ABSTRACT

Title of Document: **HARDWARE-ACCELERATED AUTOMATIC 3D NONRIGID IMAGE REGISTRATION**

YASHWANTH HEMARAJ,
MASTER OF SCIENCE
2007

Directed By: DR RAJ SHEKHAR,
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
UNIVERSITY OF MARYLAND
COLLEGE PARK, MD

Software implementations of 3D nonrigid image registration, an essential tool in medical applications like radiotherapies and image-guided surgeries, run excessively slow on traditional computers. These algorithms can be accelerated using hardware methods by exploiting parallelism at different levels in the algorithm. We present here, an implementation of a free-form deformation-based algorithm on a field programmable gate array (FPGA) with a customized, parallel and pipelined architecture. We overcome the performance bottlenecks and gain speedups of up to 40x over traditional computers while achieving accuracies comparable to software implementations. In this work, we also present a method to optimize the deformation field using a gradient descent-based optimization scheme and solve the problem of mesh folding, commonly encountered during registration using free-form deformations, using a set of linear constraints. Finally, we present the use of novel dataflow modeling tools to automatically map registration algorithms to hardware like FPGAs while allowing for dynamic reconfiguration.

HARDWARE-ACCELERATED AUTOMATIC 3D NONRIGID IMAGE REGISTRATION

By

YASHWANTH HEMARAJ

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Science
2007

Advisory Committee:
Dr Raj Shekhar, Chair
Dr Shuvra Bhattacharyya
Dr Rajiv Barua

To my Mom, Dad and Sister…

Acknowledgements

I extend my sincere appreciation and gratitude toward Dr Raj Shekhar, my advisor, who has given me this wonderful opportunity to work on the exciting area of medical image registration and for supporting me throughout this process. It is his technical expertise, academic guidance and generous financial support that have made this work possible. He has been a great motivator, always getting the best out of me and has always encouraged me to perform to my full potential. With his insightful comments, diligence and positive attitude, he has helped me develop as a researcher and as a person for which I express my heartfelt thanks. Thank you Dr Shekhar.

I would like to thank Dr Shuvra Bhattacharyya and Dr Rajiv Barua for serving on the thesis committee. I thank them for carefully reviewing my thesis, providing invaluable advice and suggestions. I would like to thank Dr Bhattacharyya for allowing me to work with him over the last couple of years. His technical insights, encouragement and motivation he has helped me develop a deep respect for innovation and scientific thinking.

I would like to thank a lot of people who have been a constant support throughout this process. I would like to thank my lab mates at the Imaging Technologies Lab (ITL), Avanti, Jianzhou, Omkar, Peng, Venkatesh and Will. It was the friendly atmosphere and the healthy discussions that we had in the lab that made every new day very interesting and memorable.

In particular I would like to thank Dr William Plishker for helping me through my research, always sharing innovative ideas and providing invaluable suggestions.

Without his efforts, the work with global chainmail and gradient descent based optimization would not have been possible at such short time. I would like to thank Dr Mainak Sen for his help through the work on dataflow modeling. Will and Mainak have been very helpful in reviewing parts of my thesis and research papers. I would also like to thank Dr Carlos Castro-Pareja for getting me started with the work on FAIR, as this work was in continuation of the work he started as a post-doc at the ITL lab.

In what has been a great journey through life, I have made many friends who will remain forever in my heart. I would like to thank my friends, Mungi, Rapa, Gayatri, Panda, Avinash, Dikpal and Nimish for the love and support they have given me.

Last, but not the least I would like to thank my parents, Shylaja Kumari and Hemaraju for being there for me always. I am not finding enough words to express how blessed I feel for having received such unconditional love and care throughout my life, always encouraging me to dream high and think big. I would like to thank my sister Deepthi and brother in law Susheel for their wonderful words of encouragement and support to realize my dreams. I would like to express my heartfelt gratitude to Suresh, Arvind, Devendra, Bharath, Keerthana and all those close to my heart for being such a wonderful support system that has made my life here beautiful.

Table of Contents

List of Tables

List of Figures

1 INTRODUCTION

1.1 Introduction to Image Registration

With the rapid advances in technology, many medical imaging technologies like computed tomography (CT), positron emission tomography (PET), magnetic resonance imaging (MRI), single photon emission computed topography (SPECT), etc. are being used in the diagnosis and treatment of different illnesses. Each of these imaging modalities has its own specific characteristics, which provide invaluable information to a physician making a diagnosis. However, even though the images may correspond to the same anatomical structure, they can be misaligned inhibiting a doctor's ability to use them optimally. Misalignment may occur because the images might have been taken at different instances of time or from different points of view. Thus a challenging task at hand is to integrate these sets of images, i.e., perform *image registration*, so that there is a correct correspondence of anatomical features, giving the physician an integrated image. In the presence of soft-tissue motion during image acquisitions, this task becomes more challenging as recovering nonlinear motion is more difficult than recovering linear motion like simple translations and rotations. Image registration provides us with an important tool in medical imaging to merge or compare these images either from the same modality or from different modalities.

Image registration is the process of finding the best transformation that best aligns the two images spatially. For example, in Figure 1, the image on the right is the same as the one on the left, but for a small rotation between them.. Image registration

allows us to calculate the transformation that is applied on the image on the right so that it matches the image on the left exactly.

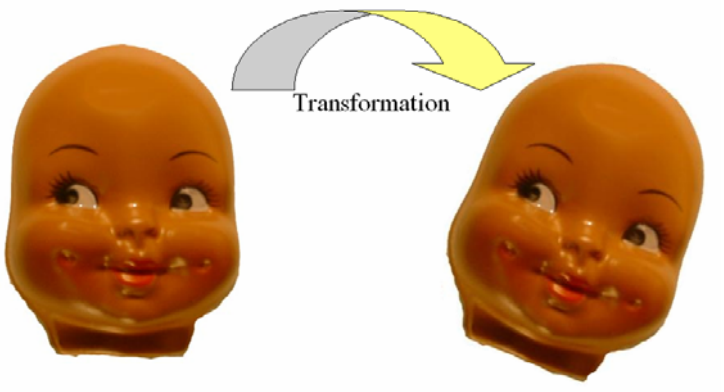

Figure 1 Image registration is the process to find the best transformation that matches the two images

The transformations that can be applied are of two types, namely, rigid registration and nonrigid registration (also referred to as elastic or deformable registration). In rigid registration, a set of linear transformations like rotation, translation, shearing and scaling are applied and in nonlinear registration, a set of nonlinear transformations like warping are applied which aligns the two images. Usually rigid registration precedes nonrigid registration as shown in Figure 2

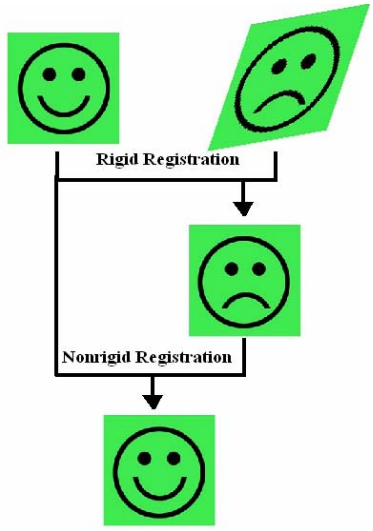

Figure 2 Flow of nonrigid registration

1.1.1 Medical Image Registration and its Applications

Image registration has the potential of being used extensively in the field of medical imaging because of the value added by registering two images of either same or different modalities. Combining images of different modalities gives physicians additional clinical information by fusing different modality images and alignment into the same spatial location before visualization. Figure 3 shows the workflow of the entire process. Some of the applications of image registration are in the field of

1) Cancer detection: Image registration plays an important role in early detection of tumors. Localization of tumor is difficult with CT and MR scans because of the low intensity contrast between the tumor and the surrounding tissues. SPECT or PET imaging makes it possible to acquire high contrast images. However, a lack of sufficient anatomic detail in SPECT or PET limit us from determining the exact position of a tumor or other lesions, and thus necessitates image registration with CT or MRI.[1, 2]

2) Radiation therapy: Radiation therapy requires the fusion of PET/MRI images with CT images. The MRI/PET images provide localization of tumor and CT images are important in the calculation of radiation dose and treatment planning. Nonrigid image registration can also recover soft-tissue deformation introduced due to breathing artifacts and body motion during continuous scanning, which makes it very useful for tumor tracking.[3, 4]

3) Image-guided interventions: Minimally invasive image-guided interventions (IGIs) are increasingly being used in modern medicine. Since these procedures are planned using preoperative images and navigated using intraoperative images, there is often a need to register these two types of images. Since IGIs increasingly involve moving and deformable organs, image registration algorithms must be able to account for soft-tissue deformations. Thus, deformable image registration is a primary and integral necessity for the continued development of IGIs. Deformable image registration quantifies tissue motion on a voxel-by-voxel basis between two temporally separated scans of the same anatomy [5].

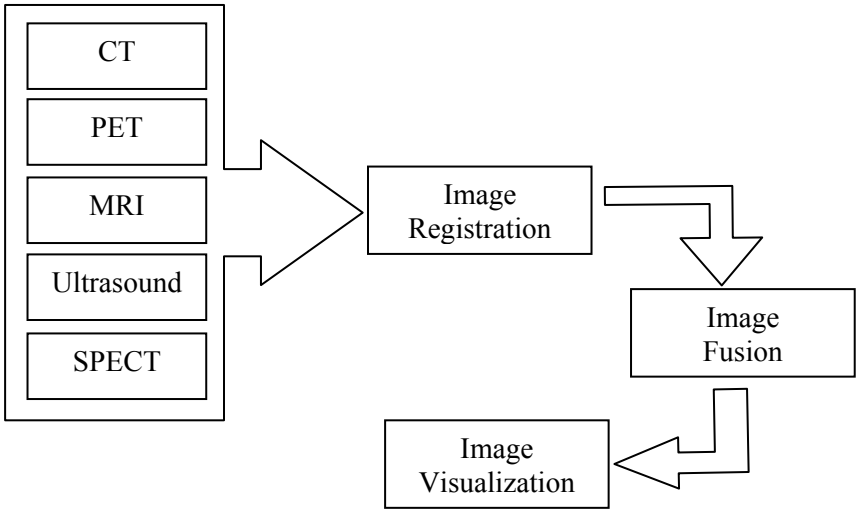

Figure 3 Workflow of image registration

1.1.2 Need for Faster Registration

Image registration can be classified as rigid and nonrigid registration algorithms. Rigid registration algorithms use global transformation, which is a combination of rotation and translation; affine registration algorithms, also a global transformation use a combination of rotation, translation, scaling or shear components. In nonrigid registration, one of the images is warped using a nonlinear transformation that best aligns one image with the other. During image guided tumor tracking, the tumor position, which varies depending on the breathing of the patients is recovered by registering the CT images taken at successive intervals of time (intra modality image registration). For such applications, the value of such deformable image registration for modeling the respiratory motion is becoming widely accepted [6]. The number of degrees of freedom in the case of affine registration is 12 (comprising of rotation, translation, shear and scaling in each of the three dimensions) but in case of nonrigid registration this number could be in the thousands. Unfortunately, the time to compute such a detailed deformation field can take a prohibitively long time through traditional computing. In such cases, where the time requirements are becoming very stringent, there is an increasing need to explore alternative computing schemes such as custom hardware to accelerate image registration.

Hybrid PET/CT scanning is now an essential tool for cancer diagnosis, staging and treatment planning and evaluation. Recently proposed nonrigid image registration algorithms have been shown to be equally effective in fusing standalone PET and CT scans as well as improving the fusion of hybrid PET/CT scans by removing breathing-induced misalignments [7]. Slow speed, however, is a limitation of the

algorithmic approach. Through hardware acceleration, the execution time of nonrigid image registration algorithms can be reduced to minute-order, allowing us to create PET/CT fusion images efficiently, from standalone scans. Also nonrigid soft-tissue misalignments that cannot be corrected with the use of hybrid PET/CT scanners can be corrected using nonrigid registration, leading to higher registration accuracy. Hybrid PET/CT scans can therefore also be processed using fast and robust registration hardware for further refinement of spatial matching. Thus, a hardware-accelerated image registration system can serve as an adjunct or as an alternative to hybrid scanners; providing PET/CT fusion capabilities to standalone PET, while enabling hybrid scanners to further refine PET/CT registration.

1.2 Computational Analysis of Image Registration

The software implementations of 3D nonrigid image registration algorithms are burdened by both computational load & memory access load. The former involves smooth B-splines interpolation at each voxel, whereas the latter comprises millions of random accesses to the memory for the evaluation of an intensity-based similarity measure used for registration during each iteration of the algorithm (chapter 2). During image registration, these steps have to be repeated numerous times (i.e., the algorithm involved numerous iterations) to arrive at the best transformation field. The number iterations is on the order of hundreds for the rigid registration part and on the order thousands for the nonrigid registration part.

While the first problem is mitigated with faster CPUs, evaluating an intensity-based similarity measure involves memory access load. Mutual information (MI), which has evolved as a robust intensity-based measure, has one of the stringent memory access

requirements (Chapter 2). The MI computation has two steps: first, each voxel of one of the two images (referred to as the reference image), is transformed into the other image space (referred to as the floating image) by applying a transformation and the intensity values in the floating image neighborhood are used to accumulate into a mutual histogram (also referred to as joint histogram) (MH). This process involves performing partial volume interpolation (Chapter 2) 'n' times, where n is the number of voxels in the reference image. The 'n' is on the order of 256^3, the typical size of a 3D medical image.. The second step involves calculating entropy from the values stored in the MH memory, which involves reading the MH memory 'm' times, where 'm' is the size of the MH memory. MH is a 2-dimensional array and thus 'm' is one-order of magnitude smaller than 'n'. Hence, compared to the second step, it is the first step that involves significant memory overhead since each coordinate that is transformed in the first step involves 1 read to the reference image memory, 8 reads to the floating image memory, and 8 read-accumulate-write operations to the MH memory.

Conventional CPUs cannot accelerate such highly memory intensive processes to a great extent since most of the accesses to the floating image and MH memories are random and do not benefit from cache-based techniques. Moreover, general-purpose processors are sequential in nature and cannot make full use of parallelism in the applications. Hardware implementations like those based on FPGAs have more freedom to fully exploit this parallelism and thus have the ability to achieve significant acceleration. It has been reported in [8] that approximately 99% or more of the registration time during rigid registration is spent on computation of MI, the

similarity metric. During nonrigid registration also an equally high percentage of time is spent in the computation of MI. Since the main bottleneck comes from the computation of MI, acceleration of MI computation though pipelining, parallel memory accesses and distributed processing is an essential path towards acceleration of image registration.

1.3 Previous Efforts at Acceleration

Image registration algorithms have evolved from rigid to affine to nonrigid in the quest to more accurately model bodily deformations. With the ever-present requirement for real-time processing, there has been an increasing demand for faster algorithms and for faster implementations capable of providing accurate results rapidly. Several groups have attempted to accelerate image registration. Many of these attempts, such as the ones suggested by Ourselin *et al.* [9] and Stefanescu *et al.* [10], are over a cluster of many CPUs with large amounts of memory. These systems are large and expensive with speedup per processor ration less than 1. Moreover, the image registration execution time is affected by dynamic factors like cache misses, making these systems' time behavior less predictable.

Similarly Warfield *et al.* [11] have implemented an image registration system for image guided neuro-surgery using 12 CPUs. The algorithm implemented is not a general algorithm, rather it is a volumetric deformation based algorithm, which is computationally less intensive and works efficiently for brain images because of high surface correspondences. More recently, Ino *et al.* [12] reported implementation of nonrigid image registration algorithm proposed by Rueckert *et al.* [13] using data distribution, data parallel processing, and load balancing techniques on a 128-CPU

cluster of PCs interconnected by Myrinet and Fast Ethernet switches. In all these systems the speedup per processor ratio is less than 1. Similarly, Rohlfing *et al.* [14] have suggested a shared-memory, multiprocessor-based computer architecture for the Rueckert's algorithm. This architecture provides accurate and fast results, but the net speedup per processor ratio was still significantly less than 1.

In general, these supercomputer-based solutions are not practical solutions in that their widespread clinical use can be cost and space prohibitive. A smaller, customized and much more compact system that provides identical results, albeit at a fraction of the cost is more appropriately suited for clinical use. Recently an implementation of a mutual information-based registration scheme on a Cell Broadband Engine (CBE) has been reported that can accelerate rigid registration [15]. Even though this method achieves rigid registration in less than 1 second, it utilizes a lot of simplifications like random sampling of image voxels. Also, this system is not able to perform nonrigid registration. Our reconfigurable nonrigid 3D registration architecture, implemented on an Altera Stratix FPGA, is unique, fast and accurate with a speedup of approximately 40x over general-purpose processors. Thus, our acceleration approach promises to play a crucial role in real-time image registration for a variety of applications like image-guided interventions. A schematic of our image registration system is shown in Figure 4.

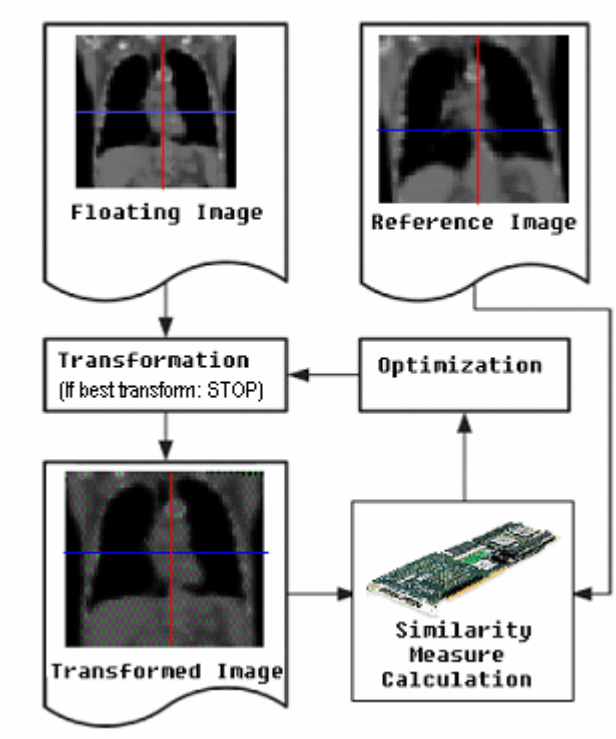

Figure 4 Image registration system

1.4 Objectives of the Thesis

The objective of this thesis is to provide key improvements on a custom processing unit built on a field programmable gate array (FPGA) that has a speedup per processor ratio of over 40, hence faster, more compact, more power conserving and significantly less expensive system resulting in a clinically viable, more economical system than the supercomputer implementations using parallel clusters of CPUs. This system is a result of the continuous development process both on the hardware as well as the algorithm side. On the hardware side we present the integration of the cubic interpolation pipeline that was presented in [16] into the existing hardware architecture that allows us to extract a more accurate deformation field required for nonrigid registration. Also the accuracy of the entropy calculation has been improved, which has resulted in the improvement of the registration times and made gradient

descent-based optimization feasible with a fixed-point implementation like ours (chapter 2). We have compared the accuracy of nonrigid registration in hardware with a software implementation of the same algorithm to validate the correctness of the hardware implementation. We find that the hardware arrived at similar solutions as those provided by the software implementation, but was over 40 time faster (Chapter 5).

During nonrigid registration, optimization of the deformation field used to warp one of the images is a challenging task. While using a free-form deformation based warping technique, we overlay a grid of control points on the reference image and optimize the deformation at these control points. In this process, we can choose to optimize the deformation at each of the control point individually or optimize the deformations at the entire control point space at the same time. On the algorithm side of our image registration system, we present in this work, a gradient descent-based optimization scheme, where the deformation field is optimized at all control points in unison in comparison with the previous implementation which used a downhill simplex-based optimization scheme to optimize each control point in a sequential manner. One of the main problems with using the free-form deformation model to model the nonrigid deformations (chapter 2) is the occurrence of folding. For example, in Figure 5, we find that point A has moved more than the physically permissible limits, causing tissues to fold upon one another. Without any preventive measures in place, there is a possibility of ending up with a solution which is not physically feasible. We had previously presented a scheme called 3D chainmail to prevent such mesh folding [16] based on a transformation applied to a single control

point. Rueckert *et al.* [13] have presented a smoothing prevention technique which constrains the motion at all the control points based on an energy measure called csmooth. This kind of folding prevention technique does not guarantee the prevention of folding, but only reduces the possibility of folding. Here we present a 3D global chainmail scheme which applies a set of constraints to the entire control point space to prevent mesh folding which makes it a better folding prevention technique in comparison with the ones like regularizing penalty term introduced by Rueckert *et al.* (chapter 4).

Our hardware makes use of several low-level optimizations that enhance the performance of the system. With the use of high-level modeling techniques like dataflow modeling, we can exploit certain dynamic behavior of the system. Finally, we present a dataflow based modeling technique with the use of dataflow graphs, which provides us an effective tool for automatically mapping image registration algorithms to reconfigurable hardware like FPGAs (chapter 6). This kind of modeling demonstrates the potential to use the dynamic behavior exposed by dataflow modeling with the low level optimizations present in the current implementation to arrive at very good implementation that provides maximum performance while consuming minimum resources.

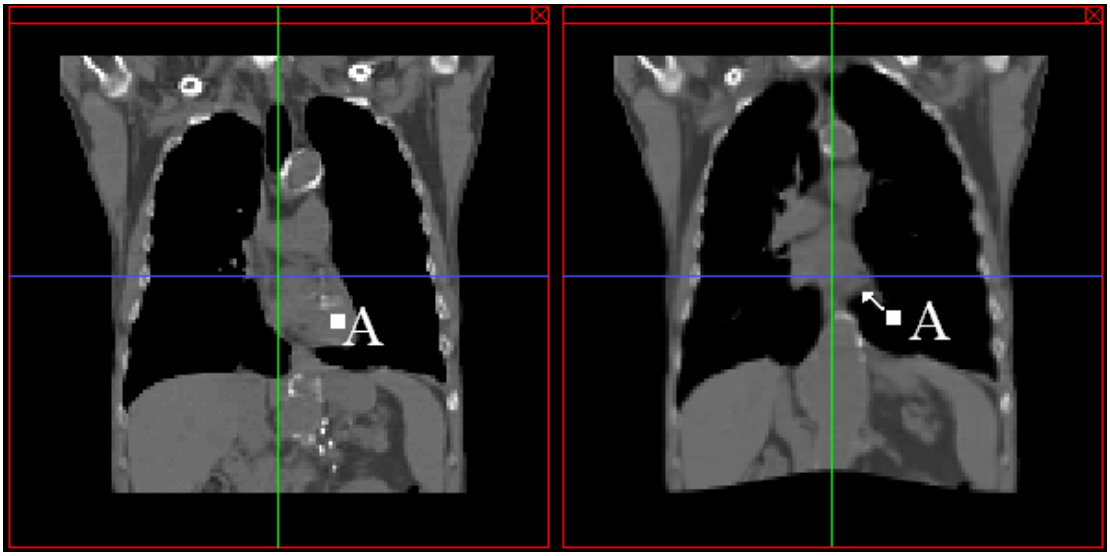

Figure 5 A synthetic example of Mesh Folding; the original image (left), and with mesh folding at point A (right) (arrow indicates the direction of folding).

2 BACKGROUND

2.1 Definition

Image registration is defined as the process of aligning two images that represent the same structure from different points of view, at different times or using different imaging techniques. The method attempts to find the transformation $\hat{T}$ that best aligns a reference image (RI), with coordinates x, y and z, and a floating image (FI).

$$\hat{T} = \arg\max_{T} \text{Similarity}\left(RI\left(x,y,z\right), FI\left(T\left(x,y,z\right)\right)\right) \quad \textbf{(1)}$$

The image registration process involves the choice of the transformation applied, choice of a similarity metric which measures how well the two images match and the choice of an optimization scheme which searches for the transformation that best aligns the two images.

2.2 Affine or Linear Registration

Affine registration or linear registration is a combination of rotation, translation, scaling and shear parameters that maps one of the images (FI) on to the other image (RI). The scaling parameters also incorporate voxel scaling necessary to compare images with different voxel sizes. Voxel scaling factors are constant for rigid registration and as such are excluded from optimization. The remaining parameters like rotation, translation and shear parameters are estimated using an optimization scheme which searches for the best transformation matrix $\hat{T}$ which maps the FI to the RI. The transformation that maps a 3D floating image to the reference image is

defined by a 4 × 4 transformation matrix T_{global}. Rigid registration involves a combination of rotations and translations only. Figure 1 provides an example of rigid registration.

$$T_{global} = \begin{vmatrix} r_{xx} & r_{xy} & r_{xz} & d_x \\ r_{yx} & r_{yy} & r_{yz} & d_y \\ r_{zx} & r_{zy} & r_{zz} & d_z \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad \textbf{(2)}$$

The location of a given RI voxel $\overline{v}_{RI}$ in the FI is given by:

$$\overline{v}_{FI} = T^{-1} \cdot \overline{v}_{RI} \quad \textbf{(3)}$$

2.3 Nonrigid or Elastic Registration

While rigid registration recovers mostly global misalignments, nonrigid or elastic registration is required to recover nonlinear deformations which cannot be recovered with rigid registration. A common method for nonrigid registration is to separate the local body deformation into a linear component and a nonlinear component. The linear component corresponds to the global motion (T_{global}) and is fixed for the entire image. For local deformation, we model the nonlinear component in our transformation model as:

$$\vec{v}_{FI} = T_{global} \times \left(\vec{v}_{RI} + \vec{v}_{local}(\vec{v}_{RI}) \right) \quad \textbf{(4)}$$

where $\vec{v}_{RI}$ is the location of a voxel in the RI; $\vec{v}_{FI}$ is its corresponding location in the FI; and $\vec{v}_{local}(\vec{v}_{RI})$ is the value of the local deformation field at $\vec{v}_{RI}$. In practice,

during registration, we transform the coordinates of the reference image even though we write the equations for the floating image. Figure 2 provides an example of nonrigid registration.

2.3.1 B-splines Based Deformation Field

It is becoming widely popular to use a free-form deformation model based on cubic B-splines for modeling local soft tissue deformations. Quoting Shekhar et al [6] "To model the local soft-tissue deformations, thin plate splines (TPS) and B-splines are used extensively. The TPS algorithm, first presented by Meyer *et al.* [17], suffers from relative sparseness of control points which affect registration accuracy and has the need for identifying landmarks for control points. The free-form deformation (FFD) model based on B-splines, developed by Rueckert *et al.* [13] addresses both these issues by placing a dense grid of automatically selected control points over the images and using these control points to deform the image." B-splines have a number of properties like a finite volume of support, and ability to model smooth transitions in the deformation field which make it suitable to model local tissue deformations. In these methods, the deformation field value at a given point is calculated using a linear combination of cubic B-splines placed on a regular grid of $n_i \times n_j \times n_k$ control points $\phi_{i,j,k}$, with $i < n_i$, $j < n_j$, $k < n_k$ and grid spacing $\delta_x(t)$, $\delta_y(t)$, and $\delta_z(t)$.

$$\vec{v}_{local}(x,y,z) = \sum_{l=-1}^{2}\sum_{m=-1}^{2}\sum_{n=-1}^{2} B_l(u)B_m(v)B_n(w)\phi_{i+l,j+m,k+n} \qquad \textbf{(5)}$$

where $i = \lfloor x/\delta_x(t) \rfloor$, $j = \lfloor y/\delta_y(t) \rfloor$, $k = \lfloor z/\delta_z(t) \rfloor$, $u = x/\delta_x - i$, $v = y/\delta_y - j$, $w = z/\delta_z - k$, and the basis functions can be written as:

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

where r is either u, v, or w and t is the current grid resolution level.

We implement our deformation model based on B-splines based deformation field along with specific algorithmic improvements proposed within our group.

2.3.2 Folding Prevention techniques

A problem with the use of FFD-based deformation field is the chance of folding of control points called mesh folding which occurs when one control point crosses over the other. This mesh folding represents a violation of the topology of the original distribution of control points causing a situation as shown in Figure 5. There have been various efforts to address this issue like using the Jacobian of the vector field [18], or constraining the control point from moving beyond a particular radius [19]. In the first case, the constraints are phrased as the determinant of the Jacobian of the deformation field. Finding a solution for such differential inequality constraints is more computationally intensive in 3D. In the second case, each control point is simply constrained to move within a local sphere of radius. This kind of radial constraints may not work well with all body organs and is not suitable for a generic registration algorithm like ours. The constraints applied should take care of relative deformations between control points rather than constraining each control point within a particular radius.

Rueckert et al. [13] have proposed a smoothing technique based on a regularizing parameter called csmooth, which reduces the possibility of folding based on the second derivative of the deformation field. However these methods cannot prevent folding but can only limit large movements of control points. This method also involves choosing parameters for which no suitable technique exists and have to be chosen manually. To prevent mesh folding, we use a robust folding prevention scheme based on 3D Chainmail, which we have presented earlier [16]. 3D Chainmail imposes a number of geometric constraints on the movement of a control point with respect to its neighboring points. These constraints, like minimum and maximum distance between adjacent control points, maximum shear between planes, control the stretching of the control points along each of the 3 principal directions and shear perpendicular to them. If the movement of a control point violates any of the limits, its neighboring vertices are moved in tandem to satisfy the limits. Thus the deformation at every control point is applied maintaining the original topology between the control points while at the same time, restricting any mesh folding artifacts.

2.4 Similarity Measures

An important problem in image registration is determining how similar the two images are; since it indicates to what degree the images are aligned. Two main approaches for determining a measure of similarity are those based on intensities of the image voxels and those based on features present in the images [20]. Feature-based measures take into account different shapes or structures in the two images, like points, curves or surfaces. Usually this kind of scheme in involves feature extraction,

or segmentation, requiring some degree of user interaction and is therefore not a good choice for fully automatic registration. The complexity of feature based registrations is usually low, but more time is spent on segmentation of different features which are also subject to errors.

Intensity-based measures, on the other hand, only operate on the voxel values making it more memory and computationally intensive, but these measures eliminate the need for feature segmentation and manual interaction making them best suited for fully automatic registration systems. Hence, intensity-based similarity measures are gaining more popularity in registration. These measures can be further classified as measures using only image intensities, measures using spatial information (i.e. intensities in a voxel's neighborhood) and histogram-based measures. However in our implementation, we use mean square error (MSE) which is the simplest of all the similarity based measures and mutual information (MI) which has been widely accepted as the most robust and accurate currently known image similarity measure for image registration [21].

2.4.1 Mean Square Error (MSE)

Mean square error (MSE) is defined as

$$\mathrm{MSE} = \sqrt{\frac{\sum_{i=1}^{N} \left(RI_i - FI_i\right)^2}{N}} \qquad \textbf{(7)}$$

where RI_i and FI_i are the reference and the transformed floating image intensity values respectively, and N is the total number of voxels in the volume of overlap. The MSE is the smallest when the images are perfectly aligned. This measure is relatively

less computationally intensive than other intensity based similarity measures like MI, but is suitable for images of the same modality only.

2.4.2 Mutual Information

Mutual Information as a similarity measure was introduced by Wells *et al.* [22]. MI between two images is a measure of the amount of information they contain about each other [23]. MI based image registration relies on the maximization of the MI between the two images and is a function of the two 3D images (RI(x,y,z) with coordinates x, y and z, FI(x,y,z)) and the transformation field $\hat{T}$ between them.

$$\hat{T} = \arg\max_{T} MI\left(RI\left(x,y,z\right), FI\left(T\left(x,y,z\right)\right)\right) \quad \textbf{(8)}$$

MI is calculated from individual and joint entropies using the following equation.

$$MI\left(RI,FI\right) = H\left(RI\right) + H\left(FI\right) - H\left(RI,FI\right) \quad \textbf{(9)}$$

The individual entropies, $H(RI)$ and $H(FI)$, and the joint entropy, $H(RI,FI)$, are computed as follows:

$$H\left(RI\right) = -\sum_{a} p_{RI}\left(a\right)\ln p_{RI}\left(a\right) \quad \textbf{(10)}$$

$$H\left(FI\right) = -\sum_{b} p_{FI}\left(b\right)\ln p_{FI}\left(b\right)$$

$$H\left(RI,FI\right) = -\sum_{a,b} p_{RI,FI}\left(a,b\right)\ln p_{RI,FI}\left(a,b\right)$$

The joint voxel intensity probability $p_{RI,FI}\left(a,b\right)$ is the probability of a voxel in the RI having an intensity a given that the corresponding voxel in the FI has an intensity

b. It can be obtained from the joint or mutual histogram of the two images. The mutual histogram represents the joint intensity probability distribution. MI-based registration aims at reducing the dispersion of values within the mutual histogram. Minimizing the dispersion of values corresponds to higher probability values in equation (10) which in turn corresponds to lower joint entropy values. Thus minimizing the dispersion corresponds to maximizing the MI.

The calculation of MI starts with the accumulation of the mutual histogram (MH) values to the MH memory while each RI coordinate is being transformed. In this stage, the following steps have to be repeated for each voxel in the RI

1) Apply a deformation field on the RI and its coordinates in the FI.

2) Load corresponding FI 2x2x2 neighborhood around the transformed FI coordinate.

3) Calculate the 8 PV interpolation weights from the fractional part of the transformed FI coordinate which correspond to the FI neighborhood obtained in step 2.

4) Accumulate interpolation weights into corresponding MH bins

This is followed by the MI calculation stage where the values stored in the MH memory are used to find the individual and joint entropies; this ensures that only voxels inside the volume of overlap figure in the MI calculation. MI as a similarity measure can be fully automatic and applicable to single or multimodality images of most organs making it suitable for application in a general image registration system.

2.5 Interpolation Techniques

During registration process, when a transformation is applied to a coordinate, the transformed location of a voxel of the RI may not coincide with the location of a voxel in the FI. In such cases interpolation is needed to account for subvoxel accuracy. In this regard nearest neighbor interpolation scheme is disregarded as it fails to achieve subvoxel accuracy. Both trilinear interpolation and partial volume (PV) interpolation, suggested by Maes *et al.* [21], can achieve subvoxel accuracy.

Trilinear interpolation involves calculating a resulting intensity level and incrementing that intensity level's MH count by one. In this method a new intensity level is introduced as a result of interpolation, causing unpredictable variations in MH values. On the other hand, PV interpolation accumulates the eight interpolation weights directly into the MH producing a mutual histogram, whose values change smoothly with small changes in the transformation, thus resulting in a smoother MI surface. Even though both these schemes involve accessing the 8 neighborhood of a FI voxel where the transformed coordinate is mapped, PV interpolation has stringent memory access requirements on the MH memory compared to trilinear interpolation because of the accessing the MH memory for accumulating all the 8 weights separately. Capek *et al.* [24], in their experiments, studied the smoothness of the MI surface when using different interpolation schemes for MI and concluded that MI, computed according to Maes, provides the smoothest MI surface among statistical voxel similarity measures.

2.6 *Introduction to FAIR Architecture*

Fast Automatic Image Registration (FAIR) architecture is a custom architecture designed by our group for FPGA implementation to accelerate image registration by accelerating MI calculation. The main bottleneck areas in image registration are the cubic B-splines interpolation and the MI calculation stages which have a lot of computational and memory access loads. In order to reduce the overall registration time we perform the above two tasks on a FPGA which contains customized architecture for acceleration. The nature of the above two tasks allows it to be programmed on a FPGA using deep pipelines and parallel units. The optimization routine is maintained on the CPU since the optimization routine contains a large amount of logical decisions to make which makes it best suited for general purpose processors.

The first generation architecture, FAIR-I, focused on acceleration of the MH calculation, by acceleration of PV interpolation. In the second-generation FAIR-II architecture, the entropies were also computed on-chip in order to reduce the communication overhead introduced in transferring the MH back and forth between FPGA and the host CPU through a peripheral component interconnect bus (PCI). We also presented a novel way of calculating the function, *plog(p)*, necessary for entropy calculation, in fixed point hardware using look-up tables (LUTs) and piecewise approximations.

2.7 *FAIR-I*

The FAIR-I architecture [8] was implemented on an external prototype board with a FPGA (Altera ACEX 1K100) for performing rigid registration. Each board

contained two processing units using two FPGAs, SDRAMs for storing images and SRAMs for the MH memory and communicated with the host computer using a PCI bus. Since the MH memory has the stringent read and write requirements (8 reads and 8 writes for each RI voxel processed), MH memory is best suited for the SRAM implementation compared to the dynamic RAMs used for storing the 2 images. In this system, PV interpolation was implemented using a 32 bit, fixed-point approach. 8 bits were used for the fractional part providing an accuracy of $1/256^{th}$ of a voxel dimension. This system was able to deliver an average speedup per processor ratio of approximately 8 compared to a software implementation on a 1 GHz Pentium III computer. Figure 6 shows the processing unit of the FAIR-I architecture.

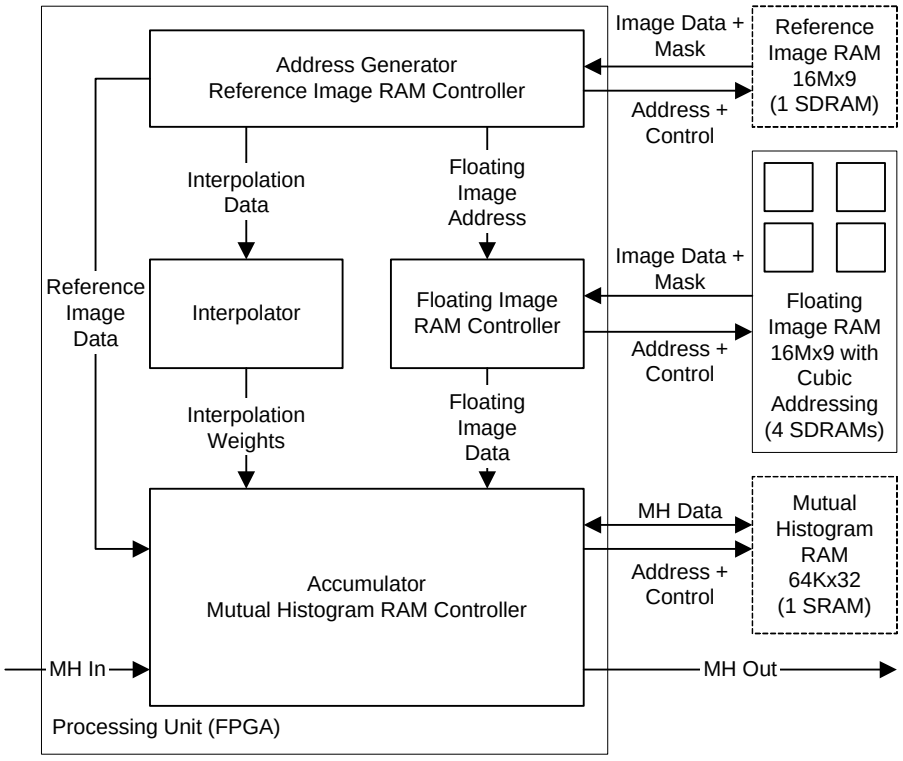

Figure 6 FAIR-I architecture Source:[8]

2.7.1 Pipelined Architecture

The FAIR architecture (Figure 6) derives its speedup from pipelining of different stages of the design. Within each design, processing of different data entries is pipelined. During PV interpolation, in the first stage, a coordinate transform is applied to all the RI voxels. Processing of these RI coordinates is also pipelined. In the next stage, the integer part of the transformed coordinate from the first stage is used to fetch the 8 neighborhood image intensities from the FI. The fractional part of the transformed coordinates provides the interpolation weights required for PV interpolation. The third stage uses the RI intensity, FI intensities and the interpolation weights to accumulate into a MH memory. The independence of each of these stages allows the design to make effective use of the pipelined architecture.

2.7.2 Parallel Memory Access Scheme

In a system with pipelines stages, the latency of the pipeline is defined by the stage with the slowest stage, and thereby we need to improve the speed of this stage. In our design the presence of memory accesses presented us with stringent memory access requirements. The RI is accessed once per voxel sequentially and therefore benefited from burst accesses. However the FI is accessed in a random order, depending on the transformation applied and therefore does not benefit from burst accesses. Also 8 neighborhood image intensities need to be read for each voxel. In the FAIR-I architecture we implemented a novel cubic memory addressing technique similar to the ones used in volume rendering hardware [25] to access this FI neighborhood.

2.7.3 Distributed Processing

With the use of multiple FPGA units, the processing of the RI voxels can be distributed across different FPGAs. MH accumulation lends itself to be parallelized by dividing the RI into smaller subvolumes. Each FPGA must contain necessary RAM to store its part of RI, complete FI and MH memories. The partial mutual histograms from different FPGAs are then merged before calculating the MI value. This is made possible by having an input port and an output port on each FPGA, where input port takes in the partial MH from the previous unit and the output port transmits the results to the next unit or to the host computer.

2.8 FAIR-II

The FAIR-I architecture accelerated rigid registration by accelerating MH calculation. In its complete design, the FAIR-II architecture concentrated on accelerating linear and elastic registration. In the first development of the FAIR-II architecture [26], a major communication bottleneck arising from the transfer of the MH to the host PC was solved by the calculation of the MI (which includes a nonlinear function *plog(p)*) on the FPGAs itself. Use of dual ported memory which can perform simultaneous read and write for the MH memory further increased the speed of MH. In its further development of FAIR-II, we present the integration of the cubic interpolation pipeline [16] for performing cubic B-splines interpolation required for nonrigid registration (presented in the next subsection). Figure 7 provides us a schematic diagram of the FAIR-II architecture.

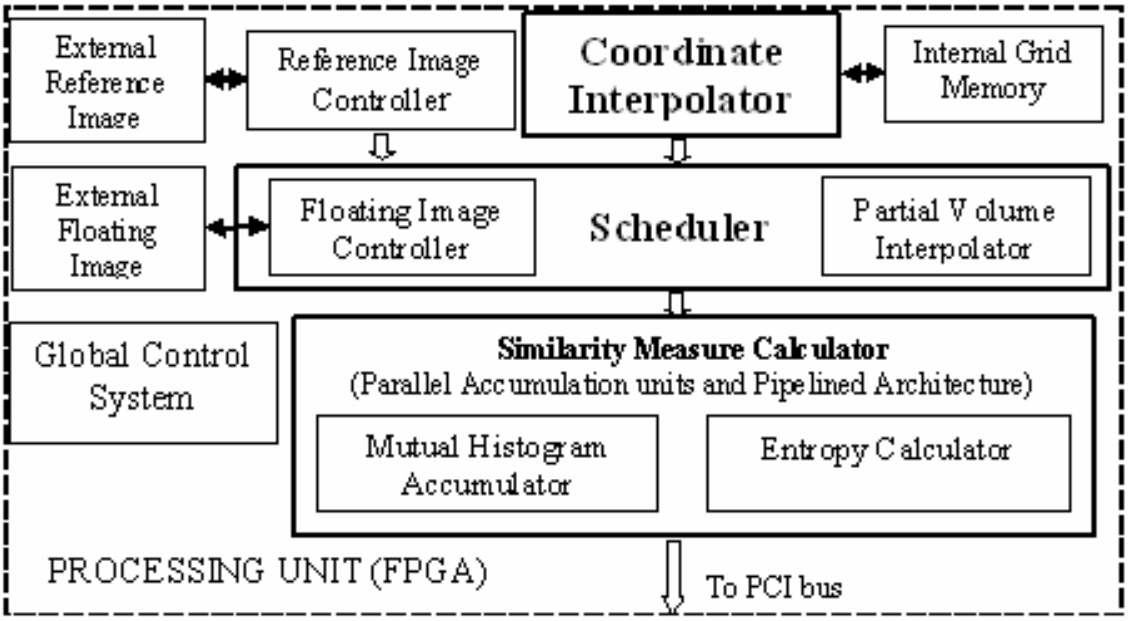

Figure 7 Schematic diagram of the FAIR-II architecture.

2.8.1 Mutual Histogram Computation

The FI is stored using a special scheme using multiple image copies to allow parallel access of 4 in-plane neighborhood values. For achieving this, we stored the FI intensities corresponding to the 4 in-plane neighborhood image intensities at every voxel position instead of just storing the intensity value of that voxel. Each image intensity value being 8 bits, and memory bus being 32 bits, the 4 in-plane values could be read in a single clock cycle, thereby allowing us to read the 8 neighborhood values in 2 clock cycles. This parallel memory access scheme multiplied the memory requirements, however reducing the memory access times. With 64 bit memory buses available on latest FPGA boards, and higher memories available we can have the entire 8 neighborhood values stored in the same manner thus reducing the FI access times by half. The MH accumulator has to use up these four values to accumulate into the MH memory, so as to prevent buffer overflow conditions. Thus we need to accumulate 4 weights to the MH memory in every clock cycle. This is achieved by

distributing the MH accumulation on four parallel partial MH accumulator entities that form the MH accumulation unit. Having 4 parallel accumulation units also mean that the MH memory size is now quadrupled. However since each copy of MH memory is of 20KB, having a block of memory of size 80KB is possible within the FPGA internal memory. The MH memories are dual ported which enable simultaneous read and write to the MH memory. To avoid possible read after write hazards in the accumulation pipeline, a possible situation being when the reference and/or FI intensities are constant in a neighborhood, the scheduler groups such same intensities values and pre-accumulates before performing the accumulation into the MH memory. The values from these partial MH memories are merged into a single MH memory before MI calculation. The FI histogram and the RI histogram are also accumulated into their respective FI and RI histogram memories. A global counter keeps count of the total number of valid voxels processed which is necessary to find the probabilities from the histogram values.

2.8.2 Entropy Accumulation

The entropy accumulation stage follows the calculation of MH. The MI, individual and joint entropies are computed from the MH as in equations (9,10). As evident from equation (10) it is necessary to evaluate the function $f(p)= plog(p)$ for the individual and joint image intensity probabilities. Calculation of a nonlinear function like $log(p)$, which is unbounded, requires infinite precision. Hardware implementation of logarithm calculation use look-up tables (LUTs) to approximate the values using series approximations. In our implementation the value of probability ‘p’ ranges from [0 1] where the function is continuous and bounded. This makes approximation of the

function *f(p)* using piecewise-polynomial function possible. FAIR-II [26] implements a Chebychev's approximation for *f(p)* which is simple to calculate and is very close to the minimax solution of the approximation function. (Appendix 1 provides the equations for Chebychev's approximation used in the implementation). Using a first order approximation for the *f(p)* we have equations (11,12) to calculate the two 32-bit coefficients, $k_1(i)$ and $k_0(i)$ stored in the position 'i' in the LUTs.

$$\begin{aligned} f(p) &= p \circ \log(p) \\ f_0(i) &= f\left(\Delta p(i+0.5) + \frac{\Delta p}{2\sqrt{2}}\right) \\ f_1(i) &= f\left(\Delta p(i+0.5) - \frac{\Delta p}{2\sqrt{2}}\right) \end{aligned} \tag{11}$$

$$\begin{aligned} k_o(i) &= \frac{1}{2\sqrt{2}}\left((\sqrt{2}-1) f_0(i) + (\sqrt{2}+1) f_1(i)\right) \\ k_1(i) &= \frac{\sqrt{2}}{\Delta p}(f_0(i) - f_1(i)) \end{aligned} \tag{12}$$

The LUTs are addressed by p_d, where $p_d = p \bmod \Delta p$. The approximate values calculated from the coefficients stored in the LUTs are given by equation (13).

$$P(p_d) = k_0 + k_1 p_d \tag{13}$$

2.9 Summary

In this section, I present an introduction to the different methods and techniques used for nonrigid image registration and the motivation for choosing them. Our primary goal is to accelerate an image registration algorithm that is able to recover nonrigid misalignments as accurately as possible, while at the same time being able to register images of mostly all parts of the body and across different imaging modalities. In implementing such a generalized algorithm, I present the different challenges and difficulties encountered during our ongoing efforts at accelerating image registration. In this chapter, I highlight the development work of a customized hardware architecture for image registration before I started working on this project. In the successive chapters, (Chapters 3,4) I present the improvements I have made to this project both on the hardware side (Chapter 3) and on the algorithmic side (Chapter 4).

3 HARDWARE IMPROVEMENTS TO THE ARCHITECTURE

3.1 Introduction

In this section, I present the key hardware improvements I have made to the FAIR-II architecture that allows us to perform nonrigid registration in hardware. In previous FAIR implementations, a global transformation unit was used on the hardware to transform the coordinates of the RI while calculating the MH. We use free-form deformations based on cubic B-splines interpolation on a regularly spaced grid to warp the image in an elastic manner during nonrigid image registration. For 3D image warping on hardware, a cubic interpolation pipeline was presented in [16] that accelerates the computation of 3D deformation fields for application where image warping is necessary. In this work, I integrate this cubic interpolation pipeline with the existing FAIR-II architecture to replace the affine coordinate transform unit that existed with the previous FAIR-II implementation. This change enabled me to apply a nonrigid deformation field to the RI and perform computation of the similarity measure entirely on hardware (Section 3.2).

During MI computation, the probability values are derived from mutual histogram. The values of the probabilities are a function of the images, size of the MH and the transformation applied and vary over the range of [0,1]. The hardware implementations of the entropy calculator depend entirely on the size of the LUTs that hold the coefficients used for Chebychev approximations (Section 2.8.2). For each of the probability value, there is a finite error that is introduced due to the limited precision of the entropy calculator. This error, accumulated over all the

probability values in the MH, causes an error in the MI value. This error in MI value causes unwanted variations in the path the optimization routine takes while finding the best deformation field. In this work, I have increased the accuracy of the entropy calculation pipeline (section 3.3) which has resulted in reduction in the registration times and leading to better results. I was able to reduce the error magnitude in the *plog(p)* calculation from 10^{-4} to 10^{-8} which also enabled gradient descent based optimization which requires very precise entropy calculation for calculating the gradients

3.2 Cubic Interpolation Pipeline for B-splines Based Deformation Fields

Rueckert *et al.* [13] suggested using B-splines to model deformation fields for nonrigid registration. B-splines based deformation field is well suited for deforming medical images during nonrigid registration because of several advantages like a finite volume of support, and their ability to model smooth transitions in the deformation field. Equations (5,6) provide the equations for finding the value of deformation field as a linear combination of cubic B-splines placed on a rectangular grid. In [16] we have presented an efficient cubic B-splines interpolation pipeline that can be used for 3D image warping. In this section I present the integration of this elastic deformation unit with the rest of the system to provide us with a complete nonrigid image registration system.

3.2.1 Control Point Grid Memory

The previous rigid registration implementation needed to store 12 parameters for the transformation vector. However supporting nonrigid registration requires the

careful consideration of how each of the potentially thousands of control points are stored. Since we needed nonsequential and fast access to the control point memory (also referred to as grid memory), the internal SRAM was used to store the control point grid memory. The size of the memory should be able to accommodate the number of control points at the finest resolution, where each entry which indicates the x, y and z displacements at each control point. The control points were stored in the two internal 512Kb RAM blocks. The integer part of the deformation at each dimension depends on the maximum permissible deformation at each control point and the fractional part determines the subvoxel accuracy of the deformation at each control points. For cubic interpolation of voxels in the boundary cases, where we need control points that lie outside the boundaries of the image, we replicated the control points that lie on the boundary by intelligent address generation technique and thus, eliminating the need for additional memory.

3.2.2 Interpolation Kernel Calculation

While the rigid registration engine only needed simple linear transformation capabilities, B-splines interpolation is needed for nonrigid registration. Using a linear combination of cubic B-splines placed on a regular grid of $n_i \times n_j \times n_k$ control points $\phi_{i,j,k}$, with $i < n_i$, $j < n_j$, $k < n_k$ and grid spacing $\delta_x(t)$, $\delta_y(t)$, and $\delta_z(t)$ we can compute the deformation at every voxel x,y,z as *given in equations (5) and (6)*

$$\vec{v}_{local}(x,y,z) = \sum_{l=-1}^{2}\sum_{m=-1}^{2}\sum_{n=-1}^{2} B_l(u)B_m(v)B_n(w)\phi_{i+l,j+m,k+n} \qquad \textbf{(5)}$$

where $i = \lfloor x/\delta_x(t) \rfloor$, $j = \lfloor y/\delta_y(t) \rfloor$, $k = \lfloor z/\delta_z(t) \rfloor$, $u = x/\delta_x - i$, $v = y/\delta_y - j$, $w = z/\delta_z - k$, and the basis functions are given as:

$$\begin{aligned} B_{-1}(r) &= (1-r)^3/6 \\ B_0(r) &= (3r^3 - 6r^2 + 4)/6 \\ B_1(r) &= (-3r^3 + 3r^2 + 3r + 1)/6 \\ B_2(r) &= r^3/6 \end{aligned} \quad \textbf{(6)}$$

where r is either u, v, or w and t is the current grid resolution level.

We can rewrite the $\vec{v}_{local}(x, y, z)$ as a combination of equations (14, 15 and 16). For cubic B-splines interpolation in hardware, we have to calculate the basis functions required for interpolation. Utilizing the fact that the control point grid structure that we overlay on the reference image is parallel to the coordinate axes, we can precompute the basis functions. Figure 8 illustrates the precomputation of the basis vectors for a 2D case. Within the control point space shown in the figure, we can see that the values of i,j remain constant for all the pixels within the shaded region. The basis functions for all the pixels in the row can be pre-computed as 'v' remains constant for all pixels within the horizontal row. Extending this to 3D, for a given plane, the basis functions for all the voxels in a given plane can be pre-computed using equation 14 and for all the voxels in row of the plane can be pre-computed using equation 15. The range of these basis functions B(u) is [0,1] which makes it suitable for hardware implementation. A simple LUT based implementation for calculating the basis functions is best suited as the maximum number of voxels in any direction is a finite value and can be stored for different values of 'r' in the

equations for the basis function given above. Using LUTs is also beneficial, as it provides us reconfigurability to change interpolation kernels.

It has been shown in [16] that dividing the image into subvolumes, where each control point forms the corner of the subvolume, and traversing through these subvolumes, minimize the number of evaluations of the B-splines basis functions and provide a more efficient hardware implementation.

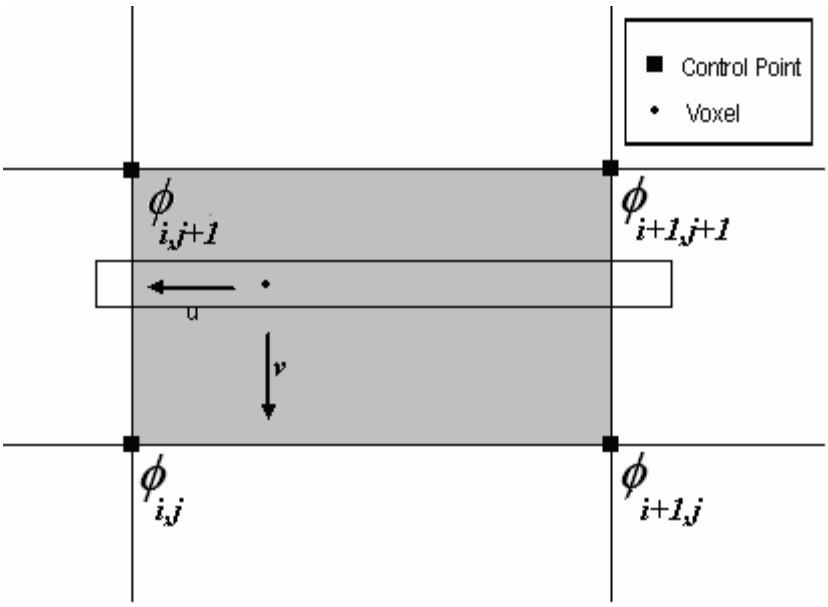

Figure 8 Precomputation of the basis vectors for a 2D case; can be extended into 3D similarly Source: [14] .

The interpolation pipeline is provided in Figure 9 and is divided into 3 stages. In stage 1, the z-coefficients given by equation (14) are calculated. These coefficients are constant for a given plane in a subvolume and thus, are held constant for the duration of processing of a plane of the subvolume in the z-coefficient buffer. While a plane is being processed, the coefficients of the next plane are being calculated. Calculation of the coefficients for the subsequent planes is paused till the coefficients

in the buffer are completely utilized. This ensures minimal latency. Similarly in stage 2, the y-coefficients for different rows in a plane are calculated as given in equation (15). The y-coefficient buffer holds the coefficients till the current row is completely processed, while at the same time, coefficients for the next row are being calculated. In the stage 3, the final deformation at every voxel is calculated using the row coefficients from the previous stage buffer as given in equation (16). Each stage contained their respective LUTs.

$$Zcoeff_{l,m} = \sum_{n=-1}^{2} B(w)\phi_{i+l,\,j+m,k+n} \text{ , in a subvolume } S_{i,j,k} \qquad \textbf{(14)}$$

$$Ycoeff_{l} = \sum_{m=-1}^{2} B(v) Zcoeff_{l,m} \qquad \textbf{(15)}$$

$$\vec{v}_{local}(x,y,z) = \sum_{l=-1}^{2} B(u) Ycoeff_{l} \qquad \textbf{(16)}$$

On top of the three stages, I have added an additional stage, stage 4, as shown in Figure 9 to the interpolation pipeline. The $\vec{v}_{local}$, from stage 3 provides the deformation field which constitutes the nonrigid portion of the coordinate transform applied to the RI. In the stage 4, this deformation field is combined with a global transformation, T_{global}, according to equation (4). The optimal global transform parameters applied at stage 4 is pre-computed during rigid registration (which precedes nonrigid registration) and stored in the internal memory of the FPGA. The output of stage 4 provides the final deformation field that is applied during the calculation of the MH.

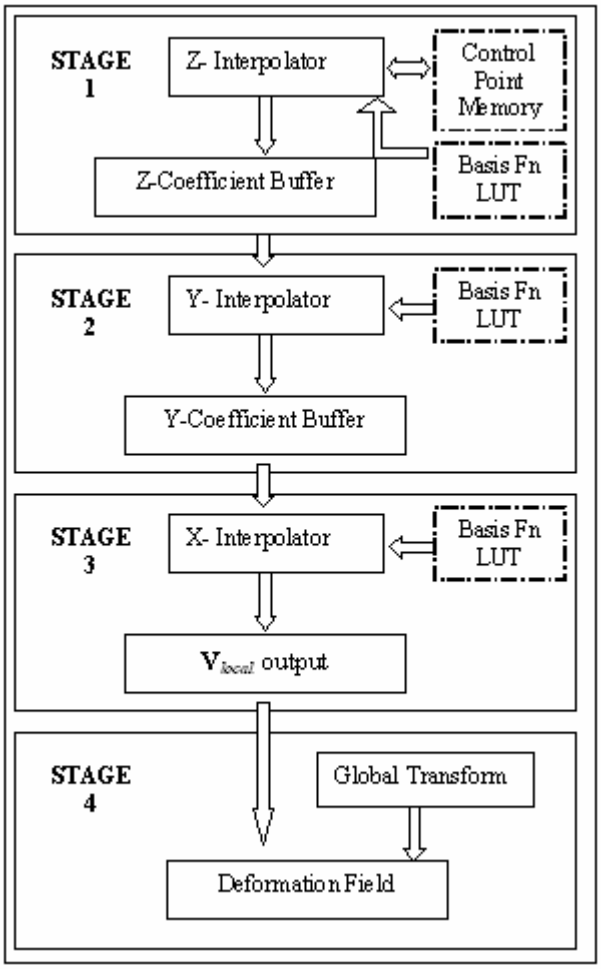

Figure 9 Coordinate transform unit with the cubic interpolation pipeline

3.2.3 Discussion

The FAIR-II with the cubic interpolation pipeline was implemented on an FPGA PCI board. While the cubic interpolation unit was able to run at 150 MHz independently [16], the FAIR-II architecture runs at a speed of 50 MHz, thus making it possible to integrate the pipeline into the system flow without any timing hazards. Also the output rate of the cubic interpolation unit as presented in [16] is one voxel per clock cycle. In FAIR-II, the voxel processing rate is one voxel per two clock cycle (without taking into account the latency and refresh cycles of the SDRAM) because of the FI neighborhood read requirements discussed in section (2.8.1). The

effective output rate of a pipelined system is limited by the output rate of the slowest unit. Thus the output rate of this pipeline is limited to one voxel every two cycles.

The control point memory is loaded with a trial deformation field obtained from the optimizer running on the host PC. Such a design allows us to optimize either one control point at a time, or all control points at the same time (chapter 4). The accuracy of the deformation field depends on the number of bits chosen for storing the coefficients in the basis function calculation LUTs and the number of bits stored in the fractional component of the grid memory; in our case these numbers being 9 bits and 2 bits respectively which correspond to a subvoxel accuracy of 0.125 voxel as shown in [16]. Using the basis functions and the grid memory, 8-bit deformation field at every voxel position is calculated through stages 1-3 and is passed from stage 3 to stage 4. This corresponds to an accuracy of 0.004 of a voxel dimension. The grid spacing (number of control points between two control points) at the finest grid resolution is 16 (minimum should be 8), thus ensuring that there are more than 64 voxels at each plane. Thus the pipeline throughput of stage 1 is not affected thereby preventing buffer under-run conditions. Similarly, by maintaining grid spacing of 16 in all three dimensions, buffer under-run conditions can be avoided in stage 2 also. Buffer overflow condition is prevented by halting the calculation of the coefficients for the subsequent plane/row till all the coefficients in the coefficient buffers are completely consumed.

3.3 Entropy Accumulation

One of the important steps in the calculation of the MI is the calculation of the individual and joint entropies. Entropy calculation relies on the *plog(p)* calculation,

where p is the probability distribution derived from the individual and joint histograms calculated during the coordinate transform stage. Since the values of probability 'p' range from [0,1] the values of the log(p) range from [$-\infty$, 0]. However, fortunately, the values of *plog(p)* are bounded within the range [$-e^{-1}$, 0] and is continuous within this range of 'p' which makes it suitable for easier hardware implementation using linear approximations. In our previous implementation, we have used LUTs based on Chebyshev approximations which are simple to calculate for continuous functions and are very close to the actual solution. In the previous FAIR-II architecture, we had a 2 LUT implementation with 64-bit 1K entries in each LUT, where each entry consisted of the two 32-bit coefficients (k_0,k_1) required for the piecewise linear approximation in the first order [27]. The range of the first LUT was from [0 0.25] while the range of the second LUT was from [0.25 1].

3.3.1 Errors Due to *plog(p)* Approximations

With the resources on the FPGA being limited which have to be shared among different components of the system, there is a tradeoff between the accuracy of the *plog(p)* pipeline and the memory availability to store the values in LUTs required for piecewise approximations. Figure 10 shows the accuracy of the *plog(p)* calculation in the FAIR-II architecture with a 2 LUT implementation; the magnitude of the error for each value of 'p' is plotted on a log scale.

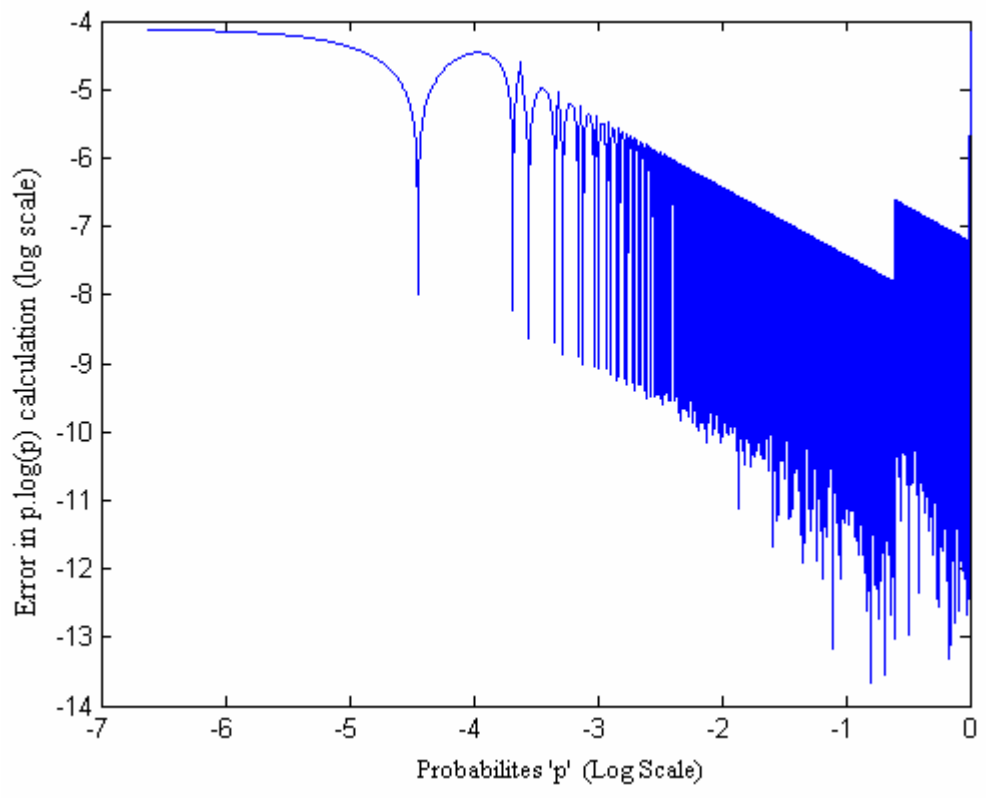

Figure 10 Error in *plog(p)* calculation with a 2 LUT implementation

The error is calculated as follows

$$\varepsilon_i(p) = P_i(p \bmod \Delta p) - f(p) \qquad \textbf{(17)}$$

where *f(p)* is the actual function value and $P_i(p \bmod \Delta p)$ is the estimated value from the LUT-based hardware implementation. We find that the error magnitude is on the range of 10^{-4}, with the error magnitude decreasing for higher values of 'p' as we move along the LUT. Whenever we switch LUTs, we see a repetition in the error pattern.

3.3.2 Error Accumulation During MI computation

The main problem in the MI computation with the above method arises from the calculation of the joint entropy. During the calculation of the MI we have to accumulate the *plog(p)* values over the entries of the individual histogram and the joint histogram as given in equation (10). The entries in the reference and floating individual histograms are the column sum and the row sum of the MH and hence are higher in magnitude compared to the joint histogram entries. Thus, as evident from Figure 10, while calculating the entropy, the error contribution from the joint histogram entries is higher compared to the error contribution from the individual histogram entries. This is because the individual histogram entries tend to be larger in magnitude than the joint histogram entries and this have lesser error contributed from the hardware computation of *plog(p)*. Also, the number of such entries are higher in a joint histogram (4096 entries in a 64 x 64-bin MH) compared to the individual histogram (64 bins each for reference and floating histogram of 64 bins). Thus we see an accumulation of errors on the right side of the negative sign of equation (9). This error plays a considerable role in the registration process as it affects the accuracy of the similarity measure. In some cases, it can totally mask minor variations in the MH entries for small changes in transformation field; causing the error to offset any minor change in the actual MI values. This alters the path taken by the optimization routine while arriving at the best deformation field forcing the optimization routine to take higher number of iterations to find the optimal value.

3.3.3 Increasing the Accuracy of the Entropy Calculation Pipeline

To reduce the error in the MI computation we could either increase the number of entries in the LUT or we could increase the number of LUTs or both. However while implementing on a FPGA, we are limited by the hardware resources on the chip which has to be shared with all other components. With careful design I was able to increase the number of LUTs to 4 with 1K 64-bit entries in each LUT. The ranges of the values stored in the LUTs vary between $[0, 2^{-13}]$, $[2^{-13}, 2^{-6}]$, $[2^{-6}, 2^{-2}]$ and $[2^{-2}, 2^{0}]$. These ranges were chosen so as to keep the error magnitude as small as possible as shown in Figure 11. We have stored more entries toward the smaller values of 'p' thus increasing the accuracy for lower values. The smallest value of 'p' that can be addressed directly with such an implementation is 1.19×10^{-7} which is of the range of the lowest probability values possible when registering a pair of 256^3 images.

Figure 12 shows the plot of the ratio of the error in *plog(p)* calculation to the value of 'p'. Here, we note that for values of 'p' lower than 10^{-8}, the error in the computation of *plog(p)* in hardware is higher than 'p' itself (blue line), causing the ratio to increase above 100%. In such cases, where the error is higher than the value of 'p', the function output is set to zero leading to lesser cumulative errors. Here, we assume that the bins with such low probabilities do not contribute significantly to the similarity measure. A MH probability value having a value less than 10^{-8} is very unlikely to contribute significantly to the MI as such a probability corresponds to the MH bin having one voxel out of 2^{26} voxels. Figure 13 shows the typical distribution of the probabilities in the MH. Thus, it is safe to assume that such small number do not affect the registration results and hence the output of the *plog(p)* calculation for

such values are made equal to zero. In Figure 12, the red line shows the plot of the ratio of error to the value of 'p' after bounding the error percentage to 100%.

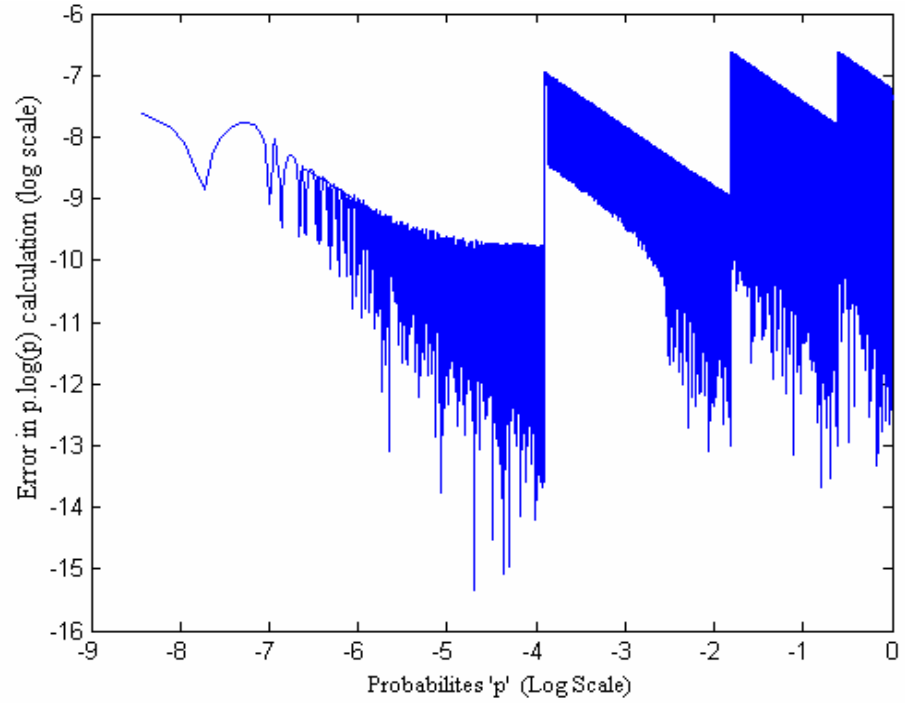

Figure 11 Error in *plog(p)* calculation with a 4 LUT implementation

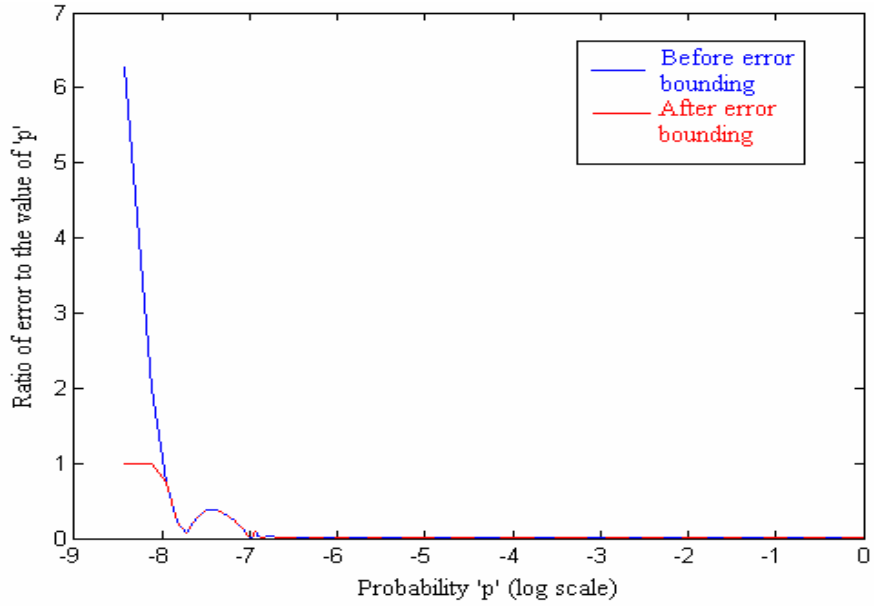

Figure 12 Ratio of the error to the value of 'p', before (blue) and after (Red) error bounding.

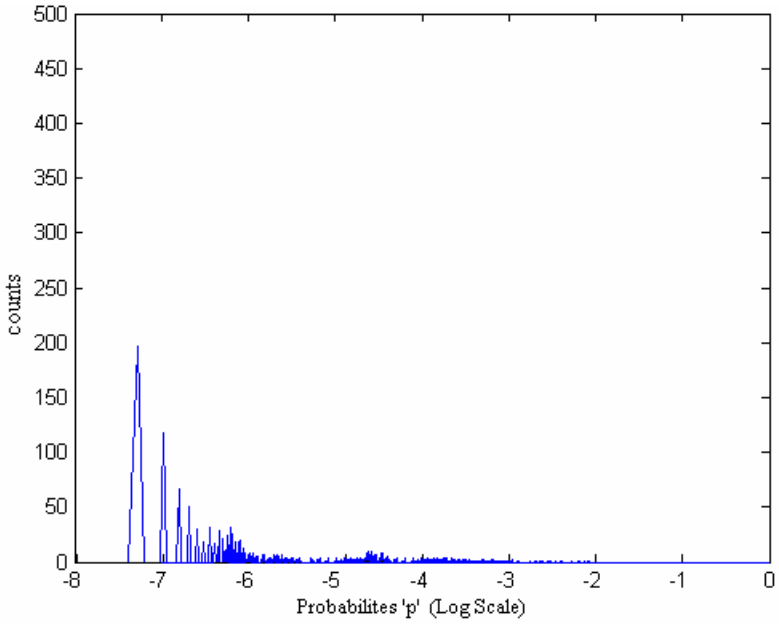

Figure 13 Plot of the histogram of the MH values, shows the distribution of the probabilities present in the MH

In Table 1 we report the error in entropy calculation between the hardware calculation and actual value from software before and after the correction to the *plog(p)* calculation for a pair of PET and CT images of dimensions 128^3 each. We registered 2 sets of images, a PET image with a CT image and 2 CT images taken at different phases of the breathing cycle to study the effect of the improvement to the registration accuracy and speed. Table 2 and Table 3compare the registration speeds for the two cases. We can see a reduction in the registration times as an effect of increased accuracy of the entropy calculator. For visual validations, Figure 14, Figure 15 and Figure 16 show two different sections of the PET image overlaid on the CT image. We can notice that the heart and tumor as indicated by the two sections in these images are mis-registered to begin with. In Figure 15, we see that the mis-registration has increased as a result of improper registration. The registration algorithm was not able to recover nonrigid deformations in the PET/CT case with the older pipeline. However, with the improved pipeline we can see the images registered

well in Figure 16. It should be noted here that, we were limited by the amount of block RAMs required for storing LUTs in the FPGA and the DSP blocks for further increasing the accuracy of the entropy calculator.

Table 1 Comparison of accuracy of entropy calculation between the software and hardware implementation for a PET/CT case of 128^3 dimension.

	MI (software)	MI (hardware)	Error
Before *plog(p)* correction	0.4338	0.3894	0.0445
After *plog(p)* correction	0.4338	0.4340	0.0002

Table 2 Speedup achieved by improved entropy calculation for the PET/CT case

	Number of iterations	Time for registration (min)
Before *plog(p)* correction	10900	21.48
After *plog(p)* correction	7892	17.30
Gain		1.24

Table 3 Speedup achieved by improved entropy calculation for the CT/CT case

	Number of iterations	Time for registration (min)
Before *plog(p)* correction	8370	39.7
After *plog(p)* correction	6331	30.0
Gain		1.32

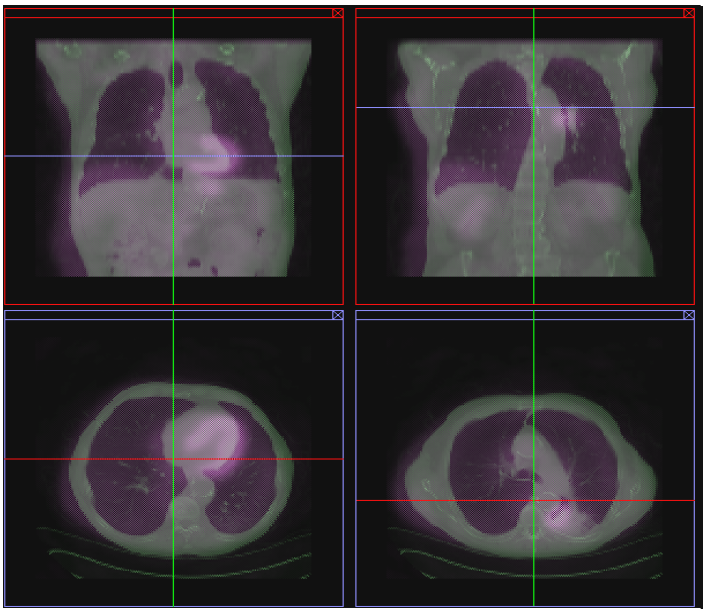

Figure 14 The PET image (pink) overlapped on the CT image (green) before registration

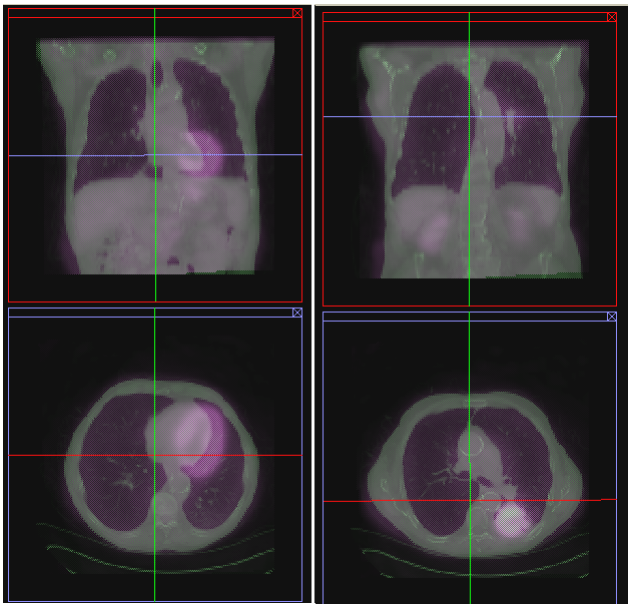

Figure 15 The PET image (pink) overlapped on the CT image (green) after registration with old entropy calculation unit

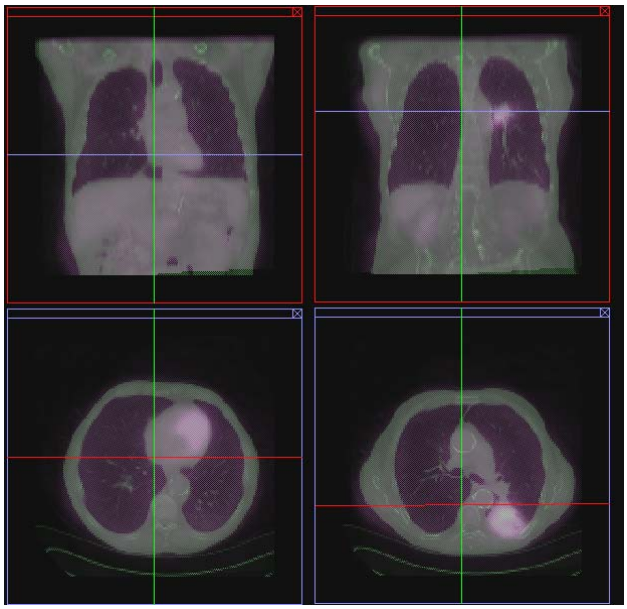

Figure 16 The PET image (pink) overlapped on the CT image (green) after registration with old entropy calculation unit.

3.4 Multiresolution in the Grid Space

During deformable image registration, like previously discussed, we overlay a uniform grid of control points. We use the deformations at these control points to obtain the deformation field required for nonrigid registration. Figure 17 shows such a grid overlaid on a 2D image. The amount of nonrigid deformations that can be recovered depends on the spacing of the control point grid on the image. With a coarse grid, it is not possible to recover all the deformations. In order to increase the accuracy of the registration, we use finer grid resolution as shown in Figure 18. With such an approach we are able to recover deformations better. However there is a limit on increasing the number of control points. Having finer grid resolutions also means that the memory required for storing the grid points in the internal memory of the FPGA increases. The number of grid points stored in the FPGA is limited by the internal memory resources of the FPGA as we require sequential and fast access to the grid memory. Also finer grid resolutions mean that there are more number of control points to be optimized and hence registration time will be longer. Thus there is a tradeoff between the registration accuracy and the registration speed. One way to reduce the registration is to have a multiresolution approach where we first register the images with a coarse grid. We register the image at lower grid resolutions by using the registration result at the coarse grid as the starting solution. I have implemented such a multiresolution approach within the grid structure to find the best deformation field. The maximum number of control points that can be overlaid on the image is 32 control points in each dimension.

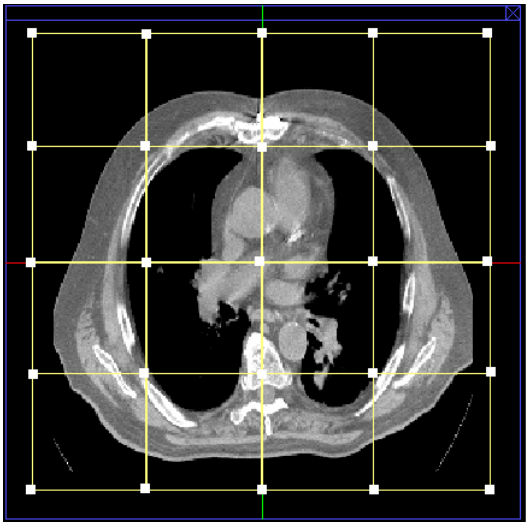

Figure 17 A uniform grid of control points overlaid on a 2D image, the square dots represent the control point positions.

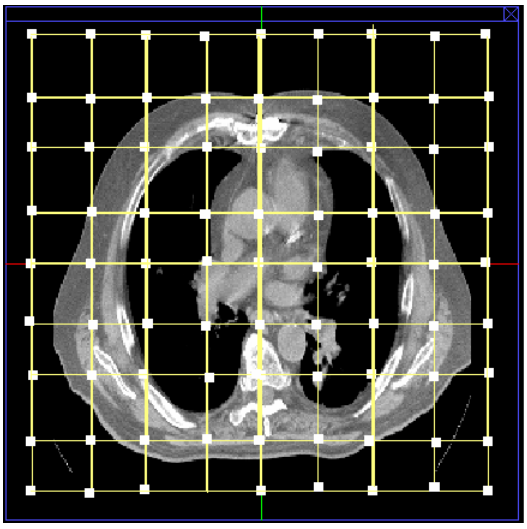

Figure 18 A uniform grid of control points overlaid on a 2D image at a finer resolution.

3.5 Calculating MSE in Hardware

As described in chapter 2, MSE as a similarity measure is suitable for images of same modality. For intra-modality image registration, equally accurate results are achieved by using MSE as the similarity measure [6]. In our hardware implementation, together with MI, I have introduced as an additional feature, the capacity for finding a MSE based similarity measure for use with images of the same

modality as given in equation (7). The MSE calculation occurs concurrently while the reference image coordinates are transformed to the floating image domain. Trilinear interpolation is used to calculate the corresponding FI intensity value. This requires the access to the entire 8 intensities around the transformed floating image coordinate. The access of the 8 FI intensities require 2 clock cycles with the memory organization of the FI in the FAIR architecture. Thus, even though we do not have the MH accumulation and the entropy calculation stages during MSE calculation, effective voxel processing rate remains the same.

3.6 FAIR-II Implementation Results

The FAIR-II architecture is implemented on a PCI board (Tsunami, SBS Technologies, Albuquerque, NM) with an Altera Stratix EP1S40 FPGA and two on-board 512 MB SDRAMs for storing the reference and floating images, which were accessed via a 32-bit bus. Two internal 512Kb RAM blocks were used to store the control point values and the lookup tables (LUTs) used for entropy calculation. The system was able to run at a frequency of 50 MHz. The speed of the overall system is currently limited by the SDRAM access speed, which runs at 50 MHz. At this voxel processing rate, the speedup in MI calculation is approximately 100 [8]. Similarly, the cubic interpolation pipeline also provides a speedup of approximately 80 [16].

When a control point is being optimized, in order to bring it into the constraints enforced by the 3D chainmail routine, more than one control point may be moved. The number of control points moved depends on the amount of deformation at that control point and the deformation at the neighboring control points. Whenever the control points are moved the control point memory on the FPGA also has to be

updated accordingly. Since there are many control points that are moved in the 3D chainmail algorithm, we have to transfer the entire grid to the board from the host PC, which leads to additional communication times, which adds to the total registration time.

The maximum size of the images in the hardware can be 512^3 and the maximum size of the control point grid that is overlaid on these images is 32^3. We have 6 bits to represent each image voxel intensity value, thus necessitating the MH to have 64x64 bins. The presence of parallel accumulation units as described in section [2.8.1] necessitates having 4 copies of MH memory in the internal memory of the FPGA. In the design of entities of FAIR-II which involves DSP components and memory components, we had to trade-off some accuracy in order to accommodate the complete design within the resources available on the FPGA. The logic cell and DSP block utilization for the FAIR-II architecture are 54% and 100% respectively. The total memory bits utilization is 55%. However, the LUTs used for the entropy calculation, which require block RAMs, cannot be further increased as the total RAM block bits utilization is 93% (3,195,072 bits out of 3,423,744). The resource utilization summary for the FAIR-II with the cubic interpolation pipeline and entropy calculator with 4 LUTs is given in Table 4. Also the resource utilization summaries for the FAIR-II with 2 LUT entropy calculator and FAIR-II with similarity measure based on MSE is also given for comparison.

Table 4 Resource utilization summary for the FAIR-II architecture

Resource Utilization	FAIR-II MSE	FAIR-II - 2 LUTs	FAIR-II - 4LUTs
Logic cells	17,824	21,264	22,687
Total combinational functions	15969	19279	20717
Total registers	8136	9324	9231
Total logic cells in carry chains	2198	3304	3250
I/O pins	355	355	355
Total memory bits	959574	1762125	1893195
DSP block 9-bit elements	112	112	112
Total PLLs	1	1	1
Maximum fan-out	8471	10189	10103
Total fan-out	72626	94866	99689
Average fan-out	3.91	4.21	4.16

3.7 Summary

Software implementations of 3D nonrigid image registration algorithms are burdened by both computational load and memory access load. The former involves smooth B-splines interpolation at each voxel, whereas the latter comprises numerous random accesses to the image memories over multiple passes of the algorithm. Together with this, the use of MI as a similarity measure accompanies high memory access load. Overall, the two main bottlenecks are spline interpolation and MI calculation stages, which have a lot of computational and memory access loads.

To reduce the overall registration time we have moved these two tasks to an FPGA, which contains a customized architecture for acceleration. The nature of these two tasks is such that they can be programmed on an FPGA using deep pipelines and parallel units. The addition of the cubic interpolation pipeline allows us to perform B-splines interpolation with minimal error, whereas the pipelined structures of various components ensure that there is minimal latency resulting in optimal speeds.

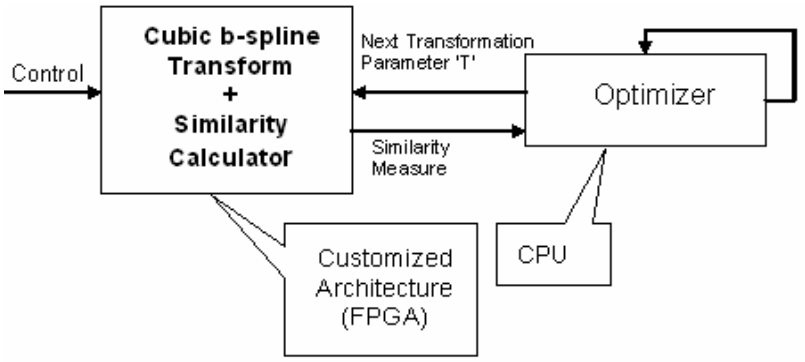

Figure 19 System architecture for the hardware accelerated nonrigid image registration system

Figure 19 shows the system architecture of the nonrigid registration system. The optimizer, residing on the host CPU, supplies a new candidate transformation to the FPGA, which applies this transformation to the RI, computes the similarity measure and then returns the similarity measure to the CPU. The CPU uses the returned value to compute the next transformation to be applied. This kind of acceleration provides us with a 100 fold increase in the voxel processing rates compared to software implementation of the same algorithm (Chapter 5). Even though the precision is affected by the limited bit widths of different components in the hardware architecture (specifically, in the computation of local and global transformation and PV interpolation), we are able to register images with subvoxel accuracy (Chapter 5).

The limited precision of the piecewise linear approximations for the *plog(p)* calculation affects MI computation. In local neighborhoods, where changes in the value of MI are small, the error in the MI calculation causes the optimization routine to take higher number of iterations than in the absence of this error. We present an implementation of a more accurate *plog(p)* calculation pipeline with 4 LUTs which reduces the error magnitude in each *plog(p)* calculation from the order of 10^{-4} to 10^{-7}.

This increases the accuracy of the MI calculation thus leading to faster and more accurate results (Table 2, Table 3). The improved accuracy of the *plog(p)* pipeline also enabled us to use gradient descent based optimization scheme as described in Chapter 4.

4 OPTIMIZATION SCHEMES AND MESH FOLDING PREVENTION TECHNIQUES

4.1 Introduction

Optimization is an important aspect of the image registration process. The optimizer provides trial search points to find the best transformation that matches the RI with the FI. The number of parameters varies with the kind of transformation applied and equal to the number of degrees of freedom in the transformation vector. For example in 3D rigid registration, there are 6 parameters that need to be optimized, 3 rotations and 3 translations along the principal axes. In case of nonrigid registration, this number increases enormously. For example, in our algorithm, where we overlay a rectangular grid on the image, we have 3 parameters (shifts along the 3 principal axes) at each of the control points. The number of parameters which require optimization increases with the number of control points. We can chose to optimize all the parameters at the same time or divide the parameters into non-overlapping subsets and optimize each set separately.

Several schemes have been proposed to optimize the parameters for image registration [13, 28]. Global optimizers like downhill simplex [29], Powell's multidirectional search [30] are useful when there are a lot of minima/maxima and is difficult to locate the global minimum/ maximum. This is particularly true in the case of noisy images like ultrasound. In such cases these search techniques explore the search space to locate the position of the optimal point. However there is a tradeoff between the complexity of the optimization algorithm and the time required to

explore the entire volume of the search space to locate the true optimum. In this section we chose one such global optimization technique, the downhill simplex and compare it with the gradient descent based optimization techniques and their suitability for fast nonrigid image registration.

The global search schemes are used for optimization of the deformation at each control point. The control points are usually optimized in a raster order. To prevent mesh folding, a major drawback of the free-form deformation based registration model, we have previously presented the 3D-chainmail method which applies a set of geometrical constraints to limit deformation of control point thereby preventing the mesh folding [16]. Our previous implementation of the 3D chainmail fits well with the optimization of individual control points using optimization schemes like simplex and Powell's. However, when optimizing the entire control point space, there is a need to have a global folding prevention scheme which applies the set of constraints to all the control points that are moved by the optimization scheme. In this section, we present a novel scheme to prevent mesh folding when multiple points are being moved in the control point space called global chainmail (GCM).

During registration, most of the time is spent in the computation of the cost function as compared to the calculation of a trial point. Also, the optimization process is usually less memory intensive and involves making a lot of decisions depending on the previous evaluations of the cost functions. This makes it very suitable for implementation on general purpose processors. Thus we have retained the optimization routine on the host PC and have supported the highly computationally and memory intensive task of calculation of the similarity measure on the FPGA. This

technique allows us the option of having a set of optimization techniques from which we can chose depending on the image characteristics. This also allows for dynamic tuning of the control parameters of the optimization routines and allows us to have measures like chainmail or GCM to prevent mesh folding.

I would like to thank Dr William Plishker for his helpful insights in the work presented in this chapter and his help in implementing the linear program solver presented here. His work on the software side of the algorithm has been of tremendous help in tackling various issues related to the hardware implementation.

4.2 Downhill simplex

In this section we discuss the simplex algorithm [29] which is a function minimization method that has the ability to crawl out of local minima to find global minima. It is a very simple system that requires only function evaluations, not derivatives. It does not use line minimizations or use derivatives of the function like Powell's multi-directional search (MDS) method [30]. It is robust in the presence of noise which makes it a desired system while registering images which contain a lot of noise like ultrasound images. A *simplex* is a n-dimensional geometrical figure, constructed from n+1 non-degenerate vertices which act as the starting points. For example, in 2D, the simplex is a triangle and in 3-D, a tetrahedron. The simplex algorithm makes it decision on the function evaluations at the n+1 points and moves to a minimum by taking a series of steps like reflection, expansion, and contraction. The simplex crawls around the parameter space and gets to the very bottom of the narrow valleys.

In our implementation, we have 3 parameters that model the deformation at each control point, one for each direction. We optimize each control point, one at a time, in a raster scan order. This simplifies the search process by reducing the dimensionality of the downhill simplex optimization. After a new trial point is given by the optimization program, we make sure that there are no folding artifacts by having the 3D chainmail algorithm [16] (single node) in place. The chainmail algorithm makes sure that the movement of the control points has not violated any of the constraints set by the user. While doing so, more than one control points are adjusted to make sure the limiting constraints are met. Thus at every iteration, the entire control point space has to be loaded onto the hardware to ensure that the hardware has the latest deformation field before evaluating the cost function. This adds for an additional communication delay.

We use the same scheme of optimization while using other single node optimization schemes like Powell's MDS. In this raster scan ordering of the control point optimizations, the optimal shift of a control point is influenced by the shifts of its neighbors. Thus, it is important that multiple passes are made, where a single pass means a complete sequential optimization of each control point describing the FFD. These optimization schemes are labeled as "greedy" algorithms because they seek the locally best solution. In our implementation we have performed 2-3 passes at each grid resolution and based on the validation results presented in this work, we have found this number to provide acceptable registration results.

4.3 Gradient Descent

Rueckert *et al.* [13] have suggested a simple iterative gradient based optimization scheme in which steps are taken in the direction of the local gradients at each control point. MI as a similarity measure lends itself suitable for gradient calculations and thus enables us to use a gradient descent (GD) based optimization scheme. The algorithm has two phases, the gradient computation phase and the trial phase. In the gradient computation phase the gradients at each of the control points are calculated as given by equation (18) from the hardware.

$$\nabla C = \frac{\partial C(\phi^l)}{\partial \phi^l} \qquad \textbf{(18)}$$

where C is the cost function, (similarity measure) and ϕ^l is the control point at location '*l*'.

In the trial point phase, suitable step size μ is applied in the direction calculated by the gradient ∇C, thus calculating the new deformation field Φ to be applied according to equation (19).

$$\Phi = \Phi + \mu \frac{\nabla C}{\|\nabla C\|} \qquad \textbf{(19)}$$

The hardware loads the entire control point space Φ after the application of the step size μ and evaluates the cost function C. Based on whether the new deformation field was a more suitable field or not, a different step size μ is applied for the same gradient vector by dividing the step size by half until a better solution point is obtained. Once a better point has been located, the gradients are recalculated with the

new deformation field as the starting point. This process is repeated till either the step size reaches a small value, or the change in the similarity measure is smaller than ε, a small constant, after which the algorithm stops. The same process is repeated for finer grid resolutions.

4.3.1 Capture of gradients

While evaluating gradients in hardware, the accuracy of the entropy calculator is very essential. The gradients are calculated using the finite difference approximations. When a small deformation is applied at each control point to capture the gradient at that point, there is a small perturbation to the MH. It is the amount of change to the MH that defines the entropy change, which in turn defines the gradient at this control point. However while calculating entropy in hardware there is a finite amount of error which has noise like characteristics as discussed in section (3.3). This error in the entropy calculation was adversely affecting the gradient computation. It had a masking effect, where the errors introduced by the *plog(p)* calculations masked the perturbations caused by the application of a small deformation at the control point during gradient computation. Thus the gradient descent optimization scheme was not able to proceed in the direction of the actual gradient, which caused the gradient descent based registration to fail. With the increase in the accuracy of the *plog(p)* calculation pipeline as discussed in section (3.3), I was able to correct this masking effect. From Table 1, we can see that the error in the gradient computation is on the order of 10^{-4} which is acceptable as the stopping criterion set for terminating the optimization routine is higher than this error margin.

4.4 Advantages of GD–based optimization scheme

Each of the optimization schemes has a set of advantages that make it suitable for implementation for a specific application. While choosing an optimization routine we have to consider several factors like the nature of two images, presence of noise, degree of deformation etc. In an automatic registration scheme like FAIR-II, by having implementation of several such optimization schemes, the user is possible to choose the optimization scheme that best suits the images that are being registered. The gradient descent based optimization scheme has several advantages in our hardware accelerated image registration scheme compared to other optimization schemes.

4.4.1 Local Gradient Computation

One of the major advantages of GD is the computation of gradients in the local neighborhood. During the first phase of optimization, gradients at all the control points are calculated. We observe that more than 90% of the time is spent in the computation of the gradients. Since the cubic interpolation involves a neighborhood of 4x4x4 control points, while calculating gradients we can calculate gradients in this finite subvolume only. For gradient computation at nodes residing on the boundaries, the boundary nodes are replicated. By calculating local gradients, we reduce the time required for gradient computation as the number of voxels processed will be less. However, at very fine grid resolutions, calculating local gradients create a very sparse MH (because of the small number of voxels involved). While calculating entropy in hardware, we observe that the percentage of error in *plog(p)* calculations is higher for small values of p. Thus if the MH becomes very sparse, computation of local

gradients is affected. This can be resolved by having a MH with smaller number of bins so that dispersion of mutual histogram is reduced.

4.4.2 Communication Times

By using a GD-based optimization scheme, the communication times are reduced. In the simplex based optimization scheme, there was a need to load the grid to the hardware through the PCI bus at the start every iteration because of the movement of more than one control points by chainmail. This leads to a lot of communication time overhead in the form of time required to load the control points to the FPGA memory where it is stored. In GD, during the second phase, where a step size is applied to the gradient vector according to equation (19), the entire grid needs to be loaded on to the hardware. However, the communication time overhead introduced by this is very small since we have a very small number of iterations through the second phase as compared to the number of iterations of complete grid transfer in the simplex case. Thus using the GD-based optimization scheme leads to lesser communication times. .

4.4.3 Parallel Implementation

In the previous implementation where a single node is optimized with simplex and chainmail (single–node) each control point is optimized in a sequential manner, with mesh folding prevented by running the chainmail algorithm at each iteration of the optimization routine. With multiple processing nodes available, this scheme is not parallelizable since the optimal point of a control point might depend on the position of the neighboring control points. Even if we are able to use multiple nodes for optimizing different control points, it is difficult to merge the deformations at all of

these control points keeping the deformations within the chainmail limits. When using multiple nodes to calculate optimal deformation at two neighboring control points, there might be potential conflicting solutions. In such cases, multiple passes may be required to optimize the control point space. However with GD-based optimization scheme, more than 90% of the time is spent in the calculation of the gradients. This becomes higher at finer deformation levels, because the total number of control points significantly increases after mesh refinement. The calculation of gradients at each control point is independent of its neighbors. Thus the calculation of gradient of the cost function at each control point can be parallelized and redistributed to different nodes. The FPGA interacts with the host PC taking in the best deformation field known previously and calculating gradients at each control points separately. I have designed the communication between the FPGA and the host PC in such a way that it preserves the independence of the control points during gradient calculation and thus lends itself well to parallelization. Ino *et al.* [12] have shown that the registration time decreases linearly with the number of nodes. Thus by having multiple FAIR-II nodes in parallel, the gradient computation can be shared across all the nodes thereby decreasing registration times further. Since FAIR-II can achieve a 40 fold speedup of registration (Chapter 5), having multiple such nodes for parallelizing the gradient calculation is instrumental in driving the registration times under a minute.

4.4.4 Disadvantages

The gradient descent optimization scheme has some drawbacks when compared to the gradient–free optimization schemes like the downhill simplex and Powell

methods, like the need to evaluate gradients and possibly the Hessian matrix. (Gradient–free optimization doesn't mean that the derivatives don't exist but only means that they are not used explicitly in the calculation of a trial point.) Also gradient descent is prone to get isolated in local minima. To tackle this situation, with the availability of fast hardware, we can register the images again with the RI and FI exchanged. Thus we reduce the probability of getting isolated in local minima. Another major drawback of using gradient descent optimization scheme is its difficulty in converging in the presence of noise. In such cases we revert back to the more robust optimization schemes like simplex.

4.5 Prevention of Mesh Folding

One of the main problems of the use of FFD based deformation field is the occurrence of folding of control points (called mesh folding) which occurs when one control point crosses over the other (section 2.3.2) . This mesh folding represents a violation of the topology of the original distribution of control points. To address this issue Rueckert *et al.* [13] proposed a smoothing penalty term called csmooth which reduces the possibility of folding.

4.5.1 Smoothing Technique

The csmooth parameter, a regularizing penalty term, is calculated as given in equation (20). The similarity measure is now the combination of the original cost function selected and the regularizing penalty term calculated and is given by equation (21). Physically, this smoothing parameter stands for the energy component of the deformed object.

$$csmooth = \frac{1}{V}\int_0^X\int_0^Y\int_0^Z \left[\begin{array}{l} \left(\frac{\partial^2 T}{\partial x^2}\right)^2 + \left(\frac{\partial^2 T}{\partial y^2}\right)^2 + \left(\frac{\partial^2 T}{\partial z^2}\right)^2 + \\ 2\left(\frac{\partial^2 T}{\partial xy}\right)^2 + 2\left(\frac{\partial^2 T}{\partial xz}\right)^2 + 2\left(\frac{\partial^2 T}{\partial yz}\right)^2 + \\ \frac{\lambda}{2}\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2}\right)^2 \end{array}\right] dxdydz \qquad \textbf{(20)}$$

where V is the volume of the image and λ represents the elastic coefficient for the body structure in consideration.

$$C' = C + \lambda(csmooth) \qquad \textbf{(21)}$$

We must note that the regularization term introduced by csmooth parameter is zero for any affine transformation and, therefore, penalizes only non-affine transformations. λ is the weighting parameter which defines the tradeoff between the alignment of the two image volumes and the smoothness of the transformation. In the previous implementations, λ has been selected experimentally and there has not been a fixed method to estimate this parameter.

This kind of correction based on csmooth parameter is suitable for applying to global GD methods and is easy to compute (second derivatives with respect to the deformations at the control points). However this method cannot prevent folding but can only hinder large movement of control points. Also the weighting parameter, λ, is difficult to adjust. For the PET/CT case which we considered earlier, we derived the deformation field for 2 values of λ, one at a very low value, $\lambda = 0.01$ and one at a very high value $\lambda = 10$.

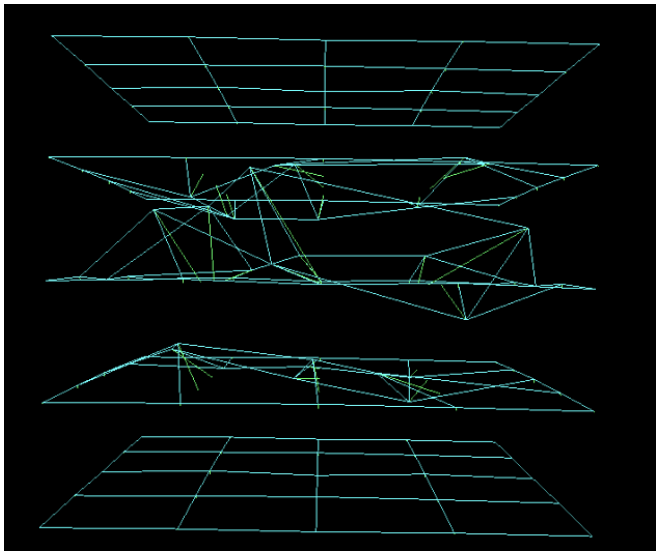

Figure 20 Deformation field with very less smoothing $\lambda = 0.01$

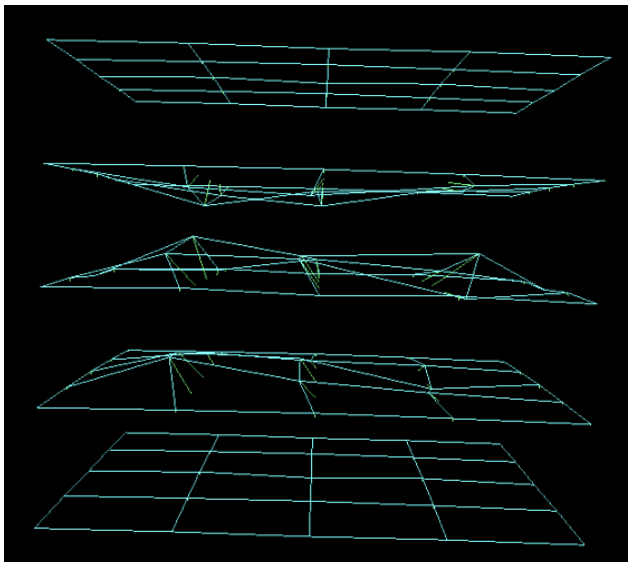

Figure 21 Deformation field with very high smoothing $\lambda = 10$.

Figure 20 and Figure 21 illustrates the deformation field across a 5x5x5 grid overlaid on the reference CT image. We can see that when the parameter is too small, we can see a high degree of deformation and mesh folding becomes evident in this case. Similarly, when the parameter is too large, it hinders the actual movement of the control points resulting in a high degree of smoothness that does not model the actual deformation field.

4.5.2 3D Chainmail

In order to prevent mesh folding in the case of single node optimization, we use a robust folding prevention scheme based on 3D Chainmail, which we have presented earlier [16]. 3D Chainmail imposes a number of geometric constraints on the movement of a control point with respect to its neighboring points. These constraints, like minimum and maximum distance between adjacent control points, maximum shear between planes, control the stretching of the control points along each of the 3 principal directions and shear perpendicular to them. If the movement of a control point violates any of the limits, its neighboring vertices are moved in tandem to satisfy the limits. Thus the deformation at every control point is applied while maintaining the original topology between the control points, thus restricting any mesh folding artifacts. Figure 22 shows an example of a control point 'A' being moved and the application of chainmail algorithm to adjust the neighboring control points so as to satisfy the limits set by the user. When extended to 3D, there are a total of 9 parameters: 6 controlling stretching (d_{minx}, d_{maxx}, d_{miny}, d_{maxy}, d_{minz}, d_{maxz}) and 3 controlling shear (s_{maxx}, s_{maxy}, s_{maxz}) which allow one control point to move freely (within the constraining limits) without hindering the motion of other control points unlike the smoothing parameter which hinders the motion of all control points equally.

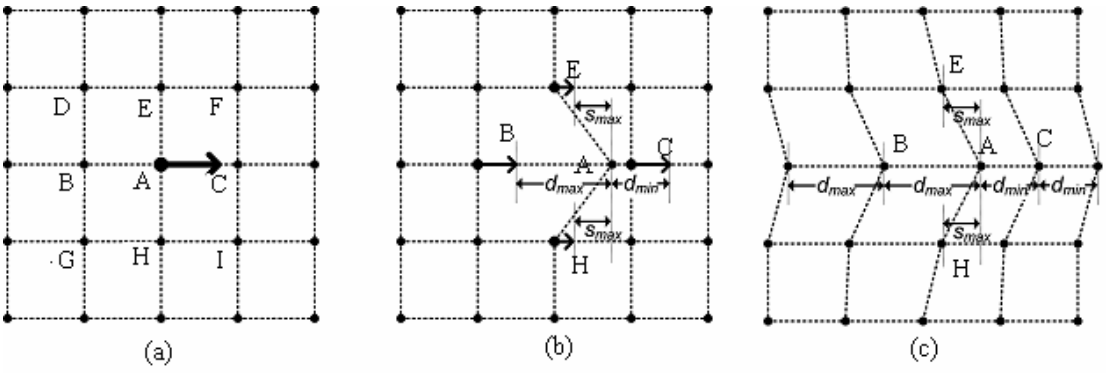

Figure 22 Local deformation propagation example.(a) Control point A affected by the optimization algorithm. (b) Propagation of the local deformation to the immediate neighbors. (c) Final grid after the deformation has propagated through all the necessary control points.

This system works well in the simplex optimization scheme where only one control point is moved at a time. While such a technique is implemented in a global optimization scheme like the GD, there are multiple points moved whenever the optimization algorithm tries to take a suitable step based on the gradients. This might lead to situations where there is a point, 'A', which is being pulled in two different directions in order to meet the constraints (for points B and C) as shown in Figure 23.

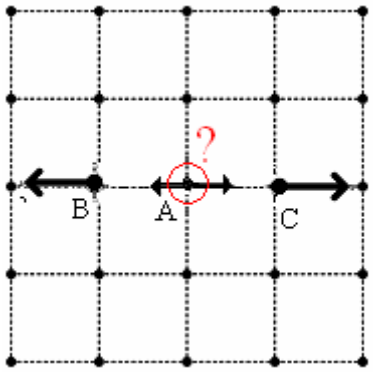

Figure 23 Multinode optimization; points B and C pulling point A in opposite directions to meet the individual constraints.

4.5.3 3D Global Chainmail

Figure 23 shows a situation where a conflict occurs when multiple points are being moved. In this case the point A is being pulled in two opposite directions in order to satisfy the standard Chainmail constraints set by the points B and C. In such cases, we can chose to either anchor the point A, without moving it or move points B and C to meet the constraints with respect to point A. Also we can split the difference proportionately. In the first case, this method locks up the control points from moving and in the second case; the problem becomes increasingly hard to split among multiple points.

In order to model the deformation field in the best way, while allowing for the maximum allowable deformation at every control point we propose a scheme called 3D global chainmail (GCM). GCM allows deformations at all the control points to satisfy the given set of constraints while moving the points as little as possible. This problem is increasingly hard at higher grid resolutions as the number of control points increases cubically with the grid size. We model an objective function that is the sum of the movements at all the control points subject to set of constraints as shown in equation (22).Since the objective function and chainmail constraints are all linear, we can use linear programming for solving the set of constraints given by equation (23) subject to the condition that the total distance of all the control points moved, D, is minimum.

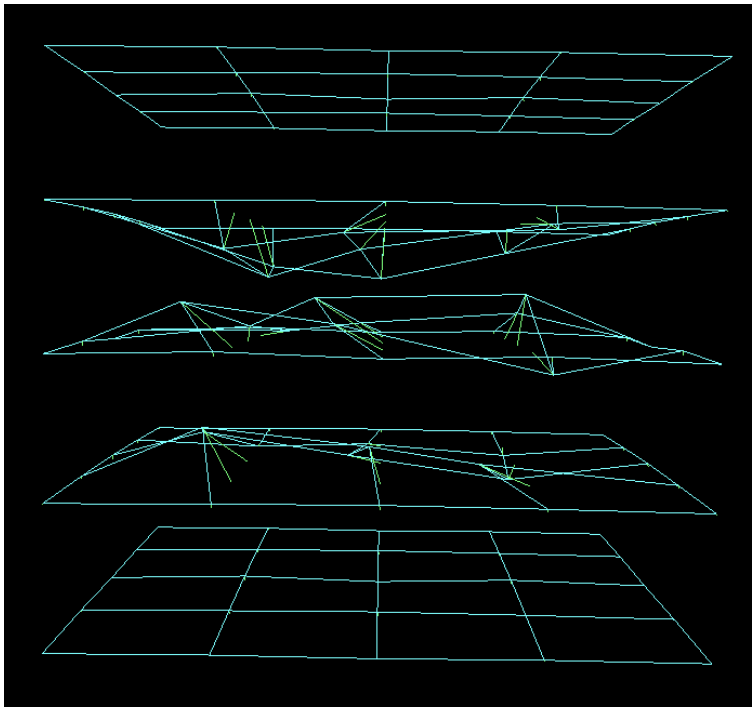

Figure 24 Deformation field with GCM

In our implementation, we use the GNU linear programming kit (GLPK) for solving the set of linear equations. The amount of time taken by the GPLK to solve the equations is very small, in the order of milliseconds. The GPLK solves the equations for movement along each of the three axes separately and gives a new deformation field in which control points are moved according to the gradients, while keeping the distance moved in order to satisfy the constraints applied to a minimum. Figure 24 shows the same grid as in Figure 21 but with global chainmail controlling the movement of the control points. In this case, we allow the control points to move about with respect to the local gradients to the maximum amount possible within the permissible limits as set with equation (23).

$$D = \sum_{\forall x,y,z} d_{x,y,z} \qquad \textbf{(22)}$$

$$
\begin{aligned}
p_{x+1,y,z} - p_{x,y,z} + GridSpaceMM_x &\le MaxD \\
p_{x+1,y,z} - p_{x,y,z} + GridSpaceMM_x &\ge MinD \\
\left| p_{x,y,z} - p_{x,y+1,z} \right| &\le MaxShear_y \\
\left| p_{x,y,z} - p_{x,y,z+1} \right| &\le MaxShear_z \\
p_{x,y,z} &\le p_{x+1,y+1,z} + GridSpaceMM_x - \varepsilon \\
p_{x,y+1,z} &\le p_{x+1,y,z} + GridSpaceMM_x - \varepsilon \\
p_{x,y,z} &\le p_{x+1,y,z+1} + GridSpaceMM_x - \varepsilon \\
p_{x,y,z+1} &\le p_{x+1,y,z} + GridSpaceMM_x - \varepsilon \\
p_{x,y,z} &\le p_{x+1,y-1,z-1} + GridSpaceMM_x - \varepsilon \\
p_{x,y,z+1} &\le p_{x+1,y-1,z+1} + GridSpaceMM_x - \varepsilon \\
p_{x,y,z+1} &\le p_{x+1,y+1,z-1} + GridSpaceMM_x - \varepsilon \\
p_{x,y,z+1} &\le p_{x+1,y+1,z+1} + GridSpaceMM_x - \varepsilon \\
\left| p_{x,y,z} - p^0{}_{x,y,z} \right| &= d_{x,y,z}
\end{aligned}
\qquad \textbf{(23)}
$$

where $p^0{}_{x,y,z}$ and $p_{x,y,z}$ are the deformations at control point (x,y,z) before and after the grid adjustment; GridSpaceMM is the grid spacing between the control points; MaxD, MinD are the maximum and minimum distances between the control point in each direction, $MaxShear_y$ and $MaxShear_z$ are the shear parameters between the adjacent planes; and ε is a positive constant.

The flow of the GD-based optimization scheme with the 3D GCM is given below.

Loop {

Loop {

Get gradient vector ∇C for current grid resolution;

Loop {

Create a potential new grid based on a step size of μ;

Bring new grid into chainmail specifications using linear program solver (GLPK); Solve separately for each dimension;

If (new grid is better) break; else $\mu = \mu$ /c; (c being a constant)

If μ is too small exit with last known good grid;

}

Recalculate the gradient vector, ∇C;

If ($\|\nabla C\| < \delta$) break; (where δ is a small positive constant)

}

Grid = Grid at a finer resolution;

If finest resolution reached, then exit;

}

4.6 Comparison of Speed

The running times vary greatly between a single node optimization scheme like simplex and multimode optimization scheme like gradient descent. In this section we study the speed comparison between the two implementations for a CT-CT registration case and PET-CT registration case, all images being of size 128^3. In Table 5, we show the speed for the simplex optimization, GD-based optimization scheme where all the voxels are used for calculating the gradients (referred for simplicity as global gradients), and GD with local gradients as discussed in section (4.4.1). For both of these cases, we can find that the simplex was faster when compared to the GD with global MI computation whereas the GD with local gradient computation was faster than the simplex optimization scheme. However, as previously discussed in section (4.4.3), we must note here that the GD-based methods lend themselves very well to parallelization during gradient computation which can bring out further reduction is speed.

Table 5 Comparison of speed for single node optimization, GD-based optimization with global gradients and GD-based optimization with local gradients

Total Registration time	Simplex (sec)	GD with Global Gradients (sec)	GD with Local Gradients (sec)
CT-CT	872	1903	640
PET-CT	1038	2092	604

4.7 Comparison of Accuracy

In this section, I compare the accuracy of the registration between a pair of representative Lung CT images taken during the inhale and exhale phases of the breathing cycle. I registered the exhale image with the inhale image using simplex optimization and GD-based optimization. I used expert-traced contours drawn on the floating and reference image to validate the results of the registration. I compared the accuracy of registration based on 3 metrics, namely overlap index, RMS distance and Haursdoff distance between contours drawn on different body structures like lung and tumor by expert on the RI and the contours transferred from the FI (drawn by an expert on the floating image) after registration. We validate the registration accuracy by matching the two contours using the above mentioned metrics. If A represents the organ/tumor volume using automatically propagated contours and B represents the lung/tumor volume using expert-drawn contours, we defined the overlap index as $(A \cup B / A \cap B)$. The RMS distance between the surface boundaries of A and B is the average of surface mismatches along radial lines emanating from the center of mass of the expert-delineated shape. For even stricter validation, we also computed the Hausdorff distance between the two volumes (the method and the measures are discussed in detail in chapter 5). Table 6 shows these results for the left lung, right lung and the tumor. We find that the results from the simplex implementation and the gradient descent implementation match closely. Both optimization schemes were able to recover a large degree of deformation between the end-exhale and end-inhale images for this representative case. For comparison, the metric values before registration are also given in the table. We can notice an improvement in all the three

metrics using both the optimization schemes. This shows that GD and simplex optimization scheme perform similarly on hardware and can be used interchangeably.

Table 6 Comparison of accuracy between GD/simplex for a representative case

Structure	Overlap Index			RMS distance			Haursdoff distance		
	Before	**GD**	**SIM**	**Before**	**GD**	**SIM**	**Before**	**GD**	**SIM**
Left Lung	0.85	0.94	0.95	4.4	2.2	2.2	34.8	21.4	16.8
Right Lung	0.90	0.92	0.93	4.6	2.8	2.5	25.4	19.3	19.0
Tumor	0.57	0.78	0.78	4.5	2.3	2.6	8.7	7.1	5.2

Before: before registration, GD: registration with GD, SIM: registration with simplex

4.8 Summary

In chapter we present the different optimization schemes that we have used in the registration system. For nonrigid registration, the number of degrees of freedom is very large which poses a challenging task to optimize all the control points in the least amount of time. We have used downhill simplex based optimization schemed in our previous works [31] for optimizing the control points on software. However the optimization schemes like downhill simplex, Powell's MDS etc are sequential in nature and cannot benefit much from the use of multiple nodes.

Gradient descent based optimization schemes work well with the same modality image registration, and also with cases where the noise levels in the image are low. Gradient descent lends itself very well for parallelization and is well suitable for hardware implementation. We discuss in this chapter the various issues that affect the computation of gradients and hardware and methods to tackle such situations.

In GD-based optimization, at every evaluation of the similarity metric, there are a number of control points that are moved. While using free-form deformation based nonrigid registration, mesh folding is a common problem that leads to unrealistic

deformations. In order to address the issue of mesh folding while using GD-based schemes, we present the 3D global chainmail scheme which applies a set of constraints to the control points while moving the control points.

Our development of the gradient descent based optimization scheme coupled with the GCM is not to replace the simplex optimization scheme but to complement it in cases where registration can be easily guided by the calculation of gradients as in the case of CT-CT registration.

5 CHARACTERIZATION OF IMAGE REGISTRATION HARDWARE

5.1 Introduction

In this section I present the registration results obtained from the FAIR-II architecture described in the previous chapters. I compare the registration results from the hardware with the registration results obtained from a similar software implementation on an Intel Xeon workstation running at 3.6 GHz. I used the result of registration to propagate expert-traced contours from one breathing phase to another. I validated the results using 4 metrics detailed in this chapter.

5.2 Experiments

In order to evaluate our hardware accelerated image registration system, we considered CT scans of different body organs acquired at different breathing phases. We selected these set of images since they had considerable amount of soft-tissue deformations introduced due to respiratory movements. Breath-hold and respiratory gated CT scans of 5 lung cancer patients and 4 abdominal cancer patients undergoing radiation treatment were considered. We registered end-exhale images with end-inhale images (referred to as exhale and inhale images), motivation being that if we were able to recover deformation between these extreme cases, deformation between any other set can be recovered easily. We recently presented a software implementation of an accurate registration-based segmentation approach, in which organ contours drawn by an expert in the exhale scan can be used to detect the organ

contours in the inhale scan [31]. Treating the results of the software implementation as the reference, we compared the results obtained by the hardware registration.

Automatic segmentation, which refers to the propagation of organ contours from one CT scan to another, was used to validate the results. Expert generated contours were used to segment the tumors and different body organs were identified by using a commercial threshold-based segmentation tool (Pinnacle, Phillips medical systems, Cleveland, OH). Treating the exhale image as RI and inhale image as the FI, the images were registered using the hardware and the deformation field obtained was used to deform the contours on the inhale image, which were then transferred on to the exhale image. The expert/Pinnacle generated contours on the exhale image were used to validate the results by comparing with the contours transferred from the inhale image.

5.3 Validation

Validation of image registration has been a challenging task because of the absence of a gold standard. Since expert generated contours are used extensively for treatment planning, use of such contours for validation is a feasible and acceptable approach. For validation we compared the organ and tumor contours generated by the registration in hardware with the expert contours. A software implementation of the same algorithm was used to generate automatic contours and the validation results from the software implementation were used to compare with the results of the hardware implementation which gave a measure of the accuracy of the hardware. We considered four metrics for validating the results, namely the mean square difference (MSD) between the unregistered and registered images, overlap index (OI), root

mean square (RMS) distance and Hausdorff distance between the surface boundaries of the registered contour and the expert contour.

5.3.1 MSD

We considered the mean square difference (MSD) as a validation measure between the exhale and the inhale images before and after registration. Also we compared the MSD between the exhale image and the registered inhale image obtained with the software implementation.

MSD is defined as

$$\mathrm{MSD} = \sqrt{\frac{\sum_{i=1}^{N} (RI_i - FI_i)^2}{N}} \qquad \textbf{(24)}$$

where RI_i and FI_i are the CT numbers of the reference and transformed floating images respectively and N is the total number of voxels in the volume of overlap. While calculating MSD, we considered the entire 12 bits in the CT number. The MSD must be minimal when the two images are perfectly aligned.

5.3.2 Overlap Index

Overlap Index gives a measure of how well the contours of the volumes considered overlap [6]. If A is the volume using expert/Pinnacle traced contours and B represents the volume of the contour propagated after registration then the overlap index is given by

$$\text{OI} = \frac{A \cap B}{A \cup B} \qquad \textbf{(25)}$$

An OI of 1.0 corresponds to the perfect overlap between the two volumes. However OI is influenced by the size of the organ, and is less sensitive to disagreement between the 2 contours in these cases.

5.3.3 RMS distance

RMS distance is given by

$$\text{RMS} = \sqrt{\frac{1}{n}\left(\sum_{i=0}^{n} (a_i - b_i)^2\right)} \qquad \textbf{(26)}$$

where a_i and b_i are vertices along the radial lines emanating from the center of mass of the volumetric regions. RMS distance between the surface boundaries of A and B measure the average of surface mismatches along these radial lines [6]. It provides a better validation tool compared to the overlap index as it can identify disagreement between contours even in the large volume organs.

5.3.4 Haursdoff distance

Hausdorff distance (HD) is defined as the maximum of the closest distances between the 2 volumes, where the closest distance is computed for each vertex in the 2 volumes[6]. Mathematically, Hausdorff distance is given by

$$\textbf{HD} = \max\left[\max_{a \in A}\left\{\min_{b \in B} d(a,b)\right\}, \max_{b \in B}\left\{\min_{a \in A} d(a,b)\right\}\right] \qquad \textbf{(27)}$$

where a is a volume A vertex, b is a volume B vertex, and $d(a,b)$ is the Euclidean distance between vertices a and b. Hausdorff distance reflects the distance of the point of A that is farthest from any point of B and vice versa. Haursdoff distance can therefore capture the worst-case mismatch as the absence of the averaging process can identify outliers. It can give a measure of the worst case misalignment between the two contours considered here. We evaluated all these measures on 5 lung cases and 4 abdominal cases and present the hardware registration results in comparison with the software registration results.

5.4 Voxel Processing Rates and Communication Overhead

Voxel processing rate is defined as the ratio of the total number of voxels processed to the total time consumed. During MI calculation, this is the ratio of the number of voxels in the RI to the time taken for one MI calculation. We compared the voxel processing rates from the hardware and compared it with the voxel processing rate of a similar software implementation on an Intel Xeon workstation running at 3.2 GHz. Table 7 shows the MI computation time in the hardware (HW) for different sized images in comparison with the software (SW) implementation of the same algorithm. With the hardware implementation, we can see a 100 fold increase in the voxel processing rates of the system (compare the hardware and software voxel processing rates in Table 7). In the hardware implementation, the reference image and the floating image (4 copies) have to be loaded on to the hardware board once before the registration begins. The time required for loading the images on to the hardware is also shown.

For nonrigid registration, we have to load the control point deformation field given by the optimization routine to the hardware at the beginning of every iteration. As previously discussed, we store the control point deformation field in the internal memory of the FPGA. Loading of the control point memory to the internal memory over the PCI constitutes a communication overhead. Table 8 shows the time required to load the control point memory (grid memory) to the hardware which constitutes the communication overhead. The grid transfer time is a function of the size of the overlaid grid and increases cubically with the resolution of the grid.

Table 7 Voxel processing rate comparison between hardware and software

Image dimension	MI computation time (sec)		Voxel processing rate (Million voxels/sec)		Image load times in hardware (sec)	
	HW	SW	HW	SW	Ref Img	Flt Img
64^3	0.02	1.83	16.38	0.14	0.72	2.71
128^3	0.12	7.28	16.78	0.28	5.45	21.49
256^3	0.78	63.59	21.48	0.26	42.41	169.73

HW: Hardware, SW: Software, Ref Img: Reference Image, Flt Img: Floating Image

Table 8 Communication overhead

Grid dimension	Grid load time (s)
5	0.0001
9	0.0149
17	0.0470

5.5 Comparison of Hardware and Software Implementation

The software implementation used for comparison with the hardware used simplex based optimization (single node optimization) with 2-3 passes at each optimization level. In the hardware implementation also, we used the same optimization scheme along with chainmail to prevent mesh folding, thus we are comparing two exactly similar implementations in hardware and software. Table 9 shows the speed comparison between the hardware and software implementations for the registration of inhale and exhale images of size 256 x 256 x 80. The table provides the number of

iteration at the two grid resolutions taken by the simplex optimization scheme and the total time required for registration at these levels. The time required for rigid registration and the time required for elastic registration are given separately and the total registration time is also provided for comparison between the hardware and the software implementations. As described, rigid registration precedes nonrigid registration in our algorithm. The rigid registration step showed a speedup of approximately 100. The speedup in the case of nonrigid registration is 40. Even though the voxel processing rate was roughly approximately 100 times faster in hardware, as shown in Table 7 (compare hardware and software numbers in the voxel processing rate column), the net speedup compared to software was less than 100, because of different factors.

The hardware implementation had fixed precision at various stages like coordinate transform, MH accumulation and entropy calculation. The loss of precision in the case of entropy accumulation on hardware affects the MI calculation directly, which leads to a higher number of iterations taken by the optimization routine. The increased number of iterations is due to the oscillations experienced by the optimization routine in small neighborhoods, where changes in MI values are very small and thus the error in MI computation hinders the optimization routine from taking the most optimal path. The speedup in the nonrigid registration was further reduced because of the grid loading at every iteration which accounts for communication time overhead. The rigid registration does not suffer from this communication latency. Overall, the hardware implementation was 40 times faster

than the software implementation on a 3.6 GHz Intel Xeon processor with 1.5 GB of RAM.

Table 9 Time comparison for a representative CT/CT case

Implementation	Rigid Registration		Nonrigid Registration		Total Time (min)
	Iterations	Time (min)	Iterations (Level1,Level2)	Time (min)	
SW	158	52.3	540, 3060	1192.0	1244
HW	123	0.5	1115, 5093	29.5	30
Speedup		104.6		40.4	41.5

Table 10 through Table 15 report the accuracy of registration-based segmentation of the CT-CT registration for all the 9 cases described above. For each case, we present the segmentation results for the left lung, right lung and tumor in the lung images and liver, left kidney, right kidney and tumor in the abdominal images. We compare the validation measures like overlap index, RMS distance and Haursdoff distance for all the different structures mentioned for the 9 CT cases. We compare the metric values obtained from registration with hardware with the metric values before registration and the metric values after registration from an equivalent implementation on software.

For visual assessment of the registration results, in Figure 25 we show the lung tumor segmentation for a representative lung image. We can see that the registration results from hardware (red) matches closely with those from the software (blue) and the expert results (green). Figure 26 shows the difference image for the same axial slice before and after registrations from hardware and software taken from the same lung image. Nonrigid image registration corrected misalignment in most major structures; any residual misalignment was near finer structures. To measure the amount of misalignment recovered, we computed the mean squared difference (MSD)

between the exhale and inhale images. Table 16 shows the MSD before and after registration for all the cases. We can see that there is a noticeable decrease in the MSD after registration, which further indicates that the images are well aligned after nonrigid registration.

Table 10 Overlap index for the Lung cases

	Right lung			Left lung			Tumor		
	Before	SW	HW	Before	SW	HW	Before	SW	HW
Lung1	0.90	0.97	0.93	0.85	0.95	0.95	0.57	0.91	0.78
Lung2	0.88	0.98	0.93	0.86	0.97	0.93	0.63	0.83	0.78
Lung3	0.88	0.96	0.94	0.90	0.97	0.91	0.32	0.73	0.27
Lung4	0.88	0.99	0.95	0.91	0.99	0.95	0.57	0.82	0.64
Lung5	0.84	0.94	0.92	0.82	0.95	0.92	0.10	0.56	0.20

Before: before registration, HW: Hardware, SW: Software

Table 11 RMS distance for the Lung cases

	Right lung			Left lung			Tumor		
	Before	SW	HW	Before	SW	HW	Before	SW	HW
Lung1	4.6	2.5	2.8	4.4	2.4	2.2	4.5	2.2	2.6
Lung2	4.9	2.2	2.2	4.1	2.6	1.8	2.7	1.8	1.9
Lung3	4.9	2.3	3.8	5.4	1.8	3.1	5.8	2.1	4.2
Lung4	5.6	1.2	2.2	3.5	1.3	2.0	5.3	2.2	3.2
Lung5	5.4	2.6	2.9	6.3	2.5	3.0	14.7	4.5	8.5

Before: before registration, HW: Hardware, SW: Software

Table 12 Haursdoff distance for the Lung cases

	Right lung			Left lung			Tumor		
	Before	SW	HW	Before	SW	HW	Before	SW	HW
Lung1	25.4	14.0	19.0	34.8	15.9	16.8	8.7	3.9	5.2
Lung2	20.1	16.3	15.2	19.1	21.2	12.3	7.1	6.5	5.1
Lung3	30.4	21.2	28.1	27.3	15.6	19.5	19.4	5.9	8.3
Lung4	27.2	26.3	21.3	23.6	25.6	12.5	18.7	18.5	12.7
Lung5	32.7	22.8	21.7	34.9	18.5	16.8	29.8	15.8	27.3

Before: before registration, HW: Hardware, SW: Software

Table 13 overlap index for the Abdominal cases

	Right kidney			Left kidney			Liver			Tumor		
	Before	SW	HW	Before	SW	HW	Before	SW	HW	Before	SW	HW
Abd1	0.71	0.83	0.81	0.67	0.89	0.79	0.81	0.92	0.87	0.54	0.77	0.64
Abd2	0.68	0.81	0.84	0.72	0.84	0.84	0.77	0.91	0.90	0.57	0.74	0.77
Abd3	0.72	0.89	0.84	0.78	0.90	0.83	0.82	0.94	0.87	0.32	0.82	0.75
Abd4	0.79	0.89	0.86	0.84	0.92	0.90	0.85	0.95	0.93	0.37	0.67	0.59

Before: before registration, HW: Hardware, SW: Software

Table 14 RMS distance for the Abdominal cases

	Right kidney			Left kidney			Liver			Tumor		
	Before	SW	HW	Before	SW	HW	Before	SW	HW	Before	SW	HW
Abd1	3.6	2.9	2.2	5.2	2.8	2.5	4.5	3.1	2.6	3.6	2.7	3.4
Abd2	5.4	3.3	3.3	8.7	2.6	2.7	9.4	3.4	3.6	4.4	3.3	4.1
Abd3	7.8	2.7	2.7	5.1	2.6	2.3	5.1	3.2	3.3	5.1	3.0	2.4
Abd4	3.5	2.8	2.7	3.1	2.7	2.7	4.0	2.5	3.1	5.6	3.6	3.5

Before: before registration, HW: Hardware, SW: Software

Table 15 Haursdoff distance for the Abdominal cases

	Right kidney			Left kidney			Liver			Tumor		
	Before	SW	HW	Before	SW	HW	Before	SW	HW	Before	SW	HW
Abd1	11.9	7.8	12.2	10.1	6.9	12.4	23.0	13.7	14.6	12.9	6.1	9.9
Abd2	17.5	8.8	11.3	16.4	6.6	9.3	20.4	9.5	11.3	18.7	9.1	9.7
Abd3	18.0	8.2	9.8	20.2	7.4	9.5	32.1	11.5	15.3	14.5	6.1	7.7
Abd4	9.5	7.1	7.0	8.8	7.2	7.9	10.7	6.6	8.0	11.0	6.8	11.0

Before: before registration, HW: Hardware, SW: Software

Table 16 MSD after and before registration in software and in hardware for all the cases

	MSD		
	Before	SW	HW
Lung1	232.1	136.3	136.7
Lung2	252.0	138.5	141.0
Lung3	214.3	133.1	134.0
Lung4	269.0	127.0	122.5
Lung5	209.1	142.0	141.1
Abd1	196.7	152.0	153.1
Abd2	104.8	62.2	66.6
Abd3	126.5	94.7	90.4
Abd4	88.7	51.4	53.0

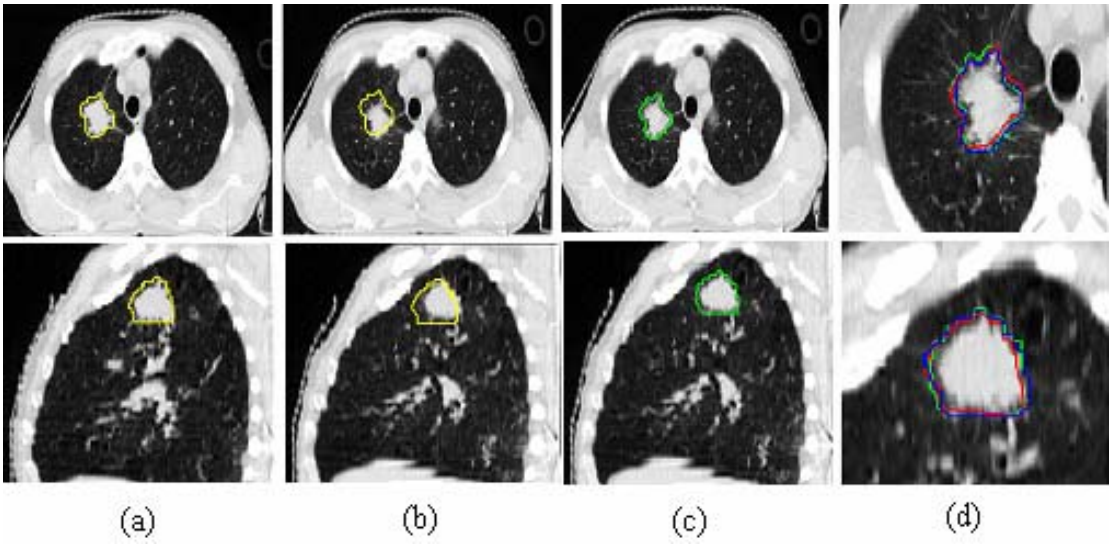

Figure 25 Lung tumor segmentation for a representative case shown in axial (top row) and sagittal (bottom row) planes: (a) inhale CT + expert contour; (b) exhale CT + contour from (a) (shows prominent misalignment of tumor); (c) exhale CT + expert contour; and (d) exhale CT + expert (green), software(blue) and hardware (red) contours. (Zoomed-in for better visualization)

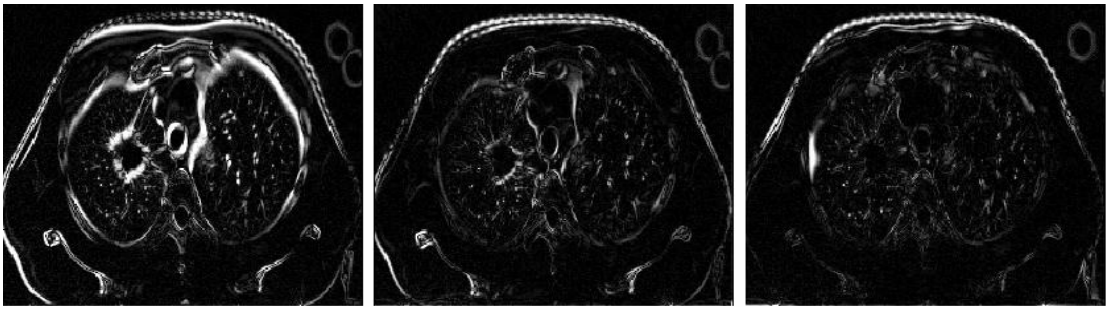

Figure 26(L-R) Difference images before registration (used for computing MSD), after registration in hardware, and after registration in software

5.6 Discussion

After looking at the validation results, we find that the hardware and software results match closely in most of the cases. Deformable image registration corrected misalignment in most major structures. Although the hardware uses finite precision arithmetic, it had minimal effect on the accuracy of segmentation (and hence registration). A remarkable improvement in all the metrics after registration for both hardware and software implementations can be seen. Any residual misalignment was

near finer structures. The misalignment at finer structures can be corrected by registration at a finer grid resolution. Due to memory limitations, sometimes it is not possible to reduce the grid resolutions to finest levels as compared to the software implementations which have very high memory resources. In such cases, like in the case of lung tumor of case 5, it was not possible to recover deformation at such finer structures completely. However, with larger FPGAs available, in the future developments, we should be able to register at very fine levels so that the accuracies of registration can be made more closely matching to the software results. Overall, the hardware implementation produced results that closely matched the software results, while providing a 40-fold improvement in speed for a single processor (i.e., FPGA chip).

The fast nature of the hardware also affords us the opportunity to register the reference and the floating images twice, after interchanging their roles, for improved accuracy and robustness. This provides for automatic validation of data as the deformation field in one case should be the inverse obtained after interchanging the images. Even though the hardware processed voxels 100 times faster, the effective speedup for the overall registration algorithm was approximately 40 for a single hardware node as a result of factors such as higher number of iterations required in hardware, communication time and the latencies in the SDRAM. The accuracy of the entropy calculator is an influencing factor in the registration speed and accuracy as the optimizer has to base its decision on the output of the entropy calculator. With more hardware resources available in newer FPGAs we are able to increase the accuracy of the hardware stages leading to better accuracies and faster convergence.

5.7 Summary

In the previous chapters, we have presented a novel custom hardware approach to accelerate nonrigid image registration. In this chapter we provide the comparison of our system (finite precision) with an equivalent software implementation (infinite precision) which runs on an Intel Xeon workstation both in terms of accuracy and speed. Our current implementation provides a 40-fold improvement in speed. Even though the hardware uses finite precision arithmetic, we observed no significant reduction in the accuracy of the registration algorithm. In fact, for the test case of CT-CT registration, the results from the hardware and the software were comparable. Further improvements, such as the use of a finer grid, can further equalize the hardware and software implementations. While precision is slightly affected in hardware due to fixed-point arithmetic, the optimizer was still able to recover the deformation, as our results show. The reconfigurable characteristic of the FPGA allows us to make algorithmic changes, which ensures flexibility in our approach to accelerate image registration. If needed, additional FPGA-based preprocessing steps can also be added to the pipeline, thus ensuring additional processing at no extra time cost [32].

6 USE OF MODELING TECHNIQUES FOR MAPPING APPLICATIONS ONTO RECONFIGURABLE HARDWARE

6.1 Introduction to DSPCAD tools

In the previous chapters we have discussed the complexity of the medical image registration application and the need for system that can perform registration at clinically viable speeds. For achieving this, we have suggested a FPGA based architecture that can achieve speedups up to 40, while delivering comparable accuracy with the software implementations. Using conventional high-level languages it is not possible to exploit various dynamic parallelism exhibited by certain systems DSP systems. Modeling of DSP applications based on the coarse-grain dataflow graphs allows designers to exploit desirable properties like dynamic reconfigurability and design reuse. There has been a growing set of tools that allow us to models DSP applications and also provide us with semantics for such modeling. In this chapter we look in to the possibility of using such automated design tools for mapping registration applications to reconfigurable hardware like FPGAs. Also we examine having a reconfigurable system which can dynamically reconfigure itself depending upon performance parameters selected through careful study of the behavior of the system. Modeling the applications like image registration with dataflow graphs provide a method to analyze such performance parameters and a framework for high level optimization of these applications. Modeling applications using dataflow modeling also provide designers the ability to arrive at more accurate solutions within given time constraints.

In this chapter, I study the use of a novel dataflow model, homogeneous parameterized dataflow (HPDF) model as presented in [33], which allows for data dependent behavior among its functional units, for mapping image registration onto a reconfigurable architecture like FPGA [34]. By the use of such model-based mapping, I explore the design to expose concurrencies and different levels of parallelism that would have been very difficult to explore otherwise. In this study I use the dataflow modeling to exploit the different levels of parallelism and data dependencies and identify performance parameters like the percentage of valid voxels (PVV) which can be used to optimize the design. Exploiting the reconfigurable nature of FPGAs, and the dynamic behavior of the registration system, we suggest the use of this metric for reconfiguration of the system to further optimize the design by exploiting parallelism as presented in [35]. The work presented in this section contributed significantly to the development of HPDF as a meta-modeling technique where we demonstrate the integration of other modeling semantics like cyclo-static dataflow (CSDF) model into the HPDF meta-modeling framework. This work was a joint effort between us and the research group of Dr Shuvra Bhattacharyya, University of Maryland, College Park. In particular, I would like to thank Dr Mainak Sen for his help during the various stages of the work presented in this chapter.

6.2 Dataflow Models

DSP applications have a varying degree of complexity and resource requirements. Modeling an application in dataflow graphs allows designers to efficiently exploit different tradeoff parameters like resource utilization, area and speed, allowing them to customize the design for a particular platform. Verification, analysis and

optimization of the final design can be done after careful exploration of the high level design structure exposed by these dataflow graphs. Also, some application structures that get obscured in the final design can also be exposed using dataflow modeling. High level optimization routines serve as efficient tools to evaluate and optimize the design for different performance parameters.

In dataflow model of computation, the computational DSP elements are called as actors. Actors in dataflow representation stand for functional units that can have varying level of complexity depending on the granularity of the design. Actors are the vertices in the dataflow graphs. An actor might be a 'C' function of several lines or may be a description of the function in hardware description language (HDL) depending on the target platform where the computation is desired. However the dataflow modeling does not take into account the functionality of the actor but relies on the amount of input data samples that it consumes in one execution (called as consumption rate) and the number of data samples it outputs in one execution (called as production rate). Actors in the dataflow graphs are connected with edges which stand for first in first out (FIFO) queues which hold tokens that are produced by the source actor of the edge until they are consumed by the sink actor of the same edge. An actor execution is called as a 'firing' of the actor and corresponds to one complete execution of the functionality of the actor. The order of actor execution is determined by the compiler or hardware or both according to the design. Each actor can have several attributes like code length (cost), execution times etc. Edges can also have different attributes like communication costs, delays etc. An edge attribute, "D" represents units of delay; each unit of delay is analogous to the z^{-1} operator in signal

processing, and is typically implemented by placing an initial data value on the corresponding dataflow edge. By examining data transfer patterns between actors, a schedule can be created that provides a way to coordinate the execution of all actors in the data flow graph.

6.2.1 Different Forms of Dataflow

Different forms of dataflow models have been proposed. The firing semantics of actors vary between these different forms of dataflow. The synchronous dataflow (SDF) model [36] enforces that a given actor must consume (or produce) a predetermined number of tokens at each firing. This restriction allows for strong compile time predictability properties. SDF has been adopted for various DSP application modeling and hardware synthesis [37]. However stringent restrictions imposed by SDF hinder modeling applications with dynamic production and consumption rates like in data dependent applications such as computer vision applications.

Other forms of dataflow graphs have been developed which allow a more flexible firing semantics. Cyclo-static dataflow graph is one such dataflow graph which can accommodate multiphase actors exhibiting different production and consumption rates, as long as the variations across the phases form statically-known, periodic patterns. Even though this model can provide additional flexibility compared to SDF these models do not permit data dependent production and consumption rates. To address data dependent production and consumption rates, HPDF graphs which allows a restricted form of data dependent behavior was suggested in [33].

6.2.2 Homogeneous Parameterized Dataflow

Homogeneous parameterized dataflow [33] models use a set of dynamically adjusted parameter values to adapt the actor behavior in a structured way to address data dependent behavior of actors. HPDF actors may change their production and consumption rates at run-time in between successive iterations of the graph, but at any given point in time, any HPDF edge will have the same data production and consumption rates for its respective sink and source actors. However HPDF models impose restrictions in the model to ensure that the HPDF subsystems are homogeneous across any level of hierarchy. This also ensures homogeneity in terms of the average rate in which the actors execute. This restricted form of data-dependent behavior supported by HPDF permits useful modeling flexibility, and also provides for efficient scheduling and resource allocation for actors, as well as verification of bounded memory requirements and deadlock avoidance.

Furthermore, since HPDF is a meta-modeling technique, hierarchical actors in an HPDF model can be refined using any dataflow modeling semantics that provides a well-defined notion of subsystem iteration. For example, a hierarchical HPDF actor can have SDF, CSDF, or HPDF actors as its constituent modules. When HPDF is applied with CSDF (referred to as HPDF/CSDF), it allows a dynamic number of phases for the actors with dynamic production and consumption rates on each phase. However the model ensures that the total number of tokens produced in a given iteration (invocation of the actor) on a given edge is equal to the total number of tokens consumed by the sink actor of the same edge, thus satisfying HPDF constraints [38].

6.3 Dataflow Representation of Image Registration

After studying the dynamic behavior of the image registration application, HPDF/CSDF approach is used for modeling this application as it best represents the lower-level, multiphase interaction between the actors. Modeling in this way allows us to describe the inherent concurrency in applications, identify potential bottleneck areas and extract potential areas that are best suited for parallelization. We have seen the complexity issues regarding the image registration application in section (1.2). In this chapter we analyze the application from a dataflow point of view to tradeoff certain design parameters like area and speed to obtain an optimal design. We identify potential areas that can be parallelized and we also suggest a reconfigurable architecture that exploits intra and inter voxel parallelization capabilities. We give valid schedules for the different actor executions which allow us to calculated buffer sizes, execution times and execution order. Representing the actor schedules in the form of "looped schedules" will allow us to represent successive repetitions of the execution sequence. Each entry in the loped schedule represents either an actor or another looped schedule (to express nested looped schedules).

6.3.1 Top-level Application Modeling

In this section, I present the hierarchical dataflow representation of the MI-based image registration algorithm using the HPDF/CSDF meta-modeling approach for modeling lower-level, multi-phase interactions between actors. Figure 27 shows our top level HPDF model of the application. The MI actor consumes one data value (*token*) on every execution. (This token can be a binary value, or a word, or a vector of data entries). The token contains co-ordinates of the RI and the FI. After 's'

executions, each consuming one token (coordinate values in this case), where ‘s’ of the order of the total number of voxels in the image, the MI actor produces the entropy between the reference and floating images. This value is then sent to the optimizer as a single token.

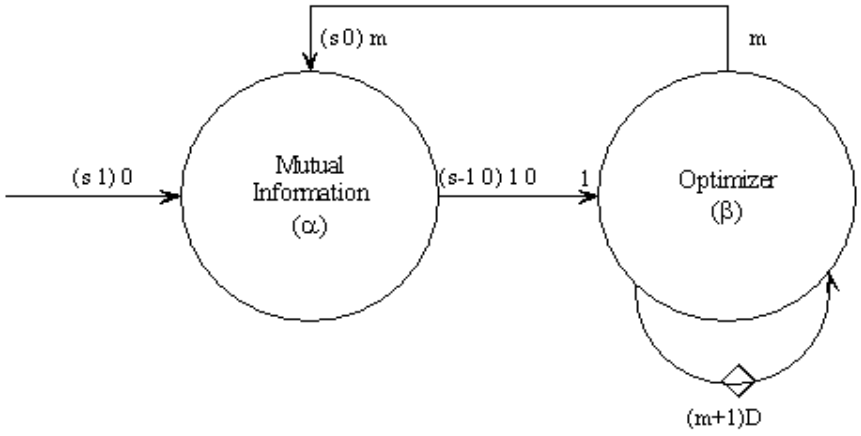

Figure 27 Top level modeling of the image registration application

Optimization of the transformation parameters also depends on the nature of the images and the amount of misalignment between the two images. In the simplex method discussed previously, in order to optimize a transformation with “m” parameters, the optimizer needs to store “m+1” previous values. The optimizer sends “m” tokens to the MI actor which corresponds to the next transformation that needs to be applied to the RI. Since “m” can vary depending on the number of parameters used to represent the desired transformation (6 for rigid, 12 for affine and thousands for nonrigid), the associated edge represents a variable-rate edge of the HPDF graph. A valid schedule for this HPDF graph is $(s\alpha)\beta\alpha$.

6.3.2 Mutual Information Subsystem Modeling

Figure 28 shows the internal representation of the hierarchical MI actor. A token here refers to each RI coordinate processed while applying the transformation (rigid or elastic). The RI controller (A) and the coordinate transform (B) consume one token each and B produces one token, which represents the transformed coordinates. A voxel is valid if it falls within the boundaries of the FI. If this voxel is valid it is passed on to the weight calculator (D) and FI controller (E). Now since all voxels may not be valid, r tokens ($r \leq s$) are produced from the "Is Valid" (C) actor, (s-r) voxels being invalid voxels. The actors D and E output tokens those are essential for PV interpolation, which is done only if the voxel is valid as indicated by C. This actor also produces tokens on the edge that connects it to the MH memory (G). For every input token in D and E, eight output tokens are produced on both the outgoing edges. The corresponding eight intensity locations in G are updated based on the tokens produced by D. After all coordinates are processed, which occurs during the first 8r phases of the MH Memory actor or equivalently after s phases of the actor B, one token of size ($q \times q$) is sent to the decomposer (Z), which in turn sends out tokens to the entropy calculator (H) actor. The actor H consumes all of these tokens, and produces a single token that contains the MI value. The actor Z was added for ease of representation and was later subsumed into the actor G during final synthesis. A valid schedule for the MI subsystem based on Figure 28 is $(sABC)(rDE(8FG))(q^2ZH)$

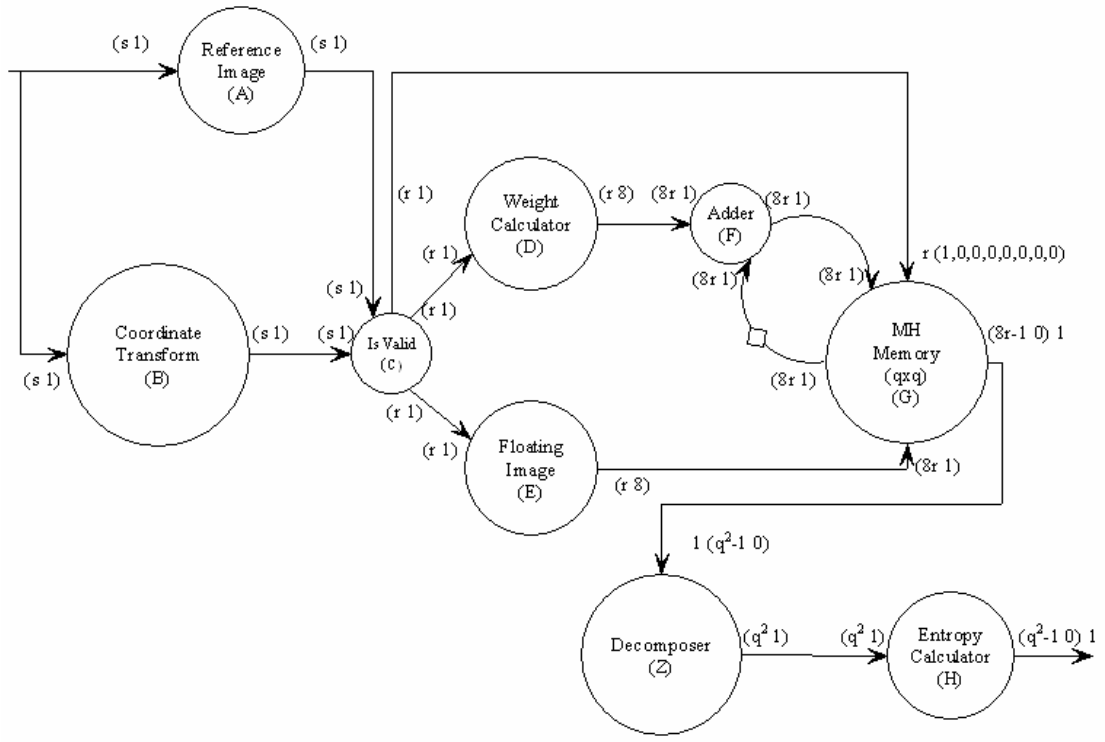

Figure 28 Dataflow modeling of the mutual information subsystem

In [34] we studied the application of such modeling for the case to rigid registration only. After studying the case for nonrigid registration, I find that the difference in the two cases is the actor B, which represents the type of coordinate transform. The subsystem modeling of the actor B might change from the conversion from rigid to nonrigid, however at the hierarchical level as seen in this section, the general dataflow remains the same. Thus the same calculations can be easily used for nonrigid registration. In the actor B, there is an additional input edge which inputs tokens from the optimizer. This corresponds to communication cost, taking in the new transformation parameter. The number of elements in the transformation vector varies from rigid to nonrigid. Figure 28 only represents the steady-state behavior of MI subsystem for simplicity. Figure 29 represents the initialization and the steady-state behavior of B - where the initial "m" tokens are used to calculate the new

transformation field and hence it updates the values inside the actor without producing any data. Considering the steady state behavior of B, we can give the schedule of the system as $(mB)(sABC)(rDE(8FG))(q^2ZH)$

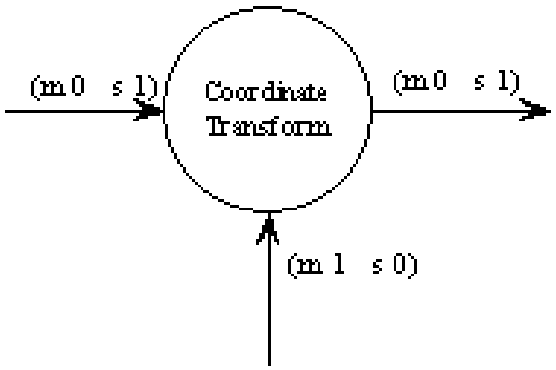

Figure 29 Steady state modeling of the coordinate transform actor

6.3.3 Parameterized Entropy Calculator

The entropy calculator, H, can be further represented by a lower level parameterized dataflow representation as shown in Figure 30. The RI and FI histograms are computed from the MH so that the voxels in the overlap areas (valid voxels) are taken into account. Row sum (I) executes once every time it gets one row (q elements) to produce one token, the RI histogram for that bin of the MH. 'q' depends on the number of bits used to represent an image intensity. For example, 'q' is 64 in this implementation and every image voxel intensity value should be at least 6 bits for using this implementation. The column sum (L) can only produce an output for every input after it has already received $q\times(q-1)$ elements corresponding to rows $(q-1)$. There are many valid schedules that can be proposed for Figure 30. As shown in [34] we can give the schedule for this as $(q-1(qZLT)IJ)(qZLTN)IJKOUV)$ which exposes a very high buffering overhead. We

can combine the above schedule with the schedule for the MI subsystem by replacing H and thus obtaining

$$(mB)(sABC)(rDE(8FG))(q-1(qZLT)IJ)(qZLTN)IJKOUV)$$

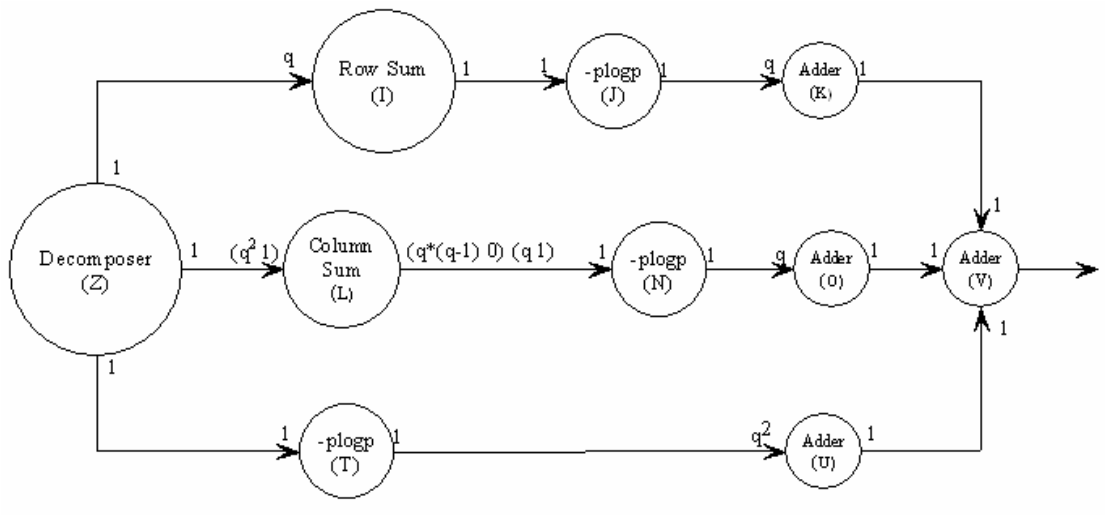

Figure 30 Parameterized entropy calculator

6.3.4 Parallel Architecture for Mutual Histogram Accumulation

By representing the system in the form of dataflow graphs, we can exploit potential parallelizable structures within this system. For example, extensive "intra-voxel" (within the processing structure for a single voxel) parallelism is possible for F and G. From Figure 28, we can see a data-rate mismatch between D,F; and E,G. This exposes a potential parallel structure as described in Figure 31. In this case we can see that there are 8 accumulate operations that are repeated 'r' times. By having multiple copies (eight in the illustration) of the actors F, G we have a parallel implementation. This reduces the buffer sizes, increases the speed of processing however at more memory and area cost. We also note that the resultant graph in Figure 31 becomes HPDF as all the parameterized actors now have the same production and consumption

rates and hence fire at the same rate. We find that there is another degree of parallelization that can be exploited referred to as inter-voxel parallelism. We found that actors A and B have "s" distinct phases, where s is the number of voxels. This involves dividing the image into subvolumes and processing these subvolumes separately by having multiple sets of such actors. In [34] we have developed an architecture that applies intra-voxel parallelism and in [35] we discuss inter and intra voxel parallelism in comparison with each other.

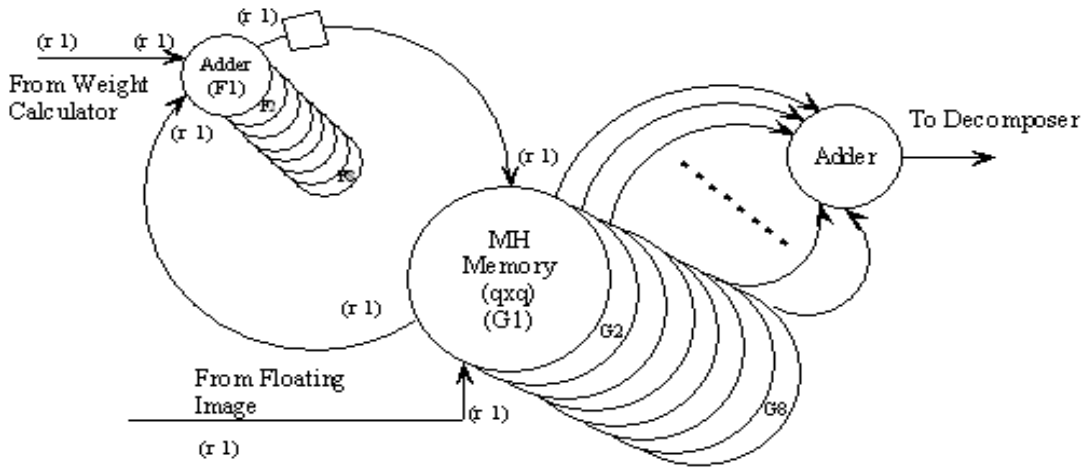

Figure 31 Parallel architecture for MH update exposing intra-pixel parallelism

6.4 Implementation

A parallel architecture was suggested as evident from the dataflow modeling that can exploit both intra and inter voxel parallelism [34, 35]. Thus, we varied the degree of parallelism as shown in Figure 31 and studied the resulting relationship between performance and area. We find that the parallelism is affected by the amount of voxels that are invalid. Invalid voxels are pretty high during the initial phases of the registration process. This happens because the optimization scheme tries larger steps through the exploration space. However when the optimizer converges onto the optimal value, smaller steps are taken and higher number of valid voxels are

expected. Also in cases where the FI has a smaller field of view than the RI, there is higher percentage of voxels that are classified invalid. Thus studying the performance of the system under different percentage of valid voxels (PVV) helped us understand the parallelizable potential of the registration application.

6.4.1 Degree of Parallelism and Relation with PVV

When a coordinate is transformed in the coordinate transformed, the actor E in Figure 28 uses the integer part of the transformed voxel coordinate as the base address in the FI space and generates the FI values (corresponding to the neighborhoods) and provides it to the MH memory for updating the MH with the weights generated by the weight calculator actor. When we have just one set of actors (floating image, weight calculator and the MH memory), actors DEFG, it takes eight firings of this set of actors to perform PV interpolation, for every input that is processed by the coordinate transform actor. As we multiply the set of actors DEFG by multiples of 2, the PV interpolation can be performed in parallel, thus reducing the time by an equal factor of 2. The limit of such parallel set of actors is 8, since after 8 units there is no additional benefit obtained. As updating the MH is a crucial part of the algorithm, such parallel execution should result in significant improvement of the whole application. However as a result of the parallelization, we can find an increase in the resource requirement of the FPGA resources and external memory.

Our system was a self timed system in which each actor required a ready signal generated by the actors preceding it to indicate its readiness for execution. When the transformed coordinate falls in the valid region, there are eight firings of the actor set F,G. However when C does not generate a signal (indicating that for the given input

coordinates, the transformation produces invalid coordinates) the iteration of the graph stops for those input coordinates and the next token is processed by the coordinate transform actor indicating a new iteration. For our implementation, any isolated invalid signal causes a two cycle penalty, but consecutive invalid signals cause only one cycle penalty for each invalid signals as there is already an invalid signal established in the pipeline.

6.5 Experimental Results

Our architecture was simulated for functional correctness and synthesis was performed with Quartus (Altera Corporation) targeting the Stratix-II family (device 1EP2S). Verilog HDL was used to develop functional modules which represented the pipelined execution of the actors. FIFO buffers were also developed in verilog with separate read and write pointers to monitor the FIFO buffer executions. The code was synthesized for different degrees of parallelism of the floating image and weight calculator actor. Next, we simulated the performance of the various configurations of the circuit with four different PVVs as 100, 90, 50 and 10 in terms of number of clock cycles. We assumed that when PVV is low, invalid signals are contiguous and they are sparse when PVV is high which has an impact on the run times as discussed in the previous section.

In this section, I present hardware synthesis results for various proposed configurations of the image registration application. Table 17 presents the synthesis results for various configurations - the columns represent the different resource allocation and maximum operating frequency for the circuitry in these configurations. The resource allocation is independent of PVV. Figure 33 shows the area of the

FPGA in terms of ALUTs and performance trade-off curve when we vary the number of parallel data paths in the MH update actor for different PVV. The area is measured by the number of ALUTs utilized in the circuit without considering the external memory, while the performance is measured by the number of execution cycles. The trend in all of the above mentioned cases reflect that the number of execution cycles decreases with increasing amounts of parallel data paths, although the corresponding area increases. Figure 32 shows the tradeoff between performance and external memory requirements. We notice that the PVV is an important metric for performance. However increasing the number of parallel data paths yields less relative performance gain at lower PVV than when at higher PVV.

Table 17 Resource utilization summary for the MH update subsystem

Number of parallel paths	1	2	4	8
External Memory	256KB	512KB	1MB	2MB
LC registers in FPGA	427	575	871	1463
DSP elements	30	30	30	30
Total FPGA area (Number of ALUTs)	598	878	1439	2588
Max Frequency of operation (MHz)	74.0	72.2	74.0	70.1

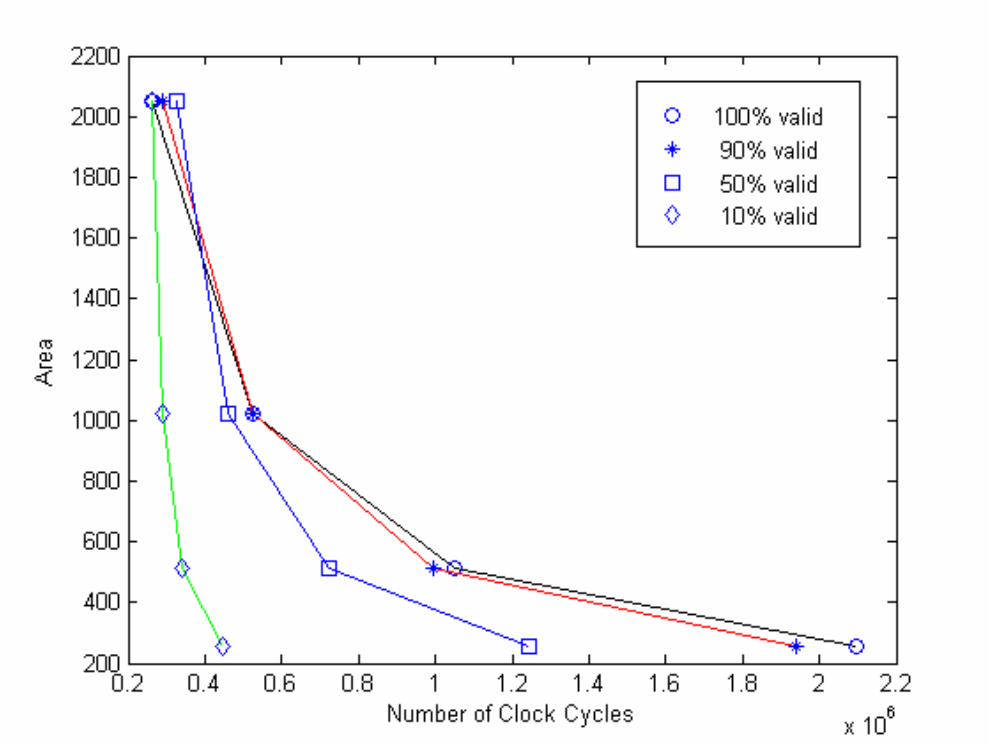

Figure 32 External memory requirement versus clock cycles required for complete execution for different PVV

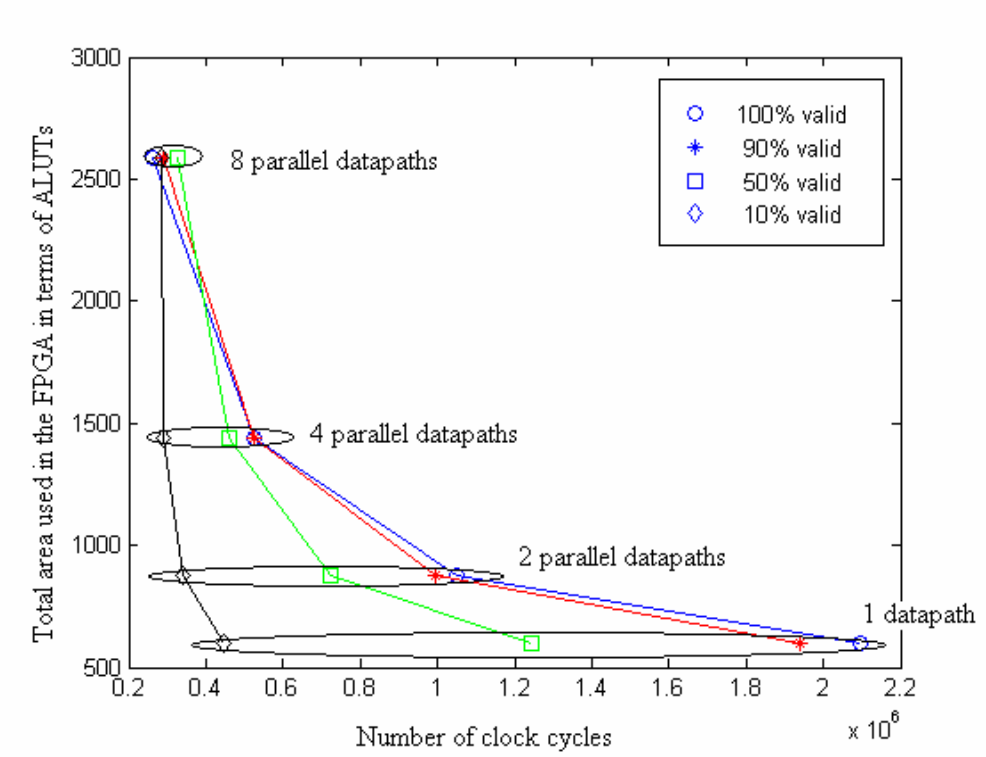

Figure 33 Total area in terms of ALUTs for different PVVs for different number of data paths.

6.5.1 Dynamic Reconfiguration

We find that with the increase in PVV, the run-time increases and memory access becomes more of a bottleneck, and gradually, it becomes more performance-effective to trade-off inter-pixel parallelism in the architecture for intra-pixel parallelism in the form of multiple (parallel) memories that alleviate the memory bottleneck. PVV is dependent on the input images. So we need reconfigurability with the knowledge of the input characteristics of the images. In the resource allocation table, Table 17, we find that the area utilization for a 1voxel- 8 data path implementation is about 8 times the area for the 1 voxel – 1 data path implementation. In Table 18, I show the comparison of performance for different PVV values, of a 1 voxel-8 data path architecture (intra-pixel parallelism) to a 8 voxel architecture with 1 data path module per voxel (inter-pixel parallelism) architecture. The units of performance in Table 18 are microseconds per voxel per co-ordinate transform and the frequencies of operation of the different memory architectures vary between 70 MHz and 74 MHz for various configurations. We must note here that the inter pixel parallelism can be increased with additional resources available on the FPGA, compared to intra pixel parallelism which cant be increased further than 8 parallel data paths (as PV interpolation requires just 8 weights to be accumulated). In [35] we suggested a similar comparison with a 7-voxel architecture than a 8 voxel architecture. Choosing 8-voxel architecture in inter pixel parallelization is also more logical as it is easier to divide a 3D image into 8 subvolumes than 7 subvolumes leading to easier control features and memory organization.

Table 18 Comparison between intra and inter voxel parallelism for different PVV values

Voxel Validity	Performance of eight 1 voxel – 1 data path	Performance of one 1voxel- 8 data path
10%	0.78	2.54
50%	2.22	2.91
90%	3.47	2.50
100%	3.75	2.33

In the Figure 34 we plot the performance of the system with the PVV for the two configurations. We find that the performance of 1 voxel-1 data path architecture is better than that of a 1 voxel-8 data path architecture, however this trend changes as the voxel validity percentage increases. Therefore, our image registration architecture monitors the PVV metric at run-time and dynamically reconfigures the architecture from inter-pixel parallelism mode to intra-pixel parallelism mode once the transition point is observed. Thus when there is a high PVV, we use the 1 voxel- 8 data path representation to achieve better performance. For the given configuration under consideration, from the Figure 34 we can find that a suitable threshold point is around 70%. In order to prevent rapid change in architecture in case the PVV oscillates around the transition point, often referred to as trashing; we assign a threshold T, such that the architecture gets reconfigured when a (70-T)% PVV state is followed by a (70 + T)% PVV state or vice-versa. T can be set by the user depending on image characteristics such that the dynamic reconfiguration happens only if necessary in terms of performance.

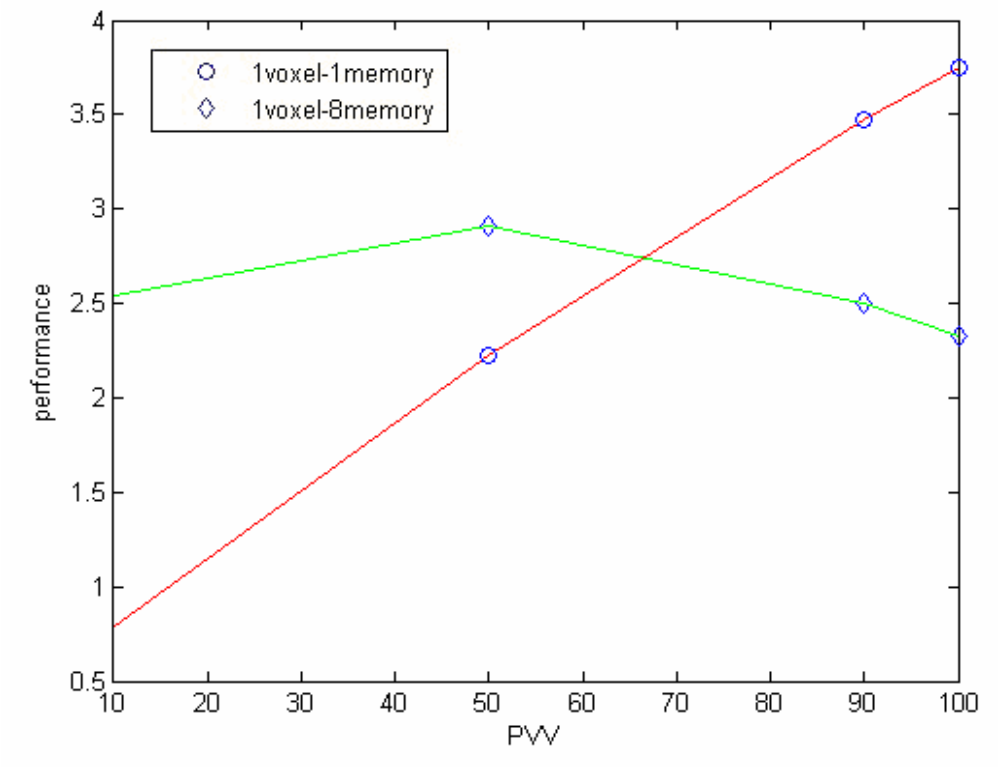

Figure 34 Comparison of performance at different PVV with the two different configurations

During reconfiguration, with intelligent design, the reconfiguration cost can be reduced with the use of simple switching logic. During the synthesis of the system, by keeping actors as common as possible between the two systems and having a composite design which switches data-buses depending on the configuration desired, the reconfiguration cost can be minimized. Actors such as coordinate transform, RI and optimizer in Figure 27 and Figure 28 can be reused across different configurations. However the actors involved in the MH update part like the floating image and weight calculator require changes with respect to the production and consumption rates during reconfiguration.

6.6 Summary

In this chapter we study the use of dataflow models for mapping image registration onto a reconfigurable architecture like FPGA. By the use of model-based mapping based on HSDF/CSDF, automatic mapping a memory and computationally intensive application like image registration onto configurable hardware by use of computer aided design (CAD) has been possible. We also explore the design to expose inter- and intra-voxel parallelism and thus arrive at optimal designs. We explored various levels of intra- and inter voxel parallelism and presented area-performance trade-offs for different parallel configurations. Our experiments quantify how increasing the number of parallel data-paths results in increased area but decreased runtime. Also we show that the parameter PVV is an important metric in exploring the design space. We also suggest the use of this metric for reconfiguration of the system to further optimize the design by exploiting parallelism.

The FAIR architecture discussed in chapters (2-4) is not a dataflow based system and has been developed using a statically configured model. However since my work with FAIR architecture was a continuation of previous work ongoing in our lab, it was not possible to redesign the complete system with the dataflow based model. Useful directions for further work include integration of the modeling insights and dynamic reconfiguration techniques developed in work with relevant design aspects of the FAIR architecture.

Use of dataflow interchange format (DIF) for modeling applications like image registration and having a library of hardware and software description entities allow us to map parts of the algorithm onto different platforms depending upon different

attributes like complexity, resource requirements and suitability to the particular platform. The concepts presented in this chapter can be extended to other DSP applications as shown in [38].

7 CONCLUSIONS

Nonrigid image registration can be accelerated using hardware methods by exploiting parallelism in the algorithm. However, any performance benefits may be mitigated by issues such as limited precision of the computational data path and communication overheads, if the system is not carefully designed. In this work, we present a hardware accelerated 3D nonrigid registration system that built upon the previous efforts at acceleration of bottleneck areas like MH computation, and MI calculation. We added a cubic interpolation pipeline to the existing FAIR architecture to perform cubic B-splines interpolation required for nonrigid registration based on free-form deformation. We evaluated the system performance both in terms of speed and accuracy compared to a software implementation running on a general purpose computer like the Intel Xeon workstation.

We validated the results obtained from the hardware by comparing it with the registration results obtained from the software implementation. Based on the validation results presented in this work we can conclude that the hardware accuracies are equivalent to the software accuracies. Even though the hardware uses finite precision arithmetic, we observed no significant reduction in the accuracy of the registration algorithm.

The hardware showed voxel processing rates approximately 100 times higher than the equivalent software implementation. However due to the communication overhead, memory access latencies and limited precision of the hardware the effective speedup achieved by the hardware is approximately 40 compared to general purpose processor, which is still the highest speedup per processing unit reported for nonrigid

registration of 3D medical images. The fast nature of the hardware also affords us the opportunity to register the reference and the floating images twice, after interchanging their roles, for improved accuracy and robustness. The reconfigurable nature of the FPGAs allows for algorithmic enhancements through software upgrades.

In this work we also presented a gradient descent optimization scheme that fits well with the hardware implementation of the registration system. The gradient descent based registration system optimize all the control points together and has several advantages like ability to work locally, ability for parallelism which make it better suitable for registration with hardware. However, with gradient descent based optimization scheme there are more than one control point that are moved which makes folding prevention more challenging. We have presented in this work, a mesh folding prevention scheme that applies a set of linear constraints to all the control points thereby preventing mesh folding. In order to find the best deformation field that conforms to the set of linear constraints applied while keeping the movement of the control points to a minimum we use a linear program solver which provides the best possible solution within milliseconds. With the use of such a mesh folding prevention scheme along with the gradient descent based optimization scheme while registering images on hardware, we arrive at similar results compared to the single node optimization schemes like simplex.

Finally we look at the registration algorithm with the help of dataflow modeling techniques based on homogeneous parameterized dataflow (HPDF) graphs. With the help of dataflow graphs we are able to exploit parallelism present at various levels like inter-pixel parallelism and intra-pixel parallelism. Based on these observations

we have suggested here a reconfigurable architecture that reconfigures itself based on the percentage of valid voxels so that performance is maximized while keeping area consumption to a minimum.

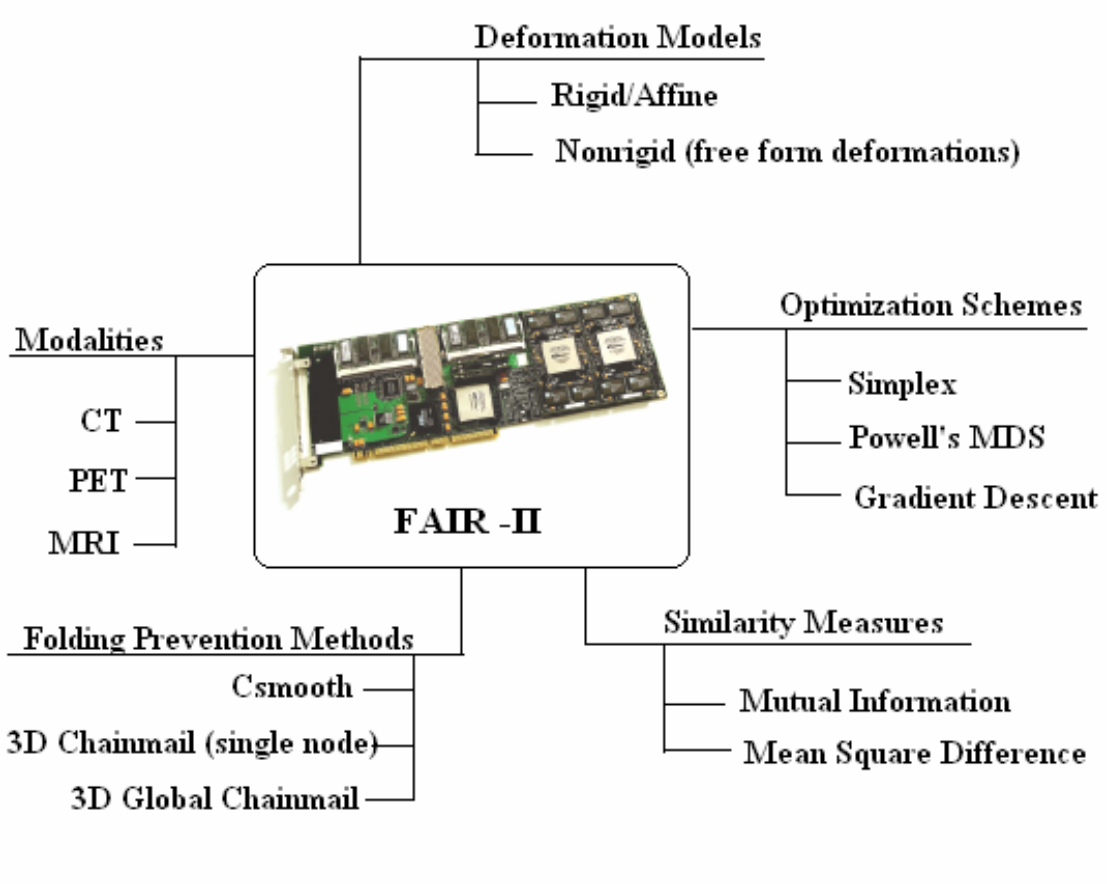

Figure 35 Overview of the FAIR registration system

7.1 Future Directions

Further improvements, such as the use of a finer grid, can further equalize the hardware and software implementations. With FPGAs with larger internal memory, the LUTs used at various stages can be made larger thus increasing accuracy. With higher DSP resources and higher memory resources, a second order approximation can be used to improve the accuracy of the entropy calculator. With a more accurate entropy calculator the registration times are driver further down while improving the registration accuracies.

With 64 bit memory buses available on latest FPGA boards, and higher memories available we can have the entire 8 neighborhood values of the FI stored at every voxel position, thus reducing the FI access times by half. Along with this, having 8 copies of the MH can effectively double the voxel processing rates in the hardware. With the use of new generation FPGAs which can run at very high speeds compared to the FPGA used and use of memories run at higher frequencies than the SDRAMs used in the current implementation can further increase the voxel processing rates. Using multi-resolution strategies like image sub-sampling during registration at lower grid resolutions allows for further increase in speed. If needed, additional FPGA-based preprocessing steps can also be added to the pipeline, thus ensuring additional processing at no extra time cost.

7.1.1 Exploiting Various Degrees of Parallelism

The future development work in this system can concentrate on exploiting the various levels of parallelism in the system in conjunction with the existing FAIR architecture. With multiple FPGA nodes available containing the similar designs,

there is a potential to use parallel nodes to exploit parallelism between the optimization schemes, i.e. allowing multiple optimization schemes running on different nodes to find the best transformation (optimization level parallelism). With gradient descent based optimization scheme, it is possible to achieve inter-node parallelism which will reduce registration times by calculation of gradients for different control points on different FPGAs as previously described. Parallel implementations of simplex based optimization scheme can also be utilized for optimizing more than one control point at any instance which will drive registration times lower (inter-node parallelism). There is a potential for parallelism by dividing the image into different subvolumes and processing parts of the image on different nodes (inter-pixel parallelism). With FPGAs with larger memory bandwidth available, the PV interpolation can be further parallelized to 8 data-paths from the existing 4 data-paths (intra pixel parallelism). With these techniques the registration times can be reduced to second-order which will make image registration an indispensable tool for a variety of time-critical medical applications like image guided interventions and surgeries.

Appendices

1. LUT-based Entropy Calculation

(Described in detail in [26])

The function $f(p)$ is approximated in the range [0, 1] by using the piecewise-polynomial function $\hat{f}_{N,m}(p)$ with *m* segments, defined in equation (28) below.

$$\hat{f}_{N,m}(p) = \begin{cases} P_0(p) & \text{for } 0 \le p < \Delta p \\ P_1(p \bmod \Delta p) & \text{for } \Delta p \le p < 2\Delta p \\ \vdots & \vdots \\ P_i(p \bmod \Delta p) & \text{for } i \cdot \Delta p \le p < (i+1) \cdot \Delta p \\ \vdots & \vdots \\ P_{m-1}(p \bmod \Delta p) & \text{for } (m-1) \cdot \Delta p \le p < 1 \end{cases} \quad \textbf{(28)}$$

where $\Delta p = 1/m$ is the segment size of $\hat{f}_{N,m}(p)$ and P_i is a polynomial of order *N*-1. To minimize the maximum approximation error, each P_i is obtained by calculating the Chebyshev approximation for $f(p)$, for $i \cdot \Delta p \le p < (i+1) \cdot \Delta p$. The Chebyshev approximation is simple to calculate for continuous functions and has the advantage that it is very close to the minimax approximation, the most accurate polynomial approximation.

Equation (29,30) is used to calculate the coefficients.

$$P_i(p \bmod \Delta p) \approx \left[\sum_{k=0}^{N-1} c_{i,k} T_k\left(\frac{2(p \bmod \Delta p)}{\Delta p} - 1\right)\right] - \frac{1}{2} c_{i,0} \quad \textbf{(29)}$$

$$c_{i,k} = \frac{2}{N}\sum_{l=1}^{N} f\left[\frac{\Delta p}{2}\cos\left(\frac{\pi(l-1/2)}{N}\right) + \Delta p \cdot (i+1/2)\right]\cos\left(\frac{\pi k(l-1/2)}{N}\right) \quad \textbf{(30)}$$

The Chebyshev polynomials are defined by $T_n(x) = \cos(n \cdot \arccos(x))$, for -1 ≤ x ≤ 1. The Chebyshev polynomials of order zero to three are shown in equation(31,32). Since each polynomial is used to approximate $f(p)$ in a specific $[i \cdot \Delta p, \quad (i+1) \cdot \Delta p]$ range, whereas the Chebyshev polynomials are defined in [-1, 1], the variable conversion shown in (32) is applied to the equations.

$$T_0(x) = 1,\ T_1(x) = x,\ T_2(x) = 2x^2 - 1,\ T_3(x) = 4x^3 - 3x \quad \textbf{(31)}$$

$$x = (2(p \bmod \Delta p) - \Delta p)/\Delta p \quad \textbf{(32)}$$

To keep the arithmetic pipeline simple, only the first, second and third-order approximations in are considered for the hardware implementation. Equation (33) defines the *i*th polynomial component of $\hat{f}_N(p)$:

$$P_i(p_d) = k_{i,3} \cdot p_d^3 + k_{i,2} \cdot p_d^2 + k_{i,1} \cdot p_d + k_{i,0} \quad \textbf{(33)}$$

where $p_d = p \bmod \Delta p$. The polynomial coefficients $k_{i,j}$ are stored in the ith entry of the LUT. They are calculated from the Chebyshev coefficients as shown in equations (34), which are derived from equations (29), (30) and (31).

$$\begin{aligned} k_{i,0} &= 0.5 \cdot c_{i,0} - c_{i,1} + c_{i,2} - c_{i,3} \\ k_{i,1} &= \left(c_{i,1} - 4 \cdot c_{i,2} + 9 \cdot c_{i,3}\right) \cdot \left(2/\Delta p\right) \\ k_{i,2} &= \left(2 \cdot c_{i,2} - 12 \cdot c_{i,3}\right) \cdot \left(2/\Delta p\right)^2 \\ k_{i,3} &= 4 \cdot c_{i,3} \cdot \left(2/\Delta p\right)^3 \end{aligned} \quad \textbf{(34)}$$

Bibliography

[1] Y.-C. Tai, K. P. Lin, C. K. Hoh, S. C. H. Huang, and E. J. Hoffman, "Utilization of 3-D elastic transformation in the registration of chest X-ray CT and whole body PET," *IEEE Transactions on Nuclear Science*, vol. 44, pp. 1606-1612, 1997.

[2] D. Mattes, D. R. Haynor, H. Vesselle, T. K. Lewellen, and W. Eubank, "PET-CT image registration in the chest using free-form deformations," *IEEE Transactions on Medical Imaging*, vol. 22, pp. 120-8, 2003.

[3] M. R. Kaus, T. Netsch, S. Kabus, V. Pekar, T. McNutt, and B. Fischer, "Estimation of organ motion from 4D CT for 4D radiation therapy planning of lung cancer," in *Lecture Notes in Computer Science*, vol. 3217: Springer-Verlag GmbH, 2004, pp. 1017-1024.

[4] G. C. Sharp, S. B. Jiang, S. Shimizu, and H. Shirato, "Prediction of respiratory tumour motion for real-time image-guided radiotherapy," *Physics in Medicine & Biology*, vol. 49, pp. 425-40, 2004.

[5] F. Sauer, "Image Registration: Enabling Technology for Image Guided Surgery and Therapy," presented at Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference, Shanghai, China, 2005.

[6] R. Shekhar, P. Lei, C. R. Castro-Pareja, and W. L. Plishker, "Automatic segmentation of phase-correlated CT scans through nonrigid image registration using geometrically regularized free-form deformation," *Medical Physics (in press)*, 2007.

[7] V. Walimbe, O. Dandekar, F. Mahmoud, and R. Shekhar, "Automated 3D Elastic Registration for Improving Tumor Localization in Whole-body PET-CT from Combined Scanner," presented at The 28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE EMBS 2006), 2006.

[8] C. R. Castro-Pareja, J. M. Jagadeesh, and R. Shekhar, "FAIR: A hardware architecture for real-time 3-D image registration," *IEEE Transactions on Information Technology in Biomedicine*, vol. 7, pp. 426-434, 2003.

[9] S. Ourselin, R. Stefanescu, and X. Pennec, "Robust registration of multi-modal images: Towards real-time clinical applications," presented at Medical Image Computing and Computer-Assisted Intervention - MICCAI 2002: 5th International Conference Proceedings, Part II, Tokyo, 2002.

[10] Stefanescu R., Pennec X., and Ayache N., "Parallel non-rigid registration on a cluster of workstations," *Proc. of HealthGrid'03*, 2003.

[11] S. K. Warfield, F. Talos, A. Tei, A. Bharatha, A. Nabavi, M. Ferrant, P. McL. Black, F. A. Jolesz, and R. Kikinis, "Real-time registration of volumetric brain MRI by biomechanical simulation of deformation during Image guided neurosurgery," *Computing and Visualization in Science*, vol. 5, pp. 3-11, 2002.

[12] F. Ino, K. Ooyama, and K. Hagihara, "A data distributed parallel algorithm for nonrigid image registration," *Parallel Computing*, vol. 31, pp. 19-43, 2005.

[13] D. Rueckert, L. I. Sonoda, C. Hayes, D. L. 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.

[14] T. Rohlfing and C. R. Maurer, Jr., "Nonrigid image registration in shared-memory multiprocessor environments with application to brains, breasts, and bees," *IEEE Transactions on Information Technology in Biomedicine*, vol. 7, pp. 16-25, 2003.

[15] M. Ohara, H. Yeo, F. Savino, G. Iyengar, L. Gong, H. Inoue, H. Komatsu, V. Sheinin, S. Daijavad, and B. Erickson, "Real-time mutaul information based linear registration on the cell broadband engine processor.," *in proceedings of IEEE International Symposium of Biomedical Imaging*, pp. 33-36, 2007.

[16] C. R. Castro-Pareja and R. Shekhar, "A cubic interpolation pipeline for fast computation of 3D deformation fields modeled using B-splines," presented at Real-Time Image Processing 2006, San Jose, CA, USA, 2006.

[17] C. R. Meyer, J. L. Boes, B. Kim, P. H. Bland, K. R. Zasadny, P. V. Kison, K. Koral, K. A. Frey, and R. L. Wahl, "Demonstration of accuracy and clinical versatility of mutual information for automatic multimodality image fusion using affine and thin-plate spline warped geometric deformations," *Medical Image Analysis*, vol. 1, pp. 195-206, 1997.

[18] E. Haber, J. Modersitzki, and, "Image Registration with guaranteed displacement regularity," *International Journal of Computer Vision*, vol. 71, pp. 361-372, 2007.

[19] W. Greene, S. Chelikani, X. Papademetris, J. Knisely, and J. S. Duncan, "A constrained nonrigid registration algorithm for application in prostrate radiotherapy," *In proceedings of IEEE International Symposium of Biomedical Imaging*, pp. 740-743, 2007.

[20] J. B. Maintz and M. A. Viergever, "A survey of medical image registration," *Medical Image Analysis*, vol. 2, pp. 1-36, 1998.

[21] F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, and P. Suetens, "Multimodality image registration by maximization of mutual information," *IEEE Transactions on Medical Imaging*, vol. 16, pp. 187-98, 1997.

[22] W. M. I. Wells, P. Viola, H. Atsumi, S. Nakajima, and R. Kikinis, "Multi-modal volume registration by maximization of mutual information," *Medical Image Analysis*, vol. 1, pp. 35-51, 1996.

[23] J. P. Pluim, J. B. Maintz, and M. A. Viergever, "Mutual-information-based registration of medical images: A survey," *IEEE Trans Med Imaging*, vol. 22, pp. 986-1004, 2003.

[24] M. Capek et al., Proceedings of, part I, pp., "“Robust and Fast Medical Registration of 3D-Multi-Modality Data Sets”," in *Medicon 2001 - IX Mediterranean Conference on Medical and Biological Engineering and Computing*. 515-518, 2001.

[25] M. Doggett and M. Meißner, "A memory addressing and access design for real time volume rendering," *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 344-347, 1999.

[26] C. R. Castro-Pareja, O. Dandekar, and R. Shekhar, "FPGA-based real-time anisotropic diffusion filtering of 3D ultrasound images," *Proceedings of SPIE (Electronic Imaging 2005)*, vol. 5671, pp. 123-131, 2005.

[27] C. R. Castro-Pareja, J. M. Jagadeesh, and R. Shekhar, "FPGA-based acceleration of mutual information calculation for real-time 3D image registration," presented at Real-Time Imaging VIII, San Jose, CA, USA, 2004.

[28] M. P. Wachowiak and T. M. Peters, "High-performance medical image registration using new optimization techniques," *IEEE Transactions on Information Technology in Biomedicine*, vol. 10, pp. 344-353, 2006.

[29] J. A. Nelder and R. Mead, "A simplex method for function minimization," *J. Computer*, vol. 7, pp. 308-313, 1965.

[30] W.H.Press, B.P.Flannery, S.A.Teukolsky, and W.T.Vetterling, "Numerical Recipes in C," *Cambridge University Press*, 1988.

[31] R. Shekhar, P. Lei, C. R. Castro-Pareja, and W. L. Plishker, "Automatic segmentation of phase-correlated CT scans through nonrigid image registration using geometrically regularized free-form deformation," *Medical Physics (in press)*, 2008.

[32] C. R. Castro-Pareja, O. S. Dandekar, and R. Shekhar, "FPGA-based real-time anisotropic diffusion filtering of 3D ultrasound images," presented at Real-Time Imaging IX, San Jose, CA, USA, 2005.

[33] M. Sen, S. S. Bhattacharyya, T. Lv, and W. Wolf, "Modeling image processing systems with homogeneous parameterized dataflow graphs," *In Proc. International Conference on Acoustics, Speech, and Signal Processing*, pp. 133-136, 2005.

[34] Y. Hemaraj, M. Sen, R. Shekhar, and S. S. Bhattacharyya, "Model-based mapping of image registration applications onto configurable hardware," *In Proc. IEEE Asilomar Conference on Signals, Systems, and Computers*, pp. 1453-1457, 2006.

[35] M. Sen, Y. Hemaraj, S. S. Bhattacharyya, and R. Shekhar, "Reconfigurable image registration on FPGA platforms.," *In Proceedings of the IEEE Biomedical Circuits and Systems Conference*, pp. 154-157, 2006.

[36] S. S. Bhattacharyya, P. K. Murthy, and E. A. Lee, " Software Synthesis from Dataflow Graphs," *Kluwer Academic Publishers*, 1996.

[37] M. Sen and S. S. Bhattacharyya, "Systematic exploitation of data parallelism in hardware synthesis of DSP applications," *In Procs of the International Conference on Acoustics, Speech, and Signal Processing*, pp. 229-232, 2004.

[38] F. Haim, M. Sen, D. Ko, S. S. Bhattacharyya, and W. Wolf, "Mapping multimedia applications onto configurable hardware with parameterized cyclo-static dataflow graphs.," *In Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pp. III-1052-III-1055, 2006.