Registering a Non-Rigid Multi-Sensor Ensemble of Images

by

Hwa Young Kim

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2009

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Image registration is the task of aligning two or more images into the same reference frame to compare or distinguish the images. The majority of registration methods deal with registering only two images at a time. Recently, a clustering method that concurrently registers more than two multi-sensor images was proposed, dubbed *ensemble clustering*. In this thesis, we apply the ensemble clustering method to deformable registration scenario for the first time. Non-rigid deformation is implemented by a FFD model based on B-splines. A regularization term is added to the cost function of the method to limit the topology and degree of the allowable deformations. However, the increased degrees of freedom in the transformations caused the Newton-type optimization process to become ill-conditioned. This made the registration process unstable. We solved this problem by using the matrix approximation afforded by the singular value decomposition (SVD). Experiments showed that the method is successfully applied to non-rigid multi-sensor ensembles and overall yields better registration results than methods that register only 2 images at a time. In addition, we parallelized the ensemble clustering method to accelerate the performance of the method. The parallelization was implemented on GPUs using CUDA (Compute Unified Device Architecture) programming model. The GPU implementation greatly reduced the running time of the method.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor Jeff Orchard, for his invaluable and helpful guidance during my research and study here at the University of Waterloo. I feel very fortunate to have him as a supervisor. He was always there to provide explanations to my questions with patience, motivation and encouragement. His keen interest in my work has greatly motivated me and has made this thesis possible.

I would also like to thank to my thesis committee members, Richard Mann and Justin Wan, for their thoughtful comments and suggestions on this thesis. My gratitude also goes to my colleagues in the Scientific Computing Laboratory for their friendship and invaluable support. I also appreciate the support I received from the Natural Sciences and Engineering Research Council of Canada.

Last but not the least, my deepest gratitude goes to my family for their endless love and support during my study here in Canada. Most especially, I would like to thank my dear husband, JungHoon, who always encouraged me as a friend although my study here meant I will be away from him for two years. Without him, this thesis would never have been completed.

Dedication

This is dedicated to my parents, parents-in-law and husband.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Image registration is the process of finding the transformation that aligns two or more images of the same object, taken at different times, from different sensors or from different perspectives. Although the images are collected from different coordinate systems, the transformation maps the points in one image to their corresponding points in the other images. As a result of registration, we can compare and distinguish the differences between images in the same coordinate system. Image registration is a common and fundamental step in image analysis and is used in various areas such as remote sensing, medical imaging, cartography, and computer vision, to name a few. During the last decades, a broad range of methods for image registration has been developed. However, the majority of methods are designed to register only two images at a time in a pairwise fashion.

Some researchers have worked on registering multiple images at the same time – called groupwise registration or ensemble registration. However, their methods are applied only to mono-modality images for specific applications such as creating an atlas in medical imaging. Recently, a clustering method that simultaneously registers several multi-sensor images was proposed [1]. This ensemble clustering method successfully performed the multi-sensor ensemble registration, and demonstrated that the registration results are more robust and accurate than pairwise methods.

The first objective of this thesis is to apply the ensemble clustering method to non-rigid registration scenarios. Therefore, this thesis works on registering non-rigid multi-sensor ensembles as an extension of the ensemble clustering method. This problem is one of the most complicated and challenging problems, and to the best of our knowledge, its solution has never been demonstrated in the literature.

Although the speed of computing has been continuously increasing, the need

for decreasing the computational time of registration methods still exists. The size, resolution and dimensionality of image data are still growing, and the complexity of registration methods is also increasing to achieve more robust and accurate registration. Thus, we consider a parallelization of the ensemble clustering method to improve its speed. The second objective of this thesis is to implement the ensemble clustering method for a Graphics Processing Unit (GPU). GPUs are a popular parallelization tool for general computing applications and provide some high-level programming languages. We use the CUDA (Compute Unified Device Architecture) programming model developed by NVIDIA to implement the method, and compare the performance of the parallel implementation with the non-parallel version.

This thesis is organized as follows. In Chapter 2, we review several multi-sensor and non-rigid registration approaches proposed in the literature. After discussing some drawbacks of pairwise methods, we explore the ensemble registration. Chapter 3 is a summary of the ensemble clustering method suggested by [1]. In Chapter 4, we present our non-rigid ensemble registration method and explain B-splines as our deformation model, regularization for a smooth deformation, our cost function, and its optimization processes. We investigate an ill-conditioning problem encountered when applying the ensemble clustering method to multi-sensor non-rigid registration, and suggest a solution using the matrix approximation and the singular value decomposition (SVD). To demonstrate our method and compare it with pairwise methods, an experiment is performed with three synthetic datasets. The registration results are given and discussed. In Chapter 5, General-Purpose computing on the GPU (GPGPU) and the CUDA programming environment are examined. We discuss our GPU implementation, experiments and their results. Finally, we conclude and describe future work in Chapter 6.

Chapter 2

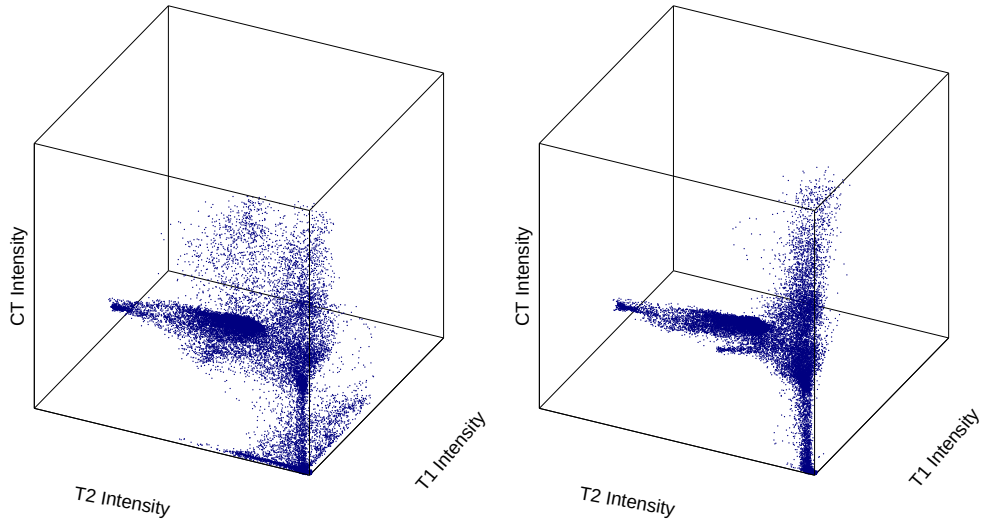
Background

2.1 Multi-Sensor Registration

Multi-sensor or multi-modality registration is the task of registering images of the same scene acquired from the different imaging devices or sensors. For example, in medical imaging we can obtain images of a body part using different modalities such as magnetic resonance imaging (MRI), computed tomography (CT), positron emission tomography (PET) and ultrasound. Other examples of multi-sensor images include satellite imagery acquired by different sensors, and several images of an object taken with different illuminations.

Multi-sensor registration is a challenging problem. Because multi-modal images have different intensity characteristics, in order to register them, we cannot use simple similarity measures such as sum of squared difference (SSD) or cross correlation (CC) in image intensities. Nevertheless, some common characteristics in the image intensities enable us to recognize that the images describe the same object or scene. This consistent correspondence between image intensities is called the *intensity mapping*. For instance, bones are rendered as white in CT, and black in MRI. Although bones have a different intensity composition in the two images, we can find similarities between the two images based on global shape. This intensity relationship between images can be visualized by the joint intensity scatter plot.

Consider registering two multi-modal images (CT and MRI) of an object. If the CT image is overlaid on top of the MR image, each pixel on the overlaid images is expressed as a 2-tuple point. Its first component is the intensity value of the pixel in the CT image and the second component is the corresponding intensity value of the same pixel in the MR image. A cloud of points is created by plotting these points for all pixels in a 2-dimensional space, where each axis represents intensity values



(a) Before Registration (b) After Registration

Figure 2.1: The change of dispersion in the JISP. The joint intensities of three images (T1-MRI, T2-MRI and CT) are plotted. (a) is the JISP before registration and (b) is the JISP after registration. The dispersion in the JISP is much decreased in (b).

from one of the images. We call this the *Joint Intensity Scatter Plot* (JISP). In the JISP, the intensity mapping between images is expressed as several coherent clusters or swaths of scatter points. A coherent cluster in the JISP often corresponds to an object in the images. For example, bone, muscle, and fat can each be represented by a cluster in the JISP of an MR/CT combination. The coherence of clusters in the JISP is disturbed as two images are moved out of registration. As a result, pixels belonging to different clusters are mixed and the coherence of clusters is broken.

Therefore, reducing the dispersion of scatter points in the JISP, or increasing the coherence of clusters in the JISP, means that two images are getting registered. The more compact each cluster is, the more accurately the two images are registered. The objective of most multi-modal registration methods is to reduce the dispersion in the JISP. Figure 2.1 shows the change of dispersion in the JISP before and after registration of three images. To quantify dispersion in the JISP, mutual information (MI) [2][3], correlation ratio (CR) [4] and normalized mutual information (NMI) [5] are often used as cost functions for multi-sensor registration. The methods using MI and NMI assume that the relationship between image intensities is probabilistic, and the correlation ratio methods assume that the corresponding intensities have a functional relationship [6]. All of them are maximized at registration and work well for both rigid and non-rigid registration.

2.2 Non-Rigid Registration

One of the basic classifications of registration methods is rigid versus non-rigid registration. Rigid registration is a method that uses only combinations of linear transformations such as translation, rotation and sometimes scaling to register images. While rigid registration takes account of only global transformations, non-rigid registration involves a broader class of transformations that allows locally deformed images (which cannot be registered only by global transformations). Figure 2.2 shows examples of rigid and non-rigid registration. Since many organs and tissues are deformable and their images do not conform to a rigid or even an affine approximation, non-rigid registration plays an important role in medical imaging. Consider registering two brain images taken from different individuals. In this inter-subject case, the images to be registered have geometric differences that cannot be matched only by rigid-body transformations. Moreover, even chest and abdomen images taken from the same individual at different times may involve non-linear transformations, because they contain highly deformable tissues. Common applications of non-rigid registration are to compare anatomy between individuals, to track changes due to growth, surgery or disease, to correct soft-tissue deformation caused by surgery, and to correct imaging warp artifacts.

The non-rigid registration methods can be classified by the transformation model that defines how images are deformed [6]. A widely used transformation model is the family of splines, including thin-plate splines (TPS) and B-splines. The thin-plate spline was established as a mathematical interpolator and introduced to the image analysis community by Goshtasby [7] and Bookstein [8]. The thin-plate spline methods treat images as a flat metal sheet that is fixed at control points in the image. Based on the location of these control points, a non-rigid transformation for the entire image is computed using the thin-plate spline as the interpolant. A drawback of the TPS is that it has a global influence on the transformation; if a control point is disturbed, then all other points in the image change. Moreover, as the number of control points increases, the computational cost rises steeply. In contrast, B-splines have a local support; when a control point is perturbed, its influence only reaches the neighborhood of the control point. B-spline based non-rigid registration techniques [9] are popular and widespread because of the B-spline's good approximation properties, computational efficiency and local support.

Another transformation model for non-rigid registration is the elastic model [10].

Elastic registration views the deforming images as an elastic body and deforms the images, using external forces to stretch the images, and internal forces to keep the deformations smooth, until the forces reach equilibrium. The registration is accomplished at the minimum energy state. However, when the images have extensive and highly localized deformations, it is hard to get good registration results with this method. To overcome this disadvantage, a viscous fluid model has been proposed [11]. Fluid registration models images as a thick fluid so that their internal forces (stress) relax as the images deform over time. Unlike the elastic method, the smoothness of the images in the fluid method does not proportionally increase as the deformation increases. As a result, this method can handle severe and large localized deformation of images.

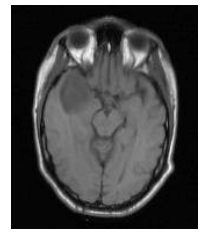
The optical flow based approaches that were originally developed in the computer vision and artificial intelligence community try to estimate the motion between two successive frames in image sequences by using the optical flow constraint equation [12]. This method is based on the assumption that the intensity value at a point in the image is uniform over small time increments. This "constant intensity" assumption is invalid for multi-sensor image pairs so that optical flow based methods have not been widely used for multi-sensor registration. For a more detailed discussion of non-rigid registration, we refer the reader to [6, 13, 14, 15].

2.3 Ensemble Registration

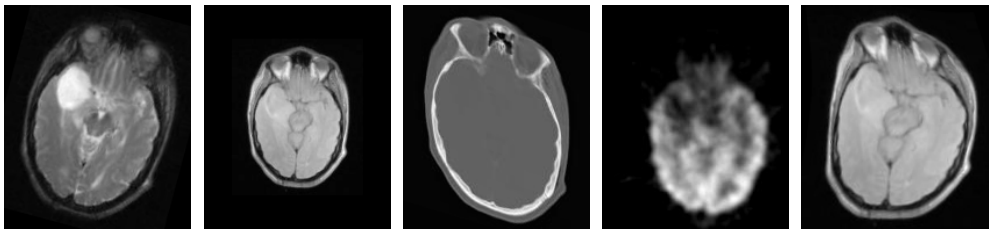
In this subsection, we discuss "Ensemble Registration" [16, 1], the task of registering more than two images simultaneously, rather than in a pairwise way. An ensemble is a set of images that contains some common content. For instance, the five images with the different modalities in Fig. 2.2 form an ensemble, as do a set of satellite images taken from different sensors.

2.3.1 Drawbacks of Pairwise Methods

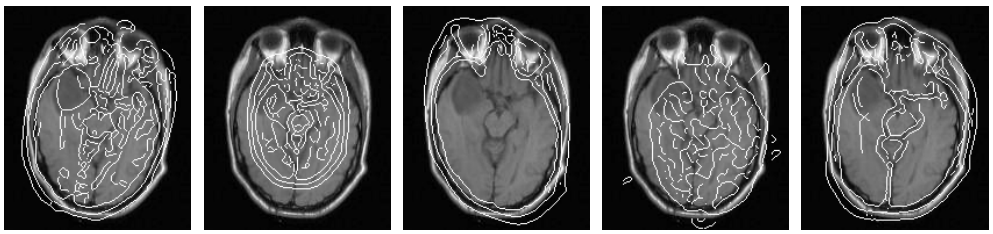
The most general approach for registering multiple images is applying some chosen registration method to two images at a time and repeating it until all the images can be brought into the same frame of reference. However, this pairwise approach can lead to problems. The first one is *selection dependency*. The registration results can be different depending on which pairs of images are selected. For example, if we try to register several multi-modal brain images (T1-MRI, T2-MRI, PD-MRI,



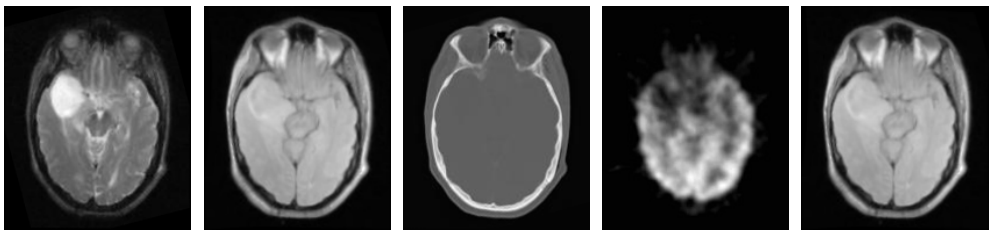
(a) Target Image



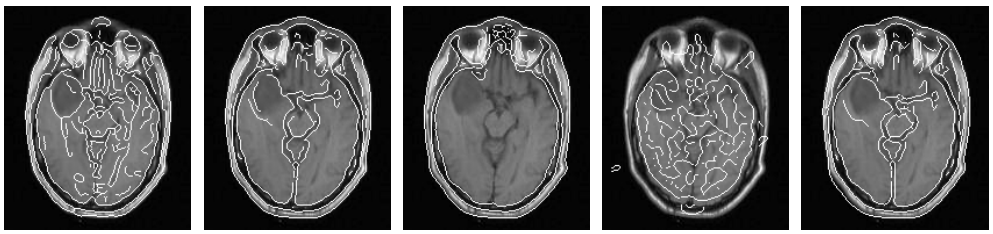
(b) Rotation (c) Scaling (d) Shearing (e) Translation (f) Non-Rigid



(g) Before Registration



(h) Registered Images



(i) After Registration

Figure 2.2: Examples of multi-sensor registration. (a) T1-MR image used for target image. (b) - (f) Transformed Images. (g) Edges of each deformed images superimposed on target image before registration. (h) Results of rigid and non-rigid registration. (i) Edges of each registered images superimposed on target image after registration. T2-MRI, PD-MRI, CT, PET and PD-MRI images (from the first colum of each row).

CT and PET) in a pairwise fashion, we need to choose which pairs of images to register. In general, the registration among different types of MR images gives good results. One should avoid registering image pairs that share very little information such as CT and PET.

Another drawback is that pairwise methods may yield a solution that lacks *internal consistency*. For example, suppose we have registration results to register image A to B, and B to C, and derive a transformation to register image A to C by composing these two registration results. In this case, we cannot say that the derived transformation for image A to C is exactly the same as the transformation that we could get if we simply register image A to C. Which transformation is correct? How do we reconcile these differences?

As a solution for these problems, a global strategy that registers all images simultaneously was proposed [16, 1]. While pairwise methods can use only a fraction of the available images at a time, the new method can use full information of all images at the same time. As a result, there is no need to choose image pairs for registration and we can use more information for registration so that the registration results become more accurate and internally consistent. We call this approach *Ensemble Registration*.

2.3.2 Ensemble Registration for Multi-Sensor Images

The first demonstration of ensemble registration was performed by Woods et al. in 1998 [17]. In order to get the complete internal consistency of the registration, they use the sum of squared differences (SSD) as cost function, which is computed by all possible pairwise SSDs. The resulting transformations are the completely internally consistent set of transformations, called "reconciled mean transformations". However, this method is only suitable for mono-sensor image registration due to the use of the SSD.

Some groupwise registration methods [18, 19] have recently emerged in the literature, motivated by computational anatomy and computational morphometry. These methods register collections of images in a population to a chosen reference anatomy to create an average shape – called an atlas. These methods avoid the need to choose a reference subject, and instead simultaneously register all subjects to a group archetype image. As a result, the methods can overcome drawbacks of pairwise registration and construct an unbiased atlas of the population. However, these methods have been demonstrated only on mono-modal images. Their

applicability to multi-sensor ensembles is unlikely.

For multi-modal ensemble registration, we can use the methods based on MI and NMI, both popular similarity measures for multi-modal registration. However, ensemble registration using MI and NMI is problematic because these methods require the joint histogram. As the number of images to be registered increases, the number of histogram bins increases exponentially. For example, for the case of registering five images, if we use $256\,(2^8)$ intensity bins per image the number of histogram bins becomes $2^{8*5} = 2^{40}$ (over 1 trillion). Therefore, the joint histogram based methods are not suitable for multi-modal ensemble registration.

Recently, a clustering method for multi-sensor ensemble registration was proposed by Orchard and Mann [1]. Without the need to form the joint histogram, this method minimizes the dispersion in the JISP by two main processes, *density estimation* and *motion adjustment*. First, the density of the scatter points in the JISP is estimated and then, while holding the density estimation fixed, the scatter points in the JISP are moved toward increasing the likelihood. The two processes are performed iteratively until the movement of the scatter points is converged. The experiments in [1] show that the results of this clustering method are robust and more accurate than the pairwise methods. In this thesis, we call this method the *ensemble clustering method*. The detail of this method is discussed in Chapter 3, because we use this method as our main tool to solve our problem, non-rigid registration of a multi-sensor ensemble of images.

2.3.3 Extension of the Ensemble Clustering Method

Although image registration has a long history and many methods have been proposed in the literature, many challenging problems still exist: "Registering of images with complex non-linear and local distortion, multimodal registration, and registering N-D images (where $N > 2$) belong to the most challenging tasks at this moment" [15]. The ensemble clustering method successfully deals with one of the most challenging tasks at this moment, multi-sensor registration, even complicating it by tacking ensembles of images, not just two at a time. Currently, we regard the ensemble clustering method as the best solution for the general purpose multi-sensor ensemble registration. This method can be used for mono- and multi-sensor image registration and it works in both pairwise and groupwise modes. However, this method is demonstrated only for rigid and affine transformations. Here, we consider adding another challenging registration problem, non-linear registration,

to the ensemble clustering method. Namely, how does the ensemble clustering method work with a large number of degrees of freedom in the transformation? The combined problem is registering a non-rigid multi-sensor ensemble of images. To solve this problem, this thesis works on registering non-rigid multi-sensor ensembles and it is an extension of the ensemble clustering method. This problem is one of the most complicated and challenging problems and to the best of our knowledge, its solution has never been demonstrated in the literature.

We give another quotation from [15]: "The major difficulty of N-D image registration resides in its computational complexity. Although the speed of computers has been growing, the need to decrease the computational time of methods persists. The complexity of methods as well as the size of data still grows (the higher resolution, higher dimensionality, larger size of scanned areas)." This statement is very true for registering non-rigid multi-sensor ensembles. The number of transformation parameters is larger than the rigid-body registration, and the number of images is also larger than the pairwise registration method. Therefore, we consider the parallelization of the ensemble clustering method to reduce the computation time.

In summary, this thesis extends the ensemble clustering method in two directions: non-rigid transformations, and parallelization.

Chapter 3

Multi-Sensor Ensemble Registration by Clustering

This section mainly summarizes the ensemble clustering method [1]. Understanding this method is an essential part because this thesis is an extension of it and we use the theory and algorithm of it as our basis. In brief, the ensemble clustering method models the density of scatter points in the JISP, forming a probability density function using a Gaussian Mixture Model. Generally speaking, one Gaussian component is used to model each cluster in the JISP. When the scatter points migrate toward the cluster centres, they move to a region of greater probability. The process of registering the ensemble of images amounts to moving the scatter points in an attempt to maximize their total probability, or likelihood. The ensemble clustering method derives a density estimate of the JISP, and the motion parameters, in tandem. These two separate, but coupled, processes are iterated until convergence.

3.1 Cost Function

Consider registering three images by using the ensemble clustering method. Each pixel corresponds to a point in the 3-D JISP, where each axis in the joint intensity space represents intensity values from one of the images. A scatter point in the JISP is expressed as a 3-tuple vector whose elements are intensity values of pixels in each image. We denote this intensity vector for pixel x as $I_x \in \mathbb{R}^D$ (D is the number of images).

The ensemble clustering method uses a statistical approach to model these scatter points in the JISP. We assume that there exists several clusters in the JISP.

Then each cluster has its own distribution and it can be expressed as a probability density function (pdf). We denote these pdfs for each cluster as ϕ, that is the density estimation of the scatter points in the JISP. As the registration proceeds, the scatter points move in the JISP as the images themselves undergo spatial transformations. The motion parameters that specify these transformations are stored in θ. Then, if we assume that the pixels in the JISP are spatially independent, a cost function can be written as a function of ϕ and θ,

$$L(\phi, \theta) = \prod_x p\left(I_x^\theta \mid \phi\right),$$

where p is a probability function and I_x^θ denotes the intensity vector for pixel x after applying the spatial transformation with parameters θ. The expression $L(\phi, \theta)$ is the probability of observing the set of intensity vectors, given the distribution specified by ϕ. Thus, the goal of the method is to maximize the likelihood cost function $L(\phi, \theta)$ by appropriate choice of ϕ and θ. For the simplicity of calculation, the method uses the logarithm of L so that the product over x turns into a sum,

$$\log L(\phi, \theta) = \sum_x \log p\left(I_x^\theta \mid \phi\right). \tag{3.1}$$

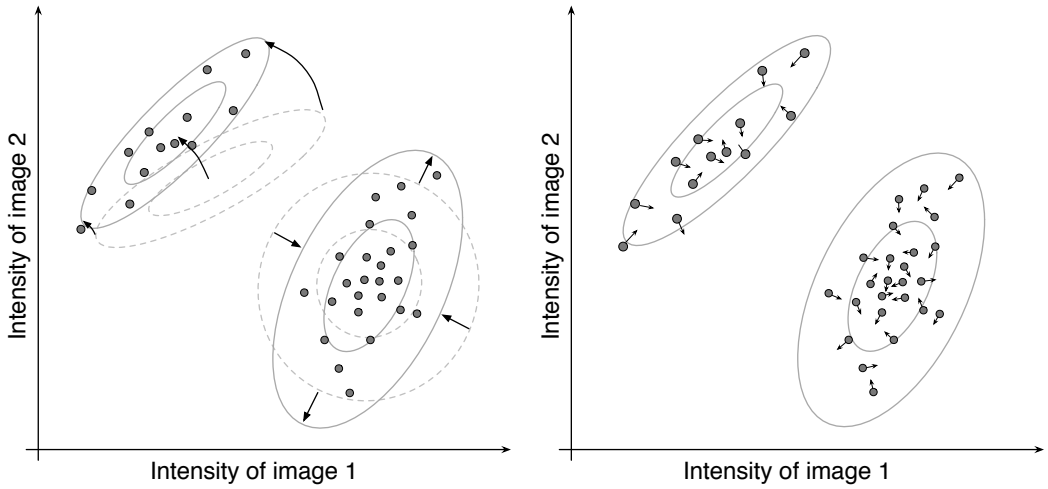
To maximize $\log L(\phi, \theta)$, the ensemble clustering method iteratively performs two processes until converging on values of ϕ and θ .

3.2 Gaussian Mixture Model (GMM)

The ensemble clustering method uses a Gaussian Mixture Model (GMM) [20] to model the density of scatter points in the JISP. The GMM is a probabilistic model for density estimation and consists of a mixture of a number of probability density functions, usually Gaussian, with different means and covariances. By applying GMM to the JISP, the clusters in the JISP are modeled as a mixture of K Gaussian components, each specified by a mean, μ_k, and a covariance matrix, Σ_k. Therefore, for a single pixel location x, the likelihood of observing the intensity vector I_x^θ is

$$p\left(I_x^\theta \mid \phi\right) = \sum_{k=1}^{K} \pi_k N\left(I_x^\theta; \mu_k, \Sigma_k\right) \tag{3.2}$$

where the kth Gaussian component is specified by μ_k and Σ_k, and π_k are the component weights, with $\sum_k \pi_k = 1$. The function N denotes the normal distribution,



(a) Density Estimation (b) Motion Adjustment

Figure 3.1: Two main processes of Ensemble Clustering Method. In density estimation, the motion parameters are held fixed while a better density estimate is computed by moving and stretching the cluster density components, as shown (a). In motion adjustment, the density estimate is held fixed and the optimal motion is determined using least-squares. As the images move, the corresponding scatter points move toward the cluster centres (on average), as shown in (b). This figure is from [1].

$$N\left(I_x^\theta;\mu,\Sigma\right)=\frac{\exp\left(-\frac{1}{2}\left(I_x^\theta-\mu\right)^T\Sigma^{-1}\left(I_x^\theta-\mu\right)\right)}{\sqrt{\left(2\pi\right)^D\left|\Sigma\right|}} \tag{3.3}$$

3.3 Two Processes

To maximize the log-likelihood cost function (3.1), the ensemble clustering method alternately performs two optimization processes. The first process is *density estimation* that models the density of the scatter points in the JISP by optimizing the cost function with respect to ϕ. The second process is *motion adjustment* that moves the images to minimize the dispersion of the JISP by optimizing the same cost function with respect to θ. Figure 3.1 describes the two processes pictorially.

3.3.1 Density Estimation

While holding the motion parameters θ fixed, this process iteratively attempts to improve the density estimate ϕ by using the *expectation-maximization* (EM) algorithm [20]. The EM algorithm has two steps; expectation and maximization.

The expectation step finds the membership of each intensity vector in the JISP among K clusters. The membership of pixel I_x^θ to cluster k is

$$\tau_{kx} = \frac{\pi_k N\left(I_x^\theta \mid \phi_k\right)}{\sum_k \pi_k N\left(I_x^\theta \mid \phi_k\right)} \qquad \text{where } \sum_k \tau_{kx} = 1 \tag{3.4}$$

The maximization step re-estimates the cluster components, μ_k, Σ_k and π_k. It means each cluster is moved according to the optimal density estimate. The new estimated cluster components are given by

$$\mu_k' = \frac{\sum_x \tau_{kx} I_x^\theta}{\sum_x \tau_{kx}}, \tag{3.5}$$

$$\Sigma = \frac{\sum_x \tau_{kx}\left(I_x^\theta - \mu_k'\right)\left(I_x^\theta - \mu_k'\right)^T}{\sum_x \tau_{kx}}, \tag{3.6}$$

$$\pi_k' = \frac{\sum_x \tau_{kx}}{\sum_k \sum_x \tau_{kx}}. \tag{3.7}$$

3.3.2 Motion Adjustment

While holding the density estimation ϕ fixed, this process finds a motion increment that moves all the scatter points toward increasing the log-likelihood cost function. The ensemble clustering method uses a Newton-type method to optimize the cost function. The following derivation outlines the process to find a small increment to the motion parameters, $\tilde{\theta}$ (called a nudge), at each iteration. It is done by optimizing the cost function, $\log L\left(\phi, \theta\right) = \sum_x \log p\left(I_x^\theta \mid \phi\right)$, with respect to the parameters θ. To optimize the cost function, we set its gradient vector to zero.

$$\frac{\partial}{\partial\theta}\log L\left(\phi,\theta\right) = \frac{\partial}{\partial\theta}\left(\sum_x \log p\left(I_x^\theta \mid \phi\right)\right) = \sum_x \frac{\frac{\partial}{\partial\theta} p\left(I_x^\theta \mid \phi\right)}{p\left(I_x^\theta \mid \phi\right)} = 0 \tag{3.8}$$

The probability function p and the normal distribution N in the gradient vector of $\log L$ are replaced by their definitions, (3.2) and (3.3) from the GMM. For notational brevity, $N_k\left(I_x^\theta\right)$ is used instead of $N\left(I_x^\theta\left(x\right); \mu_k, \Sigma_k\right)$.

$$\frac{\partial}{\partial\theta}\log L\left(\phi,\theta\right) = \sum_x \frac{-1}{p\left(I_x^\theta \mid \phi\right)} \sum_{k=1}^{K} \pi_k N_k\left(I_x^\theta\right) \frac{\partial I_x^\theta}{\partial\theta} \Sigma_k^{-1}\left(I_x^\theta - \mu_k\right) = 0 \tag{3.9}$$

To compute a small increment of motion parameters, $\tilde{\theta}$, I_x^θ is replaced with a linear approximation of a nudged version of the images, i.e. $I_x^{\theta+\tilde{\theta}} \cong I_x^\theta + \frac{\partial I_x^\theta}{\partial \theta}^T \tilde{\theta}$.

$$\begin{aligned}
\frac{\partial}{\partial \theta} \log L\left(\phi, \theta\right) &\cong \frac{\partial}{\partial \theta} \log L\left(\phi, \theta + \tilde{\theta}\right) \\
&= \sum_x \frac{-1}{p\left(I_x^\theta \mid \phi\right)} \sum_{k=1}^K \pi_k N_k\left(I_x^\theta\right) \frac{\partial I_x^\theta}{\partial \theta} \Sigma_k^{-1} \left(I_x^\theta + \frac{\partial I_x^\theta}{\partial \theta}^T \tilde{\theta} - \mu_k \right) \\
&= 0 \qquad (3.10)
\end{aligned}$$

As the final step, factoring out $\tilde{\theta}$ and collecting the remaining terms from (3.10), we get

$$\begin{aligned}
&\left(\sum_x \frac{1}{p\left(I_x^\theta \mid \phi\right)} \sum_{k=1}^K \pi_k N_k\left(I_x^\theta\right) \frac{\partial I_x^\theta}{\partial \theta} \Sigma_k^{-1} \frac{\partial I_x^\theta}{\partial \theta}^T \right) \tilde{\theta} \\
&\qquad = \left(\sum_x \frac{1}{p\left(I_x^\theta \mid \phi\right)} \sum_{k=1}^K \pi_k N_k\left(I_x^\theta\right) \frac{\partial I_x^\theta}{\partial \theta} \Sigma_k^{-1} \left(I_x^\theta - \mu_k\right) \right) \qquad (3.11)
\end{aligned}$$

or, more concisely,

$$A\tilde{\theta} = b. \qquad (3.12)$$

Notice that if there are D images and each image had M motion parameters, then the nudged version of motion parameters $\tilde{\theta}$ in (3.12) has MD motion parameters. The matrix A in (3.12) is the $MD \times MD$ system matrix and b in (3.12) is the $MD \times 1$ vector. For instance, if we register five images with three motion parameters per image ($D = 5$ and $M = 3$), then the problem turns into solving 15 linear equations containing 15 unknowns. The solution of the linear equations, $\tilde{\theta}$, is the optimal motion increment and it is used for adjusting the current estimate for θ.

Chapter 4

Non-Rigid Multi-Sensor Ensemble Registration by Clustering Method

In this Chapter, we present a method to register concurrently non-rigid multi-sensor ensembles based on the clustering method explained in Chapter 3. As an extension of the clustering method [1], our method mostly follows the framework of it, but a different transformation model is used and supplementary parts required for successful non-rigid registration are added. First, the deformation model and the regularization for non-rigid transformations are described and a cost function according to them is defined. We explain the problem that we encounter during the optimization process and how it is solved. Some implementation issues and results and discussion of experiments are given in this chapter.

4.1 Deformation Model

For the transformation model, we choose a free-form deformation (FFD) model based on B-splines. The FFD was originally suggested by Sederberg and Parry as a powerful tool for modeling 3-D deformable objects [21]. The FFD deforms an object by manipulating a 3-D parallelpiped lattice in which the object is embedded. As the lattice is deformed, the object is consistently deformed according to the deformation of the lattice. Instead of using the Bernstein polynomials for the deformation function, the bivariate cubic B-spline tensor product was used for the deformation function of FFD [22, 23]. In contrast to thin-plate splines or elastic-body splines, B-splines support local control that limits the effect of the deformation within neighborhoods of control points. This property results in computational efficiency even for a large number of control points.

Combining FFD and B-splines, we consider a 2-D image overlaid on a control lattice. We can deform the image by manipulating the underlying mesh of control points. By moving the control points independently of each other, the space between them is deformed non-rigidly so that the image on the lattice is also deformed according to the deformation of the control lattice. This deformation produces smooth and C^2 continuous transformations [22, 23].

Now, we define our deformation model by following Rueckert et al.'s formulation [9]. We denote the 2-D domain of our image as $\Omega = \{(x, y) \mid 0 \leq x < X, 0 \leq y < Y\}$. Let $c_{i,j}$ be the value of the ij^{th} control point on a control point grid (CPG) Ψ , a lattice of uniformly spaced control points. The spacing between the control points in x and y directions are denoted by δ_x and δ_y , respectively. Then at any position $\mathbf{x} = (x, y)$, the deformation D can be written as the 2-D tensor product of 1-D cubic B-splines together with the control points

$$D(x) = \sum_{l=0}^{3} \sum_{m=0}^{3} B_l(u) B_m(v) c_{i+l,j+m} \tag{4.1}$$

where $i = \lfloor x/\delta_x \rfloor - 1$, $j = \lfloor y/\delta_y \rfloor - 1$, $u = x/\delta_x - \lfloor x/\delta_x \rfloor$, $v = y/\delta_y - \lfloor y/\delta_y \rfloor$ and where B_l represents the l^{th} cubic B-spline basis function

$$\begin{aligned} B_0(u) &= (1 - u^3)/6 \\ B_1(u) &= (3u^3 - 6u^2 + 4)/6 \\ B_2(u) &= (-3u^3 + 3u^2 + 3u + 1)/6 \\ B_3(u) &= u^3/6 \end{aligned}$$

where $0 \leq u < 1$. The transformation $D(x)$ in (4.1) defines displacement of the point $\mathbf{x} = (x, y)$ after deformation. It is computed from the values of a 4×4 neighborhood of control points surrounding the point. In (4.1), i and j are the starting index of the 4×4 neighborhood of control points for $\mathbf{x} = (x, y)$, and u and v are the relative positions of x and y, respectively, within the control point cell containing the point $\mathbf{x} = (x, y)$. Figure 4.1 shows these neighborhoods and the configuration of the CPG. We present some examples of FFD deformation in Fig. 4.2. They show the change of the control-point mesh before and after moving one or several control points. As we can see, the results of moving control points are local and smooth.

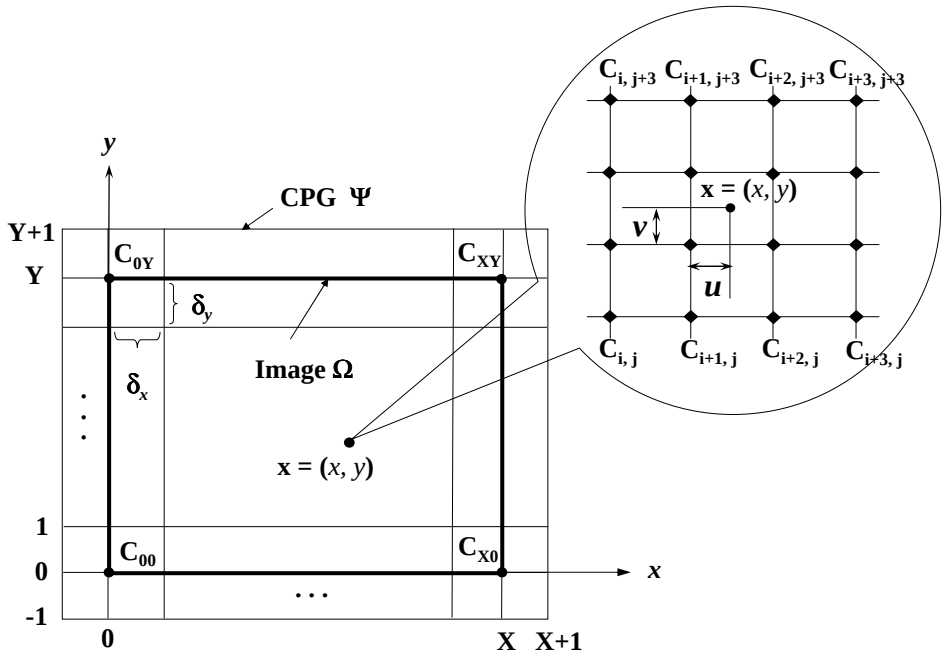
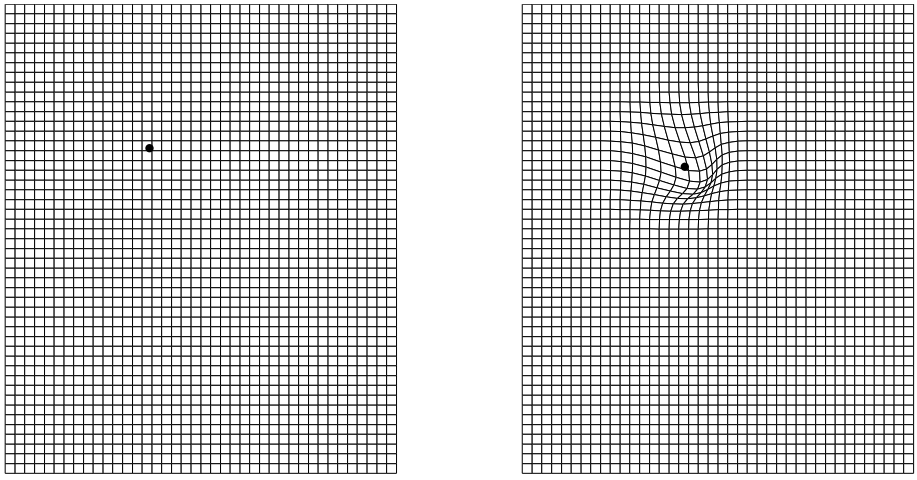
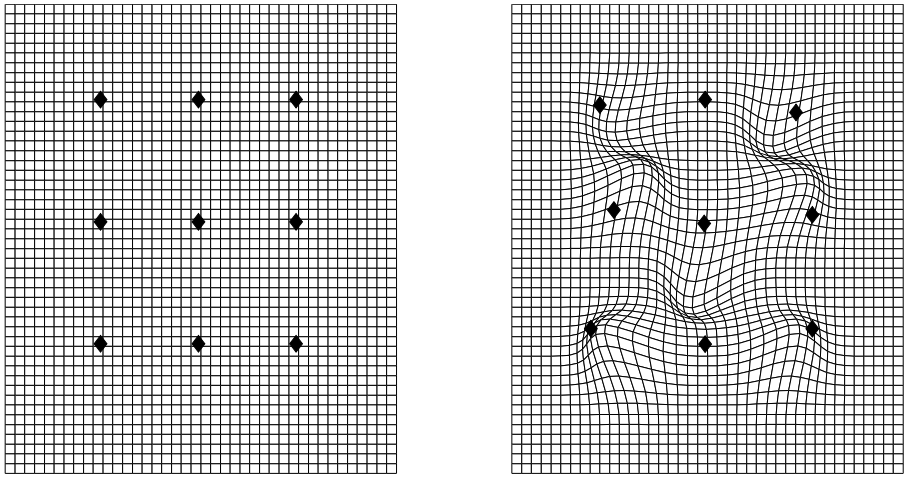


Figure 4.1: Configuration of Control Point Grid (CPG)



(a) Single Position Change



(b) Multiple Position Change

Figure 4.2: Change of CPG

4.2 Regularization

The non-rigid registration problem using high degrees of freedom in deformation is an ill-posed problem and can cause unrealistic results such as folding and tearing images. A reasonable assumption for this problem is that a physically plausible deformation should be regular and smooth or it should preserve volume and topology of images. Most applications add an additional regularization term to the cost function to achieve these goals. This term makes the problem well-posed and stabilizes the algorithm.

Our regularization term is based on the mean elastic energy – computed using four-neighborhoods of each control point. The energy E_p is defined by

$$E_p = \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{4} \|c_i - c_j\|^2 \tag{4.2}$$

where n is the total number of control points and j indexes the 4-neighborhood of a control point c_i (right, left, top and bottom). This regularization term is akin to the elastic potential energy of each control point's connections to its four neighbors.

4.3 Cost Function

Our cost function consists of two terms; the first term is the log-likelihood cost function of the ensemble clustering method (3.1), and the second term is the regularization term to constrain the image deformation (4.2). The total cost function is defined as

$$E_{total} = E_l - \lambda E_r \tag{4.3}$$

where E_l is the cost associated with the dispersion in the JISP and E_r is the cost associated with the severity of the deformation. A weighting factor λ controls the influence of the regularization term. In this formula, the second term has a negative sign, because the registration aims to maximize the log-likelihood E_l, whereas it is intended to minimize the degree of deformation E_r.

To define the first term of the cost function, following the ensemble clustering method [1], we denote a density estimation of the JISP as ϕ and a set of motion parameters that specify the displacement of images as c. Here, the set of motion parameters c corresponds to a set of deformation parameters that defines the displacement of control points on the CPG. As the control points on the CPG are

moved, images are deformed and the scatter points in the JISP are moved. If the JISP consists of D images, each point in the JISP is expressed as a D-tuple vector whose elements are the intensity values of D images for a chosen pixel. We call this vector an intensity vector and the intensity vector for pixel x is denoted as $I_x \in \mathbb{R}^D$. Then, by assuming that the pixels in the JISP are spatially independent, the log-likelihood cost function can be written as a function of ϕ and c,

$$E_l = \log L(\phi, c) = \sum_x \log p(I_x^c \mid \phi) \tag{4.4}$$

where p is a probability function and I_x^c denotes the intensity vector for pixel x after applying the deformation parameterized by c. The expression $L(\phi, c)$ is the probability of observing the set of intensity vectors, given the probability density function (pdf) specified by ϕ.

For the second term of the cost function, we use the regularization term defined in (4.2). Since the range of the function E_l is $(-\infty, 0)$, to keep the same range between the two terms of the cost function, we also take the logarithm of the regularization term. Thus, the second term of the cost function is defined as

$$E_r = \log E_p = \log \left(\frac{1}{4} \sum_{i=1}^{n} \sum_{j=1}^{4} \|c_i - c_j\|^2 \right) \tag{4.5}$$

where n is the total number of control points and c_j is the four-neighborhood of c_i (right, left, top and bottom).

4.4 Optimization

4.4.1 Modified Motion Adjustment Process

The goal of our registration method is to maximize the total cost function (4.3) by appropriate choice of a density estimation of the JISP, ϕ, and a set of deformation parameters c. To do this, two optimization processes, density estimation and motion adjustment, are alternated. The density estimation process is exactly the same as the ensemble clustering method [1], but the motion adjustment process is modified due to the deformation model and the regularization term.

The motion adjustment process is to find a motion increment that moves all images toward maximizing the total cost function (4.3). The process is the same as that described in Chapter 3 except for the additional regularization term. To

optimize the total cost function (4.3) with respect to the parameters c, we set its gradient vector to zero.

$$\frac{\partial E_{total}}{\partial c} = \frac{\partial E_l}{\partial c} - \lambda \frac{\partial E_r}{\partial c} = 0 \tag{4.6}$$

First we only consider the first term in (4.6), the gradient of log-likelihood E_l. The probability function p and the normal distribution N in the first term are replaced by their definitions, (3.2) and (3.3) from the GMM.

$$\frac{\partial E_l}{\partial c} = \frac{\partial}{\partial c} \log L(\phi, c) = \sum_x \frac{-1}{p\left(I_x^c \mid \phi\right)} \sum_{k=1}^{K} \pi_k N_k\left(I_x^c\right) \frac{\partial I_x^c}{\partial c} \Sigma_k^{-1}\left(I_x^c - \mu_k\right) \tag{4.7}$$

To compute a small increment of deformation parameters, $\widetilde{c}$ (called a nudge), I_x^c is replaced with a linear approximation of a nudged version of the images, i.e. $I_x^{c+\widetilde{c}} \cong I_x^c + \frac{\partial I_x^c}{\partial c}^T \widetilde{c}$.

$$\frac{\partial E_l}{\partial c} \cong \sum_x \frac{-1}{p\left(I_x^c \mid \phi\right)} \sum_{k=1}^{K} \pi_k N_k\left(I_x^c\right) \frac{\partial I_x^c}{\partial c} \Sigma_k^{-1} \left(I_x^c + \frac{\partial I_x^c}{\partial c}^T \tilde{c} - \mu_k \right) \tag{4.8}$$

Now we consider the second term in (4.6), the gradient of the regularization E_r. It corresponds to the gradient of log-elastic energy of the CPG (4.5).

$$\frac{\partial E_r}{\partial c} = \frac{\partial}{\partial c} \log E_p \tag{4.9}$$

As with I_x^c, we replace the log-elastic energy with a nudged version of it according to a linear approximation, i.e. $\log E_p^{c+\widetilde{c}} \cong \log E_p + \frac{\partial \log E_p}{\partial c} \widetilde{c}$. Thus,

$$\begin{aligned} \frac{\partial E_r}{\partial c} &= \frac{\partial}{\partial c} \log E_p \\ &\cong \frac{\partial}{\partial c} \left(\log E_p + \frac{\partial \log E_p}{\partial c} \widetilde{c} \right) \\ &= \frac{\partial \log E_p}{\partial c} + \frac{\partial^2 \log E_p}{\partial c^2} \widetilde{c} \\ &= \frac{\partial E_r}{\partial c} + \frac{\partial^2 E_r}{\partial c^2} \widetilde{c}, \end{aligned} \tag{4.10}$$

where $\frac{\partial^2 \log E_p}{\partial c^2}$ is the Hessian matrix. Combining the two terms, (4.8) and (4.10),

the gradient of the total cost function (4.6) becomes

$$\begin{aligned}\frac{\partial E_{total}}{\partial c} &\cong \sum_x \frac{-1}{p\left(I_x^c \mid \phi\right)} \sum_{k=1}^{K} \pi_k N_k\left(I_x^c\right) \frac{\partial I_x^c}{\partial c} \Sigma_k^{-1} \left(I_x^c + \frac{\partial I_x^{c\,T}}{\partial c}\,\tilde{c} - \mu_k\right) - \lambda\left(\frac{\partial E_r}{\partial c} + \frac{\partial^2 E_r}{\partial c^2}\widetilde{c}\right) \\ &= 0\end{aligned}$$

After expanding the brackets and reorganizing the remaining terms, we get the linear system.

$$\left(\sum_x \frac{1}{p\left(I_x^c \mid \phi\right)} \sum_{k=1}^{K} \pi_k N_k\left(I_x^c\right) \frac{\partial I_x^c}{\partial c} \Sigma_k^{-1} \frac{\partial I_x^{c\,T}}{\partial c} - \lambda \frac{\partial^2 E_r}{\partial c^2}\right) \tilde{c}$$

$$= \left(\sum_x \frac{1}{p\left(I_x^c \mid \phi\right)} \sum_{k=1}^{K} \pi_k N_k\left(I_x^c\right) \frac{\partial I_x^c}{\partial c} \Sigma_k^{-1} \left(I_x^c - \mu_k\right)\right) + \lambda \frac{\partial E_r}{\partial c} \qquad (4.11)$$

or, more concisely,

$$\begin{aligned}\left(A - \lambda G\right)\tilde{c} &= b + \lambda \frac{\partial E_r}{\partial c} \\ W\tilde{c} &= q \qquad (4.12)\end{aligned}$$

where $G = \frac{\partial^2 E_r}{\partial c^2}$, $A - \lambda G = W$ and $b + \lambda \frac{\partial E_r}{\partial c} = q$. Solving the linear system for $\widetilde{c}$ gives the optimal increment of deformation parameters, according to the linear approximations. The increment is used to adjust the current estimate for c.

4.4.2 Ill-Conditioning and Newton's Method

The objective of the second optimization process, motion adjustment, is to find the optimal increment of deformation parameter, $\widetilde{c}$, by solving the linear system in (4.12). If the number of images we register is D and the number of deformation parameters per image is M, then the size of matrix W in (4.12) is $MD \times MD$ and q is an $MD \times 1$ vector. Solving MD linear equations gives us a set of motion parameters $\widetilde{c}$, an $MD \times 1$ vector. The matrix W is largely composed of the derivatives of the image intensities with respect to each deformation parameter. Thus, when the derivatives of the image intensities are very small, the matrix becomes ill-conditioned. We explain this problem with a small linear system example. We want to solve the following linear system for x:

$$Ax = b$$

where

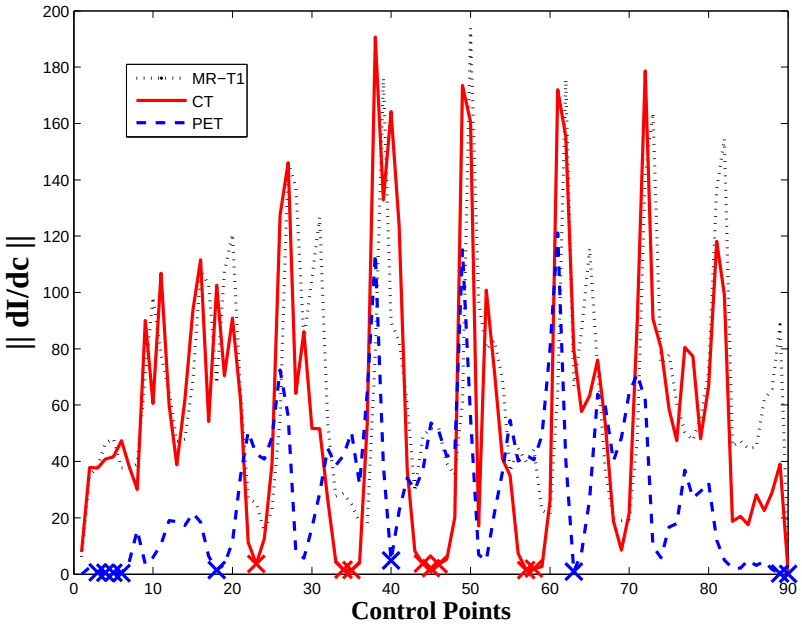
$$A = \begin{bmatrix} 0.4045 & 0.00001 & 3.2338 & 2.1708 & 1.1850 \\ 3.6088 & 0.00002 & 1.0560 & 0.3578 & 0.7074 \\ 3.8390 & 0.00001 & 2.6076 & 4.3746 & 3.9849 \\ 1.3853 & 0.00002 & 1.5130 & 3.8198 & 4.3767 \\ 2.5760 & 0.00001 & 3.8857 & 3.8976 & 3.5007 \end{bmatrix}, \quad b = \begin{bmatrix} 0.1966 \\ 0.4986 \\ 0.4901 \\ 0.0233 \\ 0.2059 \end{bmatrix}.$$

This system is ill-conditioned because all the elements in the second column of the matrix A are close to zero. After solving this system for x, we get

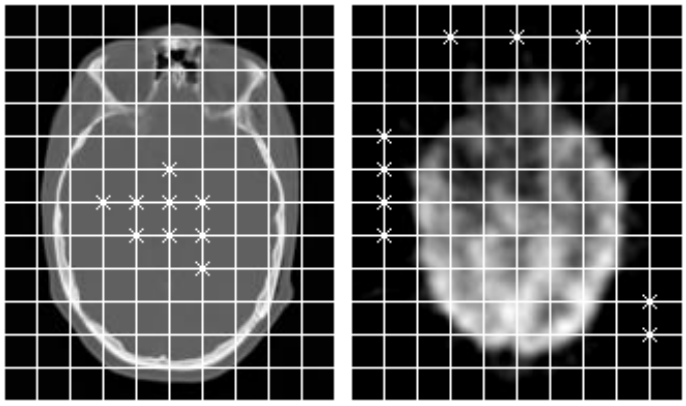
$$x = \begin{bmatrix} 0.1401 \\ 10989 \\ -0.0911 \\ 0.3457 \\ -0.3595 \end{bmatrix}.$$

The second element of x is extremely large compared with other elements. The same problem occurs when we solve the linear system (4.12). For many situation, Newton's method can converge to the root of a function very rapidly. However, when the derivative of the function is very small, the problem becomes ill-conditioned resulting in extremely large Newton steps. We encounter this problem when registering a CT or PET image with other modality images. For the case of CT images, the intensity values inside the brain are nearly homogeneous so that their derivatives are very small. As a result, some nudge values become extremely large and the control mesh is intensely distorted. Figure 4.3 shows the derivatives of three images according to each control point. For the case of CT and PET images, several very small derivative values are observed. The control points that give very small derivatives are marked in Fig. 4.3 (a) and the corresponding positions of control points are marked on the CT and PET image in Fig. 4.3 (b). The marks are located at the inside of brain for CT and in the background for PET. These parts of the images do not give enough information to find the direction for registering images.

To resolve this problem, we reduce the matrix W in (4.12) by using the singular value decomposition (SVD) [24]. The following theorem describes the SVD.



(a) Derivatives of 3 Images



(b) CT and PET image marked with control points

Figure 4.3: Derivatives of Image Intensity according to Control points

Theorem 1. Any $m \times n$ matrix A of rank r, with $m \leq n$, can be factored

$$A = U\Sigma V^T,$$

where $U = [u_1 \mid \cdots \mid u_m]$ is an $m \times m$ matrix, Σ is an $m \times n$ diagonal matrix of singular values, and $V = [v_1 \mid \cdots \mid v_n]$ is an $n \times n$ matrix such that , $U^T U = UU^T = I_m$ and $V^T V = VV^T = I_n$ with the singular values arranged in decreasing order,

$$\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \ldots \geq \sigma_m \geq 0.$$

The columns of U and V are called singular vectors and the diagonal elements σ_i are called singular values. The next theorem explains how to approximate the matrix A using the SVD in Theorem 1.

Theorem 2. Assume that the $m \times n$ matrix A has rank r. Its SVD is $U\Sigma V^T$(where U, Σ and V follow Theorem 1). Let Z be an approximation of the matrix A and its rank is $k < r$. Then the matrix approximation problem

$$\min_{\text{rank}(Z)=k} \|A - Z\|_2$$

has the solution

$$Z = A_k := U_k \Sigma_k V_k^T,$$

where $U_k = [u_1 \mid \cdots \mid u_k]$, $V_k = [v_1 \mid \cdots \mid v_k]$, and the $k \times k$ diagonal matrix Σ_k is

$$\begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_k \end{bmatrix}.$$

In essence, the matrix approximation A_k can be found by re-multiplying $U_k \Sigma_k V_k^T$. Instead of trying to solve $Ax = b$ when A is ill-conditioned, we solve a better-conditioned nearby system, $A_k x = b$. Applying the two theorems above to approximately solve $Ax = b$, the solution x can be calculated by

$$x = V_k \Sigma_k^{-1} U_k^T b.$$

For our small example, the SVD of the matrix A is

$$U = \begin{bmatrix} -0.2794 & 0.3062 & 0.7071 & 0.0750 & -0.5680 \\ -0.2046 & -0.8447 & 0.0972 & -0.3920 & -0.2855 \\ -0.5952 & -0.2204 & -0.2282 & 0.7382 & -0.0126 \\ -0.4664 & 0.3707 & -0.5721 & -0.4481 & -0.3420 \\ -0.5552 & 0.0820 & 0.3336 & -0.3082 & 0.6919 \end{bmatrix},$$

$$\Sigma = \begin{bmatrix} 12.5200 & 0 & 0 & 0 & 0 \\ 0 & 3.3704 & 0 & 0 & 0 \\ 0 & 0 & 2.6210 & 0 & 0 \\ 0 & 0 & 0 & 0.4618 & 0 \\ 0 & 0 & 0 & 0 & 0.00001 \end{bmatrix},$$

$$V = \begin{bmatrix} -0.4164 & -0.9036 & -0.0658 & 0.0760 & 0.000001 \\ 0 & 0 & 0 & 0 & -1 \\ -0.4421 & 0.1197 & 0.8488 & -0.2642 & 0.000006 \\ -0.5774 & 0.3365 & -0.1197 & 0.7342 & -0.00001 \\ -0.5457 & 0.2364 & -0.5108 & -0.6208 & 0.00001 \end{bmatrix}.$$

To approximate the matrix A, we choose $k = 4$, because the last singular value of Σ is relatively very small. The solution is

$$x = V_k \Sigma_k^{-1} U_k^T b$$

$$= \begin{bmatrix} 0.1523 \\ -0.000005 \\ -0.0144 \\ 0.1466 \\ -0.1757 \end{bmatrix}$$

Now, the remaining problem is how many singular values should be removed from Σ, namely, how to choose k. Here, we seek a balance between accuracy and stability. Including more singular values increases the accuracy, but at the risk of decreased stability. If the improvement in accuracy is too small, then the risk of including another singular value is not worth it. Figure 4.4 describes this relationship between the singular values. The following theorem gives a very useful property of the SVD [24].

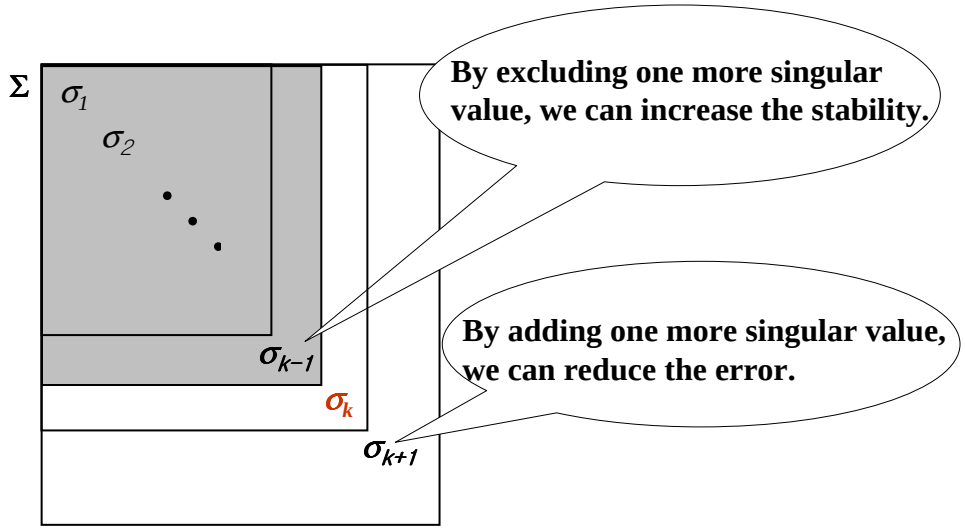


Figure 4.4: The relationship between the singular values in Σ

Theorem 3 let the SVD of $A \in \mathbb{R}^{m \times n}$ be given by Theorem 1. If $k < r = \text{rank}(A)$ and

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T,$$

then

$$\|A - A_k\|_2 = \sigma_{k+1}$$

This theorem means that the difference between the original matrix A and the estimated matrix A_k corresponds to the $k+1^{th}$ singular value. Each singular value, σ_k, is the difference between the original matrix A and the estimated matrix A_{k-1}. Therefore, we examine the change of this difference for all singular values, and choose the singular value at the moment when the difference becomes small. If the singular values are, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_m \geq 0$, then we search for the first singular value σ_i that satisfies

$$\sigma_i - \sigma_{i+1} < threshold$$

where $i = 1, 2, 3, \ldots, m-1$. The value of *threshold* is chosen experimentally.

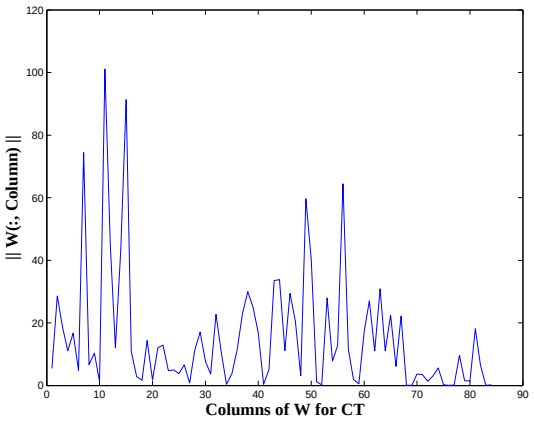
Figures 4.5 and 4.6 show the effect of this approach for CT and PET images. To generate these figures, we concurrently registered five images (MR-T1, MR-T2, MR-PD, CT and PET) by using the clustering method. We used an 7×6 control point grid with 32 pixel spacing. Since the CPG has 2 degrees of freedom (x and y axis), the number of deformation parameters per image is $7 \times 6 \times 2 = 84$. Thus,

the size of the matrix W in (4.12) was 420×420 (84×5 images). To approximate the matrix W, the value of *threshold* used was 0.001 and the reduced rank of the matrix was $k = 192$. The first plot (a) of each figure shows the norm of each column in the matrix W. The second plot (b) shows the nudge values and the image grid (sub-sampled for every 4 pixels) deformed by the nudges. The third plot (c) shows the nudge values and the deformed image grid after reducing the matrix W. The columns in the matrix W with small norm values produce extremely large nudges. Clearly, after applying the matrix approximation, the magnitude of the nudges decreases and the image grids become smoother.

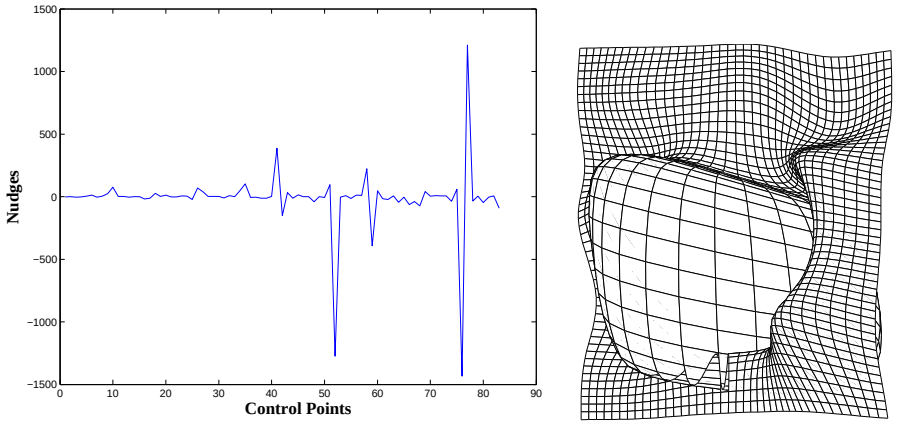
4.4.3 Regularization versus Matrix Approximation

Here, we look into the influence of the regularization and the matrix approximation, and their relationship. Both methods make the problem well-posed and stable. While the regularization maintains the topology and limits the severity of the image deformations, the matrix approximation avoids extreme deformations of the image caused by low-definition regions of the image. In order to explore the influence of each factor, and their relationship, we consider three cases; examining each term separately, and combining them together. For this experiment, we simultaneously registered five images (MRI-T1, MRI-T2, MRI-PD, CT and PET) by using our ensemble clustering method. A 7×6 control-point grid with 32 pixel spacing was used and the result images were taken at the 2^{nd} iteration. First, we applied only the regularization term to register the images without the matrix approximation. The influence of the regularization depends on the constant λ in the cost function (4.3). Thus, we varied the value of λ and observed the results. Figure 4.7 shows the deformed image grids (sub-sampled for every 4 pixels) according to changes of the constant λ. There is little change as we vary the value of λ. The extreme deformation of CT and PET images are not avoided only by the regularization.

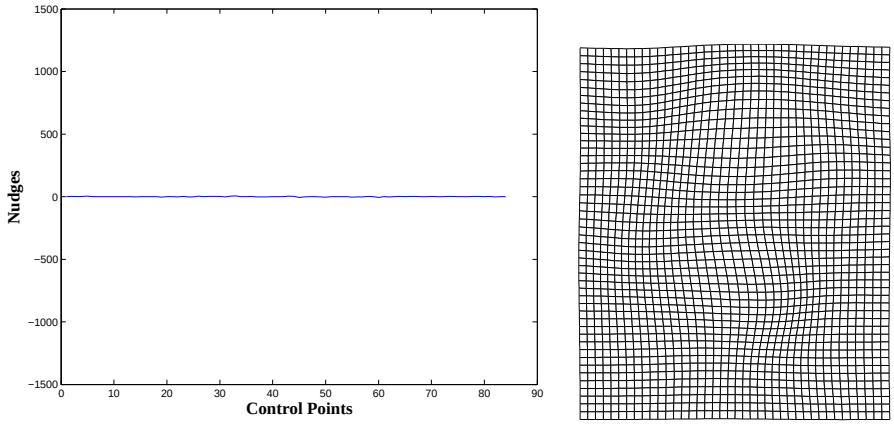
Second, we applied only the matrix approximation to the images without the regularization term. Since the outcome of the matrix approximation varies depending on the *threshold* values, we look into the results by changing the *threshold*. Note that a lower *threshold* increases the rank of the matrix approximation. In Fig. 4.8, as the *threshold* gets bigger, the displacement of the control points decreases. Because many singular values are excluded for solving equation (4.12), there is little movement of the control points on the CPG. To contrast, the small threshold makes it possible to contain more singular values so that it produces more movements of



(a) The norm of each column in matrix W

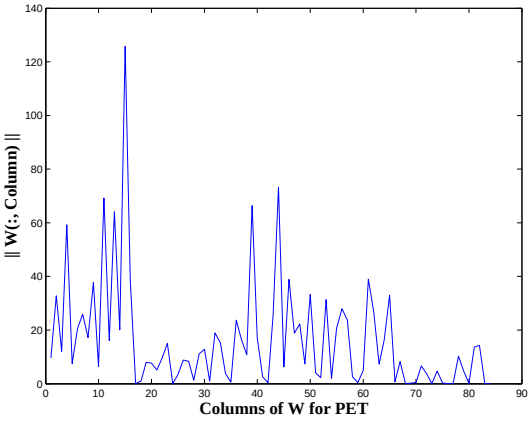


(b) Before applying matrix approximation

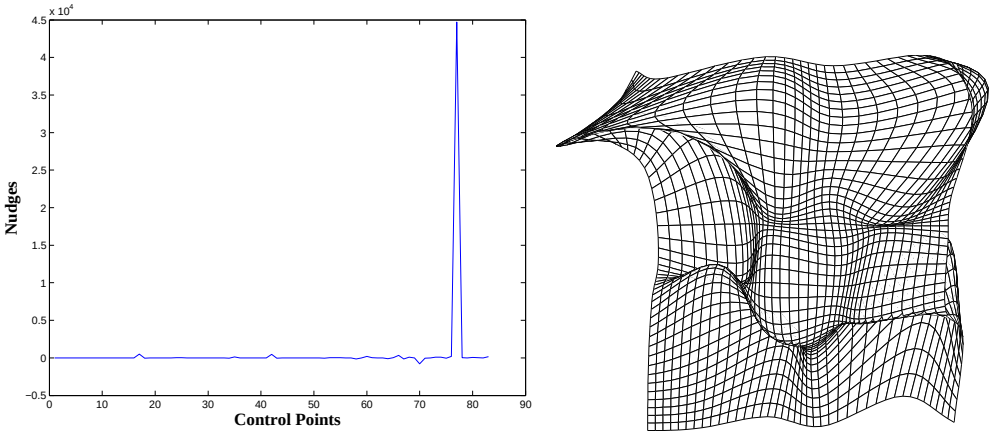


(c) After applying matrix approximation

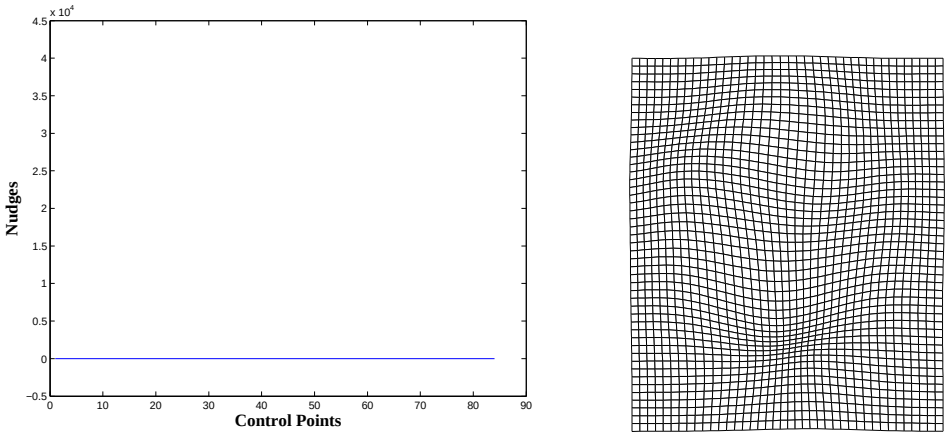
Figure 4.5: Ill-Conditioning Problem - CT image



(a) The norm of each column in matrix W



(b) Before applying matrix approximation



(c) After applying matrix approximation

Figure 4.6: Ill-Conditioning Problem - PET image

the control points on the CPG. From this result, the matrix approximation can be considered as another regularization method.

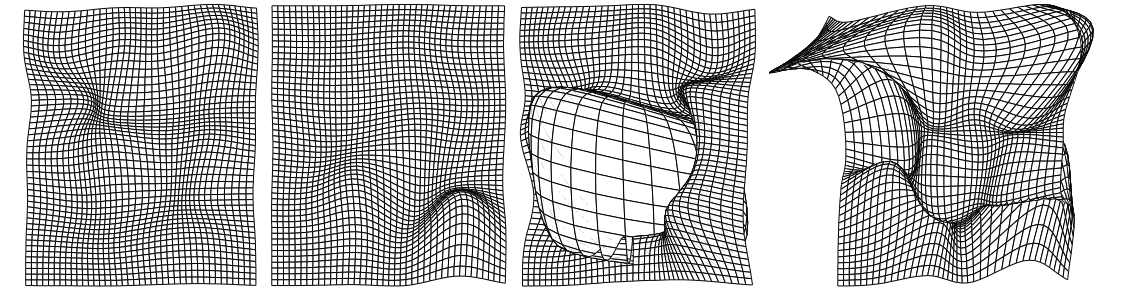
Last, based on the above results, we chose a *threshold* value of 0.0001, because it is the 1^{st} value that removed the drastic discontinuities. To test the behaviour of our regularization in conjunction with our matrix approximations, we fixed the *threshold* of 0.0001 value and changed the value of λ for regularization. Figure 4.9 shows the results. The result of this case seems regular and well-behaved. While the matrix approximation removes the excessive deformation, the regularization adjusts the movement of the control points to economize based on elastic potential energy. Therefore, we can conclude that combining the regularization with the matrix approximation is the an appropriate choice for our method.

4.5 Multi-Resolution Approaches

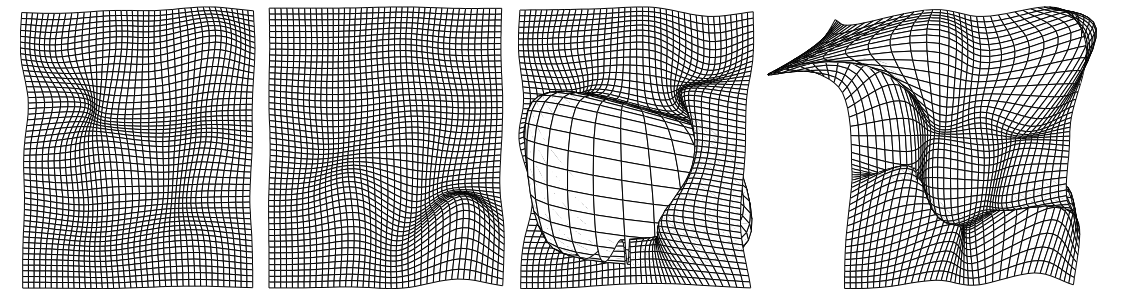
To avoid local minima and to decrease computation time, a multi-resolution framework is popularly used for image registration. The registration is first done for the down-sampled images using Gaussian blurring, and then the results at the lower resolution are used as the initial guess at the higher resolution. This procedure is iterated until the highest resolution is reached.

In our optimization, the multi-resolution strategy is used in two ways. First, the image resolution is changed into three different scales: $20\%, 50\%$ and 100%. Second, the resolution of the CPG is also changed by decreasing the spacing of the CPG. While a large spacing of CPG allows global non-rigid deformation, a small spacing allows highly-local non-rigid deformation. At the same time, the decreasing of the spacing of CPG leads to an increase in the number of degrees of freedom and computational complexity.

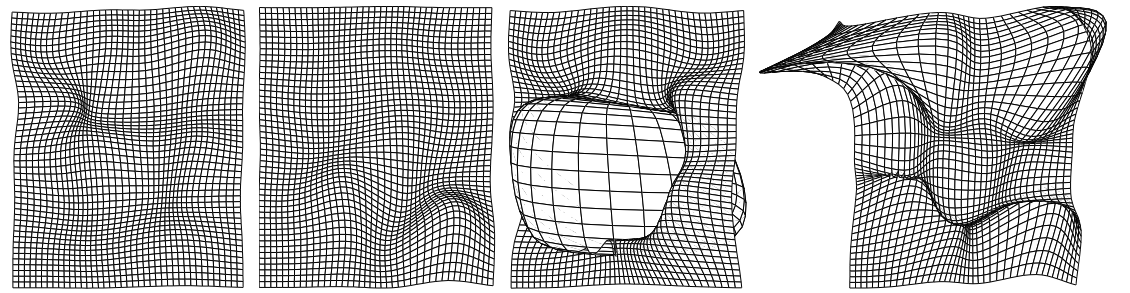
Combining two multi-resolution schemes, our algorithm uses four resolution levels. For the image resolution, the scale of images is changed to 20%, $50\%, 100\%$ and 100%. For the resolution of the CPG, the spacing of the CPG is changed to $64, 32, 32$, and 16 pixels. These spacing numbers refer to the distance, in pixels, between adjacent control points using the original resolution of the image, so that the combination of 32 spacing and 50% resolution means the control points are 16 pixels apart in the down-sampled image. The registration starts with a 20% scaled image with the CPG spaced 64 pixels, and it successively uses the higher resolution images and the finer CPGs. At the final level, the original scale of images with the CPG spaced 16 pixels is registered. In order to get the finer CPG at each level, we



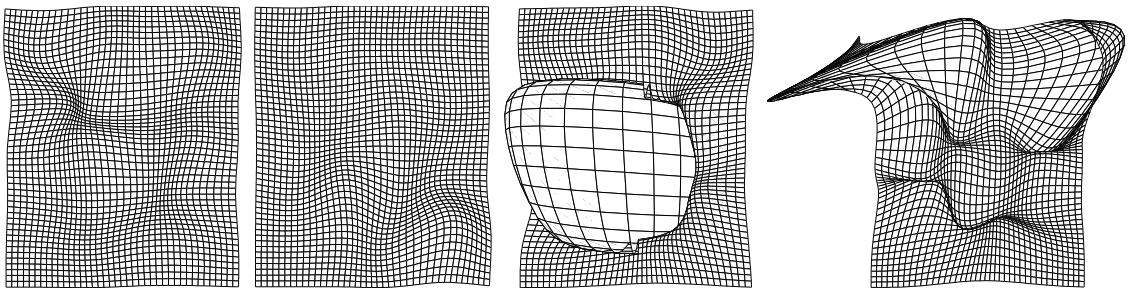
(a) $\lambda = 0$



(b) $\lambda = 0.1$

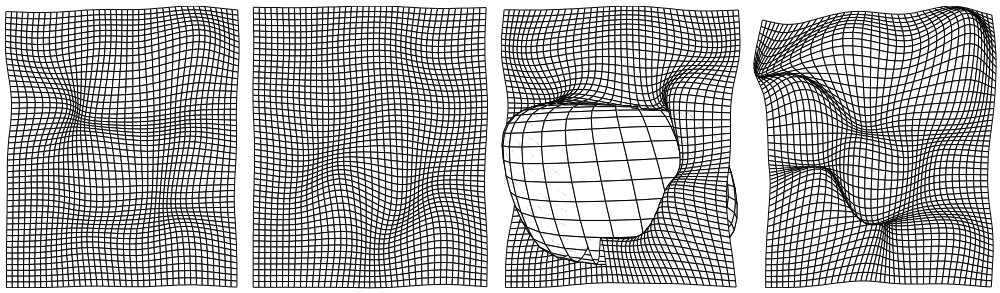


(c) $\lambda = 1$

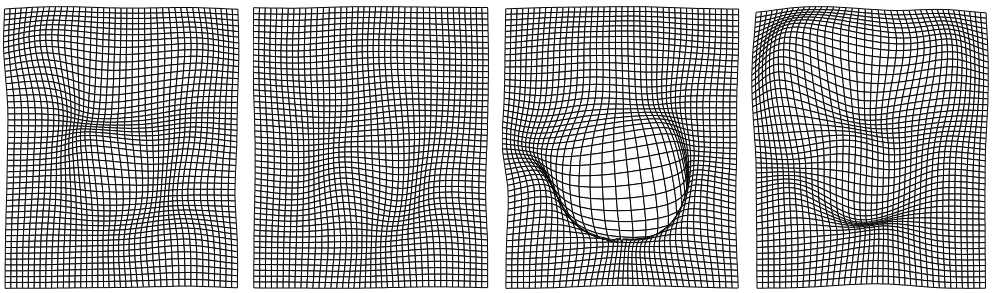


(d) $\lambda = 10$

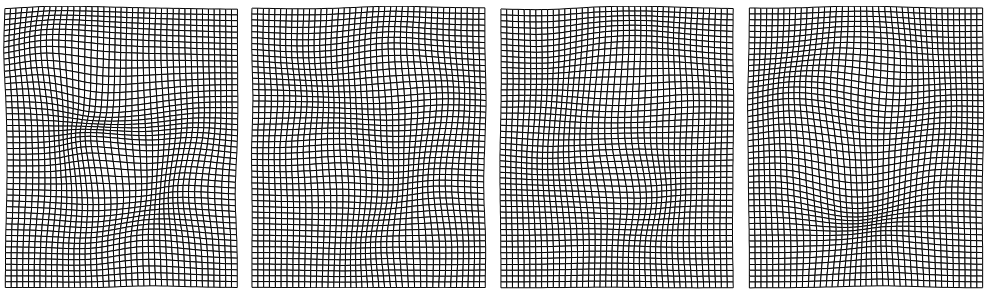
Figure 4.7: The image grids according to the change of regularization term. From left to right, MR-T2, MR-PD, CT and PET images grids.



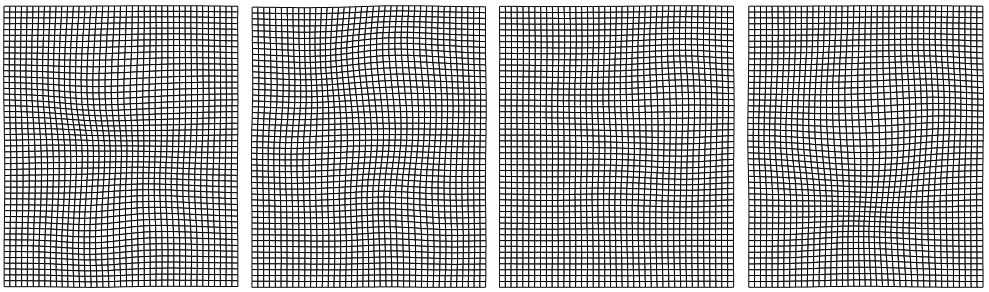
(a) $threshold = 0.00001$, $\text{rank}(W) = 357$



(b) $threshold = 0.0001$, $\text{rank}(W) = 314$

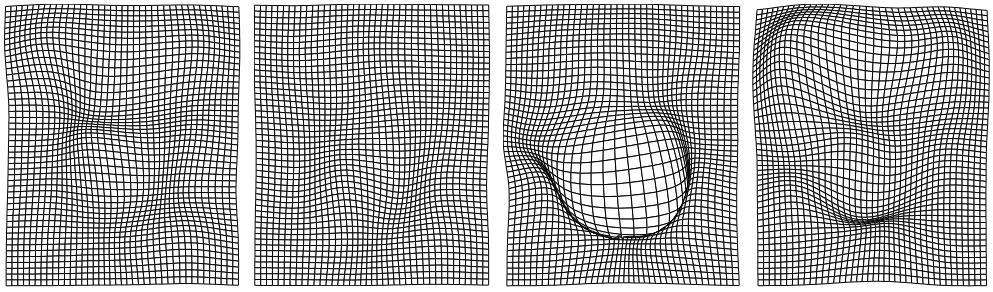


(c) $threshold = 0.001$, $\text{rank}(W) = 192$

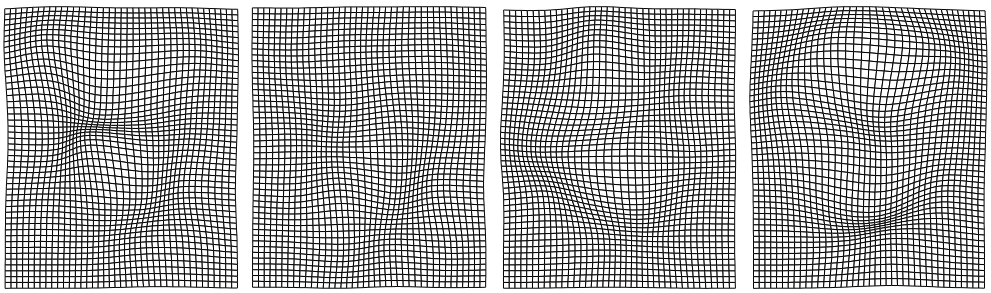


(d) $threshold = 0.01$, $\text{rank}(W) = 96$

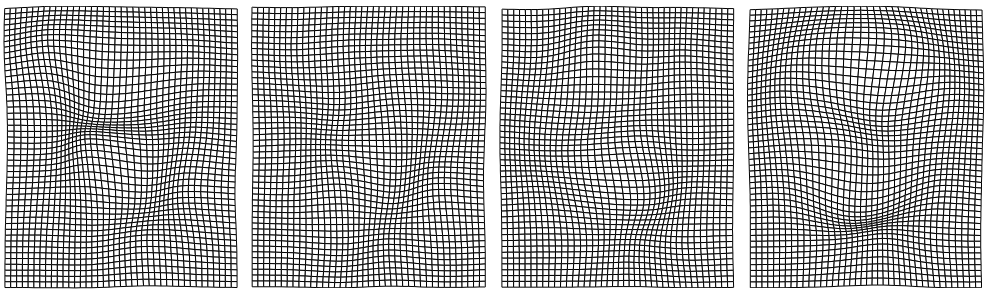
Figure 4.8: The image grids according to the change of matrix approximation term. Before applying the matrix approximation, $\text{rank}(W) = 420$. From left to right, MR-T2, MR-PD, CT and PET images grids.



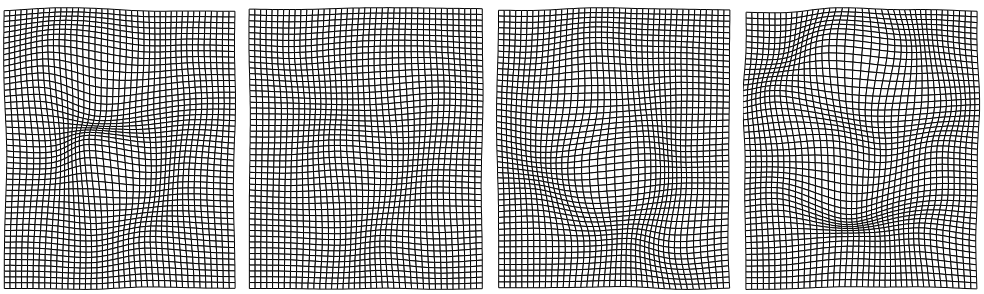
(a) $threshold = 0.0001$, $\lambda = 0$, $\text{rank}(W) = 314$



(b) $threshold = 0.0001$, $\lambda = 0.1$, $\text{rank}(W) = 261$



(c) $threshold = 0.0001$, $\lambda = 1$, $\text{rank}(W) = 321$



(d) $threshold = 0.0001$, $\lambda = 10$, $\text{rank}(W) = 420$

Figure 4.9: The image grids according to the change of regularization and matrix approximation terms. Before applying the matrix approximation, $\text{rank}(W) = 420$. From left to right, MR-T2, MR-PD, CT and PET images grids.

halve the spacing of CPG and the value of new control points is calculated from the control points at the previous level, using a B-spline subdivision algorithm [25].

4.6 Summary of Algorithm

Algorithm 1 shows the summary of our algorithm.

Algorithm 1 : Non-rigid Ensemble Clustering Registration

Input: initial ensemble I_0
Input: initial deformation parameters C_0
Input: initial GMM parameters ϕ
for each resolution level i **do**
    $I_{scaled} \longleftarrow$ scale ensemble I_0
    $C_i \longleftarrow$ increase the resolution of CPG using C_{i-1} if needed
    $I \longleftarrow$ apply deformation C_i to ensemble I_{scaled}
    repeat
        $\phi \longleftarrow$EM step – density estimation process
        $\widetilde{C} \longleftarrow$ motion adjustment process
        $C_i \longleftarrow C_i + \widetilde{C}$
        $I \longleftarrow$ apply deformation C_i to ensemble I_{scaled}
    until converged ($\widetilde{C} < tolerence$)
end for
Output: I is registered ensemble at full scale
Output: C_i holds the optimal deformation parameters
Output: ϕ holds the GMM parameters

4.7 Experiments

The purpose of these experiments is to demonstrate that the ensemble clustering method can be used for non-rigid registration, and to compare the registration results with one of the pairwise methods. To achieve this, we used three different methods for non-rigid registration. The first method is our ensemble clustering method that simultaneously registers multiple images. We chose one image from a test ensemble as a reference and deformed the other images using the FFD model

based on B-splines described in section 4.1. Then, all the images in the same test ensemble were registered concurrently.

The second and third methods are pairwise registration approaches. The second is a Nelder-Mead optimization method [26] using the Normalized Mutual Information (NMI) as a cost function. This method is a direct search method that does not use numerical or analytic gradients. The NMI is one of the popular cost functions for image registration. The details of this method will be explained in section 4.7.2. The third method is using our ensemble method in a pairwise way. Namely, we registered only two images at a time using the ensemble clustering method.

All three methods used the same multi-resolution approach described in section 4.5: $[20\%, 50\%, 100\%, 100\%]$ for image scale and $[64, 32, 32, 16]$ for spacing of the CPG. The experiment used three different image sets to generate registration trials. The first dataset is the Retrospective Image Registration Evaluation (RIRE) project's training set [27]. The second dataset is taken from the BrainWeb project at the Montreal Neurological Institute (`http://www.bic.mni.mcgill.ca/brainweb`, 1997) [28]. The third dataset is a set of face images with variable illuminations from the Extended Yale Face Database B [29]. The detail of the data processing and the parameter settings are provided in later subsections.

4.7.1 Measure of Accuracy

To assess the quality of the registration, we compute the average pixel displacement error defined as

$$Err = \frac{1}{r}\sum_{i=1}^{r} \|x_i - t_r(t_f(x_i))\|^2 \tag{4.13}$$

where r is the number of pixels within the region of interest (ROI), and $\|\cdot\|$ is the standard Euclidean norm. The forward transformation, t_f, is a ground truth transformation obtained when the test dataset is generated. The reverse transformation, t_r, is the estimated transformations obtained through the registration and should (ideally) be the inverse of t_f. Thus, Err measures the difference between the gold standard transformation and the estimated transformation. A small displacement error means better registration, with perfect registration indicated by $Err = 0$.

4.7.2 Pairwise Method

The Matlab function fminsearch attempts to find the minimum of a multivariable function using a derivative-free method. It uses the Nelder-Mead simplex algo-

rithm. We used a modified version of this function taken from the Matlab File Exchange site (`http://www.mathworks.com/matlabcentral/fileexchange/5157`). The regular fminsearch in Matlab initializes the first trials very close to each other, whereas the modified fminsearch allows initializing the step size by changing the option values. The FFD model based on B-splines was used for the image transformation. For the cost function, the negative NMI with 32 bins was employed as a similarity measure and a mean elastic energy influenced by neighborhoods, E_p in (4.2) was used as a regularization term. Thus, the total cost function is defined as

$$E_{total} = E_{\text{NMI}} + \lambda E_p \tag{4.14}$$

where E_{NMI} represents the cost associated with the image similarity measure and E_p represents the cost associated with the smoothness of the deformation. The fminsearch method gives the optimal motion parameters by minimizing the total cost function. We used the same multi-resolution framework as with the ensemble clustering method: $[20\%, 50\%, 100\%, 100\%]$ for image scale and $[64, 32, 32, 16]$ for spacing of the CPG.

4.7.3 RIRE Data

The RIRE training set consists of five different modality volumes: MR-T1 weighed, MR-T2 weighed, MR-PD weighed, CT, and PET. To begin with, the five volumes were registered using the true displacement parameters supplied by the RIRE project and scaled down to $256 \times 256 \times 26$ voxels. Then the same slice was taken from each volume, and cropped to 161×193 pixels in size. A MR-T1 weighted image was chosen as a reference and the other four images were deformed using a FFD model based on B-splines. Deformation parameters were randomly generated by a normal distribution with a standard deviation of 12 pixels and a $4 \times 3 \times 2$ control point grid with 64 pixel spacing. Ten trial sets of images were generated and registered to get reliable results. The ensemble clustering method was initialized with six Gaussian components, while the pairwise clustering method was initialized with four Gaussian components. For the parameter settings, the ensemble clustering method used 0.1 for λ in the regularization term and 0.09 for the threshold to approximate the matrix. The pairwise clustering method used 1 for λ and 0.9 for the threshold. The NMI pairwise method used 0.001 for λ. Figure 4.11 (a) and (b) show the original image set and the two masks that we used. The masks were used to identify the region of the images that were to be used in the registration

process (to populate the JISP, for example). The first mask was used at the first resolution level and the second mask was used at the other resolution levels and for the evaluation of the error.

4.7.4 BrainWeb Data

The BrainWeb provides multi-modal MR volumes that are already registered: MR-T1 weighed, MR-T2 weighed and MR-PD weighed. The 90^{th} slice was extracted from each volume without any manipulation. The size of each image is 181×217 pixels and a MR-T1 weighed image was used for a reference and the other images were deformed using our deformation model. To generate ten trial cases, deformation parameters were randomly generated by a normal distribution with a standard deviation of 12 pixels and a $4 \times 3 \times 2$ control point grid with 64 pixel spacing. Both the ensemble clustering and pairwise clustering methods were initialized with four Gaussian components. For the parameter settings, the ensemble clustering method used 0.1 for λ in the regularization term and 0.07 for the threshold to approximate the matrix. The pairwise clustering method also used 0.1 for λ and 0.07 for the threshold. The NMI pairwise method used 0.001 for λ. Figure 4.16 (a) and (b) show the original image set and the two masks that we used. The first mask was used at the first resolution level and the second mask was used at the other resolution levels and for the evaluation of the error.

4.7.5 Face Image Data

This data is a set of images of the same face taken with five very different illuminations. The images were scaled down to 161×225 pixels in size and the backgrounds of the images were removed using a mask so that only the face region of the images remained. Figure 4.21 (b) shows the five face images. The registration for this set of images is challenging, because the image F1 and F5 have few illuminated features in common. The F1 image was used for a reference and the other four images were deformed by the deformation parameters generated by a random normal distribution with a standard deviation of 12 pixels. The spacing of the CPG was 64 pixels and its dimension was $4 \times 3 \times 2$ pixels. Also, ten trial sets were generated for the experiments. The ensemble clustering method was initialized with six Gaussian components, while the pairwise clustering method was initialized with four Gaussian components. For the parameter settings, the ensemble clustering method used 0.1 for λ in the regularization term and 0.09 for the threshold to approximate the

matrix. The pairwise clustering method used 1 for λ and 0.9 for the threshold. The NMI pairwise method used 0.001 for λ. Figure 4.21 (a) and (b) show the original image set and the mask we used. The mask was used for the registration at all resolution levels and also used for the evaluation of the error.

4.8 Results and Discussion

We provide registration results for the three methods for three synthetic datasets as tables (4.1, 4.2 and 4.3). The values of each cell in the tables represent the average errors over ten trial sets, and the standard deviations of the ten trial values are given inside parentheses. We provide the entire results for the ten trials in Appendix **??**. The initial errors – the initial displacement errors – are given for the purpose of comparison.

4.8.1 RIRE Data

We provide the registration results of the RIRE ensemble in Table 4.1 and in the plot in Fig. 4.10. In Table 4.1, the mean errors indicate that the ensemble clustering method performed the registration better than two pairwise methods overall. For the MR-T2 and MR-PD images, the ensemble clustering method clearly gives better performance compared with the pairwise methods. However, the results are inconclusive for the CT and PET images. We think this is because of the ill-conditioning problems due to the small derivatives of image intensities, as explained in Section 4.4.2. This problem is not perfectly remedied for CT and PET images. For the case of the pairwise clustering method, we used the same parameters (λ for regularization, and *threshold* for matrix approximation) for all pairs of images. However, the optimal parameter values are likely different for different image pairs. Thus the registration results might be improved if we tweaked the parameters for each pair. Figure 4.11 (c) shows a deformed image set taken from the 10^{th} test set and Figure 4.12, 4.13 and 4.14 show the registration results of the ensemble clustering, pairwise clustering and pairwise NMI methods, respectively.

4.8.2 BrainWeb Data

The registration results for the BrainWeb dataset are shown in Table 4.2, and plotted in Fig. 4.15. The ensemble clustering method shows the best performance results in the entire set of images. Since the BrainWeb data consists of three MR

	T1-T2	T1-PD	T1-CT	T1-PET	Mean
Initial Error	7.43 (1.51)	6.65 (1.28)	13.1 (2.66)	6.82 (2.16)	8.50 (0.96)
NMI	5.81 (1.59)	5.29 (1.66)	12.3 (3.63)	5.58 (1.97)	7.25 (1.09)
Cluster (Pairwise)	5.89 (2.36)	2.97 (0.79)	10.8 (3.87)	9.00 (3.91)	7.17 (1.52)
Cluster (Ensemble)	3.40 (0.93)	2.28 (0.43)	11.3 (1.90)	5.87 (1.66)	5.71 (0.54)

Table 4.1: Displacement Errors (standard deviation) for RIRE Data

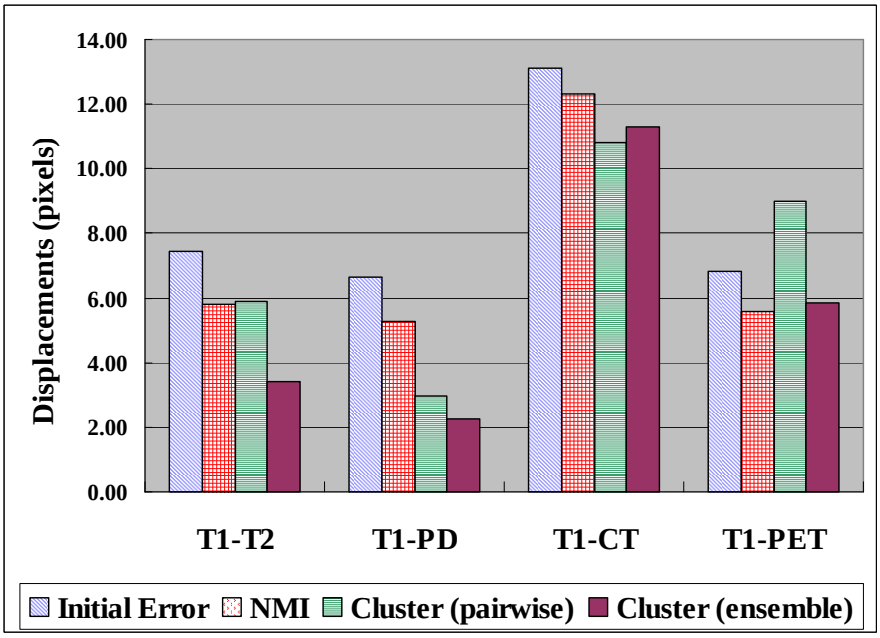
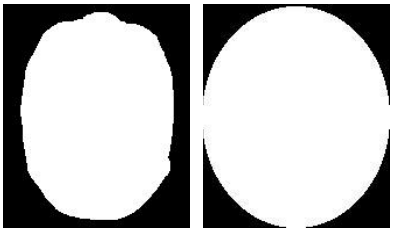
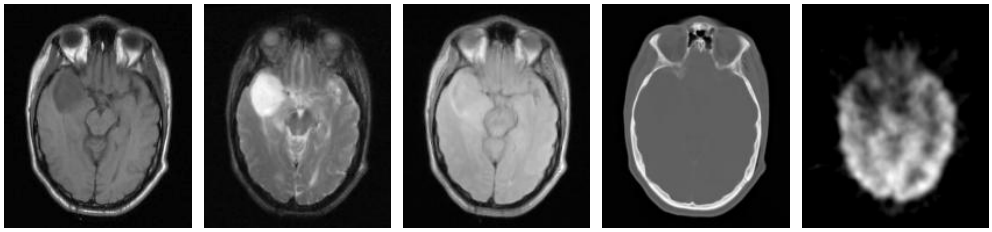


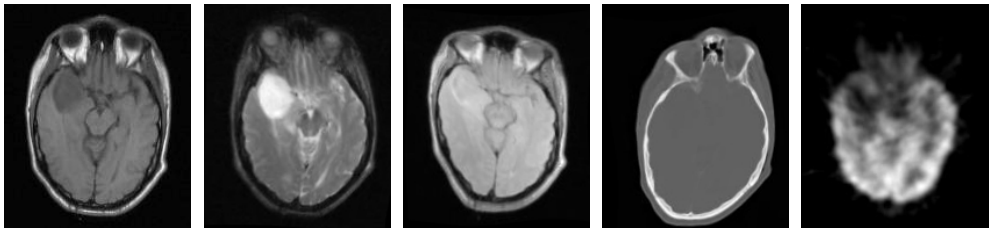
Figure 4.10: Displacement Errors for RIRE Data



(a) Masks

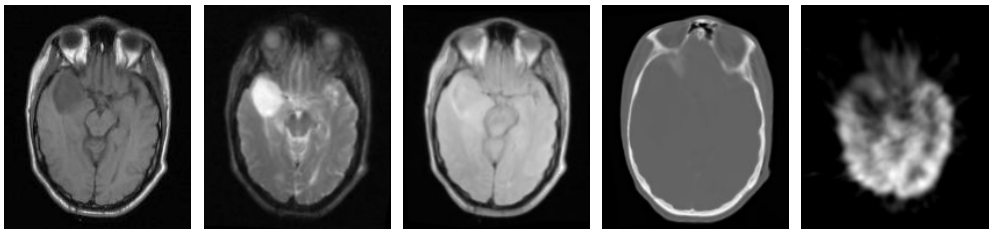


(b) Source Images: MR-T1, MR-T2, MR-PD, CT, PET

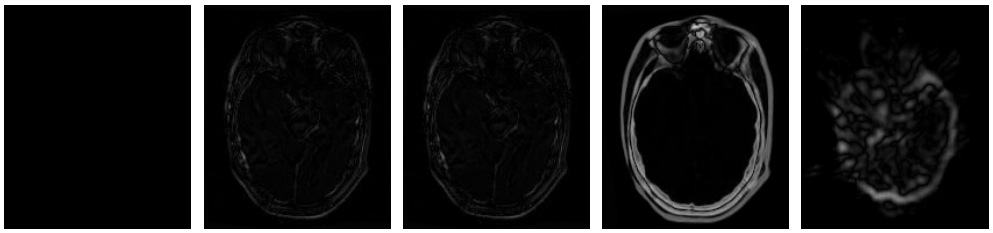


(c) Deformed Images: MR-T1, MR-T2, MR-PD, CT, PET

Figure 4.11: RIRE Data Taken from the 10^{th} Test Set.

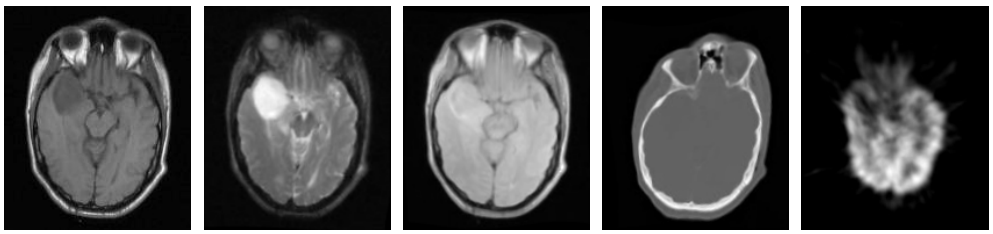


(a) Registered Images: MR-T1, MR-T2, MR-PD, CT, PET

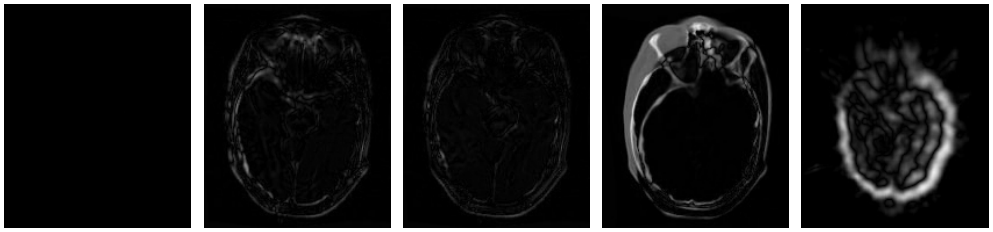


(b) Absolute Difference: MR-T1, MR-T2, MR-PD, CT, PET

Figure 4.12: Registration Results of Ensemble Clustering Method for RIRE Data Taken from the 10^{th} Test Set. (a) The registration results of the ensemble clustering method. (b) The absolute difference between the source and registered images. Rendered with the intensity range $[0-255]$.

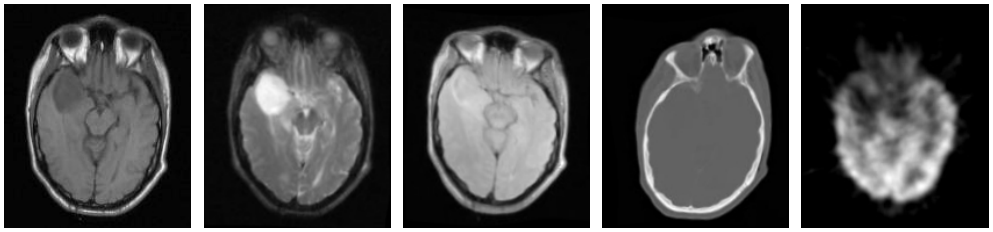


(a) Registered Images: MR-T1, MR-T2, MR-PD, CT, PET

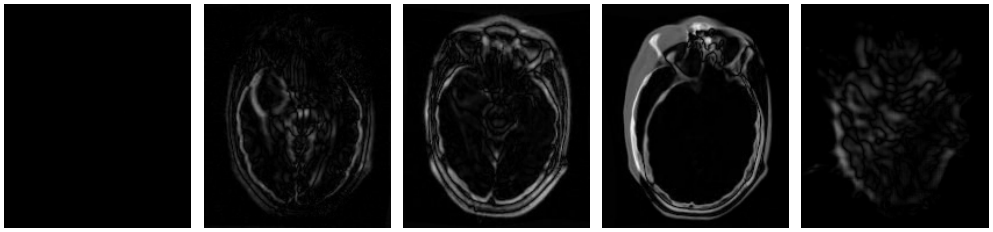


(b) Absolute Difference: MR-T1, MR-T2, MR-PD, CT, PET

Figure 4.13: Registration Results of Pairwise Clustering Method for RIRE Data Taken from the 10^{th} Test Set. (a) The registration results of the pairwise clustering method. (b) The absolute difference between the source and registered images. Rendered with the intensity range $[0-255]$.



(a) Registered Images: MR-T1, MR-T2, MR-PD, CT, PET



(b) Absolute Difference: MR-T1, MR-T2, MR-PD, CT, PET

Figure 4.14: Registration Results of Pairwise NMI Method for RIRE Data Taken from the 10^{th} Test Set. (a) The registration results of the pairwise NMI method. (b) The absolute difference between the source and registered images. Rendered with the intensity range $[0-255]$.

images (T1, T2, and PD) and these MR images tend not to have large homogeneous regions, the ill-conditioning problem seems to have less of an impact in this case. Figure 4.16 (c) shows a deformed image set taken from the 3^{th} test set and Figure 4.17, 4.18 and 4.19 show the registration results of the ensemble clustering, pairwise clustering and pairwise NMI methods, respectively.

4.8.3 Face Image Data

The registration results for the face image data are shown in Table 4.3, and plotted in Fig. 4.20. The ensemble clustering method performed the registration better than both pairwise methods except the F1/F5 pair. The images F1 and F5 have very different illumination conditions so that we would expect the pairwise methods to have little success in registering these. However, the performance of the NMI method was comparable to that of the ensemble clustering method. As it is pointed out in [1], while common content between two images ultimately gives more information for registration, extreme images (with largely disjoint content) actually yield a more compact joint histogram than images with partially overlapping content. Thus, the pairwise NMI method used the disjoint nature of the F1/F5 combination to **help** in registration. Except for this case, both pairwise methods show poor registration results and the fact that their errors are similar to the initial errors suggests that these pairwise methods actually did almost nothing. Figure 4.21 (c) shows a deformed image set taken from the 10^{th} test set and Figure 4.22 shows the registration results of the ensemble clustering, pairwise clustering and pairwise NMI methods.

	T1-T2	T1-PD	Mean
Initial Error	6.21 (2.55)	6.09 (2.82)	6.15 (2.23)
NMI	4.41 (2.29)	4.13 (2.41)	4.27 (2.11)
Cluster (Pairwise)	2.34 (1.58)	2.33 (2.22)	2.34 (1.79)
Cluster (Ensemble)	1.88 (1.19)	2.00 (1.23)	1.94 (1.14)

Table 4.2: Displacement Errors (standard deviation) for BrainWeb Data

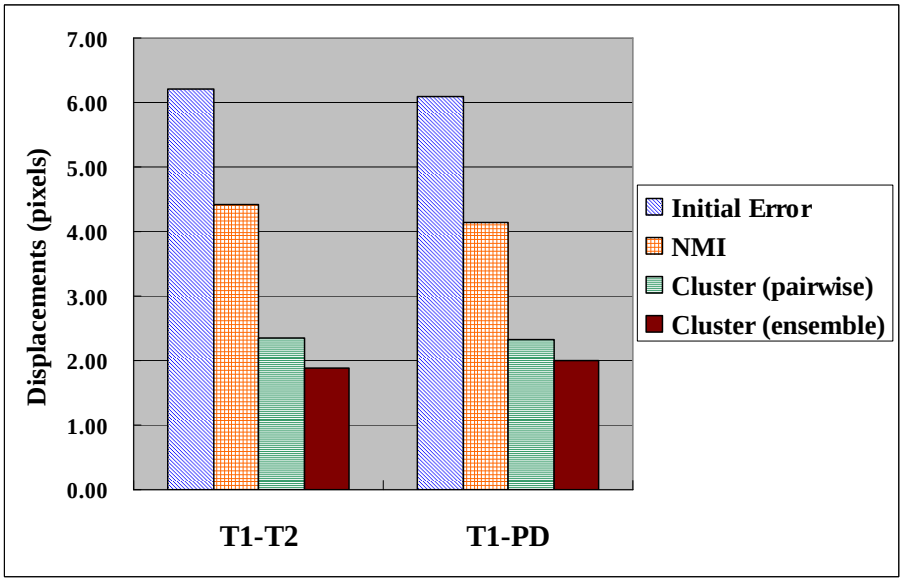
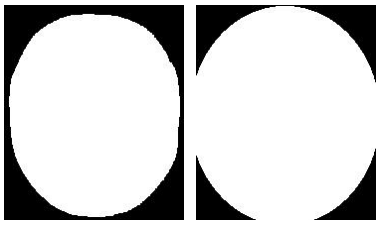
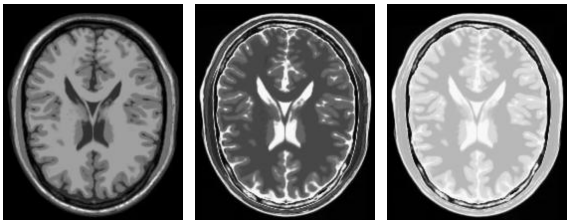


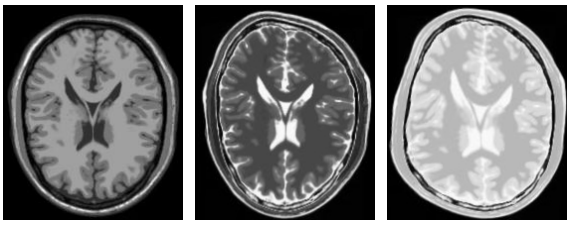
Figure 4.15: Displacement Errors for BrainWeb Data



(a) Masks

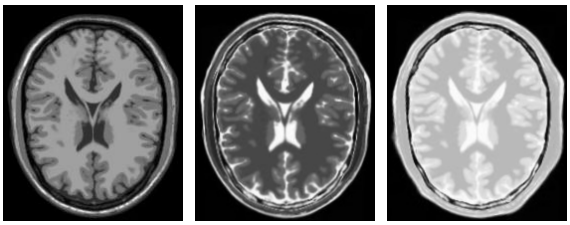


(b) Source Images: MR-T1, MR-T2, MR-PD

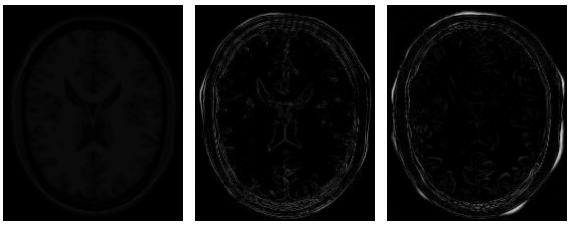


(c) Deformed Images: MR-T1, MR-T2, MR-PD

Figure 4.16: BrainWeb Data Taken from the 3^{th} Test Set.

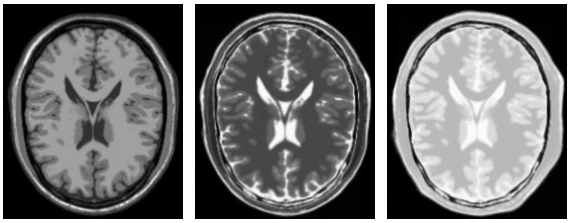


(a) Registered Images: MR-T1, MR-T2, MR-PD

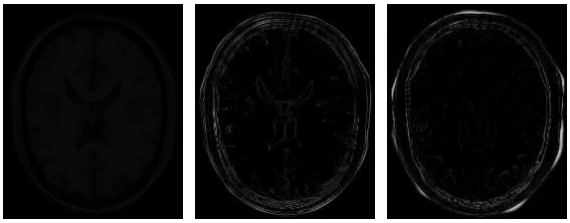


(b) Absolute Difference: MR-T1, MR-T2, MR-PD

Figure 4.17: Registration Results of Ensemble Clustering Method for BrainWeb Data Taken from the 3^{th} Test Set. (a) The registration results of the ensemble clustering method. (b) The absolute difference between the source and registered images. Rendered with the intensity range $[0 - 255]$.

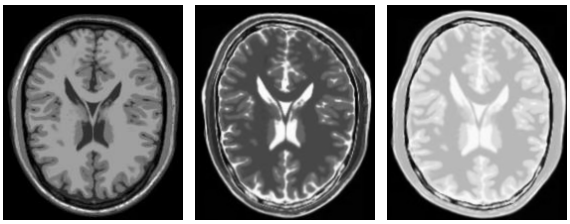


(a) Registered Images: MR-T1, MR-T2, MR-PD

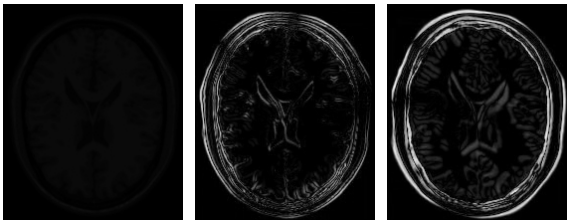


(b) Absolute Difference: MR-T1, MR-T2, MR-PD

Figure 4.18: Registration Results of Pairwise Clustering Method for BrainWeb Data Taken from the 3^{th} Test Set. (a) The registration results of the pairwise clustering method. (b) The absolute difference between the source and registered images. Rendered with the intensity range $[0-255]$.



(a) Registered Images: MR-T1, MR-T2, MR-PD



(b) Absolute Difference: MR-T1, MR-T2, MR-PD

Figure 4.19: Registration Results of Pairwise NMI Method for BrainWeb Data Taken from the 3^{th} Test Set. (a) The registration results of the pairwise NMI method. (b) The absolute difference between the source and registered images. Rendered with the intensity range $[0-255]$.

	F1-F2	F1-F3	F1-F4	F1-F5	Mean
Initial Error	7.62 (1.45)	7.07 (1.52)	13.6 (3.25)	6.88 (2.74)	8.80 (1.29)
NMI	6.93 (1.67)	6.93 (1.50)	13.5 (3.33)	5.51 (2.53)	8.22 (1.16)
Cluster (Pairwise)	6.93 (2.04)	7.09 (1.36)	13.3 (4.81)	7.85 (3.70)	8.80 (1.70)
Cluster (Ensemble)	5.35 (1.67)	3.39 (0.73)	6.41 (2.19)	5.43 (1.26)	5.15 (0.85)

Table 4.3: Displacement Errors (standard deviation) for Face Image Data

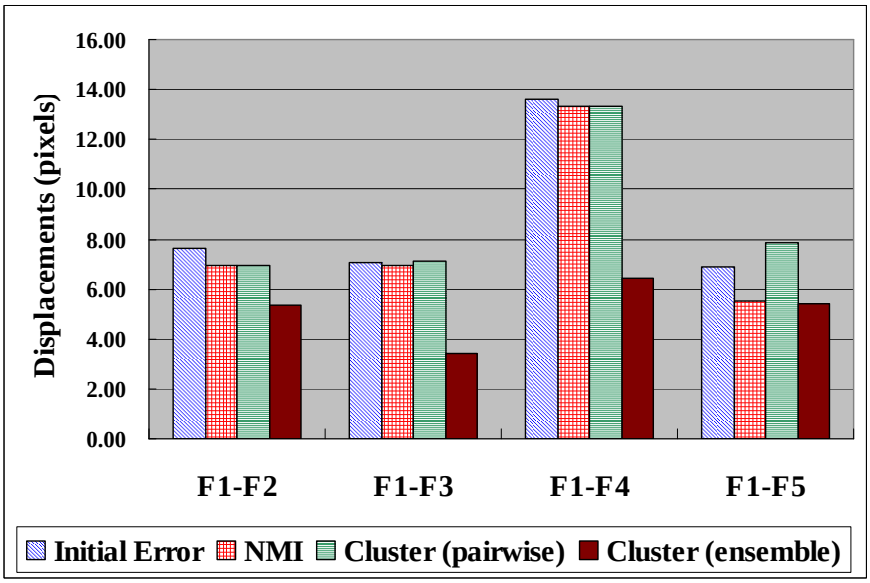
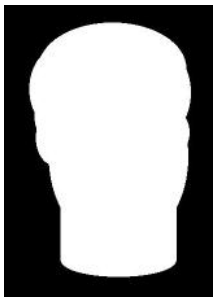
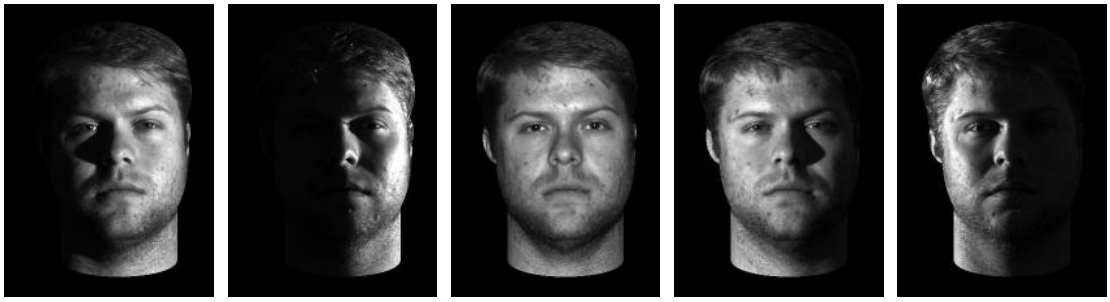


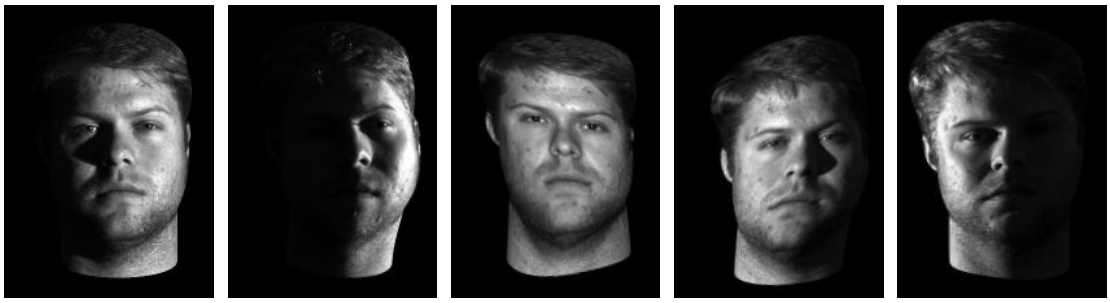
Figure 4.20: Displacement Errors for Face Images



(a) Mask

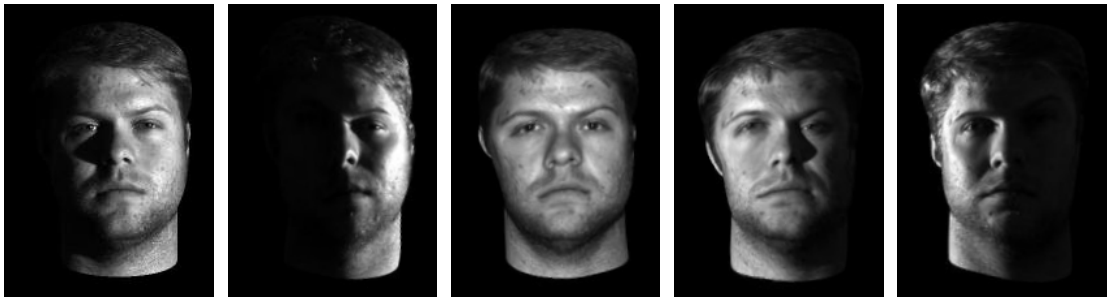


(b) Source Images: F1, F2, F3, F4, F5

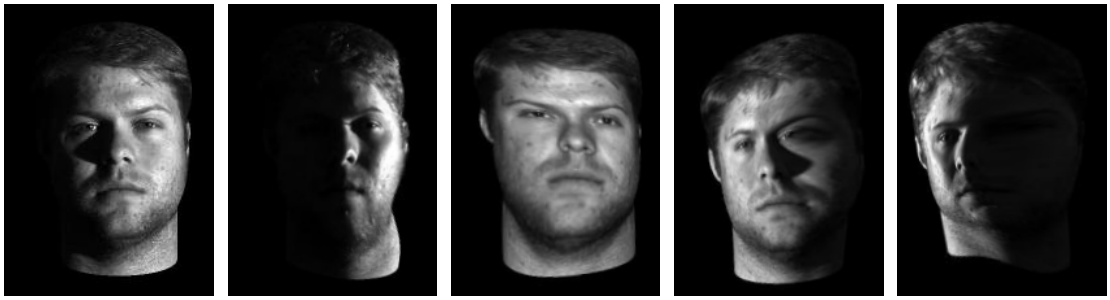


(c) Deformed Images: F1, F2, F3, F4, F5

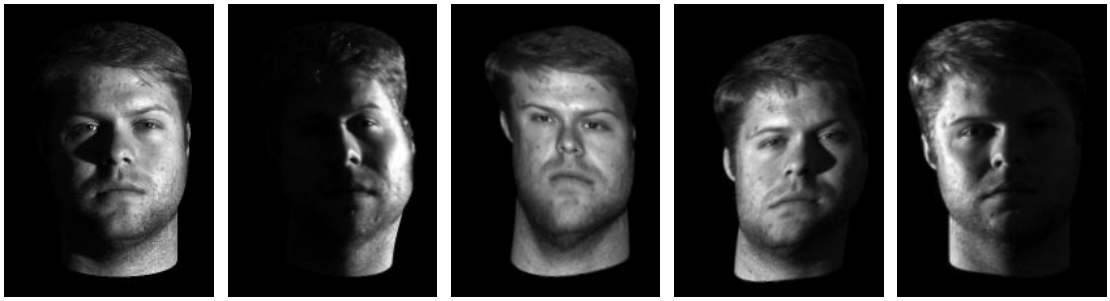
Figure 4.21: Face Image Data Taken from 10^{th} Test Set.



(a) Registered Images by the Ensemble Clustering Method: F1, F2, F3, F4, F5



(b) Registered Images by the Pairwise Clustering Method: F1, F2, F3, F4, F5



(c) Registered Images by the Pairwise NMI Method: F1, F2, F3, F4, F5

Figure 4.22: Registration Results for Face Image Data Taken from 10^{th} Test Set. (a) The registration results of the ensemble clustering method. (b) The registration results of pairwise clustering method. (c) The registration results of pairwise NMI method.

Chapter 5

Parallelization on GPU

Image registration usually requires a long computation time to search a multi-dimensional parameter space to find an optimal transformation. Non-rigid registration needs to optimize over thousands of parameters to find the best transformation. For the case of our ensemble clustering method for non-rigid registration, it usually took 3∼4 hours to register the RIRE ensemble dataset, a set of five 2-D images. To reduce the running time of the registration, we consider a parallelization of the algorithm.

Parallel computing is the simultaneous use of multiple processing elements to solve a single problem. A large problem is divided into smaller ones so that each processing element can execute parts of the problem concurrently. There are several different forms of parallelization. While task-level parallelization performs different calculations on the same or different sets of data, data-level parallelization performs the same calculation on the same or different sets of data. For image registration, data-level parallelization is the most popular approach since registration deals with repeated computation on very large image data. Single instruction, multiple data (SIMD) computers – consisting of multiple processing elements supervised by a control unit – can be used for image registration. Because SIMD computers execute the same instructions in parallel on all processing units, it is a good choice for image processing applications. Graphics Processing Units (GPUs) are a type of SIMD machine, originally optimized for 3-D graphics rendering. More recently the usage of GPUs has been extended to computationally expensive tasks in a wide variety of application domains other than 3-D graphics, called General-Purpose computing on the GPU (GPGPU) or GPU Computing [30].

In this chapter, we explore some advantages of the GPU Computing, as well as one of the GPU programming models, CUDA (Compute Unified Device Architec-

ture) developed by NVIDIA. We also implement the ensemble clustering method on GPUs using the CUDA platform and present its results.

5.1 GPGPU

Originally designed for computer graphics, GPUs have evolved into general-purpose parallel processors for non-graphics applications. A broad range of applications and tasks such as signal and image processing, physical modeling, computational engineering, game physics, computational finance and data mining have been implemented on GPUs due to several advantages over other parallel processors. First, GPUs have tremendous memory bandwidth and computational horsepower, as illustrated by Fig. 5.1. In the figure, the memory bandwidth and floating-point computation power of GPUs are nearly an order of magnitude greater than those of the CPUs. The reason why the performance of GPUs increases more rapidly than that of CPUs comes from the fundamental architectural differences. CPUs are designed for high performance on sequential tasks with instruction-level parallelism so that it focuses more on flow control and data caching. On the other hand, GPUs are designed for compute-intensive and highly parallel computation so that it has a lower requirement for sophisticated flow control and is dedicated more to data processing. This characteristic makes GPUs suitable for data-parallel computations, i.e. the same program is executed on many data elements in parallel. Image processing applications in particular – including image registration – are well suited to be implemented on this GPU architecture. Because the image blocks and pixels can be mapped to the parallel processing threads, the repeated computations for each pixel are performed in parallel on the mapped threads and it leads to substantial speed-ups. In addition, the image processing applications can benefit from the visualization functionality of GPUs.

Another advantage of GPUs is the flexibility provided by the programming processing units. Unlike the early GPUs, modern GPUs have fully programmable hardware that supports vectorized floating-point operations. Moreover, as the high level languages for GPU programming (such as High-level Shading Language (HLSL), Cg and OpenGL Shading Language (GLSL)) emerge, the programmability of GPUs improves. However, these GPU programming languages are not easy to use for those who are not familiar with the graphics APIs and graphics terms like "geometric primitives"," textures", "fragments" and "rendering passes". Recently, NVIDIA introduced the CUDA programming platform to provide easier access to GPGPU.

CUDA makes GPGPU programming possible without needing to understand GPU-specific details. The details of CUDA will be explained in Section 5.2.

Despite their several advantages, GPUs are not suited for every application. For example, word processing applications tend to be dominated by memory communication and are difficult to parallelize. Also, the GPU's lack of integers and associated operations such as bit-shifting and bitwise logical operations make it unsuitable for some applications like cryptography [31]. However, GPUs are continuously innovated and improved because of the demand of the fast-growing video game industry. Above all, GPUs are relatively cheap and are widely available. Compared with other parallel computers such as supercomputers or multi-node clusters, GPUs are very competitive for many applications. Continually increasing computation power, flexible programmability, low cost and wide availability all make GPUs an attractive parallelizing tool for general purpose computing applications.

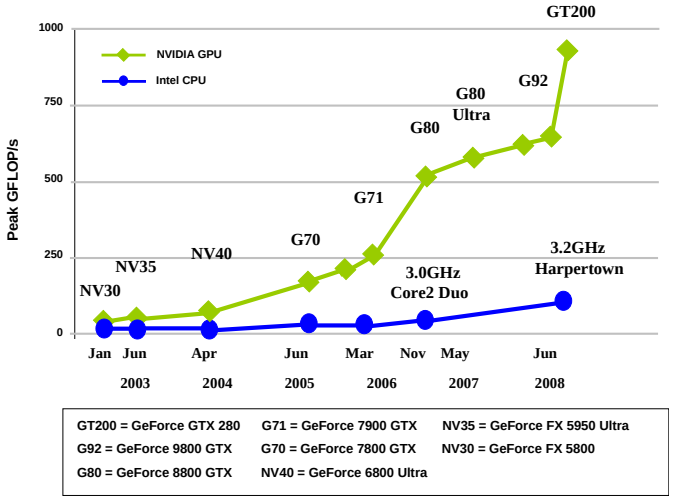
5.2 CUDA Programming Environment

CUDA (Compute Unified Device Architecture) is a parallel computing architecture for general purpose GPU computing, developed by NVIDIA. CUDA enables developers to focus on parallelization of their algorithm by hiding the underlying mechanics of GPUs and graphics. Moreover, many developers can quickly adopt the programming environment since the C/C++ language is basically used, providing a small set of extensions to handle GPUs.

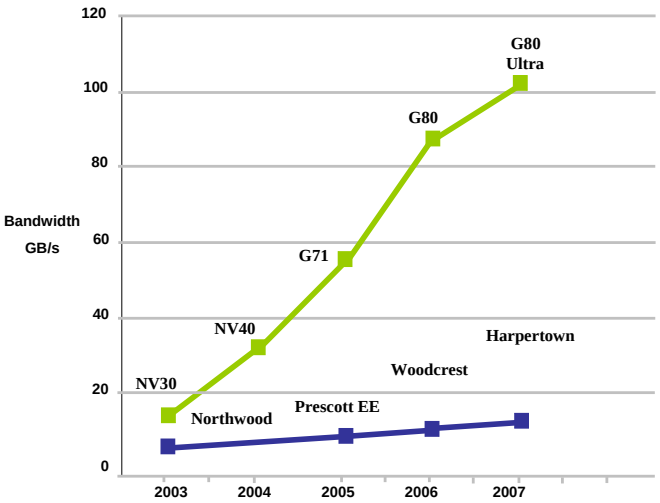
In CUDA, the parallel portions of an application are expressed as *kernels*, which are executed many times in parallel by CUDA threads on GPUs. A kernel is a piece of code executed repeatedly in many threads. A thread is the unit of parallelism in CUDA, different from a CPU thread in that it is extremely lightweight for creation and switching.

For the programming model, a kernel is executed on a grid and a grid is an array of blocks, and each block is an array of threads. Figure 5.2 (a) shows this hierarchy. Threads within a block can cooperate through shared memory and synchronize, but threads in different blocks cannot cooperate.

For the execution model, a kernel launches a grid of thread blocks and one kernel is executed on the device at a time, i.e. only one kernel is run on the graphics hardware at a time. Thread blocks are executed on multiprocessors and they do not migrate. Several concurrent thread blocks can reside on one multiprocessor limited by the multiprocessor resources. Each thread is executed by a single thread



(a) Floating-Point Operations per Second for GPU and CPU (adapted from [32])



(b) Memory Bandwidth for the CPU and GPU (adapted from [32])

Figure 5.1: Performance of CPU and GPU

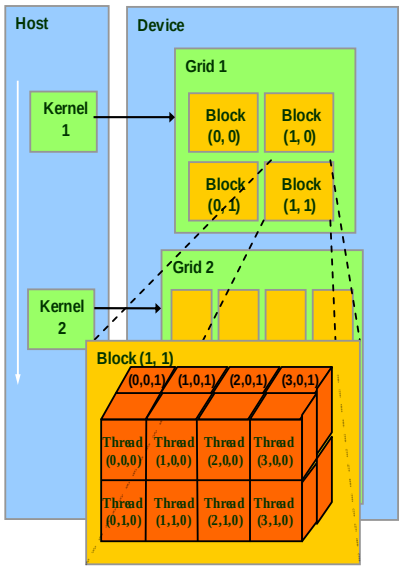
processor. The GPU hardware schedules thread blocks to run on multiprocessors to keep the processors busy. Thus, thousands of threads can be executed concurrently on GPUs.

For the memory model, Fig. 5.2 (b) shows the CUDA memory architecture. Global memory is a main means of communication between CPU (host) and GPUs (device). All threads can write and read data from the global memory. Texture and constant memory are read-only for threads. The texture memory is used for resampling images in image processing applications. Shared memory is shared among threads in a single block. While the shared memory resides on-chip, the global memory resides in device memory (DRAM). Thus, the global memory is much slower than shared memory. One of the optimization strategies is to minimize data accesses to the global memory and to maximize the use of the shared memory.

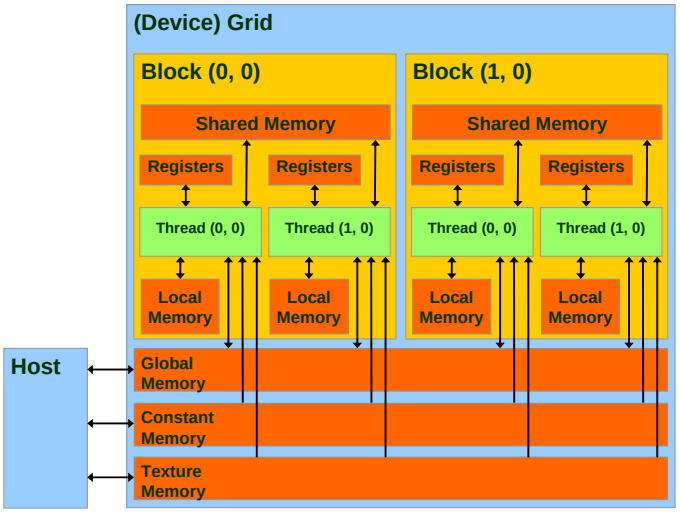
5.3 CUDA Implementation of Ensemble Clustering

Based on the above CUDA programming environment, we implemented the ensemble clustering method for rigid registration [1]. In order to map an algorithm to the CUDA programming environment, we first need to specify which portions of the algorithm will be parallelized, and isolate them as CUDA kernels. Algorithm 2 shows the algorithm of the ensemble clustering method for rigid registration. This is the same as the non-rigid case (Algorithm 1) except that rigid-body motion parameters (θ) are used instead of deformation parameters (c), and there is no need to handle the control-point grid. We chose parts of the algorithm that can be parallelized by GPUs and implemented them as kernels. The kernels are categorized into three groups.

The first group contains tasks for image transformations such as scaling images for the multi-resolution approach, and image resampling after establishing new motion parameters (lines 5, 6 and 11 of Algorithm 2). Fundamentally, these interpolate image intensity values according to the transformed coordinates of each pixel. Thus, the interpolation for new intensity values and the computation for new coordinates should be repeated for every pixel in the images. For the interpolation, we use GPU texture memory as well as some device functions, called *texture fetches*, in CUDA. After copying images into the texture memory, a *texture reference* object is defined, and bound to some region of the texture memory where the images are loaded. The texture reference has several attributes and one of them is to specify the interpolation method (either nearest-neighbour or linear). Thus,



(a) CUDA - Grid of Thread Blocks (adapted from [33]).



(b) CUDA Memory Model (adapted from [33]).

Figure 5.2: CUDA Architecture

we set attributes of the texture reference and call the texture device functions to perform the interpolation. Inside the kernel, we compute new coordinates for each pixel by applying the given transformations, and call the texture fetch functions to get new image values. Compared with a serial processor, the processing time for interpolation can be reduced hugely. This is a real benefit from GPUs.

The second group of kernels computes the four parameters of the Gaussian Mixture Model density estimation process of the ensemble clustering method (line 8 of Algorithm 2). The first parameter is the membership of each pixel to each Gaussian component (or cluster), τ in (3.4). If the number of pixels in the region of interest (ROI) is N_s and the number of clusters in the JISP is K, then the τ is calculated in an $N_s \times K$ matrix by (3.4). The other three parameters, mean μ, covariance matrix Σ, and the Gaussian component weights π, are calculated in a $K \times 1$ matrix by (3.5), (3.6) and (3.7), respectively. These matrices are designed to cooperate with the $N_s \times K$ matrix τ to re-estimate these three parameters. Therefore, this work is a kind of matrix calculation. We reorganized some of the matrices to allow efficient parallelism on GPUs, and created several kernels for these computations.

The last group of kernels computes the matrix A and the vector b in (3.12) (line 9 of Algorithm 2). They are also used for solving linear equations to find the motion increment, θ in (3.12). If the number of images is D, the number of motion parameters is M, and the number of pixels in the ROI is N_s, then the matrix A is an $MD \times MD$ matrix and the vector b is a $1 \times MD$ vector. To calculate these, we incrementally evaluate of A and b one pixel at a time. Figure 5.3 describes two implementations to calculate the matrix A and the vector b in a sequential and parallel way. For the sequential way, the $MD \times MD$ matrix A and the $1 \times MD$ vector b are calculated for each pixel inside a loop that loops over all the pixels in the ROI. This looping for all pixels is the most time-consuming part in the whole algorithm. Instead of looping, we reorganize the matrix to fit the parallel execution on GPUs like Fig. 5.3 (b). Each row of A_p (and b_p) contains the matrix A (or b) for a given pixel. Hence A_p is $N_s \times (MD)^2$ and b_p is $N_s \times MD$. Each kernel simultaneously calculates each row of A_p and b_p for each pixel in the ROI and then adds all the rows together. This approach reduces the execution time remarkably. However, the parallel method uses N_s times the amount of memory, so can cause memory problems if N_s gets large or A gets large.

After scaling all images at each resolution level, all variables and the images are transferred from CPU memory into GPU memory. The EM step and motion

adjustment are performed on the GPU. To calculate the motion increment $\widetilde{\theta}$, the matrix A and the vector b are copied to CPU memory and the linear system in (3.12) is solved on the CPU. This process is iterated until convergence.

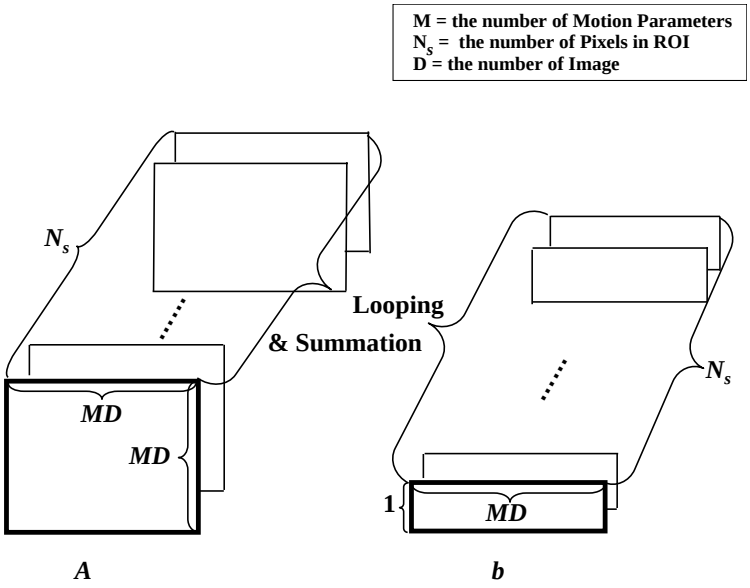
Algorithm 2 : Ensemble Clustering Registration

1: Input: initial ensemble I_0
2: Input: initial motion parameters θ
3: Input: initial GMM parameters ϕ
4: **for** each scale **do**
5: $I_{scaled} \longleftarrow$ scale ensemble I_0
6: $I \longleftarrow$ apply motion (θ) to ensemble I_{scaled}
7: **repeat**
8: $\phi \longleftarrow$EM step – density estimation process
9: $\widetilde{\theta} \longleftarrow$ motion adjustment process
10: $\theta \longleftarrow \theta + \widetilde{\theta}$
11: $I \longleftarrow$ apply motion (θ) to ensemble I_{scaled}
12: **until** converged ($\widetilde{\theta}$ is small)
13: **end for**
14: Output: I is registered ensemble at full scale
15: Output: θ holds the optimal motion parameters
16: Output: ϕ holds the GMM parameters

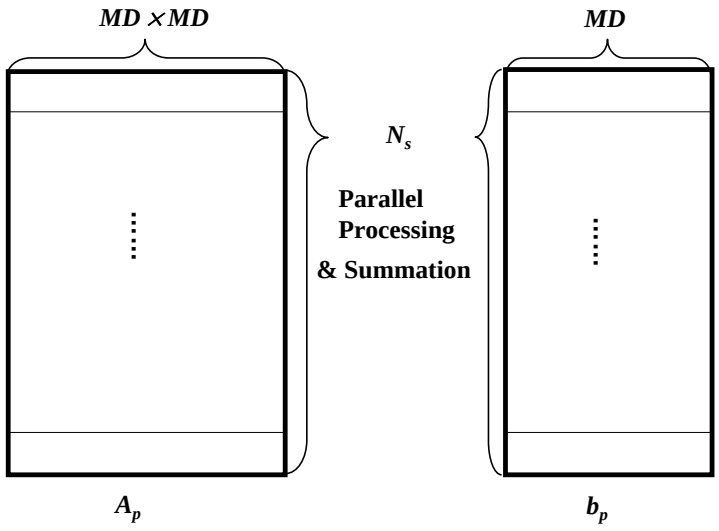
5.4 Experiments and Results

We demonstrate the efficiency of the parallelized version of the ensemble clustering method for rigid registration on GPUs and compare it with our CPU implementation. To do this, we measured the execution times to iterate through the outer-most loop (line from 4 to 13 in Algorithm 2).

For the test data, we used two different datasets. One was the Retrospective Image Registration Evaluation (RIRE) project's training set. These five images (T1-MR, T2-MR, PD-MR, CT and PET) are the same images used for non-rigid registration in Chapter 4, but this time we used 3-D volumes. The original volumes were brought into register using RIRE's given true parameters, after which the registered volumes were subsampled to a size of 80×80×32 voxels. To create synthetic data, 3-D rigid-body transformations were randomly generated by



(a) Serial Form of A and b



(b) Parallel Form of A and b

Figure 5.3: Different Implementation of A and b

uniformly choosing the three rotations and three translations from the range [-5, 5] (degrees or pixels). The second dataset is the face images (640×480) used for non-rigid registration in Chapter 4. A synthetic dataset was generated by applying randomly generated rigid-body displacements, chosen uniformly from the range [-10, 10] (pixels or degrees). For the comparison with the CPU, we used a Matlab implementation of the registration method. Matlab is very efficient for matrix computations. The computer we used for these experiments was a Dell Dimension 5150 with a 3.20 GHz Intel Pentium 4 CPU running Windows XP SP2 with 3GB memory. We used the NVIDIA Geforce 8600 GT as our GPU hardware platform.

The results of the experiment are shown in Table 5.1. The CUDA implementation accelerated the registration process by a factor of 7.5 for the RIRE dataset and a factor of 17.3 for the face image dataset.

	RIRE Data $80 \times 80 \times 32$	Face Images 640×480
CPU runtime (sec)	260.6	168.1
GPU runtime (sec)	34.7	9.7

Table 5.1: Execution Time (seconds) for CPU and GPU implementation

Chapter 6

Conclusions and Future Work

This thesis applied the ensemble clustering method to simultaneously register multi-sensor ensembles that include non-rigid deformations. Also, it provided a parallelization of the algorithm on GPUs to accelerate the performance of the method.

For the non-rigid registration, we reformulated the cost function and the optimization processes of the ensemble clustering method with a regularization term based on mean elastic energy of B-spline deformed images. During the implementation process, we faced an ill-conditioning problem in the Newton-type optimization process. The instability was largely overcome using a matrix approximation and the SVD. We demonstrated that the ensemble clustering method performs better than pairwise methods through all three synthetic datasets. Despite the fact that the ill-conditioning problem still had an influence on some images, the experiments showed that the ensemble clustering method was successfully applied to non-rigid registration of multi-sensor ensembles. Moreover, our experiments showed that the ensemble clustering registration method leveraged the concordance of all the given images and yielded improved accuracy over pairwise methods.

We parallelized the algorithm of the ensemble clustering method, and implemented it on GPUs using the CUDA programming environment. Our experiments showed that the GPU implementation is much faster than the CPU implementation, 17 times faster for the 3-D RIRE dataset, and 7 times faster for the face image dataset. While our GPU implementation was for rigid-body registration, we expect the same speed-ups for non-rigid registration.

Therefore, the first future work will be the GPU implementation of the ensemble clustering method for non-rigid registration. Since non-rigid registration needs more extensive computations due to the increased degrees of freedom in the transformations, the parallelization of the non-rigid registration method would be an

important advance.

Since we applied the non-rigid ensemble clustering method in 2-D, extending it to 3-D is another direction for future work. While we expect the method to work, we acknowledge that non-rigid transformations in 3-D have an order of magnitude more parameters, and thus might yield a more challenging optimization process.

We used a regularization term based on mean elastic energy, whereas other regularization methods can be explored to improve the performance of the registration. Investigation of alternative regularization methods might improve the capabilities of the ensemble clustering method.

One of the main stumbling-blocks of the ensemble clustering method is the ill-conditioning problem that arises when images contain homogeneous patches. In order to avoid the ill-conditioning problem, we can apply other types of optimization methods such as the gradient descent.

Lastly, we did not examine the behaviour of the ensemble clustering method with respect to the number of Gaussian components. We used the same simple paradigm for all of our experiments; we held the number of Gaussian components constant. However, we could have increased the number of components as we ramped up the resolution in our multi-resolution framework. The number of components could be thought of as another element of the multi-resolution approach. Further investigation is required to understand how the number of Gaussian components can be manipulated to improve the performance of the ensemble clustering method.

Bibliography

[1] J. Orchard and R. Mann, "Registering a multi-sensor ensemble of images," *IEEE Transactions on Image Processing (in press)*, 2009. 1, 2, 6, 8, 9, 11, 13, 16, 20, 21, 44, 55

[2] A. Collignon, F. Maes, D. Delaere, D. Vandermeulen, P. Suetens, and G. Marchal, "Automated multi-modality image registration based on information theory," *Information Processing in Medical Imaging*, pp. 263–274, 1995. 4

[3] W. Wells, P. Viola, H. Atsumi, S. Nakajima, and R. Kikinis, "Multi-modal volume registration by maximization of mutual information," 1996. 4

[4] A. Roche, G. Malandain, X. Pennec, and N. Ayache, "The correlation ratio as a new similarity measure for multimodal image registration." Springer Verlag, 1998, pp. 1115–1124. 4

[5] C. Studholme, "An overlap invariant entropy measure of 3D medical image alignment," *Pattern Recognition*, vol. 32, no. 1, pp. 71–86, January 1999. 4

[6] W. R. Crum, T. Hartkens, and D. L. Hill, "Non-rigid image registration: theory and practice." *The British Journal of Radiology*, vol. 77 Spec No 2, 2004. 4, 5, 6

[7] A. Goshtasby, "Registration of images with geometric distortions," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 26, no. 1, pp. 60–64, Jan 1988. 5

[8] F. L. Bookstein, "Principal warps: thin-plate splines and the decomposition of deformations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 6, pp. 567–585, 1989. 5

[9] 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, pp. 712–721, 1999. 5, 17

[10] R. Bajcsy and S. Kovacic, "Multiresolution elastic matching," *Computer Vision, Graphics, and Image Processing*, vol. 46, no. 1, pp. 1–21, April 1989. 5

[11] 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. 6

[12] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981. 6

[13] B. M. Dawant, "Non-rigid registration of medical images: purpose and methods, a short survey," in *Proceedings of 2002 IEEE International Symposium on Biomedical Imaging*, 2002, pp. 465–468. 6

[14] H. Lester and S. R. Arridge, "A survey of hierarchical non-linear medical image registration," *Pattern Recognition*, vol. 32, no. 1, pp. 129–149, January 1999. 6

[15] B. Zitová, "Image registration methods: a survey," *Image and Vision Computing*, vol. 21, no. 11, pp. 977–1000, October 2003. 6, 9, 10

[16] J. Orchard and L. Jonchery, "Ensemble registration: aligning many multisensor images simultaneously," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 7245, Feb. 2009. 6, 8

[17] R. P. Woods, S. T. Grafton, C. J. Holmes, S. R. Cherry, and J. C. Mazziotta, "Automated image registration: I. General methods and intrasubject,intramodality validation," *Journal of Computer Assisted Tomography*, vol. 22, no. 1, pp. 139–152, JAN-FEB 1998. 8

[18] K. K. Bhatia, J. V. Hajnal, B. K. Puri, A. D. Edwards, and D. Rueckert, "Consistent groupwise non-rigid registration for atlas construction," in *IEEE International Symposium on Biomedical Imaging: Nano to Macro*, vol. 1, April 2004, pp. 908–911. 8

[19] L. Zöllei, E. Learned-Miller, E. Grimson, and W. Wells, "Efficient population registration of 3D data," in *ICCV*, 2005, pp. 291–301. 8

[20] G. J. McLachlan and K. E. Basford, *Mixture Models, Inference and Applications to Clustering.* New York: Marcel Dekker, 1988. 12, 13

[21] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," in *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques.* ACM Press, 1986, pp. 151–160. 16

[22] S. Lee, G. Woberg, K.-Y. Chwa, and S. Shin, "Image metamorphosis with scattered feature constraints," *IEEE Transactions on Visualization and Computer Graphics*, vol. 2, no. 4, pp. 337–354, 1996. 16, 17

[23] S. Lee, G. Wolberg, and S. Shin, "Scattered data interpolation with multilevel B-Splines," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, pp. 228–244, 1997. 16, 17

[24] G. H. Golub and C. F. Van Loan, *Matrix Computation.* Baltimore: The Johns Hopkins University Press, 1996. 24, 27

[25] D. R. Forsey and R. H. Bartels, "Hierarchical B-spline refinement," in *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques.* ACM, 1988, pp. 205–212. 36

[26] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the Nelder–Mead simplex method in low dimensions," *SIAM Journal of Optimization*, vol. 9, pp. 112–147, 1998. 37

[27] J. West, J. M. Fitzpatrick, M. Y. Wang, B. M. Dawant, C. R. Maurer, R. M. Kessler, R. J. Maciunas, C. Barillot, D. Lemoine, A. Collignon, F. Maes, T. S. Sumanaweera, B. Harkhess, P. F. Hemler, D. L. G. Hill, D. J. Hawkes, C. Studholme, J. B. A. Maintz, M. A. Viergever, G. Mal, X. Pennec, M. E. Noz, G. Q. Maguire, M. Pollack, C. A. Pelizzari, R. A. Robb, D. Hanson, and R. P. Woods, "Comparison and evaluation of retrospective intermodality brain image registration techniques," *Journal of Computer Assisted Tomography*, vol. 21, pp. 554–566, 1998. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.328137

[28] C. A. Cocosco, V. Kollokian, R. K.-S. Kwan, G. B. Pike, and A. C. Evans, "Brainweb: Online interface to a 3D MRI simulated brain database," *NeuroImage*, vol. 5, p. 425, 1997. 37

[29] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman, "From few to many: illumination cone models for face recognition under variable lighting and pose," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 643–660, 2001. [Online]. Available: http://dx.doi.org/10.1109/34.927464 37

[30] GPGPU, "General Purpose Computation Using Graphics Hardware," http://www.gpgpu.org. 51

[31] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell, "A survey of general-purpose computation on graphics hardware," *Computer Graphics Forum*, vol. 26, no. 1, pp. 80–113, March 2007. 53

[32] *NVIDIA Compute Unified Device Architecture Programming Guide Version 2.0*, Jun 2008. [Online]. Available: http://www.nvidia.com/object/cuda_develop.html 54

[33] *NVIDIA CUDA Technical Training Volume I: Introduction to CUDA Programming*, 2008. [Online]. Available: http://www.nvidia.com/object/cuda_education.html 56

Appendix A

Displacement Errors

A.1 RIRE Data

Trials	Methods	T1-T2	T1-PD	T1-CT	T1-PET	Mean
1	Initial Error	7.71	6.21	13.1	6.20	8.31
	NMI	7.03	5.52	11.8	5.99	7.60
	Cluster (Pairwise)	7.23	2.18	9.10	6.12	6.16
	Cluster (Ensemble)	3.99	1.99	11.7	4.94	5.66
2	Initial Error	8.88	9.43	11.7	4.91	8.73
	NMI	5.74	8.93	8.05	4.35	6.77
	Cluster (Pairwise)	7.66	4.43	6.90	9.63	7.16
	Cluster (Ensemble)	4.95	3.24	10.1	4.77	5.77
3	Initial Error	5.28	4.68	12.4	12.2	8.66
	NMI	3.68	3.67	11.9	11.0	7.59
	Cluster (Pairwise)	1.87	2.28	8.90	15.3	7.03
	Cluster (Ensemble)	2.73	2.18	10.7	4.85	5.11
4	Initial Error	8.84	7.33	10.4	7.40	8.50
	NMI	8.93	4.43	10.0	5.81	7.30
	Cluster (Pairwise)	7.42	2.67	10.1	16.9	9.27
	Cluster (Ensemble)	4.12	2.37	10.9	4.86	5.56
5	Initial Error	8.75	6.31	13.7	5.39	8.54
	NMI	6.23	3.99	14.8	4.27	7.33
	Cluster (Pairwise)	5.80	3.10	10.6	5.51	6.25
	Cluster (Ensemble)	2.99	2.10	10.3	6.53	5.49

Trials	Methods	T1-T2	T1-PD	T1-CT	T1-PET	Mean
6	Initial Error	7.42	5.36	12.4	8.64	8.46
	NMI	7.11	4.43	12.3	5.75	7.40
	Cluster (Pairwise)	5.24	4.15	12.3	9.00	7.67
	Cluster (Ensemble)	4.34	1.67	8.81	10.6	6.34
7	Initial Error	6.95	5.58	9.44	5.52	6.87
	NMI	4.88	3.42	6.68	4.92	4.98
	Cluster (Pairwise)	8.58	2.05	5.96	5.29	5.47
	Cluster (Ensemble)	2.75	2.05	9.37	5.74	4.98
8	Initial Error	7.33	6.86	15.2	6.02	8.84
	NMI	5.57	4.90	14.4	4.24	7.29
	Cluster (Pairwise)	5.08	2.49	11.0	5.24	5.96
	Cluster (Ensemble)	3.85	1.96	14.6	5.80	6.56
9	Initial Error	8.88	7.19	19.6	7.28	10.7
	NMI	5.64	6.70	20.4	5.91	9.66
	Cluster (Pairwise)	8.37	3.55	20.8	9.35	10.5
	Cluster (Ensemble)	2.48	2.69	14.8	5.87	6.46
10	Initial Error	4.25	7.53	13.1	4.61	7.38
	NMI	3.08	6.93	12.9	3.56	6.61
	Cluster (Pairwise)	1.67	2.82	12.4	7.66	6.14
	Cluster (Ensemble)	1.84	2.58	11.6	4.78	5.20
Average	Initial Error	7.43	6.65	13.1	6.82	8.50
	NMI	5.81	5.29	12.3	5.58	7.25
	Cluster (Pairwise)	5.89	2.97	10.8	9.00	7.17
	Cluster (Ensemble)	3.40	2.28	11.3	5.87	5.71

A.2 BrainWeb Data

Trials	Methods	T1-T2	T1-PD	Mean
1	Initial Error	8.55	8.65	8.65
	NMI	7.33	5.03	6.18
	Cluster (Pairwise)	2.90	3.40	3.15
	Cluster (Ensemble)	2.02	3.08	2.55
2	Initial Error	5.20	4.31	4.75
	NMI	3.64	2.81	3.23
	Cluster (Pairwise)	1.99	1.19	1.59
	Cluster (Ensemble)	1.67	1.41	1.54
3	Initial Error	2.21	7.04	4.62
	NMI	1.38	4.40	2.89
	Cluster (Pairwise)	0.46	1.99	1.23
	Cluster (Ensemble)	0.47	2.15	1.31
4	Initial Error	10.1	12.3	11.2
	NMI	8.90	10.6	9.73
	Cluster (Pairwise)	6.15	8.46	7.31
	Cluster (Ensemble)	4.92	5.12	5.02
5	Initial Error	8.00	3.52	5.76
	NMI	4.66	2.45	3.56
	Cluster (Pairwise)	2.54	0.82	1.68
	Cluster (Ensemble)	2.22	1.16	1.69
6	Initial Error	7.02	7.59	7.31
	NMI	4.78	4.84	4.81
	Cluster (Pairwise)	3.22	2.14	2.68
	Cluster (Ensemble)	2.30	2.27	2.29
7	Initial Error	3.68	5.09	4.38
	NMI	2.34	3.47	2.91
	Cluster (Pairwise)	1.22	1.23	1.22
	Cluster (Ensemble)	1.12	1.41	1.27
8	Initial Error	3.07	3.41	3.24
	NMI	2.00	2.21	2.11
	Cluster (Pairwise)	0.61	0.71	0.66
	Cluster (Ensemble)	0.61	0.81	0.71

Trials	Methods	T1-T2	T1-PD	Mean
9	Initial Error	5.51	6.34	5.92
	NMI	3.18	4.02	3.60
	Cluster (Pairwise)	1.35	2.79	2.07
	Cluster (Ensemble)	1.31	1.61	1.46
10	Initial Error	8.74	2.53	5.63
	NMI	5.86	1.54	3.70
	Cluster (Pairwise)	2.93	0.60	1.77
	Cluster (Ensemble)	2.19	0.95	1.57
Average	Initial Error	6.21	6.09	6.15
	NMI	4.41	4.13	4.27
	Cluster (Pairwise)	2.34	2.33	2.34
	Cluster (Ensemble)	1.88	2.00	1.94

A.3 Face Image Data

Trials	Methods	F1-F2	F1-F3	F1-F4	F1-F5	Mean
1	Initial Error	7.52	6.29	12.3	6.46	8.15
	NMI	7.31	7.76	12.4	5.20	8.17
	Cluster (Pairwise)	5.71	7.43	11.4	5.57	7.53
	Cluster (Ensemble)	5.47	2.89	6.31	4.77	4.86
2	Initial Error	8.31	11.0	12.7	4.23	9.05
	NMI	7.03	10.5	12.95	3.60	8.51
	Cluster (Pairwise)	8.38	6.43	8.05	5.09	6.99
	Cluster (Ensemble)	6.19	4.98	5.30	4.30	5.19
3	Initial Error	5.10	5.18	12.0	13.8	9.01
	NMI	4.74	4.77	11.9	10.7	8.02
	Cluster (Pairwise)	4.95	5.98	9.82	6.00	6.69
	Cluster (Ensemble)	3.56	3.42	5.12	7.43	4.88
4	Initial Error	8.40	6.91	10.2	6.59	8.02
	NMI	7.51	6.18	10.0	6.15	7.47
	Cluster (Pairwise)	8.93	9.40	9.98	15.4	10.9
	Cluster (Ensemble)	5.94	2.50	4.24	5.42	4.53
5	Initial Error	8.38	6.54	14.1	5.91	8.73
	NMI	7.90	6.29	14.0	3.35	7.87
	Cluster (Pairwise)	6.25	6.32	19.1	4.52	9.05
	Cluster (Ensemble)	4.87	2.68	7.37	5.19	5.03
6	Initial Error	9.48	6.21	13.4	9.34	9.62
	NMI	8.55	5.79	12.8	8.80	8.98
	Cluster (Pairwise)	9.19	4.69	13.7	6.97	8.64
	Cluster (Ensemble)	9.20	2.83	4.71	6.57	5.83
7	Initial Error	7.30	6.19	10.1	5.53	7.28
	NMI	7.37	6.67	10.2	4.66	7.23
	Cluster (Pairwise)	7.28	7.75	10.7	6.50	8.06
	Cluster (Ensemble)	3.45	3.75	6.64	3.58	4.36
8	Initial Error	7.67	7.45	16.1	5.87	9.27
	NMI	7.01	7.02	16.1	3.33	8.37
	Cluster (Pairwise)	6.01	7.37	15.0	6.30	8.68

Trials	Methods	F1-F2	F1-F3	F1-F4	F1-F5	Mean
	Cluster (Ensemble)	5.74	3.73	11.0	6.41	6.71
9	Initial Error	9.08	8.23	22.0	7.40	11.7
	NMI	8.60	7.33	21.6	6.19	10.9
	Cluster (Pairwise)	9.34	6.82	23.7	8.55	12.1
	Cluster (Ensemble)	5.17	3.84	9.06	6.54	6.15
10	Initial Error	4.92	6.72	13.5	3.68	7.21
	NMI	3.28	7.01	13.1	3.15	6.63
	Cluster (Pairwise)	3.24	8.73	11.9	13.6	9.37
	Cluster (Ensemble)	3.92	3.28	4.42	4.12	3.94
Average	Initial Error	7.62	7.07	13.6	6.88	8.80
	NMI	6.93	6.93	13.5	5.51	8.22
	Cluster (Pairwise)	6.93	7.09	13.3	7.85	8.80
	Cluster (Ensemble)	5.35	3.39	6.41	5.43	5.15