1,0) & h(L-1,1) & \dots & h(L-1, L-1)
\end{bmatrix} \tag{2.18}$$

Roughly speaking, mutual information  $MI$  is maximum when the joint entropy  $H(T, R)$  is minimum. It happens when the two images are registered, thus the uncertainty of the joint histogram is minimized.

#### 2.6.1.1 Two-Steps Method

Traditionally, a joint histogram is estimated in two steps. First, the intensity values of the reference image  $R$  at the transformed positions are calculated, usually by intensity interpolation. A simple example of two-steps joint histogram estimation is the nearest neighbor interpolation method. The intensity values at transformed position are estimated as the intensity values of the nearest neighbors. Another example is tri-linear interpolation in 3D, which the new intensity value at each transformed point is evaluated as the tri-linear interpolated value of its eight

nearest neighbors. Rounding off to the nearest integer operation is required. Second, the corresponding joint histogram entry is incremented by one. Let  $T$  denote the template image,  $R$  denote the reference image, a pixel of position  $(i_T, j_T, k_T)$  in  $T$  transformed to a possibly non-integral position  $(i_R, j_R, k_R)$  in  $R$ . The intensity value at  $(i_R, j_R, k_R)$  is interpolated by the neighbor pixels  $(i_R', j_R', k_R')$ . The joint histogram bin  $h(T(i_T, j_T, k_T), R(i_R, j_R, k_R))$  is increased by one where

$$R(i_R, j_R, k_R) = \text{round}\left(\sum R(i_R', j_R', k_R')(i_R - i_R')(j_R - j_R')(k_R - k_R')\right) \quad (2.19)$$

Notice that equation (2.17) shows the 3D operation. In 2D, the  $k$  component should be removed.

The disadvantage of the two-steps approach is that the resulting MI is not smooth due to the rounding. Smooth MI leads to better optimization results. In some more advanced two-steps methods, for example Parzen Windows, B-spline weights summing to one are used to update a neighborhood of bins instead of a single bin to yield smoother MI.

For comparison purpose, a two-step joint histogram estimation method using Parzen windows [42] is included and modified. In the first step, intensities in the reference image are tri-linearly interpolated at the displaced positions. In the second step, B-splines are used for Parzen windowing which determine the weights distributed to neighboring histogram values. The joint histogram  $h(t, r)$  is updated by adding:

$$\beta(t - T(i_T, j_T, k_T))\beta(r - R(i_R, j_R, k_R)) \quad (2.20)$$

where  $(t, r)$  lies in the histogram neighborhood of  $(T(i_T, j_T, k_T), R(i_R, j_R, k_R))$  and  $R(i_R, j_R, k_R)$  is obtained by tri-linear interpolation of the intensities of the eight nearest pixels:

$$R(i_R, j_R, k_R) = \sum_{i', j', k' \in \mathbb{Z}^2} \tau(i_R - i_R')\tau(j_R - j_R')\tau(k_R - k_R')R(i_R', j_R', k_R') \quad (2.21)$$

where  $\tau()$  denote the triangular function:

$$\tau(x) = \begin{cases} 1 - x & \text{if } 0 \leq x \leq 1 \\ 1 + x & \text{if } -1 \leq x < 0 \\ 0 & \text{otherwise} \end{cases} \quad (2.22)$$



Since  $t=T(i_T, j_T, k_T)$ , using 1<sup>st</sup> order B-spline for the first half in the equation (2.20), the histogram function  $h(t, r)$  is updated by adding

$$\beta(r - R(i_R, j_R, k_R)) \quad (2.23)$$

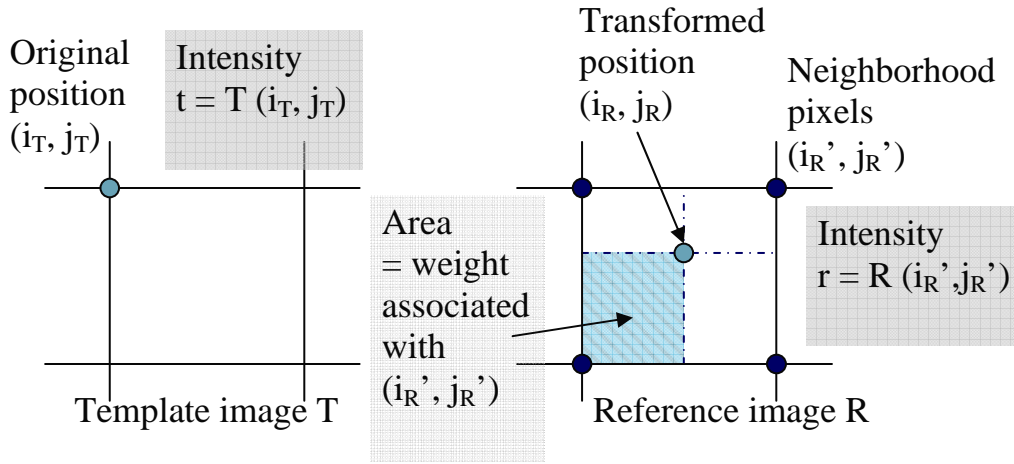


Figure 2.1 Illustration of two-steps method in 2D.

### 2.6.1.2 One-Step Method

Instead of interpolating new intensity value and updating bins in two steps, weights could be used to update the histogram bins determined by the neighborhood pixels directly. This is first known as partial volume interpolation (PVI), which was developed for 3D image registration. The weights distributed to the voxels in the neighborhood are calculated by a linear function of its distance to the transformed position. Usually, closer neighborhood pixels get larger weights. Notice that no new intensity values are introduced. Using the same notations in (2.19), the histogram bin  $h(T(i_T, j_T, k_T), R(i_R', j_R', k_R'))$  for 3D images is updated by adding:

$$\tau(i_R - i_R')\tau(j_R - j_R')\tau(k_R - k_R') \quad (2.24)$$

where  $\tau()$  is the triangular function defined in equation (2.21).

To further improve smoothness, generalized partial volume estimation (GPVE) was derived in [45]. B-spline kernel functions of 1<sup>st</sup>, 2<sup>nd</sup> or 3<sup>rd</sup> order are used to distribute weights to the neighborhood of sizes corresponding to their support. Higher order gives boarder support and smoother MI. 1<sup>st</sup> order GPVE is equivalent to PVI. Let  $\beta()$  denotes the B-spline kernel function, the GPVE histogram bin  $h(T(i_T, j_T, k_T), R(i_R', j_R', k_R'))$  for 3D images is updated by adding:

$$\beta(i_R - i_R')\beta(j_R - j_R')\beta(k_R - k_R') \quad (2.25)$$

where  $(i_R', j_R', k_R')$  are the points within the support of the corresponding B-spline kernels.

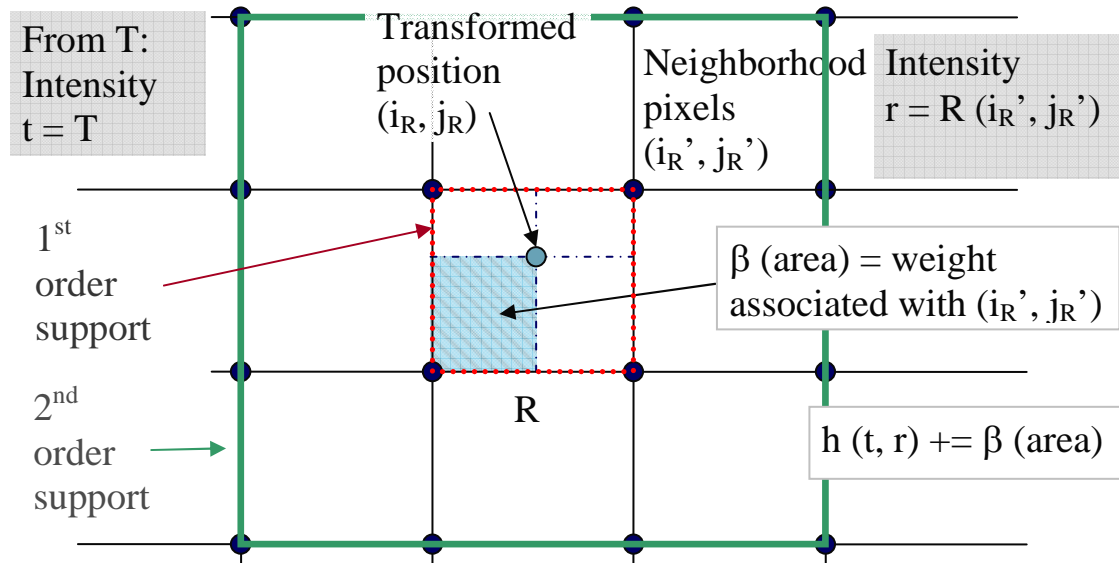


Figure 2.2 Illustration of one-step method in 2D.

### 2.6.2 Interpolation Induced Artifacts

When MI based rigid registration algorithm is tested using an identical image pair with slight changes in displacement in vertical or horizontal direction, some periodic patterns, known as the interpolation induced artifacts, occur in the PVI MI curve as illustrated in Figure 2.3 [45]. This artifact is due to the many of the grid lines may be aligned along that dimension under certain geometric transformations. The one-step PVI method gives concave shape artifacts

caused by simple linear interpolation. Presence of noise in the images further deteriorates situation. The root of the artifact in PVI is coming from splitting of a joint histogram entry into a few smaller entries. The split increases entropy at non-grid-point thus decreases joint histogram value [31].

Using GPVE, the artifacts are dramatically reduced. It is important to obtain a smooth MI curve facilitating the optimization procedure.

Though interpolation induced artifacts are traditionally discussed only in the context of rigid registration, we show that it also affect non-rigid image registration.

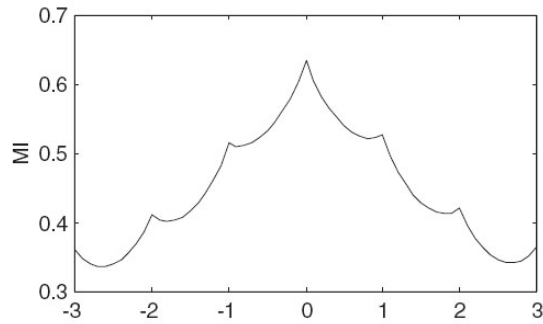


Figure 2.3 MI vs. horizontal displacement indicating interpolation artifact in PVI based rigid image registration.

### 2.6.3 Gradient Derivation

The derivatives of the image similarity measure with respect to the parameters of the transform are essential in the optimization process. Easily computed simple finite differences could estimate the derivatives with a significant computation load [48]. It is more desirable to obtain the derivatives by automatic differentiation software [21] which applies differential chain-rule numerically. When automatic differentiation software fails to differentiate due to the implicit complicated relationship between the function and its input parameters such as interpolation, analytical differentiation is the only choice left, which is exactly our case.

Gradient of MI calculation was a tedious and erroneous task due to its non-obvious relation with the displacement. The derivation of the gradient of MI with respect to the

displacement field is inspired by [15][42]. The mathematical errors in [15] are corrected in Equations (2.26) to (2.33). The gradient derivation in Chapter 3 is also redefined and corrected from [15] to get rid of the ambiguous windows function and its erroneous gradient. Explicit histogram function  $h()$  and its horizontal and vertical gradient components in 3D are explained instead in Chapter 3.

The template image  $T$  is assumed to be constant. The reference image  $R$  is re-sampled based on the displacement vector  $\mathbf{u}$ . MI depends solely on the displacement vector. Equation (2.14) is rewritten into:

$$\begin{aligned}
 MI(\mathbf{u}) &= \sum_{t,r} P_{T,R}(t,r,\mathbf{u}) \log \frac{P_{T,R}(t,r,\mathbf{u})}{P_T(t)P_R(r,\mathbf{u})} \\
 MI(\mathbf{u}) &= \sum_{t,r} P_{tr} \log \frac{P_{tr}}{P_t P_r}
 \end{aligned} \tag{2.26}$$

The Jacobian of MI is formulated as:

$$\begin{aligned}
 \frac{\partial MI(\mathbf{u})}{\partial \mathbf{u}} &= \sum_{t,r} \left[ (P_{tr})' \log \frac{P_{tr}}{P_t P_r} + P_{tr} \frac{P_t P_r}{P_{tr}} \frac{\partial}{\partial u_x} \left( \frac{P_{tr}}{P_t P_r} \right) \right] \\
 &= \sum_{t,r} \left[ (P_{tr})' \log \frac{P_{tr}}{P_t P_r} + P_t P_r \left( - (P_t P_r)^{-2} \frac{\partial P_t P_r}{\partial u_x} P_{tr} + \frac{(P_{tr})'}{P_t P_r} \right) \right] \\
 &= \sum_{t,r} \left[ (P_{tr})' \log \frac{P_{tr}}{P_t P_r} - \frac{(P_r)'}{P_r} P_{tr} + (P_{tr})' \right] \\
 &= \sum_{t,r} (P_{tr})' \left[ \log \frac{P_{tr}}{P_t P_r} + 1 \right] - \sum_{t,r} \frac{(P_r)'}{P_r} P_{tr}
 \end{aligned} \tag{2.27}$$

In addition to that, the last term disappears:

$$\sum_{t,r} \frac{(P_r)'}{P_r} P_{tr} = \sum_r \frac{(P_r)'}{P_r} \sum_t P_{tr} = \sum_r \frac{(P_r)'}{P_r} P_r = \sum_r (P_r)' = 0 \tag{2.28}$$

Note that the summation of all the changes in the histogram always equals zero.

$$\sum_{t,r} (P_{tr})' \left[ \log \frac{P_{tr}}{P_t P_r} + 1 \right] = \sum_{t,r} (P_{tr})' \left[ \log \frac{P_{tr}}{P_r} - \log P_t + 1 \right] = \sum_{t,r} (P_{tr})' \log \frac{P_{tr}}{P_r} \tag{2.29}$$

The second term equals zero since probability of template image is constant:

$$\sum_{t,r} (P_{tr})' \log P_t = \sum_t \log P_t \sum_r (P_{tr})' = \sum_t \log P_t (P_t)' = \sum_t \log P_t(0) = 0 \quad (2.30)$$

The last term equals zero too. Again summing up the changes in the histogram results in zero:

$$\sum_{t,r} (P_{tr})' = 0 \quad (2.31)$$

The gradient of MI with respect to displacement vector  $\mathbf{u}$  is:

$$\frac{\partial MI(\mathbf{u})}{\partial \mathbf{u}} = \sum_{t,r} (P_{tr})' \log \frac{P_{tr}}{P_r} \quad (2.32)$$

From Equation (2.14), we have:

$$P_{T,R}(t,r) = \frac{h(t,r)}{\sum_{t,r} h(t,r)} = \frac{h(t,r)}{N} \quad (2.33)$$

Please continue to read Chapter 3 to see how the gradient of specific histogram estimation is derived.

## 2.7 GPGPU and CUDA Programming

Graphics Processor Unit (GPU) is good at compute-intensive, highly parallel computation. It was originally designed for graphics rendering only. Due to the fast growing development in GPU technology, its power increases sharply every year, whereas CPU power increases moderately every year. General-Purpose computation on GPUs (GPGPU) is getting more popular and applicable in many areas of research. GPU gives immediate performance speed up for most data parallel suitable problems, especially those with high ratio of arithmetic operations to memory operations.

### *2.7.1 Image Processing using GPU*

Increasing interests in using GPU is observed in the medical image processing area. In [17], the authors were able to obtain a speed up of 44 in learning-based non-rigid multi-modal registration. In their work, OpenGL and OpenGL Shading Language (GLSL) were used for GPU programming. Their main procedure included Gaussian filtering and joint histogram

computation. Separable Gaussian filter was used to process 3D data. Vertex buffer object is used to store immediate joint histogram result. Parallel matrix multiplication and summation was reported in interactive organ segmentation [19]. GPU was used to compute partial derivatives in level-sets for 3D segmentation [27], 10 to 15 times speed up was observed. GPU was also used to accelerate tomographic reconstruction in [46].

### *2.7.2 CUDA Programming*

In older days, OpenGL and DirectX were the only graphics API to communicate with GPU. There are more user friendly high level GPGPU programming environments such as BrookGPU [2], Sh [4], CUDA [3] and CTM [1]. CUDA is chosen due to its popularity and availability of documentation.

Compute Unified Device Architecture (CUDA) is a new hardware and software architecture for communicating on the GPU as a data-parallel computing device without the need of mapping them to a graphics API. The GPU is viewed as a compute device capable of executing massive threads in parallel and as a coprocessor to the main CPU. The CPU is viewed as the host. CPU has host memory and GPU has device memory. Host memory needs to be copied into device memory before GPU execution. An application is executed many times by threads independently on different data. A program compiled into an instruction set of the device called a kernel is downloaded to the device.

From the software perspective, a batch of threads is organized as a grid of thread blocks. A thread block is a batch of threads that cooperate together efficiently sharing data through some fast shared memory and synchronizing their execution to coordinate memory accesses. Blocks of same dimension and size that execute the same kernel can be batched together into a grid of blocks. Threads in different thread blocks cannot communicate and synchronize. The number of concurrent running blocks depends on the parallel capabilities of the device.

From the hardware perspective, the GPU is a set of Single Instruction, Multiple Thread (SIMT) architecture multiprocessors with on-chip shared memory. A grid of thread blocks is executed on the device for execution on the multiprocessors. Each multiprocessor processes batches of blocks one batch after the other. A block is processed by only one multiprocessor. The number of concurrent running blocks depends on the resources available. The issue order of the blocks within a grid of thread blocks is undefined and there is no synchronization mechanism between blocks, so threads from two different blocks of the same grid cannot safely communicate with each other through global memory during the execution of the grid.

CUDA provides a more user-friendly programming environment for data-parallel computing than traditional parallel programming. No scheduling is involved. The data-parallel code and the thread configuration would handle all parallel processing. The performance of GPU depends on the reads and writes manner on the different types of memory, arithmetic intensity, the way data is partitioned and the configuration of threads [3]. To maximize the performance, the limitations of GPU should be considered and thread configurations should be tuned.

### 2.8 Breast MRI

Early detection of breast cancer is one of the important factors to the cure of the disease [25]. Dynamic contrast-enhanced breast MRI is a less invasive tool to detect cancerous tissues than traditional surgery. The contrast agent gadolinium dimeglumine is taken by the patient prior the imaging process. A time series of MRI are then acquired. Cancerous tissues show strong and rapid contrast enhancement, normal tissues show weak or no response. Accurate breast cancer detection relies on the classification of subtle pathological changes over time [5]. During a substantial imaging period, it is impossible for the patient to hold the breath and stay still. Motion correction techniques are used to align the breast MRI sequences to compensate the breathing, cardiac and other motions.

Three factors should be considered in the registration of dynamic contrast-enhanced breast MRI: softness of breast tissues, contrast-enhanced features and the huge amount of high-definition image data. In [36], the local motion is corrected by a free-form deformation B-splines deformation model and normalized mutual information is used for quality control. The correction is controlled by minimization of the cost functional comprising the similarity functional and the regularization functional which penalizes smoothness of the deformation field. This regularization term is replaced by an incompressibility constraint for soft tissues in [31]. The constraint helps preventing the mutual information driven registration, which is designed to minimize intensity inconsistency, from shrinking contrast enhancing features. This is implemented by enforcing unity deviations of the Jacobian determinant of the deformation. In [18], a multi-resolution optical flow with brightness consistency assumption relaxed and subjected to a regularized best-fit within a family of transforms is addressed. A bio-mechanically based elastic breast registration was developed [35] to fit the real breast tissues movement by using position of skin and muscle surface as the only boundary conditions instead of regulating by a smoothness term or regularization term. It leads to the investigation of the possibility of other suitable models such as a div-curl system to model the motion. In optical flow algorithm, the curl represents the amount of rotation of objects or the camera; the divergence represents the rate of approach of objects [40].



CHAPTER 3  
PROPOSED WORK  
3.1 Overview

The goal is to properly register the unchanged template image  $T$  with the reference image  $R$  by choosing a set of parameters  $f$  and  $g$ , which control the deformation. In the reconstruction,  $T(i_T, j_T, k_T)$  corresponds  $R(i_T+u_x, j_T+u_y, k_T+u_z)$ , where  $\mathbf{u} = (u_x, u_y, u_z)$  is the displacement field in x, y, and z dimensions respectively. In the div-curl system, divergence  $f^1$  and curl  $f^2, f^3, f^4$  regulate the intermediate vector field involved in the process. A Runge-Kutta method approximates the displacement field from the intermediate vector field.

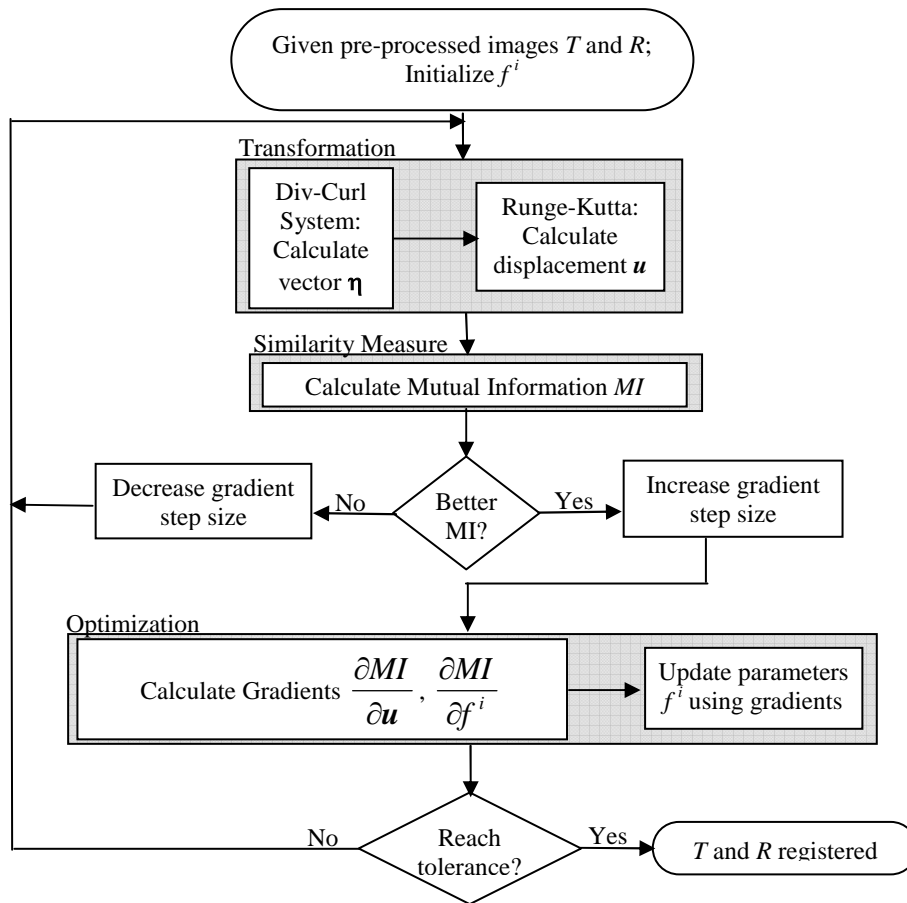


Figure 3.1 Algorithm Overview

A general registration algorithm can be decomposed into three main components: the transformation model, the similarity measure and the optimization process [13]. The transformation model in use is the div-curl system and Runge-Kutta estimation which takes divergence and curl as inputs and output deformation field. The similarity measure is Mutual Information (MI) which quantifies the wellness of the registration. The optimization process has various components including gradient calculation and multi-resolution. Figure 3.1 illustrates the overall procedure.

### 3.2 MI Gradient w.r.t. x, y, z Derivation

In this section, the derivation of MI gradient of different histogram estimation methods is extended from section 2.6 and published in [11].

Intuitively, in GPVE, the change in a displacement  $\mathbf{u}$  leads to changes in the B-spline weights distributed to the histogram entries of the neighboring pixels and thus the joint probability mass function. A formal derivation follows.

$$\begin{aligned} \text{For } \forall t = T(i_T, j_T, k_T), \quad \forall r = R(i_R', j_R', k_R') \\ h(t, r) = \sum_{i, j, k \in \mathbb{Z}^2} \sum_{i', j', k' \in \mathbb{Z}^2} \beta(i_T + u_x - i_R') \beta(j_T + u_y - j_R') \beta(k_T + u_z - k_R') \end{aligned} \quad (3.1)$$

where the notations are same as in Chapter 2. An entry represents the count of all pixels in  $T$  having intensity value  $t$  and all pixels belong to any neighborhood of coordinates  $(i_R, j_R, k_R)$  in  $R$  having intensity value  $r$ . The count may not be an integer due to the weights of B-spline spreading throughout neighborhood. Explicitly, the partial derivatives become:

$$\begin{aligned}
(P_{tr})' &= \left( \frac{\partial P_{tr}}{\partial u_x}, \frac{\partial P_{tr}}{\partial u_y}, \frac{\partial P_{tr}}{\partial u_z} \right) \\
\frac{\partial P_{tr}}{\partial u_x} &= \frac{\partial}{\partial u_x} \left[ \frac{1}{N} \sum_{i,j,k \in \mathbb{Z}^3} \sum_{i',j',k' \in \mathbb{Z}^3} \beta(i_T + u_x - i_R') \beta(j_T + u_y - j_R') \beta(k_T + u_z - k_R') \right] \\
&= \frac{1}{N} \sum_{i,j,k \in \mathbb{Z}^3} \sum_{i',j',k' \in \mathbb{Z}^3} \frac{\partial}{\partial u_x} \beta(i_T + u_x - i_R') \beta(j_T + u_y - j_R') \beta(k_T + u_z - k_R') \quad (3.2) \\
\frac{\partial P_{tr}}{\partial u_y} &= \frac{1}{N} \sum_{i,j,k \in \mathbb{Z}^3} \sum_{i',j',k' \in \mathbb{Z}^3} \beta(i_T + u_x - i_R') \frac{\partial}{\partial u_y} \beta(j_T + u_y - j_R') \beta(k_T + u_z - k_R') \\
\frac{\partial P_{tr}}{\partial u_z} &= \frac{1}{N} \sum_{i,j,k \in \mathbb{Z}^3} \sum_{i',j',k' \in \mathbb{Z}^3} \beta(i_T + u_x - i_R') \beta(j_T + u_y - j_R') \frac{\partial}{\partial u_z} \beta(k_T + u_z - k_R')
\end{aligned}$$

As in [43][44], the differentiation property of B-spline is:

$$\frac{\partial \beta^n(x)}{\partial x} = \beta^{n-1}\left(x + \frac{1}{2}\right) - \beta^{n-1}\left(x - \frac{1}{2}\right) \quad (3.3)$$

where  $n$  denotes the order of B-spline kernel in use.

Finally, the derivatives are:

$$\left\{ \begin{aligned}
\frac{\partial MI(\mathbf{u})}{\partial u_x} &= \sum_{t,r} \log \frac{P_{tr}}{P_r} \frac{1}{N} \sum_{i,j,k \in \mathbb{Z}^3} \sum_{i',j',k' \in \mathbb{Z}^3} \beta'(i_R - i_R') \beta(j_R - j_R') \beta(k_R - k_R') \\
\frac{\partial MI(\mathbf{u})}{\partial u_y} &= \sum_{t,r} \log \frac{P_{tr}}{P_r} \frac{1}{N} \sum_{i,j,k \in \mathbb{Z}^3} \sum_{i',j',k' \in \mathbb{Z}^3} \beta(i_R - i_R') \beta'(j_R - j_R') \beta(k_R - k_R') \\
\frac{\partial MI(\mathbf{u})}{\partial u_z} &= \sum_{t,r} \log \frac{P_{tr}}{P_r} \frac{1}{N} \sum_{i,j,k \in \mathbb{Z}^3} \sum_{i',j',k' \in \mathbb{Z}^3} \beta(i_R - i_R') \beta(j_R - j_R') \beta'(k_R - k_R')
\end{aligned} \right. \quad (3.4)$$

Values of B-spline function and differentiation of B-spline function are looked up from a pre-calculated table to speed up calculation.

Similarly, when PVI is in use, the  $\beta()$  function will be replaced by a triangular function  $\tau()$  in Equation (2.22) and the derivative of a triangular function is straight forward.

When Parzen window in equation (2.20) is used, the equation (3.4) will become:

$$\left\{ \begin{array}{l}
\frac{\partial MI(\mathbf{u})}{\partial u_x} \\
= \sum_{i,r} \log \frac{P_{tr}}{P_r} \frac{1}{N} \sum_{i,j,k \in \mathbb{Z}^3} \beta'(r - R(i_R, j_R, k_R))(-1) \sum_{i',j',k' \in \mathbb{Z}^3} \tau'(i_R - i_R') \tau'(j_R - j_R') \tau'(k_R - k_R') \nabla R \\
\frac{\partial MI(\mathbf{u})}{\partial u_y} \\
= \sum_{i,r} \log \frac{P_{tr}}{P_r} \frac{1}{N} \sum_{i,j,k \in \mathbb{Z}^3} \beta'(r - R(i_R, j_R, k_R))(-1) \sum_{i',j',k' \in \mathbb{Z}^3} \tau'(i_R - i_R') \tau'(j_R - j_R') \tau'(k_R - k_R') \nabla R \\
\frac{\partial MI(\mathbf{u})}{\partial u_z} \\
= \sum_{i,r} \log \frac{P_{tr}}{P_r} \frac{1}{N} \sum_{i,j,k \in \mathbb{Z}^3} \beta'(r - R(i_R, j_R, k_R))(-1) \sum_{i',j',k' \in \mathbb{Z}^3} \tau'(i_R - i_R') \tau'(j_R - j_R') \tau'(k_R - k_R') \nabla R
\end{array} \right. \quad (3.5)$$

This could be understood as the change in a displacement field  $\mathbf{u}$  leads to changes in its weights associated to neighboring pixels and change in its intensity value, thus leads to changes in the B-spline weights distributed to the histogram neighboring intensities and finally the joint probability mass function.

### 3.3 MI Gradient w.r.t. $f^i$ Derivation

The original gradient calculation proposed in [22] as mention in section 2.4 provides an approximation of the gradient values by applying inverse filtering. The choice of choosing a suitable mask size for inverse filtering is quite experimental. Convolution with inverse filter is slow even FFT is used instead of traditional convolution when data size or filter size is huge. Since the inverse filtering of the Laplacian operator is mathematically equivalent to solving Poisson equation, the same Poisson FFT solver developed for solving div-curl system replaces the presence of inverse filtering.

$$B = A \otimes m^{-1} \Leftrightarrow A = B \otimes m \Leftrightarrow \Delta B = A \Leftrightarrow B = \Delta^{-1}(A) \quad (3.6)$$

where  $\Delta^{-1}()$  denotes the Poisson solver;  $A$  is the input and  $B$  is the solution to the Poisson equation.

The gradient equation (2.10) becomes

$$\nabla_{f^j} MI = \nabla_{\frac{\partial F^1}{\partial f^j}} (\Delta^{-1}(\nabla_{\phi^x} MI)) + \nabla_{\frac{\partial F^2}{\partial f^j}} (\Delta^{-1}(\nabla_{\phi^y} MI)) + \nabla_{\frac{\partial F^3}{\partial f^j}} (\Delta^{-1}(\nabla_{\phi^z} MI)) \quad (3.7)$$

### 3.4 Multi-resolution Topology Preserving Regridding

In this framework, starting with the lowest suitable grid size along with the lowest appropriate image size and the lowest feasible intensity level, the algorithm moves to the next level once a predefined tolerance is reached. The resultant displacement field in the previous level is then interpolated as the initial values in the next level but grids are reset. Initializing in coarse-to-fine fashion not only saves time, it also helps avoid local maxima from macro view. Using a brute force approach, the complexity of computing a  $m_G \times n_G \times k_G$  ( $=N_G$ ) grid for an image of size  $m_I \times n_I \times k_I$  ( $=N_I$ ) is  $O(N_G^2 N_I)$  for each iteration. Due to the analytical derivative of mutual information and the use of gradient descent method, the complexity is dramatically reduced to  $O(N_I) + O(N_G \log N_G)$  for each iteration, a near linear complexity. In Table 3.1, the components in the system are analyzed.

Table 3.1 The complexity of the components in the framework.

Component	Major Operations	Complexity
Div-Curl Calculation	Gradient of div, curls in x, y, z direction and FFT Poisson Solver	$O(N_G \log N_G)$
Runge-Kutta Approximation	Interpolation to estimate immediate displacement field	$O(N_G)$
Mutual Information Calculation	Update of histogram bins according to the image pixels value; Entropy computation (L: number of intensity levels)	$O(N_I) + O(L)$
Derivate $\frac{\partial MI}{\partial u}$ Calculation	Update of changes in histogram bins	$O(N_I)$
Derivate $\frac{\partial MI}{\partial f^i}, i = 1, 2, 3, 4$ Calculation	Convolution of $\frac{\partial MI}{\partial u}$ with pre-calculated filters	$O(N_G)$

Since all the components are of complexity related to the grid size, image size and intensity level, multi-resolution could greatly reduce the complexity with sacrifice in fine

accuracy which could be compensated in the next level. This triple multi-resolution scheme is used in 2D experiments as shown in Figure 3.2 and only grid refinement is implemented in 3D as shown in Figure 3.3.

With the introduction of multi-resolution, a simplified version of regridding takes place to allow smooth transition of similarity measure during resolution changes. The total displacement from the previous resolution is interpolated as the initial displacement in the current resolution. If grids are not reset between resolutions, spikes will be observed in MI graph as shown in later experimental results.

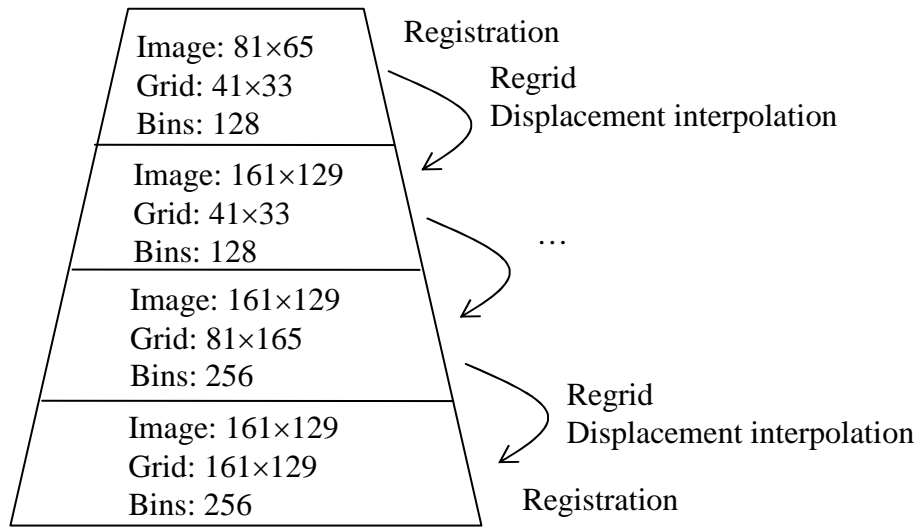


Figure 3.2 Triple-multi-resolution scheme for 2D image registration.

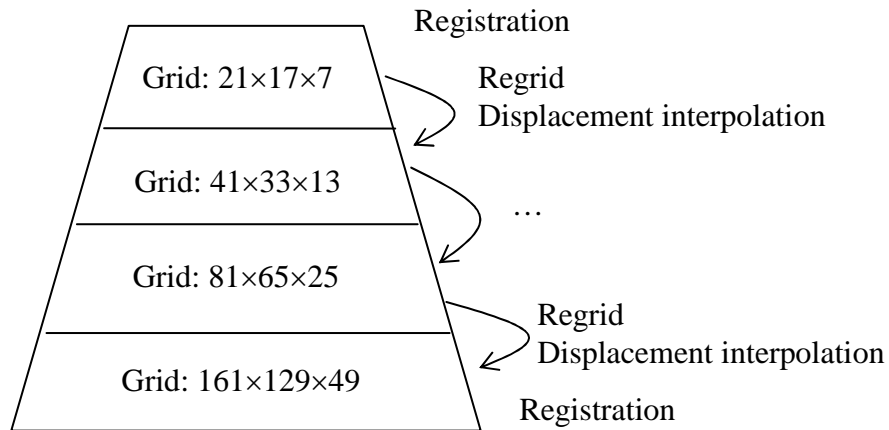


Figure 3.3 Multi-resolution scheme for 3D image registration.

### 3.5 Compressibility-aware Monitoring

Many human tissues are elastic and incompressible. To model this characteristic correctly in medical imaging, it is highly desirable to allow images of tissues taken within a short period of time to locally deform but at the same time its volume remain stable or within a limit. It is technically challenging to enforce the volume-stability in existing transformation models. The non-rigid registration approach in [34] imposed constraints on the Jacobian and its derivatives for noninvertible transformation. In [37], an incompressibility constraint regularized the transformation to preserve volume.

In the adaptive grid generation, the monitor function  $f$  is a mean to control the ratio of areas between the domains in the transformation. Positive  $f$  ensures no folding in the grids. Setting monitor function to one ensures incompressibility no matter how much the grid deforms. Notice that the required parameter is adjusted directly instead of being a constraint additional to the system. The monitor function  $f$  is related to the div-curl system that the divergence is set to  $f - 1$ . In this case, there is no need to adjust  $f$  but only the curl function  $g$  throughout the registration procedure.

Relaxation of monitor function to fall inside an upper and a lower bound would make suitable for other kind of applications. This powerful compressibility-aware algorithm naturally

maintains the constant volume without additional constraint or regularization term, which matches the requirement of soft tissues modeling.



## CHAPTER 4

### PROPOSED EFFICIENT IMPLEMENTATION USING CUDA

#### 4.1 Method

The complete non-rigid image registration algorithm is re-written from Matlab and C code to CUDA code. The CUDA implementation is developed side-by-side comparing on resultant values with Matlab execution to ensure accuracy. Originally, only the Poisson FFT solver was implemented in CUDA and executed from Matlab. Detailed profiling shows large amount of latency in memory copying which offsets the speedup except for a huge problem size. This proof of concept envisions the migration of the whole system to CUDA environment and the pursuit for speedup.

The CUDA implementation architecture is designed for GeForce 8800GTX and its specification is shown in Table 4.1. Different GPUs may vary in memory size and parallel capability. After input images are read from file, they are stored on the host memory and copied to the device memory. All the other computations are working on device memory to avoid memory copying time. Before any operation, global memory is allocated for all arrays and buffers. They will be re-used for the rest of the program. Reads and writes to global memory are kept minimal. Constant memory is used to store B-spline, B-spline gradient lookup table and convolution kernels. Shared memory is used within a block whenever possible because accessing shared memory is much faster than global memory. Coalesced access to global memory is preferred since it is could be ten times faster than un-coalesced reads and writes. Quicker device functions for mathematics operations are used when applicable. Usage of logical operation is minimal. 3D data are stored in 1D array. All other massive simple arithmetic operations not mentioned in the following section are executed in SIMT manner. After every calculation is completed, results are copied back to files for analysis.

Table 4.1 Nvidia GeForce 8800GTX Specification.

Number of Multiprocessors	16
Global Memory	768MB
Constant Memory	64KB
Shared Memory per Block	16KB
Warp Size (in threads)	32
Maximum Number of Threads per Block	512
Maximum Number of Threads in x, y, z dimension	512, 512, 64
Maximum Number of Grids in each dimension	65535
Maximum Number of Active Blocks per Multiprocessor	8
Maximum Number of Warps per Multiprocessor	24
Maximum Number of Active Threads per Multiprocessor	768

#### 4.1.1 Data Partition

Since thread scheduling is automatic, it is important to partition data evenly to keep threads occupied at the same time. There are two types of data size defined in the non-rigid registration system: image size and grid size. Image size equals the overlap of reference and template volume in Figure 4.1. Grid size is not to be confused with the CUDA grid of threads. It is the non-rigid grid as mentioned in the algorithm in Figure 4.1. All grids mentioned in the following refer to this grid unless specified. Grid size varies from  $3 \times 3 \times 3$  to full image size.

There are mainly four configurations of thread management named as “pixel 1D”, “grid 1D”, “pixel 3D” and “grid 3D” used throughout the implementation. When the x, y, z position of the data does not affect the operation, for example, adding up two 3D volumes, the image data of problem size  $m_i n_i k_i$  is partitioned into 1D block of 192 threads to be processed in CUDA as shown in Figure 4.2. Similar arrangement for grids is illustrated in Figure 4.3. When the x, y, z position of the data matters, for example, convolution, the data is partitioned into 3D block of  $4 \times 4 \times 6$  threads. Problem size is  $m_i, n_i, k_i$  in x, y, z-direction respectively. The choice of 192 threads per block is due to the maximum of 768 active threads per multiprocessor and 128 to 256 threads per block would give efficient execution. There are 6 threads in the z-direction for 3D block and 4 threads per x and y direction because there is no 3D grid, only 2D grid for

thread management in CUDA. One kind of implementation is to handle all z-direction data in one block as shown in Figure 4.4. Similar arrangement for grids is illustrated in Figure 4.5.

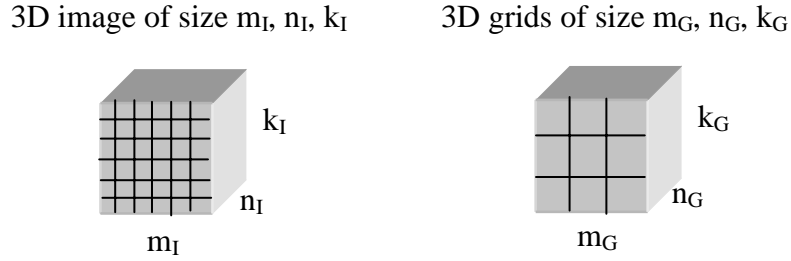


Figure 4.1 Image size and Grid size of 3D Image

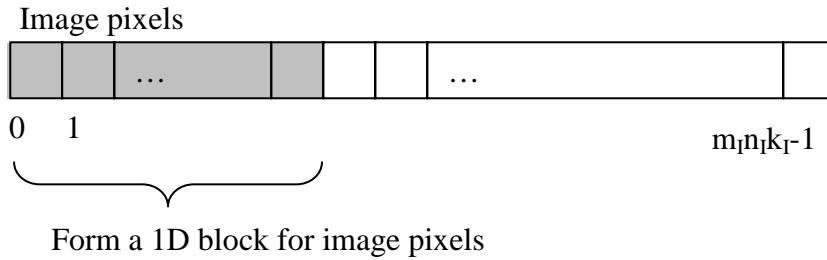


Figure 4.2 3D image data lined up as 1D array and form numerous 1D blocks

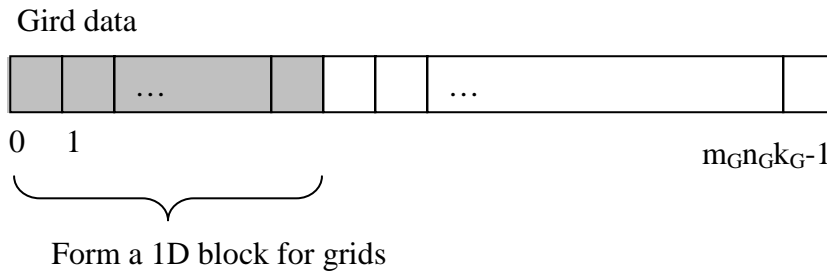


Figure 4.3 3D grid data lined up as 1D array and form numerous 1D blocks

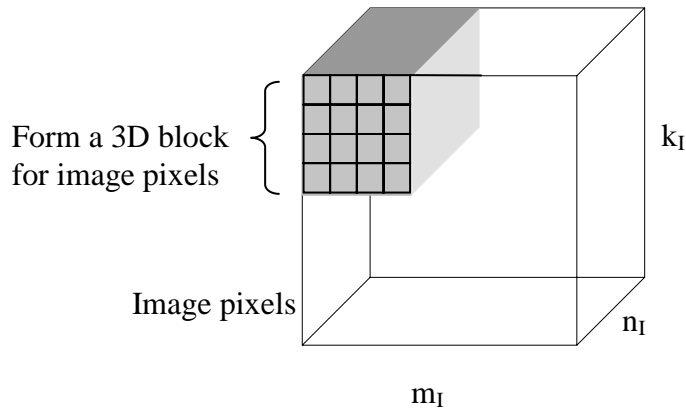


Figure 4.4 Image data lined up as 3D arrays and form numerous 3D blocks

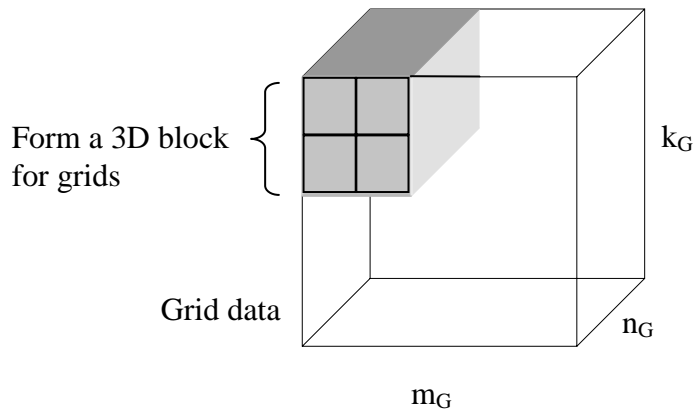


Figure 4.5 3D grid data lined up as 3D arrays and form numerous 3D blocks

#### 4.1.2 Parallel Program

All routines in the system are executed in data parallel manner; data are synchronized after each routine. Original algorithm in Figure 3.1 is modified for GPU implementation as shown in Algorithm 4.1. Most of the operations are highly parallel elements. The input parameters divergence  $f^1$  and curl in x-direction  $f^2$ , curl in y-direction  $f^3$  and curl in z-direction  $f^4$  are initialized to 1, 0, 0 and 0 respectively. All gradients are initialized to zeros. Previous mutual information value  $MI_{old}$  is set to zero in the beginning.

Div-curl calculation is a grid-level procedure which consists of gradient computation and solving the Poisson equation. It takes the divergence and curls as input and computes the

intermediate vector field as output. The gradient of  $f$  w.r.t.  $x, y, z$  is simply the difference of the next data value and the previous value in  $x$ -direction,  $y$ -direction or  $z$ -direction. In other words, calculating the gradient in  $x$ -direction for a data value at  $(i, j, k)$  involved reading from  $(i-1, j, k)$  and  $(i+1, j, k)$ . There are always two reads and one write operation. Using the current simple approach, the same input data is always read twice from the global memory. In future, shared memory should be used to store the data accessed by the block to reduce the number of reads from global memory similar to the efficient separable convolution for 2D convolution using GPU presented in [33]. "Grid 3D" thread configuration is used for gradient calculation. The gradients of divergence and curl (in  $x, y, z$  direction) components are summed up as the input of the Poisson equation. A 3D FFT Poisson solver is implemented by simply applying six 1D FFTs and a multiplication. In CUDA, CUFFT library is provided for FFT calculation. The challenge in applying 1D FFT is to re-order 3D data in consecutive  $x$  direction for  $x$ -direction FFT, consecutive  $y$  direction for  $y$ -direction FFT and same for  $z$  direction FFT. After applying FFT in  $x$ -direction,  $y$ -direction,  $z$ -direction, the immediate result is multiplied by the Poisson wave numbers. Another round of FFTs in  $x, y, z$  direction gives the solution [14].

Runge-Kutta method is also a grid-level procedure which takes the intermediate vector field from div-curl calculation as input and approximates the intermediate displacement field. Main operations in Runge-Kutta are interpolation of intermediate vector field and arithmetic update to the displacement field. In tri-linear interpolation, an output value is interpolated by the nearest eight neighboring values of the evaluated 3D position. Eight unpredictable reads and one write are involved. Multiple reads of the same value from the global memory slow down the interpolation operation. There is no straight forward solution for this since the input data is too big to fit into the 16KB shared memory and the threads within a block may not read the same set of values. "Grid 3D" configuration is used for Runge-Kutta.

Mutual information joint histogram estimation is a pixel-level procedure. For each pixel in the template image, it is compared against the neighborhood of the transformed pixel. If 3<sup>rd</sup> B-

spline and GPVE is in use, the neighborhood could be of size 27 to 64. It translates to 27 to 64 random reads globally from the reference image and the constant B-spline table, 27 to 64 floating point multiplications and 27 to 64 floating point writes globally to a random histogram location. The 256×256 floating point joint histogram is too big to fit into the 16KB shared memory. The input image in experiment 161×129×49 is too big to fit in shared memory too. This is a bad scenario for SIMT operations since data are not written to predicted locations.

Algorithm 4.1. 3D Image Registration for images T and R	
1	Initialize $F_{\text{Parallel}} (f^1, f^2, f^3, f^4)$
2	Initialize Gradient $_{\text{Parallel}} (G_{f1}, G_{f2}, G_{f3}, G_{f4})$
3	$MI_{\text{old}} = 0$
4	While tolerance level not reached
5	Set $f^i$ to $f^i - G_{f1} * \text{tolerance}$ , for $i = 1$ to 4
6	Synchronize Data ()
7	$(V^x, V^y, V^z) = \text{Div-Curl}_{\text{Parallel}} (f^1, f^2, f^3, f^4)$
8	Synchronize Data ()
9	$(u^x, u^y, u^z) = \text{Runge-Kutta}_{\text{Parallel}} (f^1, V^x, V^y, V^z)$
10	Synchronize Data ()
11	$MI = \text{Calculate MI}_{\text{Parallel}} (T, R, u^x, u^y, u^z)$
12	Synchronize Data ()
13	If $MI > MI_{\text{old}}$
14	Set $f^i$ to $f^i$ , for $i = 1$ to 4
15	Synchronize Data ()
16	$(G_x, G_y, G_z) = \text{Calculate MI Gradient w.r.t. } x,y,z_{\text{Parallel}} (T, R, u^x, u^y, u^z)$
17	Synchronize Data ()
18	$(G_{f1}, G_{f2}, G_{f3}, G_{f4}) = \text{Calculate MI Gradient w.r.t. } f^i_{\text{Parallel}} (G_x, G_y, G_z)$
19	Synchronize Data ()
20	$(G_{f1}, G_{f2}, G_{f3}, G_{f4}) = \text{Normalize}_{\text{Parallel}} (G_{f1}, G_{f2}, G_{f3}, G_{f4})$
21	Synchronize Data ()
22	Else
23	Decrease tolerance level
24	End if
25	End while
26	Output $u^x, u^y, u^z$

There is a highly concurrent 256-bins-histogram presented in [31]. It is a histogram calculation instead of joint-histogram calculation and it handles integer histogram values only. In

[38], the authors proposed a GPU based solution for integral joint histogram. Each thread maintains a portion of joint histogram instead of a portion of input data. Histogram updates by different threads do not coincide. Performance decreases as more histogram bins are used. It is impractical in our situation due to floating point and the numerous writes. Scaling down joint histogram to  $64 \times 64$  bins to fit into the  $16KB$  shared memory would probably increase the efficiency but dramatically lose accuracy.

Under the requirements of GPVE, 3<sup>rd</sup> B-spline and  $256 \times 256$  joint histogram bins, a  $N$ -threads and  $N$ -sub-joint-histograms approach is proposed. These sub-joint-histograms are stored in the global memory of the GPU because it is impossible to fit even one of them into the  $16KB$  shared memory. The input data is partitioned for  $N$  possible concurrent threads to execute. Each thread processes its share of image data and writes to its sub-joint-histogram. After computation of all sub-joint-histograms, they are summed up in parallel as one joint histogram. The choice of  $N$  equaling 256 is experimental. More sub-joint-histogram may speed up but it requires even more global memory for storage. It may not speed up due to more time required for summation of sub-joint-histogram. Other procedures in mutual information are entropy calculation and summation of 2D histogram to 1D probability density function. The data in histogram bins are lined up for 1D block of 192 threads to execute.

Calculation of gradient of mutual information w.r.t.  $x$ ,  $y$ ,  $z$  direction is a similar procedure as the joint histogram estimation except that the result is always written to a specific location. The complexity only remains in the 27 to 64 unpredicted and uncommon reads of reference image, B-spline tables and the multiplications. Gradient in  $x$ ,  $y$  and  $z$  directions are written according to the input location. "Pixel 3D" configuration is used for this operation.

Calculation of gradient of mutual information w.r.t. input parameters  $f^1$ ,  $f^2$ ,  $f^3$  and  $f^4$  consists of convolution of a redistribution filter as shown in Figure 4.6, Poisson solver and gradient computation. The redistribution filter concentrates pixel-level gradients to grid-level gradients. In the original Matlab implementation, a redistribution filter is applied to the whole

input pixel-level gradients and grid-level gradients are then extracted by skipping on the results. To further reduce the complexity, separable 1D filter as shown in Figure 4.7 is applied in x, y, z direction with skipping pixels. When gradient in x-direction is calculated, filtering is only done at skipped x-direction. When gradient in y-direction is calculated, filtering is only done at skipped x and y-direction. When gradient in z-direction is calculated, filtering is done at skipped x, y and z-direction. There is no repeated multiple reads to the same input image. There is always fixed number of writes to the result but the number of reads depend on the grid size. Poisson solver replaces the inverse filtering method in the original Matlab implementation. "Grid 3D" configuration is used here.

Normalization of the gradients consists of finding the maximum value and division of the maximum value.  $N$ -ary search approach is employed. Each thread reads  $N$  data at a time, compares and stores the bigger values. The problem size decreases by  $N$  times each time. It will take  $X$  reduction steps to finish  $N^X$  input data. Once the maximum value is obtained, all the input values are averaged out in parallel. Other reductions involved problem follows this approach. "Grid 1D" configuration is used in this step.

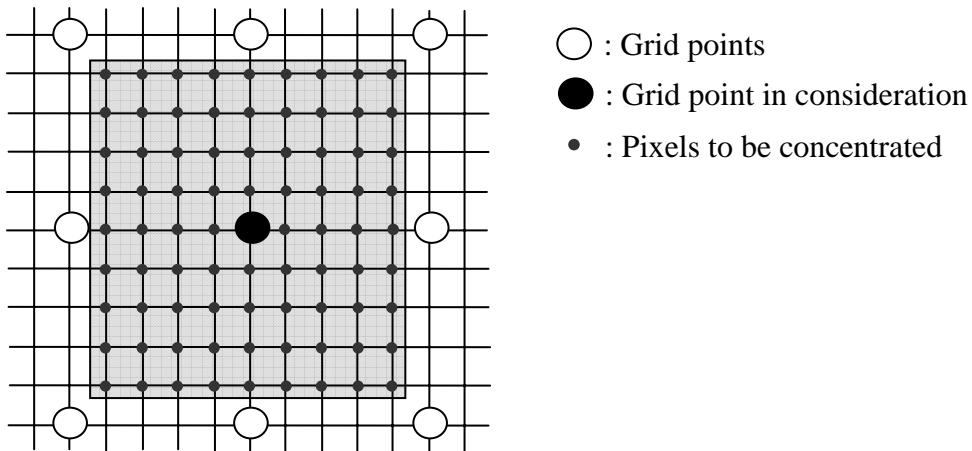


Figure 4.6 2D redistribution filter.



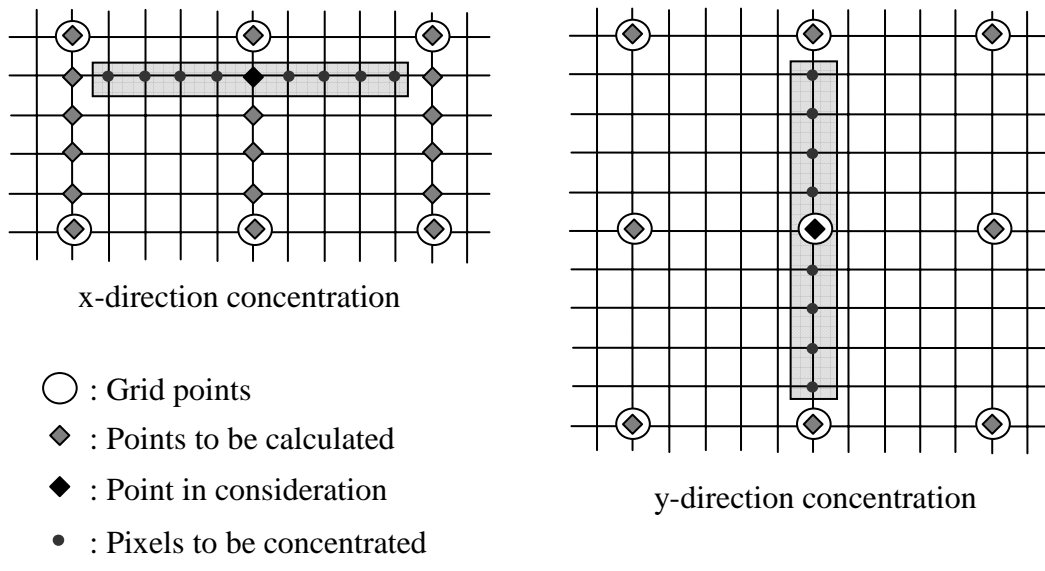


Figure 4.7 Two separable 1D redistribution filters.

## CHAPTER 5

### EXPERIMENTAL RESULTS VALIDATION

#### 5.1 Setup

2D experiments are conducted on a Pentium 4 3.2GHz computer with 3GB Ram. The whole non-rigid registration system is implemented in Matlab with various routines for interpolation, MI calculation and derivatives of MI calculation re-written in C. 3D experiments are conducted on the same computer with GPU acceleration and written in CUDA.

Details of the dataset can be found in Chapter 6. In the following, input image data are originated from a sub-set of the dataset.

##### *5.1.1 Contrast Enhanced Simulated 2D Image Data*

Ground truth is required for error measurement but it is impossible to have ground truth available in clinical application. Simulation of the deformation yields the ground truth. In the simulation, the input image is treated as the reference image  $R$ . The template image  $T$  is simulated from distorting  $R$  with 36 control points evenly distributed in a B-spline thin plate model [7]. The thin plate model is a fair choice since it is not the same as the registration model. The resultant distortion is recorded as ground truth of the displacement field. Optionally, to test on efficiency on contrast enhanced images, some random regions in the template image  $T$  are then arbitrarily adjusted to higher intensity level. In this way, the estimated displacement field could be compared with the ground truth. A central slice of size  $161 \times 129$ , which contains the right breast, is extracted from the 3D breast MRI at time step  $0$  is used as input. The initial images are shown in Figure 5.1.

##### *5.1.2 Cropped 2D Image Data*

A manually cropped central slice of size  $161 \times 129$ , which contains the right breast, is extracted out of 3D breast MRI at time step  $0$  and time step  $1$ . The initial images are shown in

Figure 5.2 with the right image as the difference between the rigid transformed reference image  $R$  and template image  $T$ .

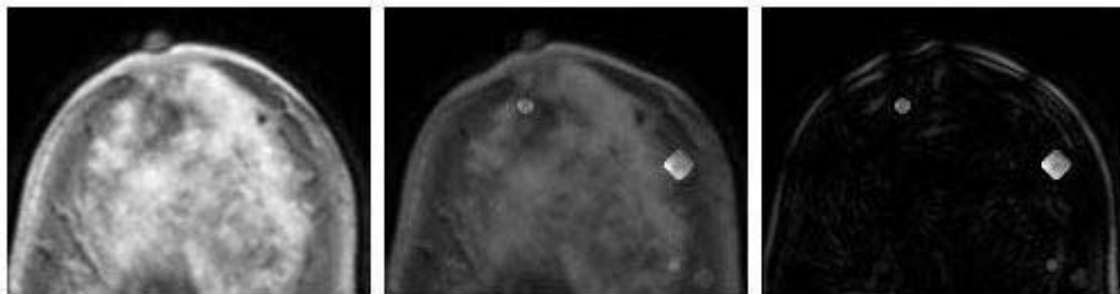


Figure 5.1 From left to right: The reference image  $R$ , the template image  $T$  with simulated distortion and contrast-enhanced cancerous regions and the subtracted image between  $T$  and  $R$ .

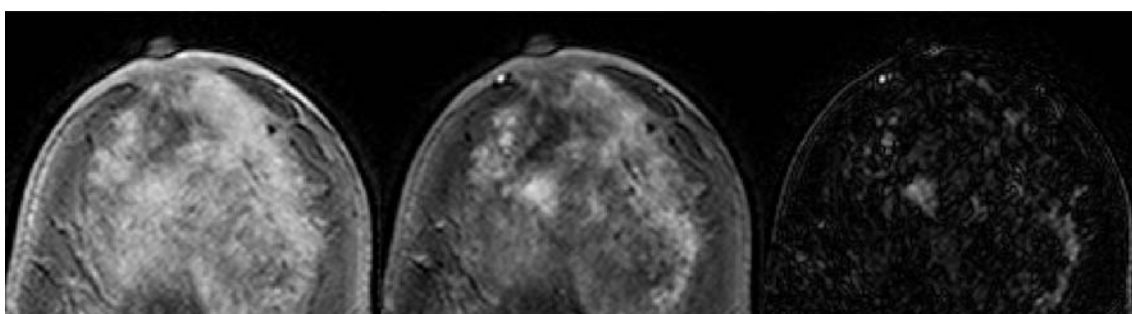


Figure 5.2 From left to right: The reference image  $R$  (time step  $0$ ), the template image  $T$  (time step  $1$ ) and the subtracted image between  $T$  and rigid-transformed  $R$ .

### 5.1.3 Identical 2D Image Data

A central slice of size  $129 \times 129$ , which contains the left breast, is extracted out of 3D breast MRI at time step  $0$ . Both template image  $T$  and reference image  $R$  are set to the same image.

### 5.1.4 Simulated 3D Image Data

A volume of size  $161 \times 129 \times 49$ , which contains the right breast, is extracted out of 3D breast MRI at time step  $0$  and deformed by a B-spline thin plate model specified by 36 control points with random displacements  $d$ . Ground truth is collected in the transformation.

### 5.1.5 Cropped 3D Image Data

A volume of size  $129 \times 129 \times 33$ , which contains the left breast, is extracted out of 3D breast MRI at time step 0 and time step 1 are registered. A smaller volume is used due to the low speed of running on Matlab.

### 5.2 Error Measurement

To validate the accuracy of the algorithm, one of the best methods is to measure the correctness of the transformation. The displacement field of the each pixel represents the local deformation in non-rigid registration. If ground truth of the displacement field is known, the error is calculated from the difference between the ground truth and reconstructed displacement field. The area of interest contains only non-background region. Contrast-enhanced regions should be excluded if present.

Absolute error and warping index are the most common error measuring methods. Let  $u$  be the ground truth displacement field and  $u'$  be the estimated displacement field, absolute error is computed by  $e = |u - u'|$ . Warping index  $w = \sqrt{\|u - u'\|^2}$  defined in [41] gives another estimate of the error. Often, the average and maximum values denote the overall correctness and the worst estimation respectively.

The 2D Matlab implementation is first applied on the "Contrast Enhanced Simulated 2D Image Data". No regriding or monitoring is in use. The change in MI and average error are recorded in the process. Initially, the input parameters divergence is set to 1 and curls are set to 0, indicating static velocity field. Our algorithm is able to converge at different resolutions over time. As shown in Figure 5.3 and Figure 5.4, MI improves logarithmically with average error minimizes gradually. The program proceeds to the next level when the stopping criteria, step size getting below a tolerance threshold. When the algorithm jumps from one resolution level to another level, spikes are observed in the MI curve and the average error curve. When the algorithm iterates more, MI converges to a maximum position whereas average error converges to a minimum. These spikes are explained by the interpolation of  $f$  and  $g$  using previous level

results. Without the input  $f$  and  $g$  from the previous level, the high MI value at the starting point of the finest level would not be possible. The program gradually recovers to the previous maxima or minima after more iterations or resolution levels. The topology preserving regriding in section 2.3 is implemented and results are shown next. Visually, the registration gets better as shown in Figure 5.5. Grid and image resolution refine and the contrast-enhanced features becomes clearer over time. After the registration, contrast-enhanced features are clearly distinguished from the simulated movement as shown in Figure 5.5. No shrinking of these features is observed.

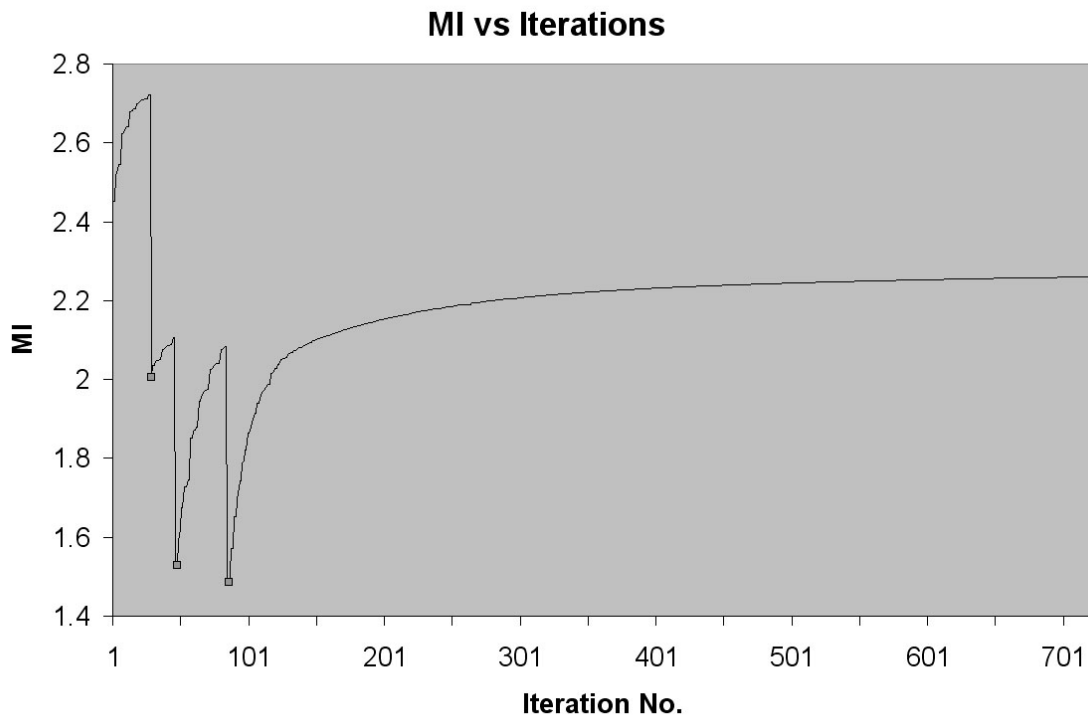


Figure 5.3 Changes in mutual information versus iterations.

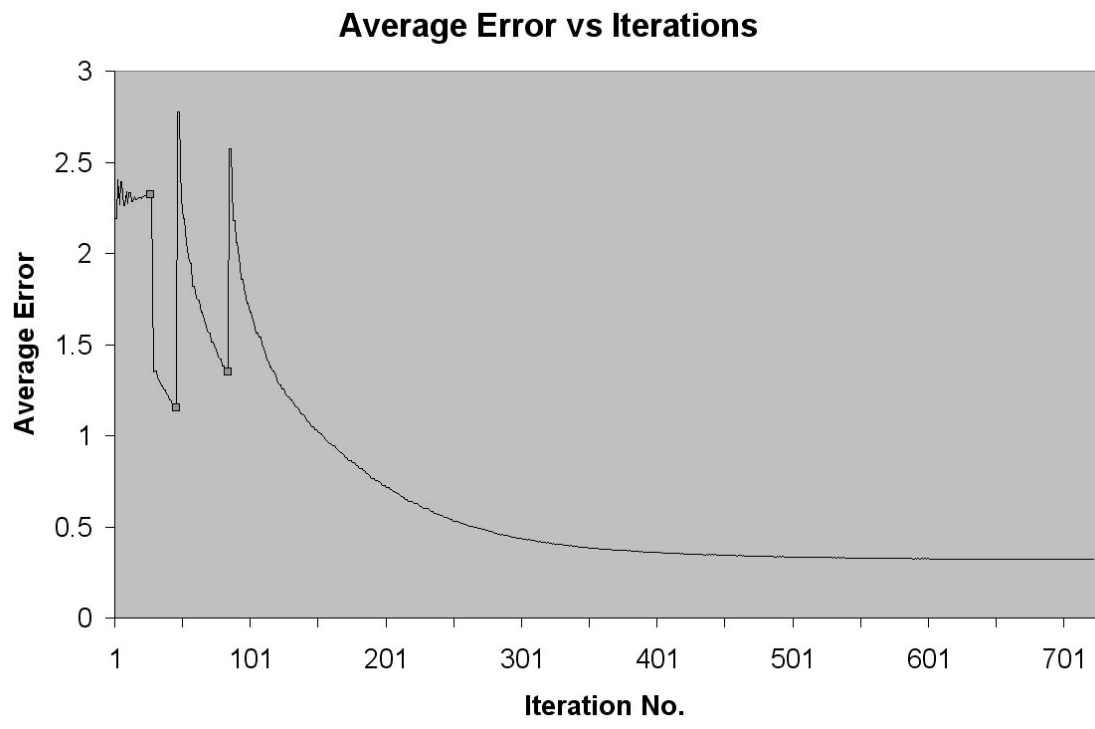


Figure 5.4 Average error versus iterations.

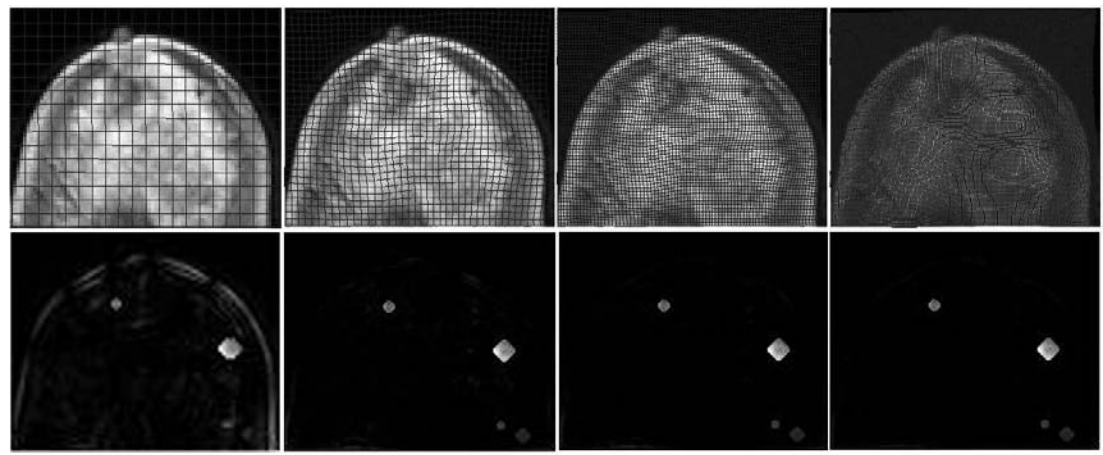


Figure 5.5 Top: Deformation grid at different resolution level; Bottom: difference of the reconstructed reference image  $R$  and the template image  $T$ . Grid resolution improves from left to right.

Table 5.1 A configuration of different resolution levels and corresponding resultant number of iterations, mutual information and average error.

Image Size	Grid Size	No. of Bins (in histogram)	No. of iterations	MI	Average Error
81×65	41×33	128	28	2.7206	2.3199
161×129	41×33	128	21	2.1058	1.1544
161×129	81×65	256	41	2.0501	1.2550
161×129	161×129	256	603	2.2603	0.3211

Table 5.1 shows the results of a possible configuration under a constant tolerance level  $0.001$ . The left three columns are the settings in different resolution levels and the right three are the results accordingly. Multi-resolution is desirable because the number of iterations and the time it takes to reach a certain tolerance level is much less than a finer level.

In the zoomed-in Figure 5.6, the smooth reconstructed grid on the right is almost the same as the simulated grid. No grid folding is observed. Our algorithm gives almost perfect deformation estimation.

The resultant divergence and curl are shown in the contrasted Figure 5.7 where white implies higher value, gray implies almost zero and black implies negative value. The gradient of MI with respect to  $f$  and  $g$  are shown respectively in Figure 5.7. The gradient of MI with respect to displacement field in  $x$  and  $y$  direction are shown respectively in Figure 5.7. Larger divergence and curl values are observed in more distorted areas. Notice that MI is insensitive to the simulated cancerous regions and sensitive to the edges and motion areas.

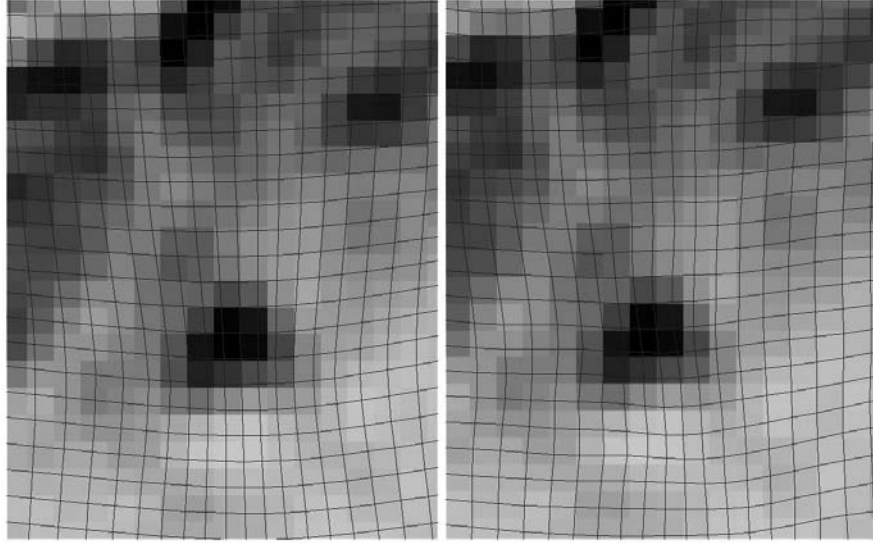


Figure 5.6 From left to right: The ground truth of the deformation grid and the reconstructed deformation grid.

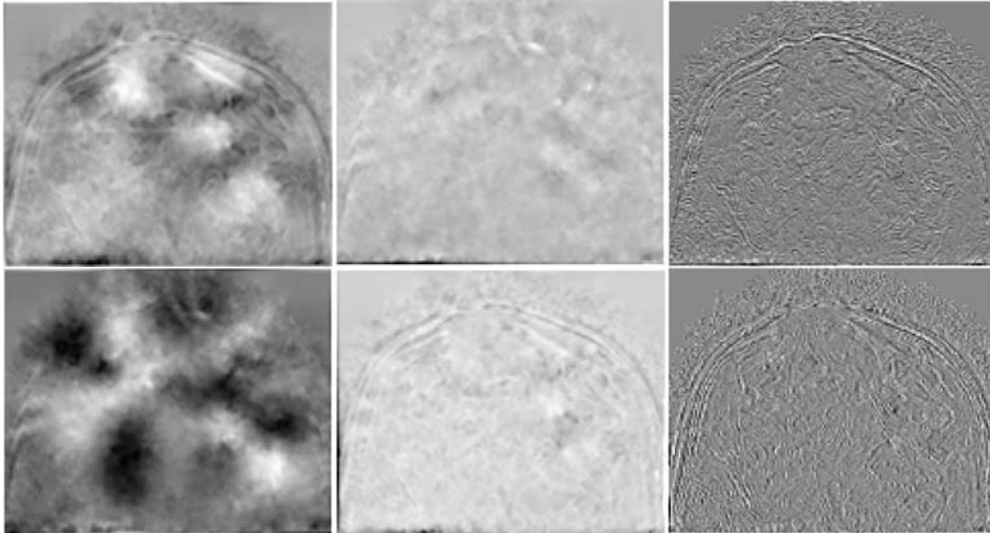


Figure 5.7 From left to right: The resultant divergence (top) and curl (bottom); the gradient of MI w.r.t. divergence (top) and curl (bottom); and the gradient of MI w.r.t. displacement in  $x$  (top) and  $y$  (bottom) directions.

The 3D CUDA implementation is then applied on the “Simulated 3D Image Data”. Regriding and compressibility-aware monitoring are in use. During the registration process, the difference of ground truth and transformation displacement field is captured in the average



warping index. Registration process is given ten chances to improve MI and a maximum of 200 iterations in each resolution starting from grid distance of 8, 4, 2 and finally 1. Monitor function is set to at least 0.9 to disallow shrinking of volumes and allow minor error in real data. Figure 5.8 shows the changes of average warping index in different iterations of the registration for a number of maximum allowed random displacements  $d$ . Figure 5.9 shows the corresponding MI values achieved. Markers in Figure 5.9 represent the grid resolution refinement using regriding. Average warping index is decreasing over time and MI value is increasing for different  $d$ . This is desirable since the maximization of the similarity measure leads to displacements with more accuracy. This validates the correctness of the CUDA implementation. Regriding helps average warping index curves reducing and MI curves increasing from one resolution to another. Overtime, average warping index of different amount of initial errors were able to reach to as low as 0.5 or lower and MI curves were able to converge to a common value of 2.3. Only single-multi-resolution scheme of different grid distances 8, 4, 2 and 1 on all dimensions is experimented here due to incomplete 3D implementation on the triple-multi-resolution strategy.

These results suggest that MI maximization corresponds to correctness of registration. Since compressibility-aware monitoring limits the ability to deform, average warping index and MI slow down converging after some iterations. Regriding significantly stimulates the MI maximization process with straightly decreasing error across the next resolution level.

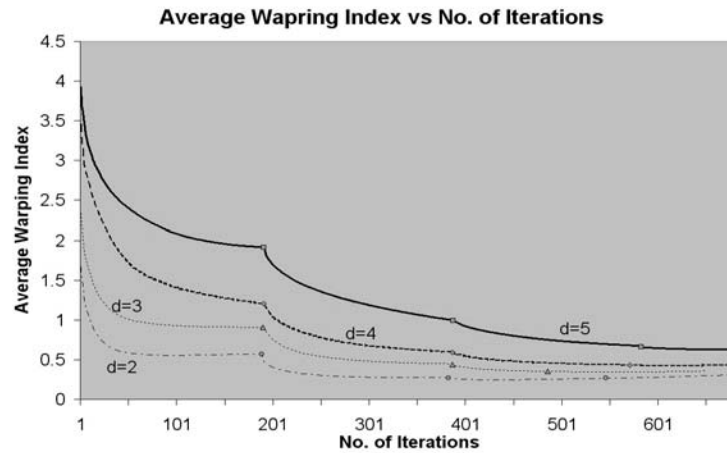


Figure 5.8 Average warping index values at different number of iterations for maximum allowed random displacements  $d=2, 3, 4, 5$ . Markers indicate grid resolution refinement.

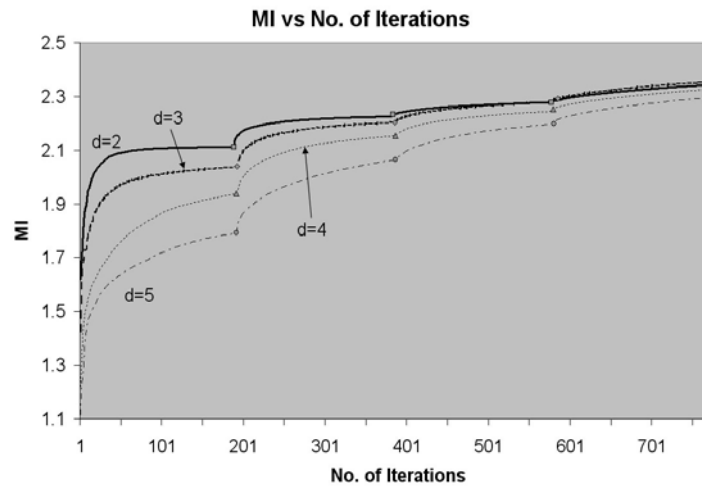


Figure 5.9 MI values at different number of iterations for maximum allowed random displacements  $d=2, 3, 4, 5$ . Markers indicate grid resolution refinement.

### 5.3 Comparisons on Different Joint Histogram Estimation for Mutual Information

The 2D Matlab implementation is applied on the “Identical 2D Image Data” to explore the correctness of different histogram estimations.

GPVE, PVI and Parzen windows joint histogram estimation methods with gradient optimization are implemented and compared for 2D images. In Figure 5.10, the MI and gradient

of MI are compared. Two identical images at time step  $0$  are used as input of the joint histogram estimation. One pixel moves in the horizontal direction  $x$  from  $-1.5$  to  $1.5$ . The corresponding fluctuations in MI and gradient of MI are then recorded. All MI graphs are center at zero meaning zero displacement in that point gives maximum MI. The analytical gradient is calculated by the previous derivations in the methodology section. The finite difference gradient is computed by subtract the previous MI value from the current one. It can be seen that the shape of the analytical gradient is very close to the finite difference one. Here, modified PVI which uses 3<sup>rd</sup> order B-spline for initial 20 iterations to overcome its artifacts. In PVI, grid points tend to move towards integral positions. If the original version is used, the graph of PVI will not move at all since its value at integral position is always greater than at nearby non-integral positions.

On an average of 50 runs, the table 5.2 shows the time required for each method to perform joint histogram estimation and calculation of gradient of mutual information for  $129 \times 129$  input images. PVI is the fastest whereas Parzen windows method is the slowest in general.

Figure 5.11 shows the difference between reconstructed and original image. Again, the difference image obtained by GPVE gives the best contrast of the desired regions.

The MI of GPVE, PVI and Parzen windows are on different scale. The percentage increase is calculated by this formula:

$$\frac{\text{MI} - \text{Initial MI}}{\text{Maximum Possible MI} - \text{Initial MI}} \times 100\% \quad (4.13)$$

where initial MI is the MI before non-rigid registration begins and maximum possible MI is the MI when both images are identical. The maximum possible MI for GPVE, PVI and Parzen windows are 3.4668, 5.8971 and 4.4995 respectively when the template image  $T$  is used as both inputs for MI. The graph below shows GPVE gives the largest amount of percentage increase over the iterations. PVI stops converging after 250 iterations.

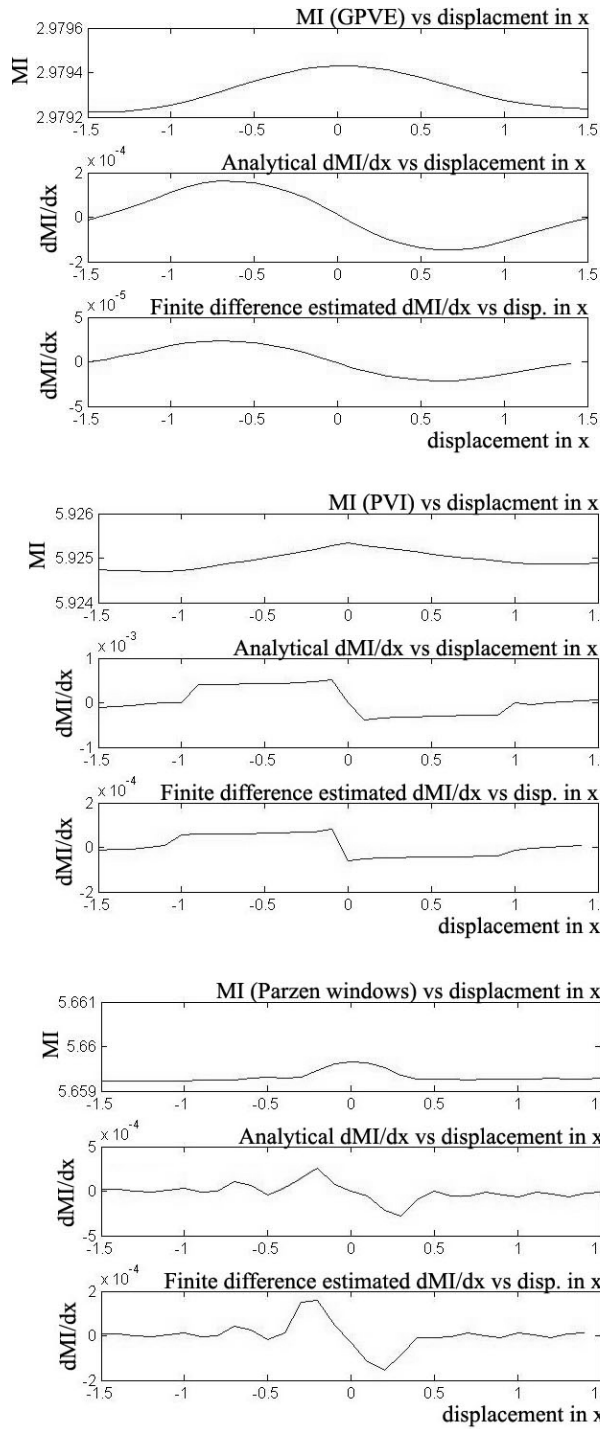


Figure 5.10 Comparisons of various methods GPVE (top), PVI (middle) and Parzen Windows (bottom) in MI and gradient of MI with respect to displacement in  $x$ . The top graph is MI, the middle graph is the analytical gradient and the bottom graph is the gradient estimated by finite difference.

Table 5.2 Time comparison of GPVE, PVI and Parzen windows on joint histogram estimation and gradient calculation.

	Joint Histogram Estimation	Gradient of MI
GPVE	0.02416 s	0.09368 s
PVI	0.01593 s	0.07322 s
Parzen Windows	0.03418 s	0.1029 s

Based on the joint histogram estimation method used in MI, there are several variations exist in the literatures. GPVE, PVI and Parzen windows are compared here. When a new position is found, the PVI method distributes linear weights to the pixel neighborhood without calculating a new intensity value. The GPVE method uses B-spline weights which gives smoother updates. In Parzen windows, intensity of new position is bi-linearly-interpolated and the new intensity value is distributed to the histogram neighborhood using B-spline weights. (There are also other variations of Parzen windows which are excluded in this paper. The choice is made due to the simplicity of bi-linear interpolation and smooth histogram update.) From the experiment, GPVE method performs the best in term of convergence. The percentage increase in MI is the largest among the methods in the experiments. PVI method is the fastest in term of computation time but it fails to converge to a higher value over time. It could be explained by the tendency of pixels to move to integral positions to achieve better MI. Parzen windows method converges over time but it is about 50% slower than GPVE in the joint histogram estimation. The converging power of Parzen windows method is much less than GPVE method as shown in figure 5.12. It could be concluded that the smoothness of the joint histogram estimation function result in different converging behaviors.

Gradient of various MI are derived and implemented for use in the gradient descent optimization method. From Figure 5.7, the analytical gradients are very close to the finite difference one meaning the correctness of the analytical results. The curves of GPVE are smooth due to the one-step distribution of 3<sup>rd</sup> order B-spline weights. PVI method uses bi-linear interpolation, thus updates do not spread smoothly to neighbors. The Parzen windows method

adapted in this paper uses bi-linear interpolation to calculate the intensity value and B-spline to distribute updates in the histogram. It makes the curve partially smooth and partially rough.

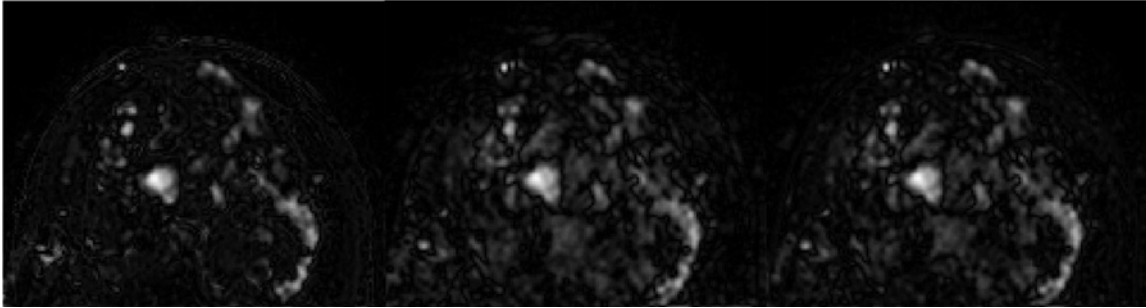


Figure 5.11 Difference image of template image and reconstructed reference image by GPVE (left), PVI (middle) and Parzen windows (right).

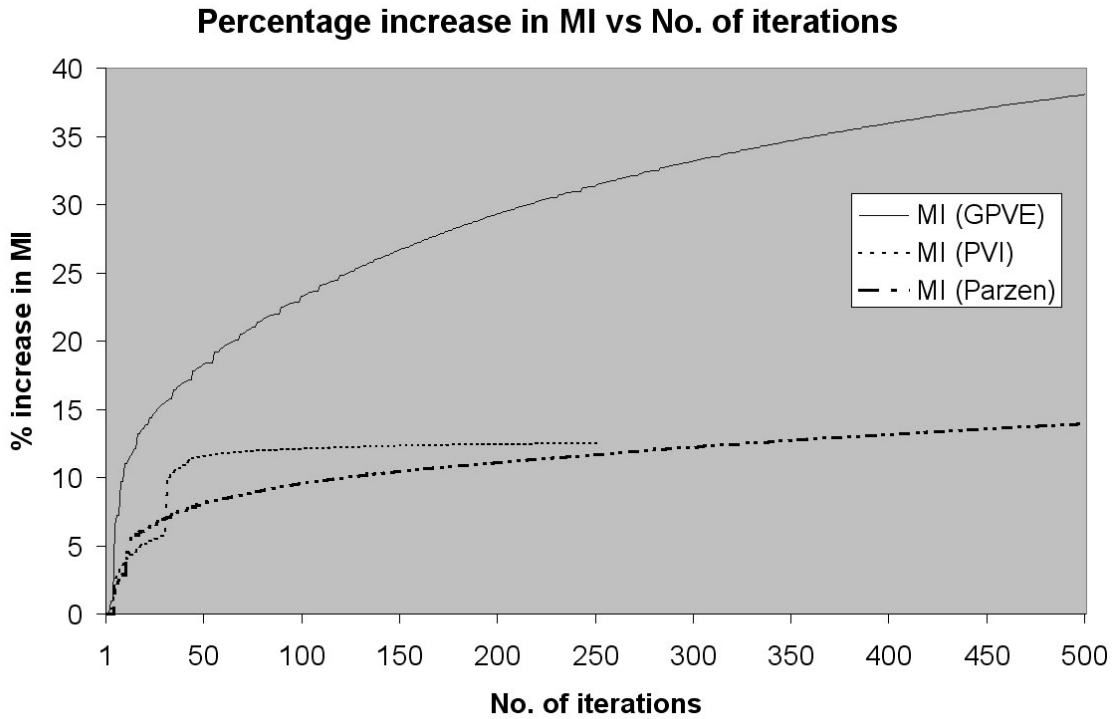


Figure 5.12 Percentage increase in MI versus number of iterations for GPVE, PVI and Parzen windows.

#### 5.4 Comparisons on Different Similarity Measures

MI with GPVE and Sum of Square Difference (SSD) are compared on performance. In MI based registration, GPVE joint histogram estimation, entropy calculation and gradient of MI w.r.t.  $x$ ,  $y$ ,  $z$  and  $f$  are the major operations. In SSD based registration, reference image  $R$  is re-sampled according to the local displacements found. Subtraction with pixels in  $T$  followed by squaring and summing up yields the SSD. Gradient w.r.t.  $x$ ,  $y$ ,  $z$  and  $f$  are then computed accordingly. GPU implementation is employed. Multi-resolution of grid distance 8, 4, 2, 1 combined with regriding after each resolution and compressibility-aware monitoring is applied to “Simulated 3D Image Data“. Both SSD and MI based are given ten chances of no improvement and maximum of 200 iterations per resolution setting. The table 5.3 shows required time of an average of 50 runs of the routines on images of size  $161 \times 129 \times 49$ . The grid distance is denoted by  $h$ . Smaller  $h$  implies denser grids. SSD is undoubtedly faster than MI due to its simplicity in calculation and bottom neck in MI calculation using data-parallel approach. Figure 5.13 shows the convergence of both SSD and MI over time. The simplified version of topology preserving regriding prevents similarity measures from re-bouncing after changing resolutions marked by a special icon in the graphs. Figure 5.14 plots the comparison of average warping index recorded in both SSD and MI based registration. The trends of graph suggest MI graph is able to get to a smaller error value than SSD over number of iterations which suggests the reliability of MI on real world data. Although both curves are against the same iterations axis, the time consumed in SSD is much shorter than MI. The overall time taken for MI to finish this registration is about 30 minutes and the time for SSD is around 1 minute. When tolerance is set to a lower value, less iteration occur and less time is consumed. In another experiment, tolerance is set to five chances of no improvement and maximum of 50 iterations per resolution, MI registration takes 3 minutes and SSD registration takes 15 seconds to finish. The tolerance level could be adjusted to acceptable time limitation for practical application.

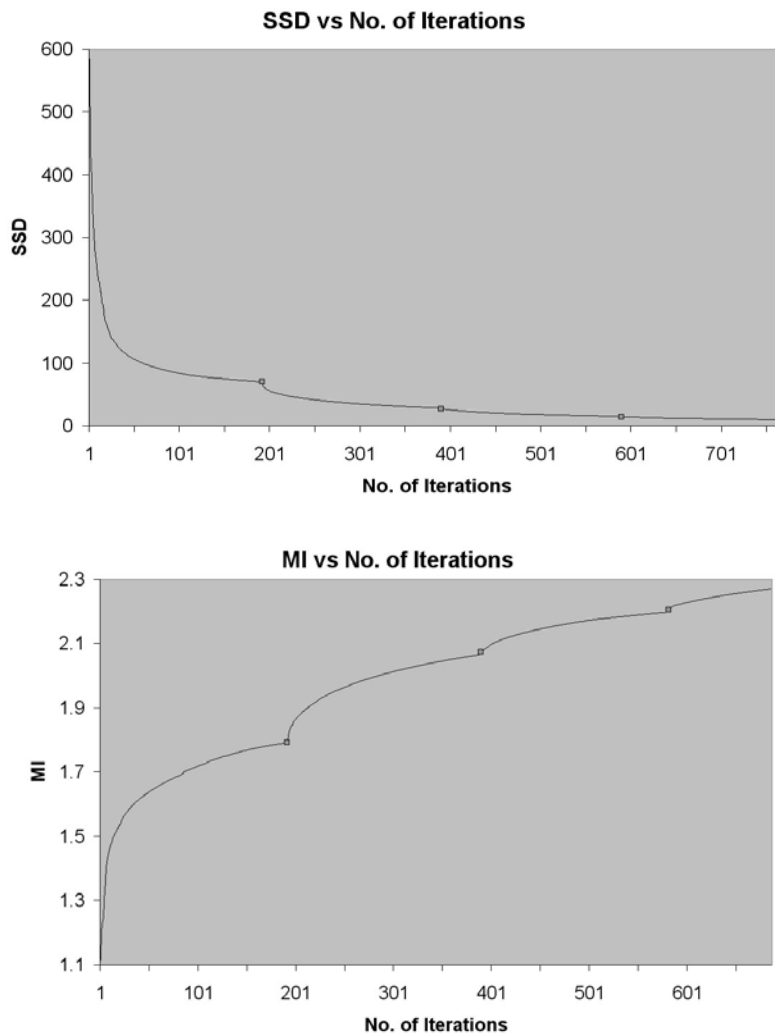


Figure 5.13 SSD and MI vs. number of iterations. Markers show the regridding points.

MI and SSD are both similarity measures based on the intensity of the image. MI measures the similarity of the statistical data of the image. SSD measures directly the difference in the pixel values. To maximize MI, both input images should have identical statistical data which does not require the input images to be exact in intensity values, which is the case for SSD minimization. The advantage of using SSD is of course the speed due to the simplicity of the operation. Thanks to the incompressible nature of the div-curl model, when SSD drives the pixels to position that minimizes SSD, the monitor function limits the position of the grids to



avoid grid folding. It could be deduced from Figure 5.14, average warping index of SSD registration will converge to a higher error value than MI due to the possibility of small amount of noise presence in real world data. MI can go much lower in error because of its insensitivity to lighting and contrast difference. Hence, MI is a more suitable similarity measure for this kind of breast MRI.

Table 5.3 Time comparison of MI (GPVE) and SSD operations.

	Basic Computation	Gradient Computation			
		$h=8$	$h=4$	$h=2$	$h=1$
MI (GPVE) (in ms)	1180.24	1623.66	1900.05	2048.25	2311.05
SSD (in ms)	4.88	12.45	16.34	50.37	210.16

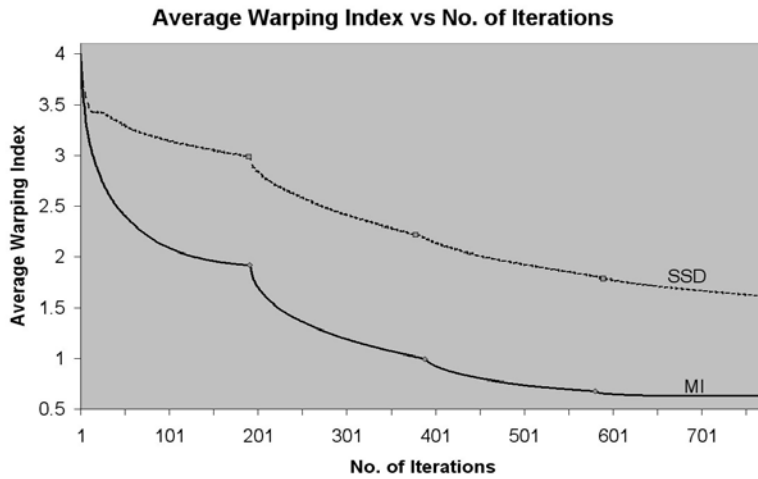


Figure 5.14 Comparisons of average Warping Index of SSD and MI vs. number of iterations. Markers show the regriding points.

### 5.5 Comparisons on GPU and CPU Implementation

In this section, the performance of CPU implementation and GPU implementation are examined. The CPU configuration consists of a Pentium 4 3.2GHz computer with 3GB Ram. Code is mainly written in Matlab. Mutual information, gradient of mutual information, interpolation are written in C. The GPU configuration consists of an Nvidia GeForce 8800 GTX processor and the same CPU as coprocessor. The GPU version is completely rewritten in CUDA which resembles all the main procedures in the Matlab version. In the following, the

maximum number of units handled by GPU per second and the relative GPU to CPU speed up are measured and discussed. Specification of GPU is shown in Table 4.1. The “Cropped 3D Image Data” is taken as the input. Both implementations are given the problem of same size and complexity.

To measure the processing power of GPU, a metrics similar to GFLOPS/s is computed. When problem size increases, GPU performance should be pushed up to peak when most threads are non-idle. Figure 5.15 show the maximum number of mega-grids performed by the GPU per second (M Grid/s) for specific grid-level operation when total number of grids increases. Under the test image of size  $129 \times 129 \times 33$ , parallel version of Runge-Kutta achieves around 650 M Grid/s and gradient normalization achieves around 730 M Grid/s. Their performance is much better due to their high data independency. Div-Curl calculation can reach a maximum of 3.5 M Grid/s due to the relative complicated FFT. Gradient of MI w.r.t.  $f^1, f^2, f^3, f^4$  is at maximum of 2.5 M Grid/s. Its varying performance is due to the mixed grid-level and pixel level operations. The grid-level graph does not show the corresponding pixel-level tasks it is handling. Its lower performance value is due to multiple data reads problem described in the previous section. Figure 5.16 shows the maximum number of mega-pixels performed by the GPU per second (M Pixel/s) for specific pixel-level operation when total number of voxels increases. Image interpolation is observed at maximum of 160 M Pixel/s despite its eight reads of global memory. Gradient of MI w.r.t.  $x, y, z$  has 4 M Pixel/s no matter what image size is used. The value for MI calculation is not listed here due to the fixed 256-threads implementation.

Although the running on CPU does not mean running on one thread in GPU, the speed up comparison aims to provide an idea of the magnitude of performance increase by switching from CPU to GPU. Keeping in mind GPU development grows more rapidly than CPU, an investment in a new GPU of similar price to a new CPU is worthy. Table 5.4 shows the recorded speed up when the grid size increases. The time required to execute a kernel on CPU is

assumed to be roughly proportion to the problem size. Generally, when data size increases, threads work more productively, speed up increases. Especially in the normalization of gradients, a maximum speed up of 221 is observed. It is due to all input data and output data are independently divided among all the threads. Runge-Kutta achieves a speedup of 191. Div-curl has a speed up of 30. Gradient of MI w.r.t.  $f^1, f^2, f^3, f^4$  has a speed up of 42. Again, fluctuations are due to mixed grid-level and pixel-level operations present. Figure 5.17 displays the data in Table 5.4 in a graph which clearly shows the speed up relationship. Table 5.5 presents the speed up of pixel-level operations. Due to high data interdependency, MI calculation, Gradient of MI w.r.t.  $x, y, z$  and image interpolation have a near constant maximum speed up of around 4, 170 and 15 respectively. It is very challenging to further improve those numbers unless at the algorithm level. This could be extended to a new research topic. Overall, the speed up of GPU to CPU for the whole registration process is about 40 times.

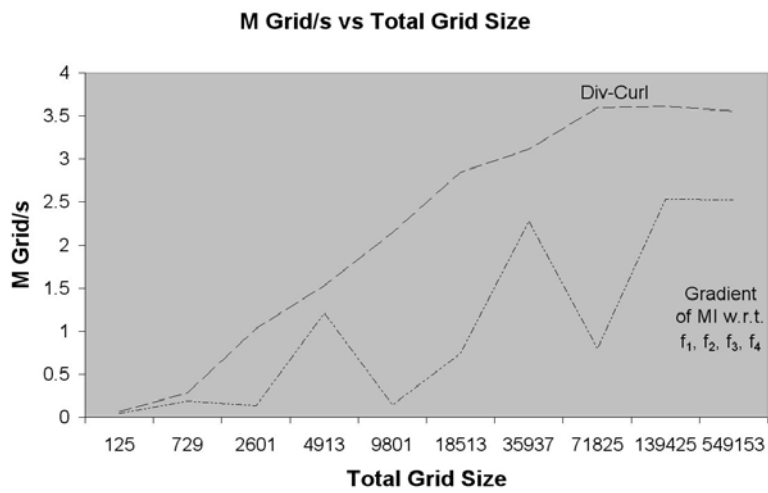
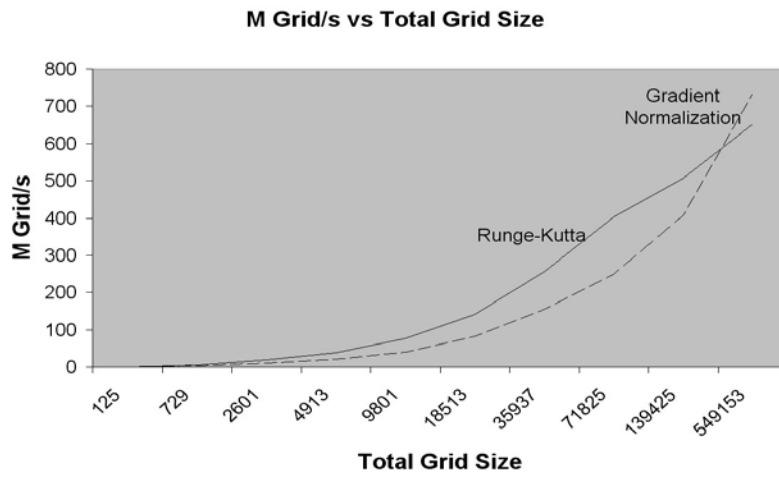


Figure 5.15 Amount of M Grid/s of data processed by GPU vs. Total Grid Size of Runge-Kutta method and gradient normalization.

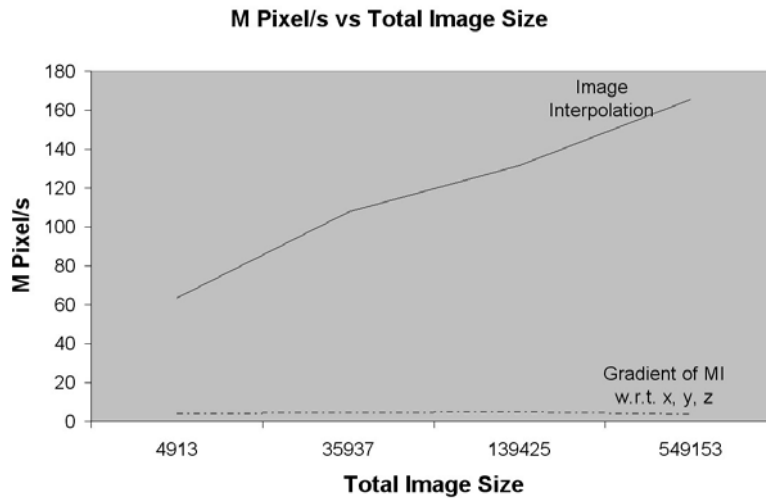


Figure 5.16 Amount of M Pixel/s of data processed by GPU vs. Total Image Size.

Table 5.4 Speed up of GPU to CPU at different resolution levels for grid-level operations.

Grid Size	Total No. of Grids	Speed up observed			
		Div-Curl	Runge-Kutta	Gradient Normalization	Gradient of MI w.r.t. $f$
5×5×5	125	3.34	4.72	0.66	9.94
9×9×9	729	4.79	6.96	1.56	8.40
17×17×9	2601	10.21	10.76	3.97	26.26
17×17×17	4913	11.87	10.43	8.07	19.48
33×33×9	9801	13.21	35.02	15.04	21.53
33×33×17	18513	16.36	30.86	30.47	26.45
33×33×33	35937	17.57	59.51	56.68	36.61
65×65×17	71825	23.08	124.21	94.20	34.21
65×65×33	139425	24.65	139.75	135.34	41.92
129×129×33	549153	29.91	190.69	221.15	41.83

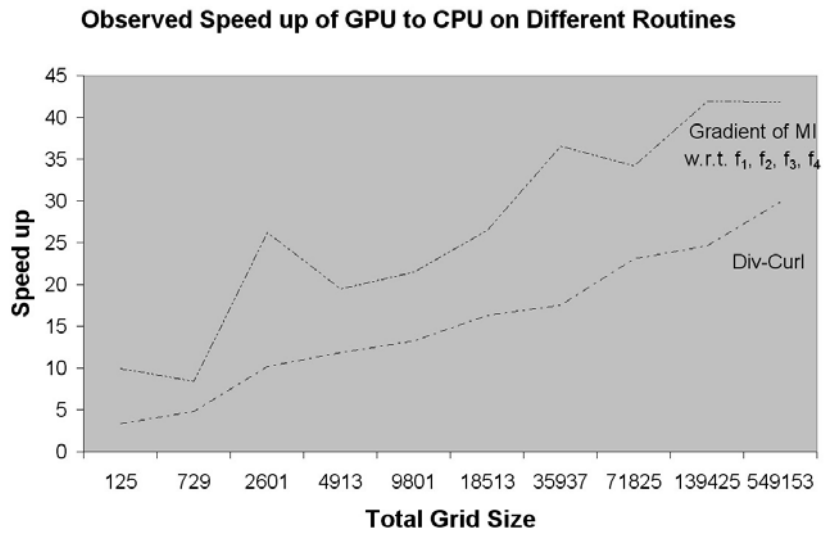
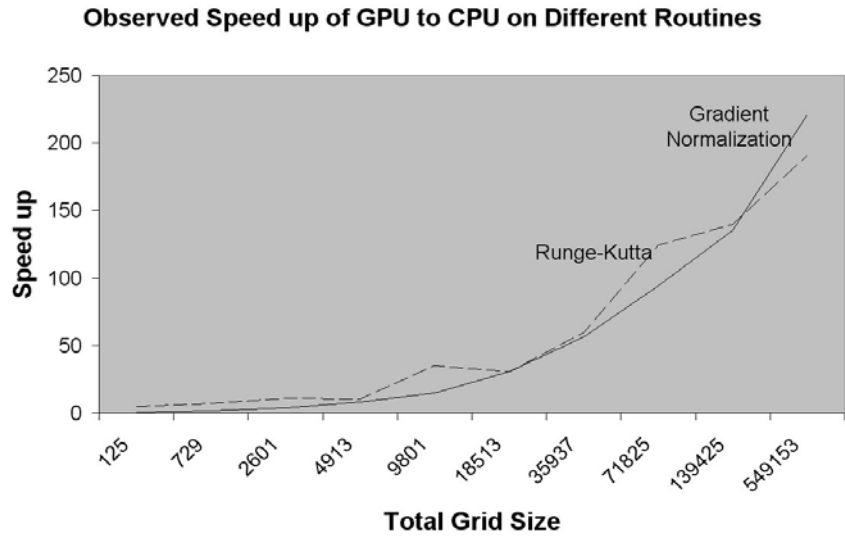


Figure 5.17 Observed Speed up of GPU to CPU vs. Total Grid Size.

Table 5.5 Speed up of GPU to CPU at different resolution levels for pixel-level operations.

Image Size	Total No. of Pixels	Speed up observed		
		MI	Gradient of MI w.r.t. $x, y, z$	Image Interpolation
17×17×17	4913	2.44	149.31	9.93
33×33×33	35937	4.02	163.23	15.94
65×65×33	139425	4.77	178.90	15.41
129×129×33	549153	4.20	140.82	15.97

## 5.6 Compressibility-aware Monitoring

“Cropped 3D Image Data” is used to demonstrate the power of compressibility-aware monitoring mentioned in section 3.5. Jacobian of the transformation with and without monitoring are recorded after registering reference image  $R$  at time  $1$  to template image  $T$  at time  $0$  as shown in Figure 5.18 and Figure 5.19 respectively. Total image size is  $1,017,681$  voxels. In the experiments, the monitor function is set to greater than  $0.9$  to disallow shrinking of contrast enhanced voxels and allow some errors in real world data. In Figure 5.18, with monitoring, Jacobian values always lie above  $0.9$ . In Figure 5.19, without monitoring, Jacobian values could even go negative or get extremely big. This suggests uncontrolled volume changes when monitoring is absent.

As addressed in [34], volume calculation is not a trivial task. A segmentation mask is generated according to the contrast difference between the rigid-aligned reference image and the template image. This segmentation mask is then used to contain the volume of interest for counting changes in volume. Since their segmentation method is a rough approximation, an even simpler method is employed here. Four contrast enhanced sub-volumes are hand-picked for volume changes observation. If intensity value of a voxel is greater than a specific threshold, the voxel is considered part of the sub-volume. Figure 5.20 shows the number of voxels included for four different contrast enhanced volumes during different iterations for registering template image at time  $0$  and reference image at time  $1$ . Minor fluctuations are observed but generally volume changes are within  $10\%$ . This is reasonable since the Jacobian values collected in the registration with monitoring are in the range of  $0.9$  to  $1.15$ . The volume changes from without monitoring registration are not collected since hand-picked region may not have extreme volume changes to be exhibited.

“Simulated 3D Image Data” is experimented to MI and error behavior. In Figure 5.21, the MI curve is compared with and without the applying compressibility-aware monitoring. The corresponding average warping index is shown in Figure 5.22. From the MI comparison curves,

registration without monitoring has more freedom in deformation and is able to obtain higher MI value than registration with monitoring. From the average warping index comparison curves, though the curves have different convergence shapes, they are able to achieve the same low error value at the end. An example of near uniform deformation is shown in Figure 5.23.

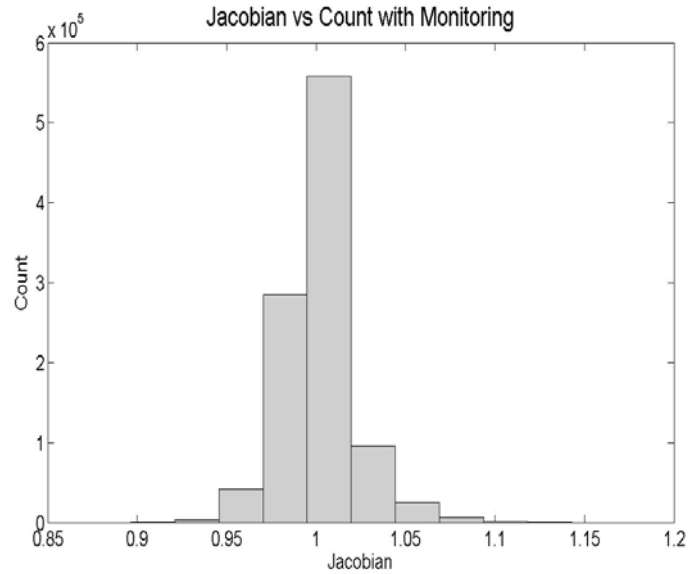


Figure 5.18 Jacobian vs. Count for registering reference image at time 1 to template image at time 0.

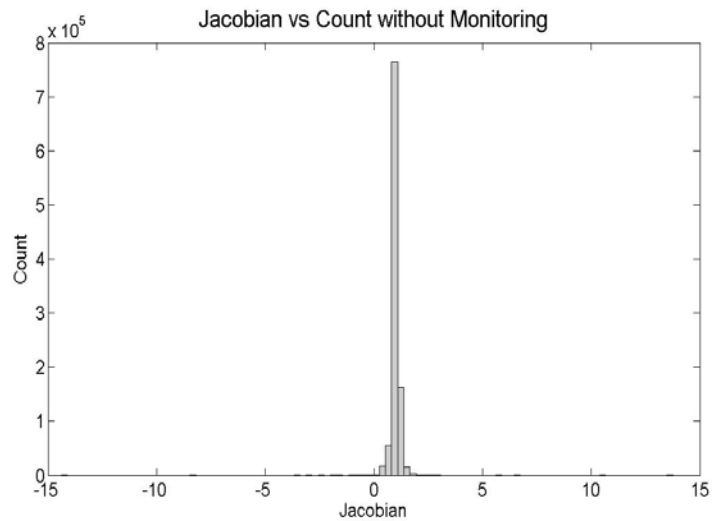


Figure 5.19 Jacobian vs Count for registering reference image at time 1 to template image at time 0. A number of extreme values are observed.



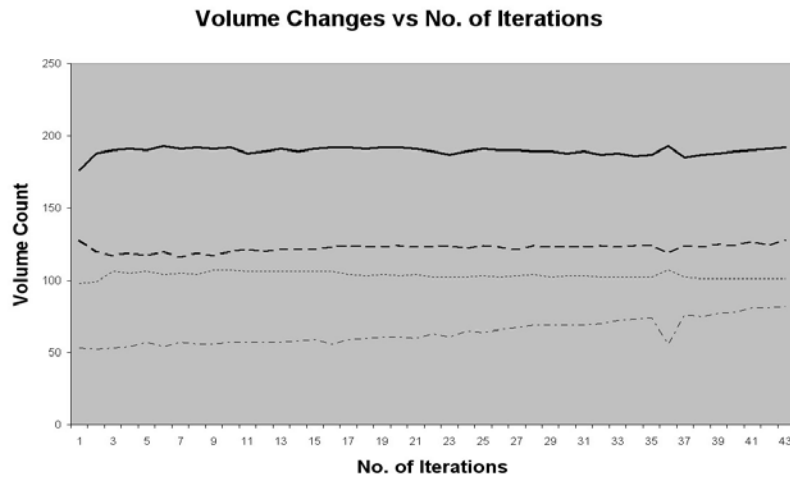


Figure 5.20 Volume changes during different iterations in registering template image at time 0 with reference image at time 1.

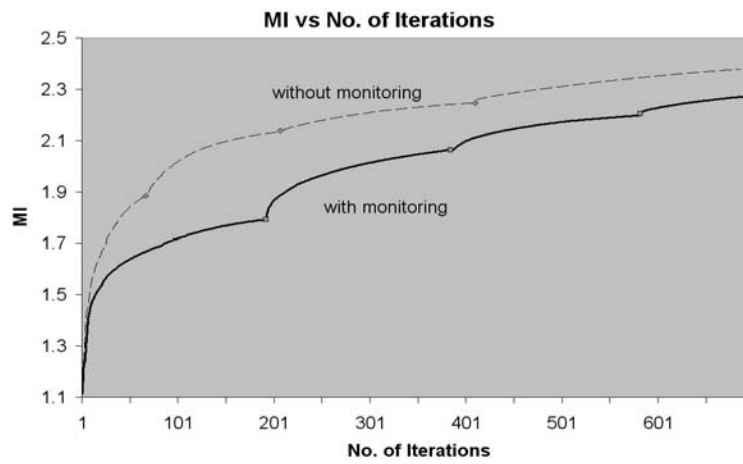


Figure 5.21 Comparison on the effect of with and without monitoring on MI vs. no. of iterations. Markers show grid refinement points.

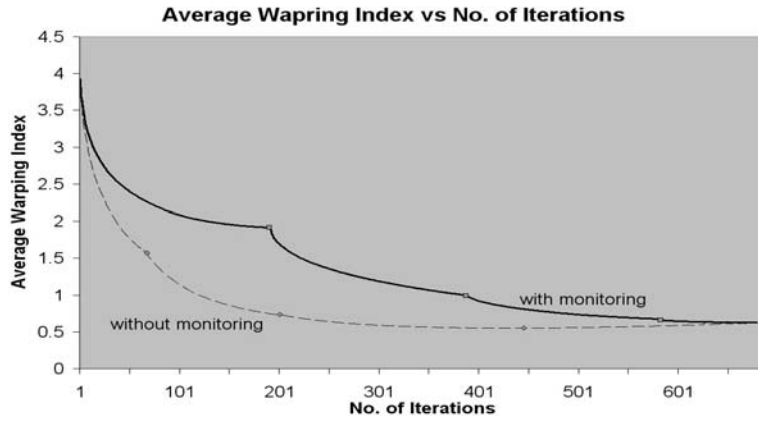


Figure 5.22 Comparison on the effect of with and without monitoring on average warping index vs. no. of iterations. Markers show grid refinement points.

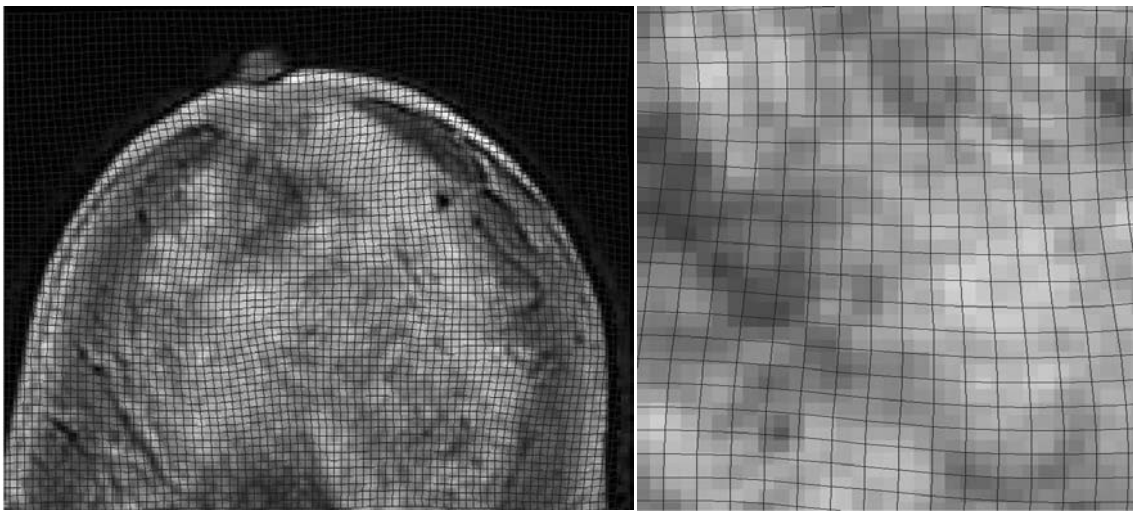


Figure 5.23 Deformed grids of near uniform sizes.

CHAPTER 6  
EXPERIMENTAL RESULTS USING CLINIC BREAST MRI

6.1 Overview

An optimization framework for non-rigid breast MRI registration is presented. The non-rigid div-curl system properly models the local deformation in breast tissues. Mutual information provides a robust mean to measure similarity in real world situation where there is no guarantee in constant illumination and noise condition. The monitor function is naturally aware of the incompressibility by limiting the ratio of area of before and after reconstruction to be constant. Various optimization techniques facilitate the use at scanner consoles or interactive workstations.

6.2 Setup

The dataset used in this experiment is contributed by Dr. Peng at the Department of Radiology at The University of Texas Health Science Center at San Antonio. The patient was lying when MRI were taken. The original dataset consists of 50 slices of 2D 512×512 breast MRI taken at 11 time steps. The whole imaging process takes around 400 seconds. Each time step is matched to the actual timeline as shown in Table 6.1. Each MRI contains a slice of two breasts and the chest region.

Table 6.1 Actual Time for Different Time Steps

Time Step	0	1	2	3	4	5	6	7	8	9	10
Actual Timeline (in s)	0	84	119	154	190	225	261	296	331	367	402

All breast image slices are manually cropped to one breast of size 161×129. The experiment takes 49 slices as input. These input dimensions are designed for proper grid

allocation. There are 11 time steps in the input data. The data volume at time step 0 is considered as the template image  $T$  and the data volume at time step 1 to 10 are the reference images  $R$  to be reconstructed. The 3D volumes are globally aligned to time step 0 using mutual information based rigid registration. Alignment to image volume at step 0 provides a more convenient way for massive contrast enhancement analysis but the enhanced intensity values in the reference image may go out of range of the normal 256 levels of intensity value. To solve this problem, any voxel with intensity value greater than the maximum intensity value in  $T$  is considered contrast enhanced. All contrast enhanced voxels are considered registered in MI calculation. In other words, the displacements of the contrast enhanced voxels are controlled by the similarity of joint histogram of the surrounding non-contrast enhanced voxels of the reference image to the template image. In the experiment data, the maximum intensity value of the template image was 154. This is called the “MI-Match” method in the following context. SSD is another candidate for similarity measure when contrast enhanced regions are considered matched. This is called “SSD-Match” in the results. Alternately, input images could be normalized first but accuracy would be lost. This is investigated as the “MI-Norm” method.

Tolerance level is ten chances of no improvements and maximum of 100 iterations per resolution. Multi-resolution scheme of grid distances 8, 4, 2 and 1 with regriding is in use. Monitor function is set to be at least 0.9 in compressibility-aware monitoring to allow some real world error.

GPU computation is applied to the dynamic contrast enhanced 3D breast MRI data. Experiments are conducted on the same computer with GPU acceleration and written in CUDA as covered in Chapter 5.

### 6.3 Results

The overall registration results from time steps 1 to 10 are discussed here. Our algorithm is able to converge at over time. As shown in Figure 6.1, MI improves straightly as time increases in all registration at time steps 1 to 10 in the MI-Match method.

The Figure 6.2 shows the contrast difference of the central slice in time step  $0$  coordinates before and after registration at different time steps of different methods. The reference image  $R$  taken at time step  $1$  to  $10$  which is reconstructed to register the template image  $T$  taken at time step  $0$ . The absolute difference between the reconstructed  $R$  and  $T$  are shown in Figure 6.2. Contrast at some regions amplifies a lot where contrast at others remains about the same. Some edges of the breast are gone in the after registration images. MI-Match and MI-Norm have similar visual results. This implies MI is insensitive to contrast changes in MI-Norm method and the adaptive grid system is able to allocate grids correctly using the non-contrast-enhanced region for MI maximization guide in MI-Match. The only concern of MI-Match and SSD-Match is when majority of the volume is contrast enhanced region. There may not be enough non-contrast regions to quantify the deformation for similarity optimization. This problem is exhibited in the SSD-Match visualization where the contrast regions in the center could not be correctly identified.

Figure 6.3 shows the joint histogram of the registration at different time steps in MI-Match method. Sharper areas along the diagonal indicate better alignment. The joint histograms in all time steps condense to the diagonal as registration proceeds. The condensation implies registration aligning more voxels over time. The sharp white diagonal shown in registrations at time step  $2$  to  $10$  are due to contrast enhanced matching assumption for MI-Match discussed in the setup. Figure 6.4 shows the joint histogram of the registration at time steps  $1$  and  $2$  in MI-Norm and SSD-Match methods. In MI-Norm method, intensity values are normalized to range of  $0$  to  $255$ . The un-continuities resulting in blank vertical and horizontal lines are due to missing intensities value after normalization. After time step  $1$ , intensity value range is broadening due to presence of contrast enhancement. The matching diagonal moves towards the vertical axis. Joint histograms in other time steps are similar. In SSD-Match method, the joint histogram graph is normal as higher resolution registration gives sharper diagonal.

MI-Match is the preferred method in the following. The percentage of intensity enhancement of every voxel in the  $161 \times 129 \times 49$  breast MRI are recorded. The top 10,000 percentage intensity enhancement of the total 1,017,681 is plotted in Figure 6.5. Figure 6.6 shows the zoomed in top 500 enhancements. The zoomed-in plot gives a vague suggestion for possible threshold to distinguish malignant tissues from benign tissues. The threshold value chosen from the graph is around 200% beyond time step 2. Figure 6.7 shows the intensity enhancement of possible malignant tissues after the threshold was set. This provides an initial guess on the cancerous volume.

The intensity enhancement of four sample points is illustrated in Figure 6.8. Point A and B jump dramatically at time step 2 and remain high throughout the curve. Point C and D do not exhibit high percentage of increase during the imaging process. Roughly comparing with the general contrast agent uptake curve, point A and B are possible lesion tissues whereas point C and D are normal tissues.

Our algorithm successfully aligned the dynamic contrast-enhanced breast MRI. Motion across different time steps is corrected by global rigid and local non-rigid transformation while contrast-enhanced features remain the about same size. Corresponding pixels at different time steps are then related and intensity increment for every pixel could be found. Different kinds of tissues give different responses along the imaging timeline. A model to classify tissues into carcinoma, fiber-adenoma, normal tissue, muscle and fat based on the response curves should be used after the registration process. Further investigation is required to find an appropriate model for classification.

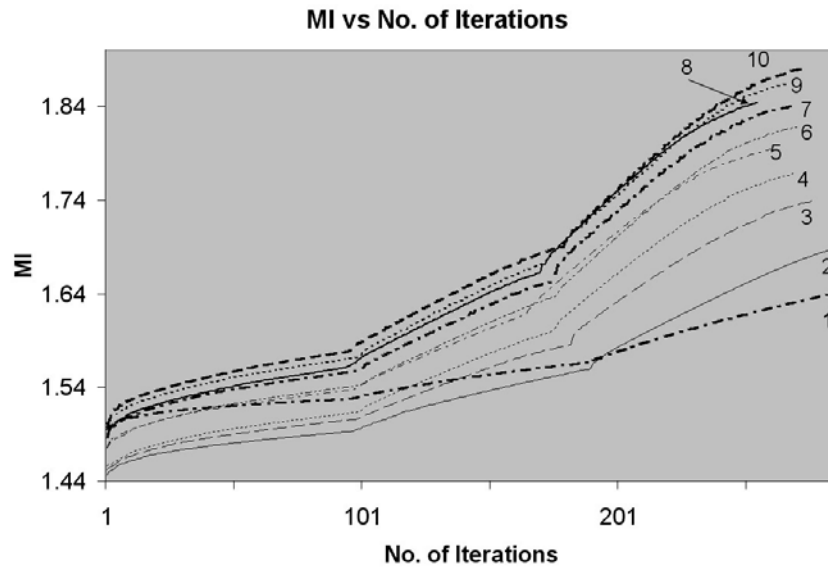
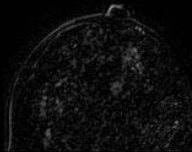
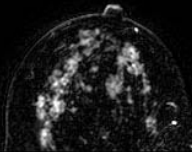
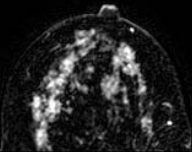
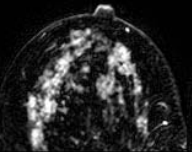
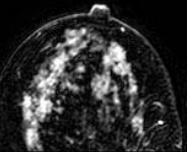
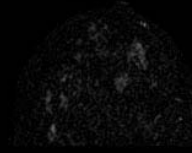
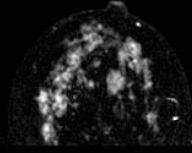
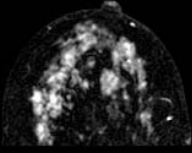
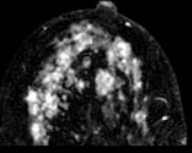
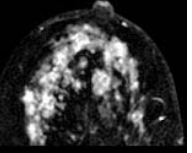
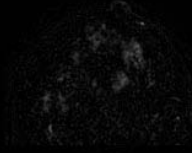
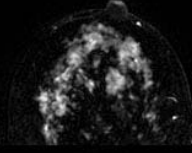
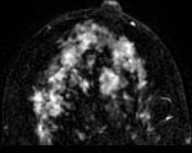
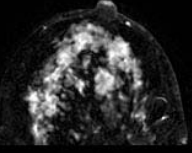
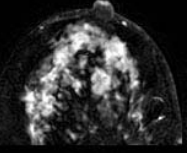

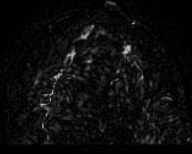
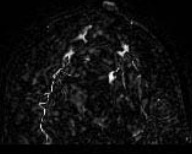
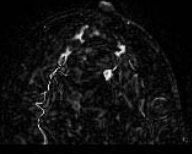
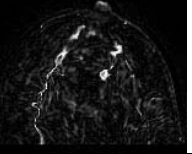
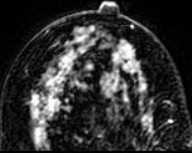
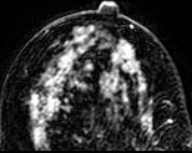
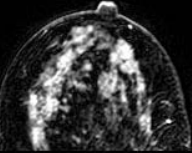
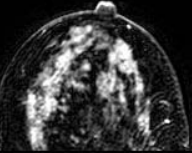
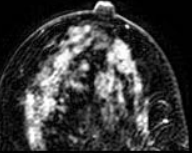
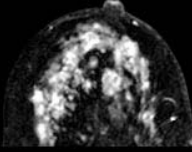
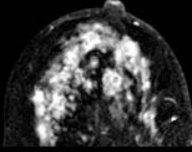
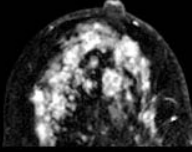
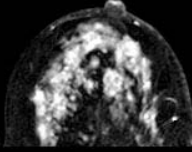
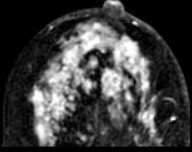
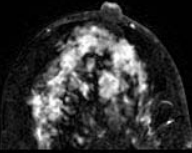
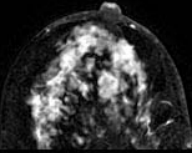
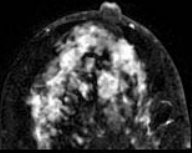
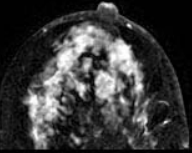
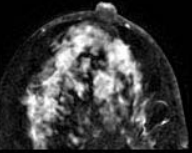
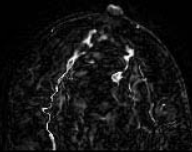
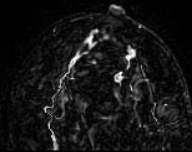
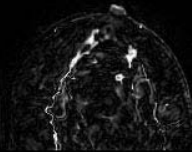
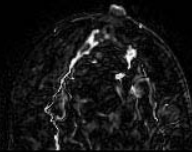
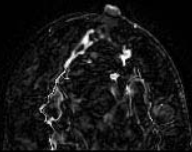


Figure 6.1 MI vs. no. of iterations for registrations at time step 1 to 10 of MI-Match.

	Time Step 1	Time Step 2	Time Step 3	Time Step 4	Time Step 5
Before					
After (MI-Match)					
After (MI-Norm.)					
After (SSD-Match)					

	Time Step 6	Time Step 7	Time Step 8	Time Step 9	Time Step 10
Before					
After (MI-Match)					
After (MI-Norm.)					
After (SSD-Match)					

MI-Match: MI-based registration assuming contrast-enhanced regions as matched.

MI-Norm: MI-based registration with normalized intensities.

SSD-Match: SSD-based registration assuming contrast-enhanced regions as matched.

Figure 6.2 Differences of the central slice before and after registration at different time steps for different methods.



Time Step	Before Registration	After Rigid Reg.	After Non-rigid Resolution			
			h=8	h=4	h=2	h=1
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						

Figure 6.3 Joint histograms of registration at different time steps of MI-Match.

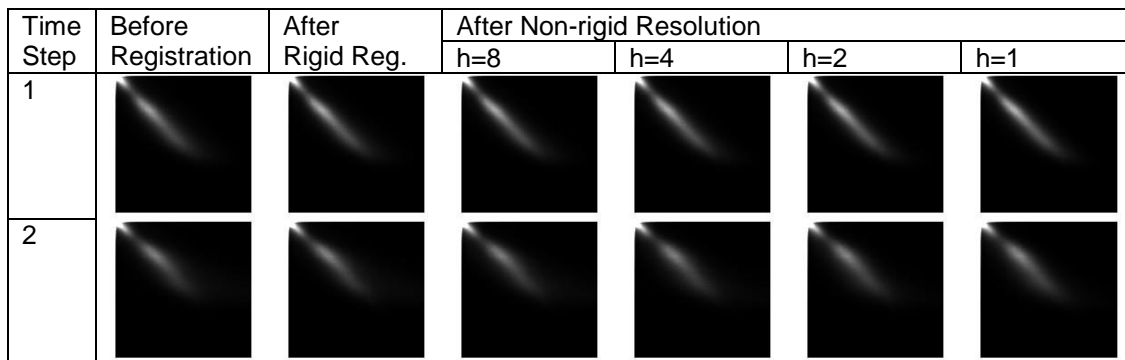
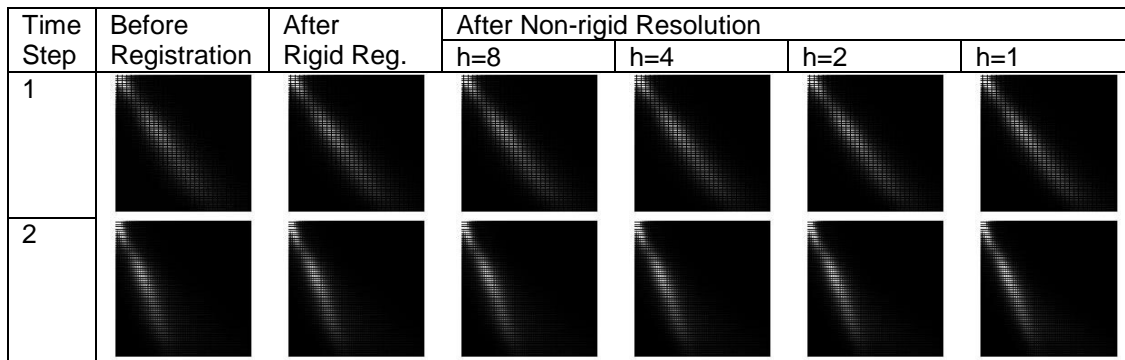


Figure 6.4 Joint histograms of registration at different time steps of MI-Norm (top) and SSD-Match (bottom).

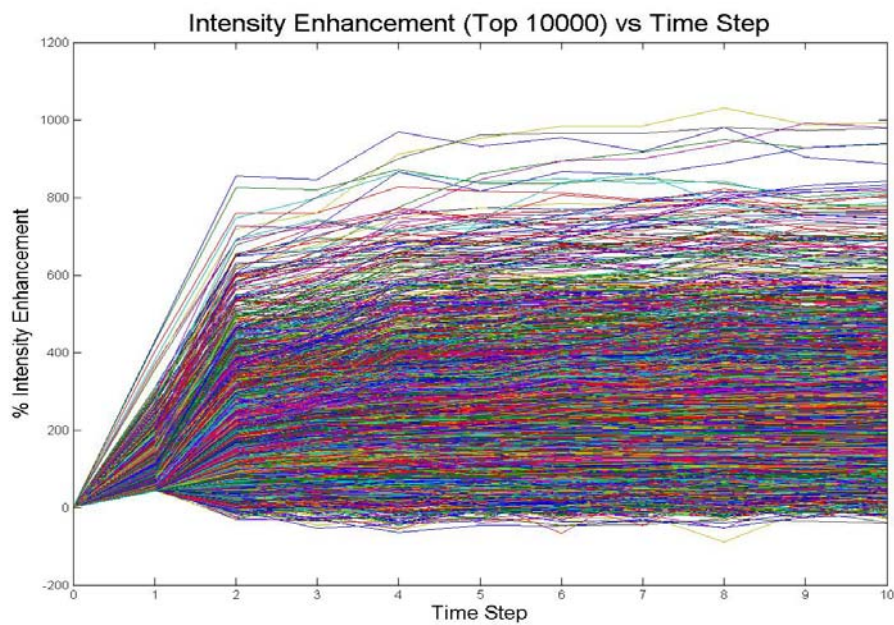


Figure 6.5 Top 10,000 intensity enhancement vs. time steps.

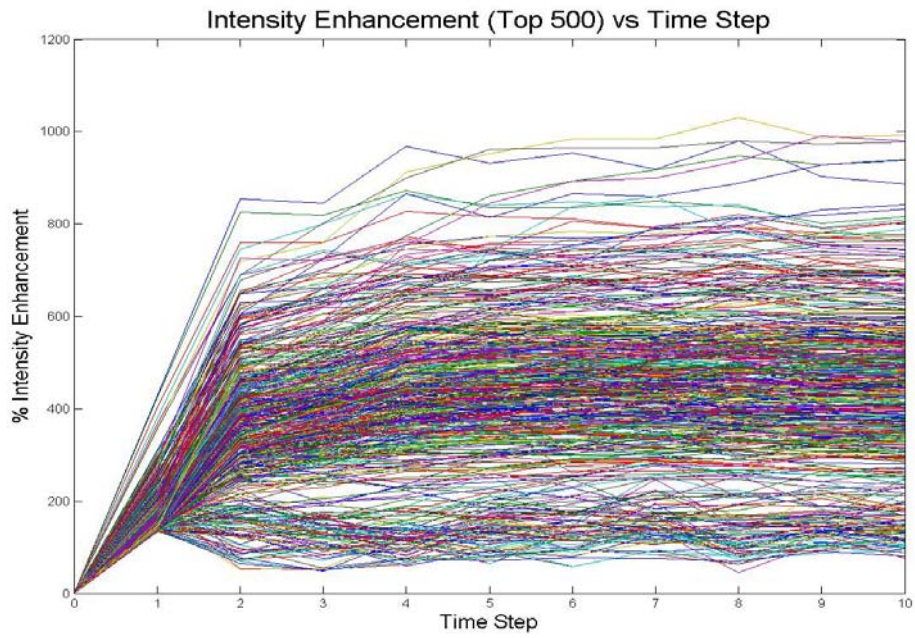


Figure 6.6 Top 500 intensity enhancement vs. time steps.

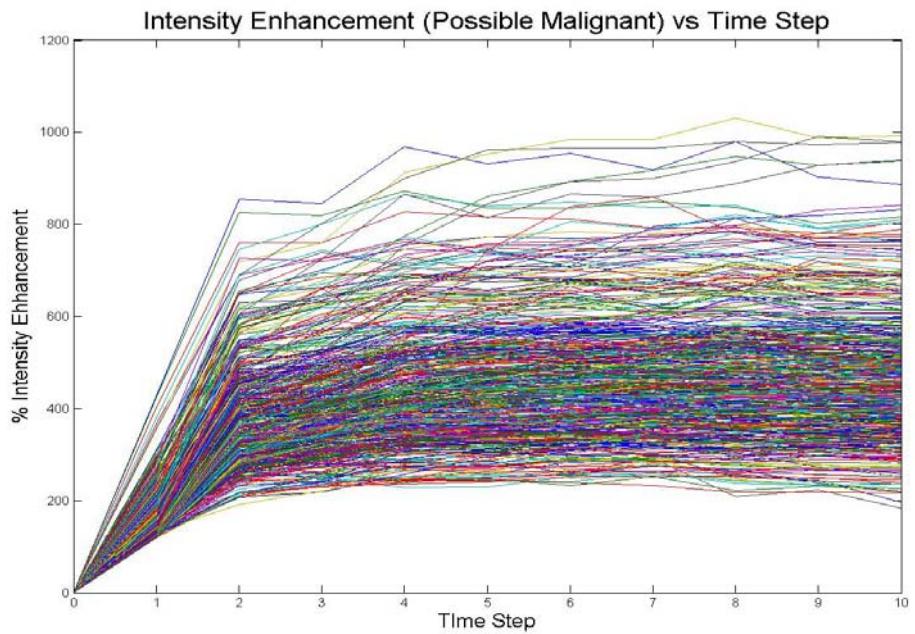


Figure 6.7 Intensity enhancement of possible malignant tissues after setting a threshold.

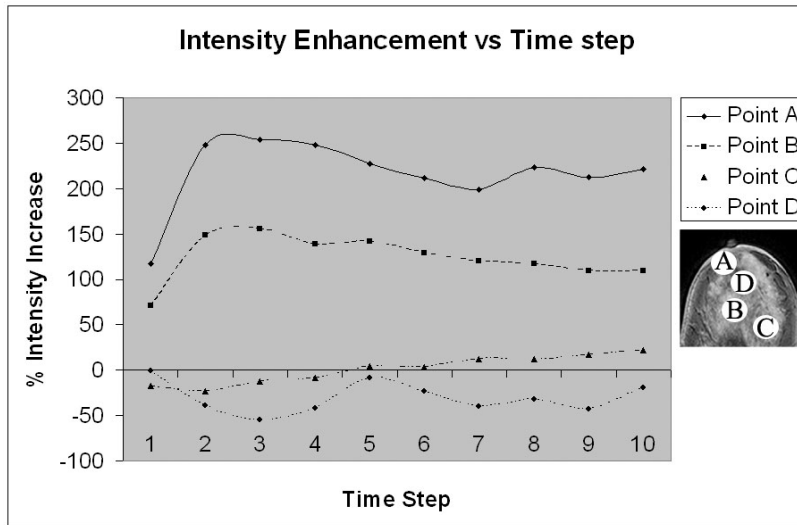


Figure 6.8 Intensity enhancements of four sample points. Point A and B are possible lesion regions. Point C and D are normal tissues.

## CHAPTER 7

### CONCLUSIONS

A new approach to non-rigid image registration using mutual information is presented. Adaptive grid generation models the non-rigid deformation by adjusting the divergence and curl parameters. Its compressibility-aware monitor function allows flexible control of the transformation area or volume ratio. Setting the ratio to unity resembles the incompressibility of human tissues. Mutual information (MI) as a similarity measure is robust against noise and different illumination similarity measure. Artifacts of different histogram estimations are investigated. Gradient of different MI variation are derived for the optimization scheme. Newly improved gradient information provides more accuracy in gradient descent optimization scheme. Multi-resolution strategy with regridding cuts down the registration time which makes it feasible for clinical practice. Our algorithm is well scalable to handle 3D real image data and capable to register between difference modalities. The algorithm is validated by simulated deformation to compensate the absence of ground truth. The framework is applied to dynamic contrast enhanced breast MRI which requires the algorithm to be able to handle local motion of soft tissues and contrast enhanced region. This is a positive match with the compressibility-awareness of adaptive grid generation and robustness of MI. With GPU optimization, the performance of the system is pushed to a newer peak. More than 40 times speedup is observed in the current configuration. If finer tuning is conducted on an elite GPU, even higher performance could be expected. The experiments show promising results for the breast MRI. It is possible to include this algorithm in computerized breast cancer detection, which is an essential module in the next generation of clinical procedure.

## CHAPTER 8

### FUTURE WORKS

In the future, the GPU implementation could be more fine-tuned to fully utilize all GPU resources. Further investigation on data-parallel MI calculation should be well researched. Scalable implementation should be prepared for better GPU being manufactured in near future. It is possible to extend this work to other contrast enhanced medical images. Currently, 3D visualization of the final results has not been completed yet. This work would become a powerful clinical tool if incorporated with 3D rendering and model fitting the contrast enhancement results.

## REFERENCES

- [1] ATI's CTM: <http://ati.amd.com/companyinfo/researcher/Documents.html>
  
- [2] BrookGPU: <http://graphics.stanford.edu/projects/brookgpu/>
  
- [3] NVIDIA's Compute Unified Device Architecture (CUDA):  
[http://www.nvidia.com/object/cuda\\_home.html](http://www.nvidia.com/object/cuda_home.html)
  
- [4] Sh, A High-level Metaprogramming Language for Modern GPUs: <http://libsh.org/>
  
- [5] P. Armitage, C. Behrenbruch, M. Brady and N. Moore, "Extracting and visualizing physiological parameters using dynamic contrast-enhanced magnetic resonance imaging of the breast," Medical Image Analysis, vol. 9, 2005, pp. 315-329.
  
- [6] R. Bajcs, S. Kovacic, "Multiresolution elastic matching," Comp Vision Graphics Image Processing 1989, vol. 46, pp. 1-21.
  
- [7] F. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 11, no.6, 1989, pp. 567-585.

- [8] X. X. Cai, D. Fleitas, B. Jiang and G. Liao, "Adaptive Grid Generation Based on the Least-Squares Finite-Element Method," *Computers and Mathematics with Applications*, vol. 48, 2004, pp. 1077-1085.
- [9] H. Chen, C.Y. Hsieh and G. Liao, "Non-rigid Image Registration Using Adaptive Grid Generation: Preliminary Results," in *2007 4th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 2007, pp. 580-583.
- [10] H. Chen, T. Lin, C. Hsieh, H. Hsiao, M. Chu and G. Liao, "Large Deformation Image Registration Using Regridding," submitted to *IEEE Transactions on Pattern Analysis and Machine Intelligence* in 2008.
- [11] M. Chu, H. Chen, C. Hsieh, T. Lin, H. Hsiao, G. Liao and Q. Peng, "Adaptive Grid Generation Based Non-rigid Image Registration using Mutual Information for Breast MRI", to be appeared in *Journal of Signal Processing Systems*, 2008.
- [12] G. E. Christensen, R. D. Rabbitt, and M. I. Miller, "Deformable Templates using Large Deformation Kinematics," *IEEE Transactions on Image Processing*, vol. 5, no. 10, pp. 1435–1447, 1996.
- [13] W. R. Crum, T. Hartkens and D. L. G. Hill, "Non-rigid image registration: theory and practice," *British Journal of Radiology*, 2004, pp. 140-153.



- [14] J. Demmel, "Solving the Discrete Poisson Equation using Jacobi, SOR, Conjugate Gradients, and the FFT," online lecture notes, March 1996. Available at:  
<http://www.cs.berkeley.edu/~demmel/cs267/lecture24/lecture24.html>.
- [15] N. Dowson and R. Bowden, "A Unifying Framework for Mutual Information Methods for Use in Non-linear Optimisation," in ECCV 2006, pp. 365-378.
- [16] L. Dougherty, J. Asmuth and W. Geftter, "Alignment of CT lung volumes with an optical flow method," Acad Radiol 2003, vol. 10, pp. 249–54.
- [17] Z. Fan, C. Vetter, C. Guetter, D. Yu, R. Westermann, A. Kaufman and C. Xu, "Optimized GPU Implementation of Learning-Based Non-Rigid Multi-Modal Registration," Proc. SPIE Medical Imaging 2008, vol. 6914, 69142Y.
- [18] M.S. Froh, D. C. Barker, K. K. Brock, D. B. Plewes and A. L. Martel, "Piecewise-Quadrilateral Registration by Optical Flow – Applications in Contrast-Enhanced MR Imaging of the Breast," in MICCAI 2006, pp. 686-693.
- [19] L. Grady, T. Schiwietz, S. Aharon and R. Westermann, "Random Walks for Interactive Organ Segmentation in Two and Three Dimensions: Implementation and Validation", Proceedings of MICCAI 2005, vol. 2, 2005, pp. 773-780.
- [20] D.J. Griffiths, Introduction to Electrodynamics, reading, 1989, Prentice Hall.

- [21] P.D. Hovland and B. Norris, "Users' Guide to ADIC 1.1", Argonne Technical Memorandum ANL/MCS-TM-225
- [22] H.Y. Hsiao, H. Chen, T. H. Lin, C.Y. Hsieh, M. Y. Chu and G. Liao, "A new parametric nonrigid image registration work based on Helmholtz's theorem," Proc. SPIE Medical Imaging 2008, vol. 6914, 69142W (Mar. 11, 2008).
- [23] C.Y. Hsieh, H. Chen, T. H. Lin, H.Y. Hsiao, M. Y. Chu and G. Liao, "On the development of a new non-rigid image registration using deformation-based grid generation," Proc. SPIE Medical Imaging 2008, vol. 6914, 69140W (Mar. 11, 2008).
- [24] B. Jiang, The Least-Squares Finite Element Method, reading, 1998, Springer.
- [25] C. K. Kuhl and H. H. Schild, "Dynamic Image Interpretation of MRI of the Breast," Journal of Magnetic Resonance Imaging, vol. 12, 2000, pp. 965-974.
- [26] J. Kybic and M. Unser, "Fast Parametric Elastic Image Registration," IEEE Transactions on Image Processing, vol. 12, no. 11, 2003, pp. 1427-1442.
- [27] A. E. Lefohn, J. E. Cates and R. T. Whitaker, "Interactive, GPU-Based Level Sets for 3D Segmentation", Proceedings of MICCAI 2003, vol. 2, 2003, pp. 564-572.
- [28] G. Liao and D. Anderson, "A new approach to grid generation," Applicable Analysis, vol. 44, no. 3, 1992, pp. 285-298.

- [29] G. Liao, X. Cai, D. Fleitas, X. Luo, J. Wang and J. Xue, "Optimal control approach to data set alignment," *Applied Mathematics Letters*, no. 21, 2008, pp. 898–905.
- [30] F. Maes, A. Collingon, D. Vdermeulen, G. Marchal, P. Suetens, "Multimodality Image Registration by Maximization of Mutual Information," *IEEE Transactions on Medical Imaging*, vol. 16, 1997, pp. 187-198.
- [31] J.P.W. Pluim, J.B.A. Maintz and M. A. Viergever, "Interpolation artifacts in mutual information-based image registration", *Computer vision and image understanding*, vol. 77, pp. 211-232.
- [32] V. Podlozhnyuk, "Histogram calculation in CUDA," Technical report, Nvidia 2007.
- [33] V. Podlozhnyuk, "Image convolution in CUDA," Technical report, Nvidia 2007.
- [34] T. Rohlfing, C. R. Maurer, Jr., D. A. Bluemke and M. A. Jacobs, "Volume-Preserving Nonrigid Registration of MR Breast Images Using Free-Form Deformation With an Incompressibility Constraint," *IEEE Transactions on Medical Imaging*, vol. 22, no. 6, 2003, pp. 730-741.
- [35] L. Roose, W. Mollemans, D. Loeckx, F. Maes and P. Suetens, "Biomechanically Based Elastic Breast Registration Using Mass Tensor Simulation," in *MICCAI 2006*, pp. 718-725.

- [36] D. Rueckert, L. I. Sonoda, C. Hayes, D. L. G. Hill, M. O. Leach and D. J. Hawkes, "Nonrigid registration using free-form deformations: Application to breast MR images," , IEEE Transactions on Medical Imaging, vol. 18, no. 8, 1999, pp. 712–721.
- [37] M. Sdika, "A Fast Nonrigid Image Registration With Constraints on the Jacobian Using Large Scale Constrained Optimization," IEEE Transactions on Medical Imaging, vol. 27, 2008, pp. 271-281.
- [38] R. Shams and N. Barnes, "Speeding up Mutual Information Computation Using NVIDIA CUDA Hardware," in DICTA 2007, pp. 550-560.
- [39] R. Strzodka, M. Droske and M. Rumpf. "Image Registration by a Regularized Gradient Flow - A Streaming Implementation in DX9 Graphics Hardware." Computing, 73(4):373–389, 2004.
- [40] D. Suter, "Motion Estimation and Vector Splines," IEEE Proceedings CVPR '94, 1994, pp. 939-942.
- [41] P. Thévenaz, U. Ruttimann and M. Unser, "A pyramid approach to subpixel registration based on intensity," IEEE Transactions on Image Processing, vol. 7, 1998, pp. 1-15.
- [42] P. Thévenaz and M. Unser, "Optimization of Mutual Information for Multiresolution Image Registration," IEEE Transactions on Image Processing, vol. 9, 2000, pp. 2083-2099.

- [43] M. Unser, A. Aldroubi and M. Eden, "B-Spline Signal Processing: Part I-Theory," IEEE Transactions on Signal Processing, vol. 41, no. 2, 1993, pp. 821-833.
- [44] M. Unser, A. Aldroubi and M. Eden, "B-Spline Signal Processing: Part II-Efficient Design and Applications," IEEE Transactions on Signal Processing, vol. 41, no. 2, 1993, pp. 834-848.
- [45] P. K. Varshney and M. K. Arora, Advanced Image Processing Techniques for Remotely Sensed Hyperspectral Data, Springer, 2004, ch. 3.
- [46] F. Xu and K. Mueller "Accelerating Popular Tomographic Reconstruction Algorithms on Commodity PC Graphics Hardware." IEEE Transactions on Nuclear Medicine, vol. 52, no. 3, 2005, pp. 654- 663.
- [47] X. Xu and R. D. Dony, "Fast Fluid Registration Using Inverse Filtering for Non-rigid Image Registration," in ISBI 2006, pp. 470-473.
- [48] T.S. Yoo, Insight into Images: principles and practice for segmentation, registration, and image analysis, Wellesley, Massachusetts: A K Peters, Ltd., 2004.

## BIOGRAPHICAL INFORMATION

Yvonne M. Y. Chu received the B.Eng. and the M.Sc. degree in Computer Science in 1999 and 2001 respectively from the Hong Kong University of Science and Technology, Hong Kong, China. She is now pursuing a PhD degree in Computer Science and Engineering at the University of Texas at Arlington, Texas, United States. She has been working in various multimedia editing, video coding and medical image processing projects. After graduation, her career plan is to work in medical image research and development.