

RICE UNIVERSITY

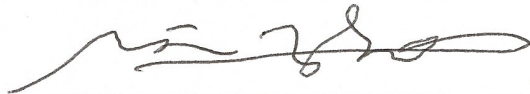
**Radial MILO: A 4D Image Registration Algorithm
Based on Filtering Block Match Data via
 ℓ_1 -minimization**

by

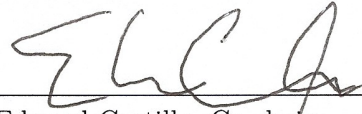
Arturo Vargas

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE
Master of Arts

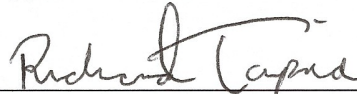
APPROVED, THESIS COMMITTEE:



Yin Zhang, Chair
Professor of Computational and Applied
Mathematics



Edward Castillo, Co-chair
Adjunct Professor of Computational and
Applied Mathematics



Richard Tapia
University Professor
Maxfield-Oshman Professor in
Engineering



Timothy Warburton
Professor of Computational and Applied
Mathematics

Houston, Texas

April, 2015

ABSTRACT

Radial MILO: A 4D Image Registration Algorithm Based on Filtering Block Match
Data via ℓ_1 -minimization

by

Arturo Vargas

Minimal ℓ_1 Perturbation to Block Match Data (MILO) is a spatially accurate image registration algorithm developed for thoracic CT inhale/exhale images. The MILO algorithm consists of three components: (1) creating an initial estimate for voxel displacement via a Mutual Minimizing Block Matching Algorithm (MMBM), (2) a filtering step based on ℓ_1 -minimization and a uniform B-spline parameterization, and (3) recovering a full displacement based on the filtered estimates. This thesis presents a variation of MILO for 4DCT images. In practice, the use of uniform B-splines has led to rank deficient linear systems due to the spline's inability to conform to non-structured MMBM estimates. In order to adaptively conform to the data an octree is paired with radial functions. The ℓ_1 -minimization problem had previously been addressed by employing QR factorization, which required substantial storage. As an alternative a block coordinate descent algorithm is employed, relieving the need for QR factorization. Furthermore, by modeling voxel trajectories as quadratic functions in time, the proposed method is able to register multiple images.

Acknowledgments

I would like to thank the members of my committee, Dr. Edward Castillo, Dr. Yin Zhang, Dr. Richard Tapia, and Dr. Timothy Warburton; especially to my advisors Dr. Edward Castillo and Dr. Yin Zhang who first introduced me to image registration and ℓ_1 optimization. Their time and patience is what made this possible. Dr. Richard Tapia's speeches always provided inspiration and motivation, and Dr. Timothy Warburton's computational science courses provided the tools to implement Radial MILO effectively. I would also like to thank the CAAM 600 professors Dr. Sorensen, Dr. William Symes, and Dr. Jan Hewitt for the thesis writing course, the lessons learned were invaluable.

I thank my family as well as my friends from Fullerton, California, Jonathan Bernal, Matthew Hendricks, and Eamon Donovan for their support. I thank my CAAM cohort and CAAM friends who have made these last few years wonderful. To Veronica Landa, who helped me revise my thesis and stayed up late to hear me talk about image registration. I dedicate this thesis to my brother and sisters. I hope they find something that sparks their interest and inspires them to strive for excellence. Finally, a big thanks to the NSF for providing the funding to carryout this project. NSF GRFP grant number: DGE-1450681.

Contents

Abstract	ii
Acknowledgments	iii
1 Introduction	1
2 Medical Image Registration	4
2.1 Types of Transformations	4
2.2 Medical Image Processing on GPGPU's	5
2.3 Image Registration Benchmark Tests	7
2.4 <i>Minimal ℓ_1 Perturbation to Block Match Data (MILO) Algorithm</i> . .	8
2.5 Contributions: Radial MILO	9
3 Mathematical Concepts for Image Registration Algorithm	
Development	11
3.1 Voxel Displacement Modeling	11
3.1.1 Lagrangian Coordinate Framework	12
3.1.2 Recovering Coefficients	13
3.2 Block Matching	13
3.2.1 Block Matching as an Image Registration Algorithm	15
3.2.2 Mutual Minimizing Block Matching Algorithm	16
3.3 B-splines	16
3.3.1 B-splines in Image Registration	19
3.3.2 B-splines with MMBM data	19
3.4 On ℓ_1 -minimization	20

3.4.1	Compressive Sensing	20
3.4.2	De-Noising Basis Pursuit/LASSO	21
3.5	$2^{\hat{d}}$ Tree Data Structures	24
3.6	Radial Basis Functions	27
3.6.1	Local Approximations: Moving Least Squares	27
3.6.2	Image Registration Based on Radial Functions	28
3.7	Conforming Parameterization	28
4	Methods and Implementation	30
4.1	Segmenting a Thoracic CT Image	31
4.2	GPGPU Computing Model	33
4.3	Block Matching on the GPGPU	34
4.4	The $2^{\hat{d}}$ data structure class	37
4.5	Radial Functions	39
4.5.1	Overview of Globally Supported Radial Functions	39
4.5.2	A Class of Compactly Supported Functions: Wendland's Functions	40
4.5.3	Discussion	41
4.6	Filtering MMBM Data via ℓ_1 -minimization	41
4.6.1	Block Coordinate Descent	42
4.6.2	Filtering Algorithm via ℓ_1 -minimization	44
4.6.3	Moving Least Squares	45
4.7	Radial MILO Algorithm	46
5	Numerical Experiments	47
5.1	Image Description	47
5.2	Block Matching Parameters	48
5.3	Registration Results	54
5.3.1	Spatial Accuracy: Expiratory Phases	55

5.3.2	Spatial Accuracy: Maximum Inhalation and Exhalation . . .	56
5.4	Review of DIR-LAB Reported Algorithms	59
5.4.1	Searching Based Methods	59
5.4.2	Optical Flow Based Methods	60
5.4.3	Demons Based Methods	61
5.4.4	Trajectory Based Methods	62
5.4.5	Variational Methods	63
5.5	Visual Results	64
6	Conclusion	66
	Bibliography	68

Chapter 1

Introduction

Images play a key role in many different fields. For example, remote sensing employs collections of images for environmental modeling, weather forecasting, and integrating geographic information systems [1, 2, 3, 4, 5]. Medical science uses image sequences to monitor tumor growth and verify treatment [6, 7, 8, 9, 10]. Whether the application is image mosaicing or feature tracking, an underlying model must be found in order to relate the images. Finding the underlying model is the focus of image registration. Image registration can be broadly defined as:

*Definition 1 **Image Registration** : the determination of a spatial transformation that best relates information depicted within a sequence of images with respect to an assumed motion and/or intensity model.*

Depending on the image and the goal of the registration, different approaches to image registration can be taken. Zitova and Flusser [11] group the objectives of image registration into four main categories:

- **Multiview Analysis:** Given images from different view points, the goal is to create a larger image, for example a mosaic.
- **Multitemporal Analysis:** Given images taken at different times, the goal is to construct an intermediate image or estimate displacement.
- **Multimodal Analysis:** Given images acquired by different sensors, the goal

is to combine the images to create a more detailed image.

- **Scene to Model Registration:** Given a model image and a related image (scene) with similar content, the goal is to find how the scene may fit into the model. For example, in remote sensing, images are registered from arial or satellite data onto maps or geographic information systems.

During the last few decades, image acquisition devices have undergone rapid development creating an influx of diverse types of images, and invoking research in automatic image registration. Automatic image registration is desirable as it would minimize the need for human intervention to register images. Various researchers have put great effort into creating image registration algorithms [11, 12, 13, 14]. Due to differing image content and goals, there is no universal image registration algorithm. Image registration algorithms do, however, share a common anatomy [11, 12, 13, 14].

Mathematically, image registration seeks to find the optimum transformation, W , to relate a **reference** image, R , and a set of **target** images at time, T_i , under a given metric, J :

$$\mathbf{W}^* = \arg \min_W \sum_i J(R, T_i, W). \quad (1.1)$$

Image registration algorithms are typically composed of the following:

1. **Deformation Model:** Class of voxel-to-voxel maps
2. **Objective Function:** A function that serves to measure how well a model is able to relate images under a given metric.
3. **Optimization Scheme:** An approach to find the "best" member of the class

In this thesis I consider image registration for Thoracic Four Dimensional Computed Tomography images (4DCT). Thoracic 4DCT captures sequences of 3D im-

ages of the thorax during respiration. These images can show how the pulmonary parenchyma changes with air content. When we breathe our lungs expand and contract in a non-uniform fashion, thus obtaining physiological or quantitative motion data from these image sets requires image registration [14, 15].

I begin by providing an overview of image registration methods and discuss the use of the general purpose graphics processing unit (GPGPU) in medical image analysis. I then discuss the use of benchmark images to judge the spatial accuracy of registrations, and introduce the *Minimal ℓ_1 Perturbation to Block Match Data* (MILO) algorithm. MILO is a spatially accurate algorithm for multitemporal analysis of three-dimensional computed tomography (3DCT) images that has provided the inspiration for this thesis [16]. I then provide a discussion on each of MILO’s components and their uses in other image registration algorithms. Lastly I address MILO’s weaknesses, particularly that it has not been extended to 4D, and present Radial MILO, a 4DCT image registration algorithm.

Chapter 2

Medical Image Registration

2.1 Types of Transformations

The goal of multitemporal registration is to recover an optimal spatial transformation that describes the displacement of voxels for a reference image, [11]. This means that the type of motion captured by the sequences of images will influence the type of modeling needed. In order to capture the intricate deformations of the lungs that occur during the breath cycle, models that allow for complex deformations must be used. There are three main modeling approaches for deformable image registration that appear in the literature:

- **Optical Flow Based Deformation:** Optical flow is described by Horn and Schunck as the "distribution of apparent velocities of movement of brightness patterns in an image" [17]. Optical flow methods calculate the motion between two image frames for every voxel position. Horn and Schunk's work on determining optical flow has led to various registration algorithms [18, 19, 20, 21].
- **Free-form Deformation:** Free-form deformations parameterize a spatial transformation. They describe the displacement field as a linear combination of basis functions. Defining $\phi(\mathbf{x})$ as displacement field,

$$\phi(\mathbf{x}) = \sum_{i=1}^N \alpha_i B_i(\mathbf{x}). \quad (2.1)$$

Each B_i denotes a basis function where $\alpha_i, \mathbf{x} \in \mathbb{R}^{\hat{d}}$ (\hat{d} denotes dimension) are coefficients and voxels, respectively, and N is the number of basis functions. Various registration algorithms have been designed with varying types of basis functions, such as B-splines, radial basis functions, etc. [11, 16, 18, 22, 23].

- **Physics Based Deformations:** Physics based deformations have an underlying stiffness model [18, 24, 25]. The strength in incorporating physics is allowing different tissues in images to have different physical attributes.

Because medical images are being collected at much faster rates, the need to process images has also increased. More notably the need for faster algorithms has become apparent as the resolution of the images has increased. To illustrate computational demands of images, consider the difference between 2D, 3D, and 4D images composed of data type unsigned short integers (2 bytes). In terms of data size a 512×512 image is roughly 0.5 MB, a 3D volume image of $512 \times 512 \times 128$ is 67 MB and a $512 \times 512 \times 128 \times 10$ 4D image is around 671 MB. Higher resolution images have more voxels (pixels in 2D) and require larger storage. As data sets get larger the algorithm complexity increases, and with it computational cost. The growing collection of larger images has led to an increasing demand for fast and spatially accurate image processing, data analysis, visualization, and interactivity tools [26].

2.2 Medical Image Processing on GPGPU's

In 2007, with the release of Nvidia's Compute Unified Device Architecture (CUDA), computational scientists were given access to simpler ways to program GPGPU's. To computational scientists, GPGPU's provided a way to perform thousands of computations in parallel [26, 27]. Many image processing algorithms port easily to the

GPGPU because they act on individual voxels. For example, when applying a filter to an image, it is applied to every voxel of the image and requires no communication with neighboring voxels; a single instruction is assigned to different data.

A drawback back of CUDA is that it is only designed to work with Nvidia GPGPU's. As such, several industries came together to create a standardized application programming interface (API) called the Open Compute Language (OpenCL) [26, 27, 28]. The purpose of the language is to allow programmers to manage parallelism and data delivery in massive quantities through parallel processors. The OpenCL model goes beyond GPGPU's, allowing the programming model to be mapped to homogenous or heterogenous, single or multiple-device systems consisting of CPU's, GPGPU's, Field-Programmable Gate Arrays (FPGA), and potentially other future devices.

The need to quickly register large quantities of images and the rapidly improving performance of GPGPU's have brought researchers to develop GPGPU based algorithms with parallelizable computations. A major benefit of using a GPGPU for image processing is that images can be stored in specialized texture memory. Texture memory has the advantage of offering highly efficient access to image data in scenarios where memory access patterns exhibit a great deal of spatial locality and offers texture interpolation when needed [18, 28].

With the development of techniques and tools for image registration, a method for accessing spatial accuracy was needed. This necessity introduced the notion of benchmark images.

2.3 Image Registration Benchmark Tests

In 2011, Murphey et. al. [29], published the results of the Evaluation of Methods for Pulmonary Image REgistrations 2010 (EMPIRE10). This challenge served as a public platform for the comparison of registration algorithms designed for thoracic CT image pairs. The EMPIRE10 challenge sought to create a fair playing field to compare algorithms by having a third party provide images for registration and judge the quality of the registration. Judging is based on an algorithm’s alignment of lung boundaries and major fissures, correspondence to annotated landmark pairs, and singularities in the deformation field [29]. However, having a third party perform the evaluation, meant that it could only be done when there is a judge available. To address this both drawback the Deformable Image Registration Lab (DIR-LAB) and Vandemeuebroucke et. al. [30], offer free online collections of CT images that can be used to test image registration algorithms at www.dir-lab.com and at www.creatis.insa-lyon.fr/rio/popi-model, respectively. Expertly located landmark points are annotated throughout the images and treated as ground truth [15, 16, 30, 31, 32]. This type of benchmark testing allows for self-testing of a variety of registration algorithms by eliminating the need of a third party to judge the quality of the registration. Due to its accessibility I employ the image sets provided by the DIR-LAB (Table 5.1). For future work other image sets will be employed.

The overall best registration algorithm at the EMPIRE10 challenge, greedy diffeomorphic registration (GSYN), was developed by a team from the University of Pennsylvania’s Image Computing and Science Laboratory [29, 33]. Their algorithm employs the data processing library, ANTs [34], and consists of (1) an affine registration step and (2) a diffeomorphic step. The affine registration step provides an initial global alignment between lungs, and the diffeomorphic step introduces more

degrees of freedom to improve precision when mapping the displacement of voxels. By definition a diffeomorphism is a smooth bijective mapping with a smooth inverse. Diffeomorphic transformations can be categorized as a physics based model, governed by the Lagrange Transport Equation [14]. The EMPIRE10 challenge is still active and researchers can submit their results online at <http://empire10.isi.uu.nl/submit.php>. When compared to MILO [16], GSYN performed overall less accurately on DIR-LAB data.

2.4 Minimal ℓ_1 Perturbation to Block Match Data (MILO)

Algorithm

The MILO algorithm is a spatially accurate image registration algorithm developed for temporal analysis of thoracic CT inhale/exhale images. The algorithm has achieved high spatial accuracy in CT image pairs and has been shown to out-perform, or perform as competitively as, the leading registration algorithms based on DIR-LAB benchmark images [16]. Although MILO delivers satisfactory results, there are drawbacks. Therefore, the focus of this thesis is to explore a variant of this algorithm for 4D image registration.

MILO is based on three components, (1) a Mutual Minimizing Block Matching Algorithm (MMBM), (2) an ℓ_1 filtering step, and (3) recovering a full parameterization based on filtered data points. MMBM estimates describe specific voxel displacements via an exhaustive search and a voxel intensity metric. A limitation of the MMBM algorithm is the fact that an optimal point match, in terms of the similarity metric, may not be optimal in terms of spatial accuracy [16], thus filtering is necessary. MILO uses MMBM data in conjunction with uniform B-spline basis functions to

recover an initial displacement field. Under the assumption that there is a small amount of spatially inaccurate data relative to the number of points, ℓ_1 -minimization techniques are used to identify potentially inaccurate data. A full displacement field is then recovered based on the remaining points [16].

Though the MILO algorithm has performed well, in practice the algorithm has resulted in some issues:

1. It employs a uniform B-spline mesh which may result in a rank deficient matrix if MMBM data does not lie within the support of a basis function.
2. The ℓ_1 -minimization problem is addressed by employing QR factorization, which requires substantial storage.
3. The modeling approaches limit the registration to pairs of images.

2.5 Contributions: Radial MILO

As a variant of the MILO algorithm, I present Radial MILO. Radial MILO contributes the following:

1. An alternative to a uniform mesh, a $2^{\hat{d}}$ tree data structure is used to spatially partition the estimates from the MMBM algorithm. A spatial parameterization is constructed by employing a linear combination of radial basis functions centered at the occupied leaves of the tree, thereby conforming to the MMBM data.
2. To address the ℓ_1 minimization problem, Radial MILO employs a block coordinate descent, relieving the explicit need for QR factorization.

3. Lastly, voxel displacements are modeled as quadratic trajectories to create a 4D image registration algorithm.

Chapter 3

Mathematical Concepts for Image Registration Algorithm Development

3.1 Voxel Displacement Modeling

Both the MILO and Radial MILO algorithms can be classified as free-form image registration algorithms. As such MILO and Radial MILO employ a spatial parameterization for a voxel's displacement and trajectory, respectively. Mathematically, for a pair of images, a free-form registration algorithm seeks to recover a displacement field, $\phi(\mathbf{x})$, for voxels, \mathbf{x} , for the following spatial transformation, ψ .

$$\psi(\mathbf{x}) = \mathbf{x} + \phi(\mathbf{x}), \quad \psi : \mathbb{R}^3 \rightarrow \mathbb{R}^3. \quad (3.1)$$

By simply parameterizing displacement, Equation (2.1), the spatial mapping, Equation (3.1), is limited to only offering the new location of the voxel on the target image. Recalling that the goal of this thesis is to perform 4D image registration, the most straight-forward approach would be to interpolate the displacement fields provided by 3D image registration algorithms. An issue with this approach is the fact that the sequences of images are not incorporated into a global model (i.e. the problem is completely decoupled). Furthermore, errors in the pairwise image registration would propagate as the displacement fields are interpolated [15].

However, since 4D images are simply a collection of 3D images, registering across a 4D image set is the recovery of each voxel's spatial trajectory as a function of time.

As such, this approach lends itself well to a **Lagrangian** coordinate system. Figure (3.1) illustrates the difference between modeling displacement versus modeling a voxel trajectory.

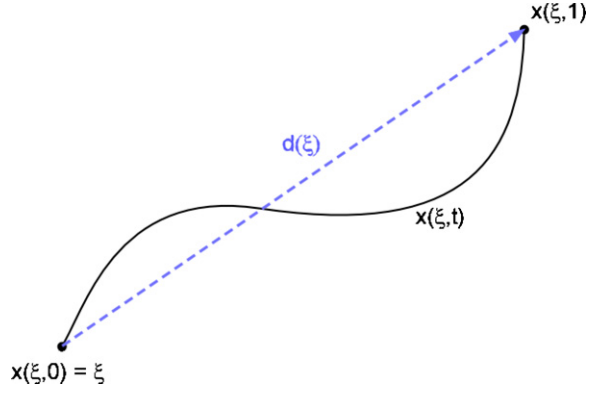


Figure 3.1 : Displacement, \mathbf{d} , versus trajectory of a voxel, \mathbf{x} , at the initial position, ξ , from time 0 to 1 [15].

3.1.1 Lagrangian Coordinate Framework

The Lagrangian coordinate system models the trajectory of individual voxels as a function of time. For example, the method in [35] employs a polynomial model in terms of time, t ,

$$\Upsilon(\mathbf{x}, t; q(\mathbf{x})) = \begin{bmatrix} q_1(\mathbf{x})t^N + q_2(\mathbf{x})t^{N-1} + \dots + q_N(\mathbf{x})t \\ q_{N+1}(\mathbf{x})t^N + q_{N+2}(\mathbf{x})t^{N-1} + \dots + q_{2N}(\mathbf{x})t \\ q_{2N+1}(\mathbf{x})t^N + q_{2N+2}(\mathbf{x})t^{N-1} + \dots + q_{3N}(\mathbf{x})t \end{bmatrix} + \mathbf{x}. \quad (3.2)$$

The spatial mapping, Υ , defines the trajectory path for the voxels, \mathbf{x} , in the reference image. The terms, $q_k(\mathbf{x})$ and t represent coefficients and time, respectively. Similar to displacement, each coefficient, $q_k(\mathbf{x})$, can be parameterized,

$$q_k(\mathbf{x}) = \sum_{i=1}^N \alpha_i B_i^k(\mathbf{x}). \quad (3.3)$$

In a study done by Castillo et. al. [35], it is demonstrated that there is no significant difference in employing a quadratic, cubic, quartic, or a quintic function when registering images throughout the expiratory phase. As would be expected, they also show a linear model does significantly worse in comparison for 4D modeling. Since the DIR-LAB provides propagated landmarks for the expiratory phase for assessing accuracy, Radial MILO is used to register only over that phase, and is therefore paired with a quadratic function.

3.1.2 Recovering Coefficients

By modeling displacements as a linear combination of basis functions, free-form image registration reduces to finding a best set of coefficients for a subset of voxels in the reference image. This approach lends itself to two different methods for recovering coefficients. One can either pose an optimization problem with an assumed metric and optimize over the set of coefficients, or find optimal corresponding voxels throughout the images and interpolate. A novelty of the original MILO algorithm is the use of constrained optimization to find corresponding voxels from the reference and target images. The MILO algorithm poses a constrained optimization problem that seeks estimates to minimize the block match similarity metric, subject to an acceptable least squares residual based on a B-Spline parameterization [16].

3.2 Block Matching

The block matching algorithm has origins in video compression [36, 37, 38] and has demonstrated utility in medical image registration [16, 39, 40]. Block matching seeks

to find corresponding voxel locations in pairs of related images. The algorithm partitions a reference image into blocks (*reference blocks*) of $n \times m \times z$ voxels, chosen based on distinctive features of the reference image. Each block is then assigned to a search window in the target image. In order to compare candidate blocks it is necessary that the search window be at least the same size as the reference block, thus the search window consists of $(n + 2 \times p_0) \times (m + 2 \times p_0) \times (z + 2 \times p_0)$ voxels, where p_0 denotes the maximum allowed displacement (*search window radius*).

There are a few variations on the block matching algorithm [38], but the most intuitive and accurate is the Full Search Algorithm (FSA). FSA compares the reference block with each possible candidate block of the same size reference block in the search window, potentially $(2 \times p_0 + 1)^3$ candidates, according to a given metric. The choice of metric is dependent on the image content and potential voxel intensity changes between images. Clearly, this is a computationally intensive process; however, the block matching algorithm is well suited for parallel computing. Thus, using the GPGPU, the block matching algorithm can be executed in a reasonable amount of time [38, 16].

Adopting the notation used in [16], block matching can be expressed as the following discrete optimization problem:

$$\arg \min_{\mathbf{d} \in B_{p_0}} F(R(\mathbf{x}), T(\mathbf{x} + \mathbf{d})) \quad (3.4)$$

where B_{p_0} denotes $\{\mathbf{d} \in \mathbb{Z}^{\hat{d}} : \|\mathbf{d}\|_{\infty} \leq p_0\}$ as a search window, F is the employed metric, R and T are the reference and target images, respectively, and \mathbf{x} is a voxel (center of the reference block). An illustration denoting the mechanics of block matching is found in Figure 3.2.

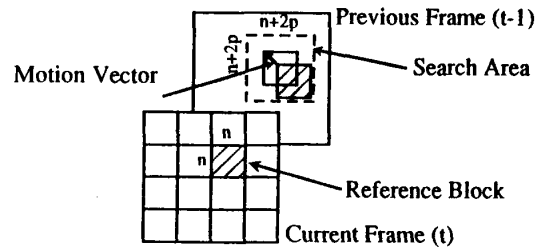


Figure 3.2 : Illustration of Block Matching process, [36]

3.2.1 Block Matching as an Image Registration Algorithm

Since the objective of pairwise image registration is to find voxel displacements, it is quite natural to employ block matching for image registration. Various image registration algorithms, including MILO, are built on the block matching algorithm [41, 40, 42]. An example is Castillo, et. al.'s three step algorithm based on compressible flow, LCF [35]. The algorithm first formulates a non-linear least squares problem based on a compressible flow model. The objective function is then minimized using a block matching scheme. Next, the points are filtered via a least median of squares approach. By fitting to the median, up to 50% of outliers can be removed [43]. Finally, moving least squares is used on the filtered data points to recover the full displacement field. See Table (5.3.2) for a comparison against Radial MILO.

However, because block matching finds displacements using a voxel intensity metric, the minimizer for the metric may not be spatially accurate with respect to the physical transformation. In addition, erroneous points may be introduced by the choice of the reference block size or search window size. Choosing an appropriate reference block is crucial to obtaining meaningful data. For example, choosing a reference block with uniform pixel intensity and assigning a search window with uniform

pixel intensity will result in useless data as there would be several optimum displacements. The choice of the metric is also important for success; if changes in intensity were to occur between images, the metric should account for that. The fact that erroneous estimates can enter block match data calls for the incorporation of filtering techniques.

3.2.2 Mutual Minimizing Block Matching Algorithm

As a first round of filtering, MILO uses a variation of the block matching algorithm, referred to as Mutual Minimizing Block Matching (MMBM). The MMBM algorithm is simply a two-phase FSA algorithm that provides a filter for potential erroneous points. The standard block matching algorithm finds an associated displacement vector for every voxel, while the MMBM algorithm only keeps the solution of the block match if it gets mapped back to the reference image after switching the role of the reference and target images. However, this approach can lead to spatially scattered estimates; moreover there is still no guarantee that the resulting points are spatially accurate, motivating the need for an additional round of filtering. For example, consider Figure 3.3, which demonstrates the displacement estimates provided by the MMBM algorithm. Visually, one can tell that some of the arrows are spatially inaccurate.

3.3 B-splines

As discussed, a common approach to free-form registration is to create a parameterization of the displacement field [11, 16, 22]. A curve generated by B-splines is simply a collection of compactly supported piecewise polynomials that are joined along knots.

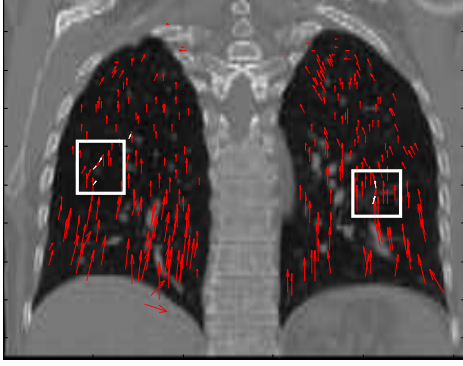


Figure 3.3 : Reference image, showing spatially inaccurate displacements.

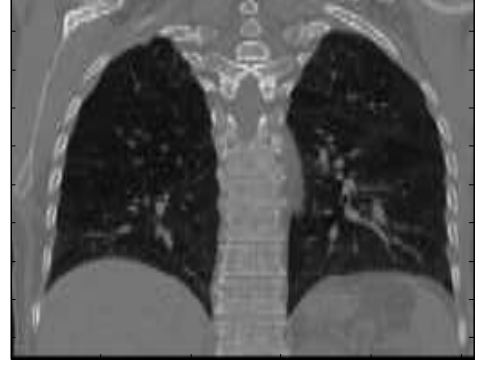


Figure 3.4 : Target image

Definition 2 In one dimension a **knot vector** is a non-decreasing set of coordinates, $k_x = \{\xi_0, \xi_1, \xi_2, \dots, \xi_{n+p^\circ+1}\}$. Where n is the number of basis functions used to construct the B-spline curve, and p° is the polynomial order.

Each piecewise polynomial is referred to as a spline basis function. The beauty of splines is that although they are composed of piecewise polynomials, a linear combination of degree p° basis function will result in a global $C^{p^\circ-1}$ function, thus making it a perfect candidate for inherently smooth transformations [16, 44]. Moreover, the compact support leads to computationally efficient algorithms. B-spline basis functions are defined recursively, starting with a constant B-spline.

Definition 3 **Constant B-spline**

$$S_{j,0}(x) = \begin{cases} 1 & : x \in [\xi_j, \xi_{j+1}) \\ 0 & : x \notin [\xi_j, \xi_{j+1}) \end{cases}$$

B-spline basis functions of order p° are defined by the following recurrence relation:

Definition 4 **B-spline basis function of order p°**

$$S_{j,p^\circ}(x) = \left(\frac{x - \xi_j}{\xi_{j+p^\circ} - \xi_{j^\circ}} \right) S_{j,p^\circ-1}(x) + \left(\frac{\xi_{j+p^\circ+1} - x}{\xi_{j+p^\circ+1} - \xi_{j+1}} \right) S_{j+1,p^\circ-1}(x).$$

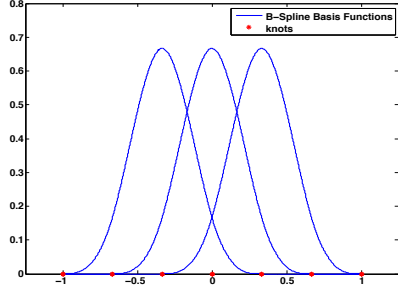


Figure 3.5 : Example of 1D B-spline basis functions

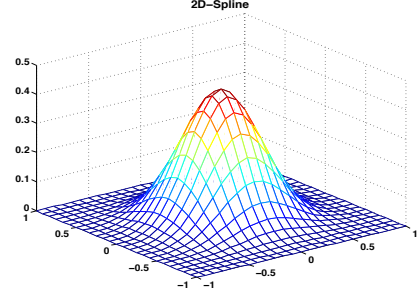


Figure 3.6 : Example of 2D B-spline basis functions

A notable feature of splines is that as the order of the polynomial increases so does the region of compact support. Moreover, given a set of basis functions, a curve can be written as a linear combination of splines, for example:

$$f(x) = \sum_{i=1}^N a_i S_{i,p^\circ}(x). \quad (3.5)$$

For higher dimensions, surfaces can be generated by taking tensor products of one-dimensional splines. As an example, consider the 2D B-spline:

$$f(x, y) = \sum_{j=1}^N \sum_{i=1}^N a_{i,j} S_{i,p^\circ}(x) S_{j,p^\circ}(y). \quad (3.6)$$

3.3.1 B-splines in Image Registration

Two benefits of using splines is that they can describe non-rigid inherently smooth transformations [45, 46], and that they are defined over knots, which can be made nonuniform by altering the spacing of the mesh [46, 47, 48]. Although B-splines provide great flexibility, the associated image registration optimization problem is non-linear and non-convex. The non-convexity implies that the optimization problem can have local minima. Finding a global minimizer then becomes a more difficult problem since gradient based methods, such as Gauss-Newton and BFGS, may get stuck in local minima if a "good" initial guess is not provided [49].

Another issue associated with using gradient based methods is that the objective function is typically required to be twice differentiable [49]. Since an image is a discretely sampled signal, image data must be interpolated and smoothed; however, over-smoothing could potentially destroy fine image structure [50].

Alternatively, if one were to use splines as basis functions for interpolation they would need to find corresponding voxels in the reference and target images. Since many image registration algorithms are based on B-splines, several techniques have been developed for addressing the choice of estimates [14, 16, 51]. The most intuitive way of choosing voxel locations is by manually selecting landmark points in the reference image and target image. From a practical point of view it may be too labor intensive to identify enough landmark points and track them [14], thus automatic feature selection is desirable.

3.3.2 B-splines with MMBM data

In the case of the MILO algorithm, block matching provides a simplistic approach for recovering the displacement vector, \mathbf{d} , of a single voxel, \mathbf{x} , as the solution to

the following problem (written here using a sum of squared difference metric, other metrics may be employed),

$$\min_{\mathbf{d} \in B_{p_0}} \sum_{\mathbf{x}_j \in \Omega_i(\mathbf{x}_i)} [R(\mathbf{x}_j) - T(\mathbf{x}_j + \mathbf{d})]^2. \quad (3.7)$$

The following is the nonlinear least squares objective function associated with spline basis functions, where α is the set of coefficients from Equation (2.1):

$$\min_{\alpha} \sum_{\mathbf{x}_j \in \mathbf{X}} [R(\mathbf{x}_j) - T(\mathbf{x}_j + \phi(\mathbf{x}_j; \alpha))]^2. \quad (3.8)$$

The difference between solving (3.7) versus the nonlinear least squares problem, (3.8), is that the solution of (3.7) will only provide a displacement for a single voxel; solving equation (3.8) will provide a full displacement field. Recalling that the MMBM algorithm does not guarantee that the filtered points are spatially accurate, as demonstrated in Figure (3.3). The original MILO algorithm proposes automatically determining spatially inaccurate points by employing ℓ_1 -minimization [16].

3.4 On ℓ_1 -minimization

Under the assumption that there are few inaccurate estimates relative to the total number of estimates, an ℓ_1 optimization problem can be posed with the goal of detecting the spatially inaccurate estimates in the data set. Recovering sparse solutions has been useful in compressive sensing and machine learning.

3.4.1 Compressive Sensing

In compressive sensing, a signal is said to be K -sparse if it is a linear combination of only K basis vectors [52]. Rather than computing every coefficient and keeping the K largest in magnitude, under certain conditions, compressive sensing offers a

way to reconstruct sparse signals (or signals deemed sparse after some transformation) by solving an ℓ_1 -minimization problem [52, 53, 54]. Compressive sensing uses a measurement matrix, Θ , that acts on a signal, $v \in \mathbb{R}^N$, to recover measurements, y ,

$$\Theta v = y. \quad (3.9)$$

The measurement matrix $\Theta \in \mathbb{R}^{\hat{M} \times N}$ such that \hat{M} denotes the number of measurements and $\hat{M} < N$. Under the assumption that the desired v is the sparsest vector that satisfies Equation (3.9) the so-called zero-norm can be used to pose the following optimization problem:

$$\min_{v \in \mathbb{R}^n} \|v\|_0 \text{ s.t. } \Theta v = y. \quad (3.10)$$

Formulation (3.10) is computationally intractable since the problem breaks down to selecting a minimum number of non-zero values that satisfy the linear constraint. The sparsity promoting ℓ_1 norm is a common relaxation for Equation (3.10). The following formulation is referred to as Basis Pursuit:

$$\min_{v \in \mathbb{R}^n} \|v\|_1 \text{ s.t. } \Theta v = y. \quad (3.11)$$

Compressive sensing shows that under certain situations, solving equation (3.11) is equivalent to solving Equation (3.10) [52, 53, 54]. Unfortunately for the MILO algorithm, there is currently no theory guaranteeing recoverability on spatially inaccurate points.

3.4.2 De-Noising Basis Pursuit/LASSO

In order to accommodate noisy data, equality constrained optimization problems are relaxed by incorporating the constraint into the objective function [52, 53, 54]. For example Equation (3.11) is relaxed to:

$$\min_{v \in \mathbb{R}^n} \|v\|_1 + \frac{1}{\lambda} \|\Theta v - y\|_2^2. \quad (3.12)$$

Outside of compressive sensing, formulation (3.12) is known as the least absolute shrinkage and selector operator (LASSO). The statistical learning community introduced the LASSO formulation to address two weaknesses of the ordinary least squares (OLS) model. The first weakness is OLS estimates often have low bias but large variance. The second issue is interpretation of the data; in situations where there are numerous coefficients it is common to want to find a subset that has the most influence [55].

Unlike the LASSO and Basis Pursuit formulation, the MILO algorithm formulates a linear system based on more estimates than basis functions, allowing the parameterization to "fit" to the data. MILO is able to detect spatially inaccurate estimates by filtering MMBM estimates via ℓ_1 minimization. MILO's ℓ_1 formulation employs an overdetermined linear system, $A\alpha_j = d_j$, where the k^{th} linear equation of A is given by evaluating a voxel, \mathbf{x}_k , at the parameterization and setting it equal to its displacement, d_j^k ,

$$A_{k,:} = \sum_i^N \alpha_i^j B_i(\mathbf{x}_k) = d_j^k. \quad (3.13)$$

The variable α^j denotes the set of coefficients for a spatial component, j. By introducing the variable, p_j , and employing the ℓ_1 norm to formulate the following optimization problem for each spatial dimension, j:

$$\min_{p_j, \alpha_j} \|p_j\|_1 \text{ s.t. } A\alpha_j - d_j - p_j = 0 \quad j \in \{1, 2, 3\}, \quad (3.14)$$

the MILO algorithm can use the parameterization to fit to the majority. Measurements with large magnitude elements of p_j are deemed spatially inaccurate (outliers). As currently written, formulation (3.14) is not in the same form as the LASSO/Denoising Basis Pursuit problem (3.12), since $A \in \mathbb{R}^{M \times N}$ where $M > N$ and noise in the

data must be accounted for. For simplicity the spatial indices j , are dropped keeping in mind that there is an optimization problem for each spatial component. In order to reformulate, a QR decomposition is used:

Proposition 1 Consider the QR decomposition of the matrix $A = QR$, $Q = [Q_1, Q_2] \in \mathbb{R}^{M \times M}$, $R \in \mathbb{R}^{M \times N}$. The following optimization problem,

$$\min_{p, \alpha} \|p\|_1 \text{ s.t. } A\alpha - d - p = 0$$

is equivalent to the following:

$$\min_p \|p\|_1 \text{ s.t. } Q_2^T(d + p) = 0 \quad (3.15)$$

Proof 1 By multiplying the constraint by $Q_2^T \in \mathbb{R}^{(M-N) \times M}$ one can derive an under-determined linear system with one decision variable, p .

After applying the proposition, relaxing the constraint leads to:

$$\min_p \|p\|_1 + \frac{1}{\lambda} \|Q_2^T(d + p)\|_2^2, \quad (3.16)$$

which will be referred to as the ℓ_1 filtering problem, a similar formulation to Equation (3.12). Spatially inaccurate data is identified by the non-zero elements in p for a suitable λ . By removing the outliers, a full displacement field can be computed using the trusted estimates. An issue with reformulating the ℓ_1 filtering problem is that it comes with an associated cost in storage. Given $A \in \mathbb{R}^{M \times N}$, where $M > N$, the associated QR factorization leads to a matrix, $Q \in \mathbb{R}^{M \times M}$, $Q = [Q_1, Q_2]$, such that the matrix of interest is $Q_2 \in \mathbb{R}^{M \times (M-N)}$. To demonstrate how this approach requires more storage consider $A \in \mathbb{R}^{100 \times 2}$; the matrix of interest $Q_2 \in \mathbb{R}^{100 \times 98}$, requires more storage than A , creating a limiting factor on the number of estimates that MILO can use. More details on the choice of λ will be discussed in Chapter 4.

3.5 $2^{\hat{d}}$ Tree Data Structures

Although B-splines work well with the MILO algorithm, solely defining a B-spline mesh over an image can lead to rank deficient matrices in formulation (3.14) if estimates are not within the compact support of splines (Figure 3.7). In order to avoid rank deficient linear systems, MILO extrapolates additional estimates (Figure 3.8).

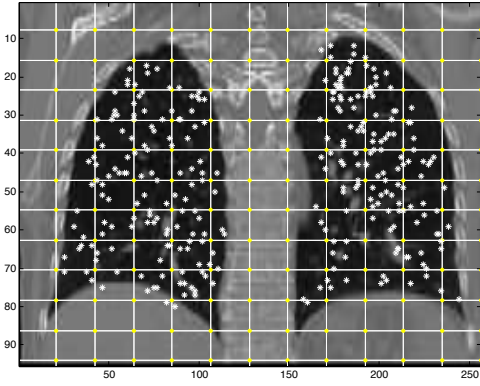


Figure 3.7 : B-Spline Mesh (white and yellow), MMBM estimates (white)

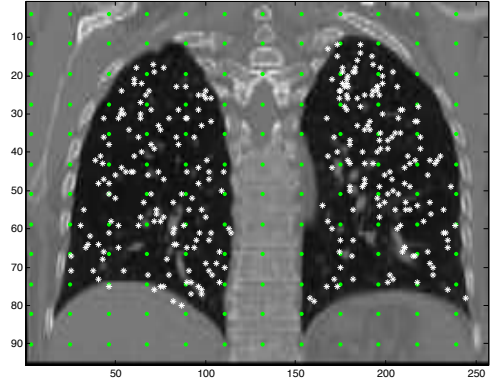


Figure 3.8 : MMBM estimates (white), extrapolated data (green)

As a variation to the original algorithm, Radial MILO moves away from using basis functions that require a mesh and instead employ a mesh-less parameterization. A $2^{\hat{d}}$ tree is used to partition the image according to the spatial distribution of displacement estimates. Radial functions are then defined over the occupied leaves of the $2^{\hat{d}}$ tree. As an example of a $2^{\hat{d}}$ tree partitioning an image, consider Figure (3.9).

Like all trees a $2^{\hat{d}}$ tree data structure has a root node, but specific to the $2^{\hat{d}}$ tree, each node will have either zero or $2^{\hat{d}}$ children for a fixed \hat{d} . For $\hat{d} = 1, 2, 3$, $2^{\hat{d}}$ trees are referred to as a binary tree, quadtree, and octree, respectively. $2^{\hat{d}}$ trees are typically used to spatially partition points in $\mathbb{R}^{\hat{d}}$. The purpose of each node is to represent a bounding box in $\mathbb{R}^{\hat{d}}$. At the root node, the bounding box encompasses all the points

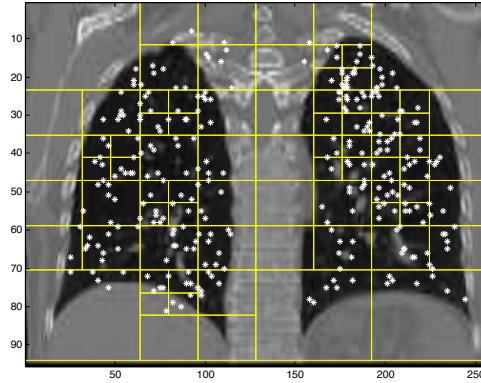


Figure 3.9 : A $2^{\hat{d}}$ conforming to MMBM estimates

of the data set. In order to create a $2^{\hat{d}}$ tree it is necessary to define a capacity value for the maximum number of points allowed in each bounding box. If the number of points in a given node is greater than the capacity value, the node splits into $2^{\hat{d}}$ children and redistributes the points among the children [56, 57, 58]. Figure 3.10 provides an illustrative example of a quadtree partitioning data.

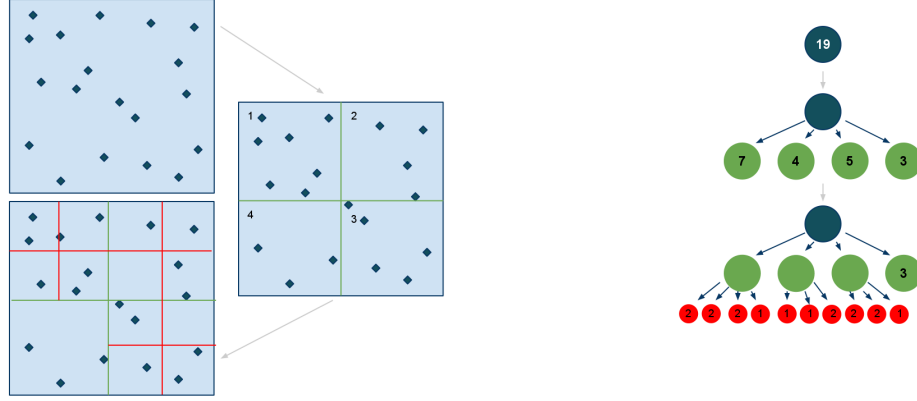


Figure 3.10 : An example of a quadtree with a box capacity of 3. Left image is the spatial partitioning. Right image is the tree data structure. Image credit: [57] .

As an example of an application of a $2^{\hat{d}}$ tree, consider their use in approximating images. Image estimates can be an attractive alternative to using high resolution images. For example, applying a filter on a $512 \times 512 \times 512$ image results in more than 10^8 voxels having to be evaluated. By employing a $2^{\hat{d}}$ tree one can reduce data processing by creating an estimated image based on voxel intensity. The underlying idea is to partition the image based on the varying voxel intensity. Regions that have close to zero variability could be represented by a single large box. Regions with large variability would require further partitioning [59, 60].

Haber, et al.[60] provide an example of a registration algorithm that uses a $2^{\hat{d}}$ tree structure to reduce the dimensions of an image. The registration is then carried out using free-form registration methods. $2^{\hat{d}}$ trees have also been used for adaptive discretization of partial differential equations and adaptive based image registration algorithms [61, 62]. Radial MILO uses $2^{\hat{d}}$ tree data structures to determine the centers

of radial functions instead of using them to create an estimation of the image.

3.6 Radial Basis Functions

As an improvement on MILO, Radial MILO uses radial functions instead of B-splines.

Formally, a radial function can be defined as [63]:

Definition 5 A function $\rho_c : \mathbb{R}^{\hat{d}} \rightarrow \mathbb{R}$ is said to be **radial** if there exists a function, $\hat{q} : [0, \infty) \rightarrow \mathbb{R}$, such that $\rho_c(x) = \hat{q}(\|x - c\|)$ for all $x \in \mathbb{R}^{\hat{d}}$.

Examples of globally supported radial basis functions include: thin-plate splines, Guassians, and multi-quadratics. Radial functions have the advantage of being simple and easy to implement for interpolation in $\mathbb{R}^{\hat{d}}$ when compared to B-splines. Unlike B-splines, radial functions do not require taking tensor products in order to interpolate to higher dimensions. Furthermore radial functions offer more flexibility when dealing with unstructured data since they can be centered in an unstructured manner.

3.6.1 Local Approximations: Moving Least Squares

In some cases there may be interest in approximating a function at a single point when only neighboring data is available. Moving least squares (MLS) is a type of approximation that is based on employing neighboring data to create this approximation. The idea behind MLS approximation is to solve a weighted least squares problem with neighboring data assigning weights based on distance to the approximation of interest. MLS has been used in image registration and in surface reconstruction [16, 64, 65]. The strength of MLS is in its ability to handle unstructured noisy data. Unlike interpolation, MLS is essentially a "fit" onto the given data. Formally, MLS can be defined as, [58]:

Definition 6 Moving Least Squares Approximate: Given certain data sites: $\mathbf{X} = \{\hat{x}_1, \dots, \hat{x}_N\} \subseteq \Omega \subseteq \mathbb{R}^{\hat{d}}$ and their function values $\mathbf{Y} = \{\hat{y}_1, \dots, \hat{y}_N\}$. For $x \in \Omega$, the value, $s_{y,\mathbf{X}}(x)$, of the moving least squares approximate is given by $s_{y,\mathbf{X}}(x) = g^*(x)$ where g^* is the solution of:

$$\min_g \left\{ \sum_{i=1}^N [y_i - g(x_i)]^2 w(x, x_i) : g \in \mathbf{P}_m(\mathbb{R}^{\hat{d}}) \right\}.$$

The function, $w(x, x_i)$, is typically a radial function, such as a Gaussian, [16, 64, 65].

In the context of the MILO and Radial MILO algorithm, MLS is employed to compute a full displacement field once spatially inaccurate data has been removed.

3.6.2 Image Registration Based on Radial Functions

Radial functions can also play pivotal role in image registration [11, 18, 23]. For example, consider Shushanria et. al.'s [23] free-form registration algorithm. The algorithm employs manually selected landmarks and uses a radial function based parameterization to recover a full displacement field. Landmarks are grouped into clusters using the k-means algorithm. The support radius is chosen based on the distances between the landmarks and chosen in such a way that neighboring functions overlap. The main drawback of this approach is its need of manual intervention for determining landmarks.

3.7 Conforming Parameterization

The novelty of the Radial MILO algorithm is the use of a $2^{\hat{d}}$ tree to conform to an unstructured distribution of displacement estimates. In order to create a spatial parameterization of the voxel trajectories, radial functions are used as basis functions. The centers of the radial functions are taken to be the centers of the occupied boxes

provided by the $2^{\hat{d}}$ tree. Since the choice of functions and parameters will influence the transformation accuracy, [23, 64, 66], the choice of radial functions for Radial MILO was accomplished by surveying various candidate functions as discussed in the following chapter.

Chapter 4

Methods and Implementation

As in the MILO algorithm, Radial MILO is composed of three steps. The first step is to apply the MMBM algorithm to a set of reference voxels to estimate a displacement for each target image. The second step is to filter out potential outliers in the MMBM data. The last step is to recover the displacement field using MLS. This chapter provides an explanation of each step and implementation details, such as: using image segmentation to define a region of interest within the reference image, using a GPGPU to combat the computational burden of the MMBM algorithm, an object oriented approach to construct a $2^{\hat{d}}$ tree, employing block coordinate descent to potentially filter out spatially inaccurate data, and recovering the full displacement field using MLS. Figure (4.1) provides a walkthrough of the different components in employing the Radial MILO algorithm, and Figure (4.2) illustrates a set of voxels used in the MMBM algorithm.

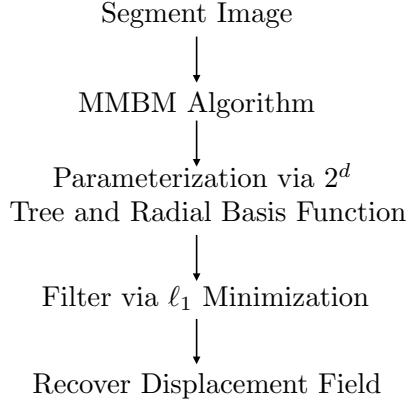


Figure 4.1 : Image Registration employing Radial MILO

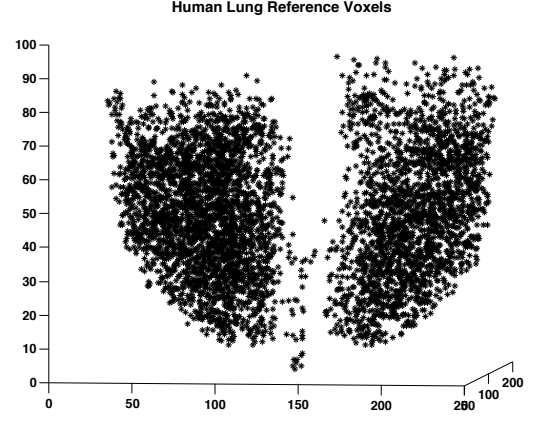


Figure 4.2 : After segmentation, reference voxels are sampled from the reference image

4.1 Segmenting a Thoracic CT Image

As with image registration, image segmentation is an active area of research. Medical image segmentation aims to extract object boundary features, allowing for analysis of biological organs [67, 68, 69]. The need to segment large amounts of medical images makes automatic image segmentation desirable [70]. In this thesis, Radial MILO is used to register ten sets of 4DCT lung images. Image segmentation is used to extract the lungs, allowing for uniform sampling of reference voxels.

In order to segment the benchmark images, a thresholding approach is used. Threshold image segmentation partitions an image based on voxel intensity. The process is manually conducted via MATLAB's image processing toolbox. First a mask volume is generated by identifying voxels between intensity values (1, 800). This process is dependent on the image and requires tuning of parameters to retrieve satisfactory results. Empty regions in the lungs are then filled by MATLAB's **fill** command. Any lingering border information is removed via MATLAB's **imclearborder**

command. The function **imclearborder** suppresses structures that are lighter than their surroundings and are connected to the image border. Figures 4.3 - 4.6 provide an example of applying the MATLAB commands to a CT image. Further details about MATLAB's image processing toolbox can be found in MATLAB's documentation.

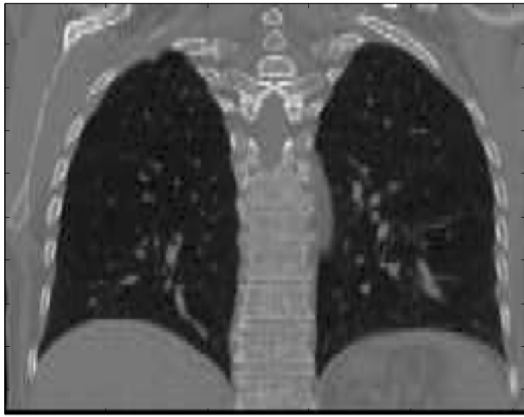


Figure 4.3 : Unsegmented image



Figure 4.4 : Threshold mask

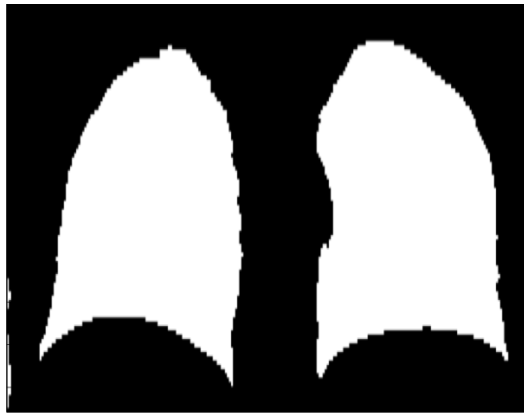


Figure 4.5 : Image after fill command

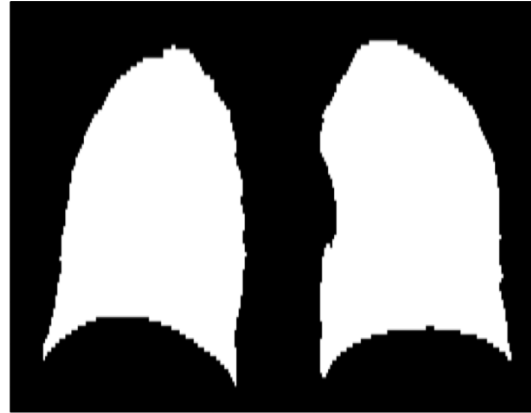


Figure 4.6 : Image after clearborder command

4.2 GPGPU Computing Model

As demonstrated in [35], the employment of the GPGPU, in place of a CPU, can substantially reduce compute time. Application programming interfaces (API), OpenCL, and CUDA can all be used to program GPGPU's; both OpenCL and CUDA employ the same memory hierarchy model. Other programming models such as OpenACC employ compiler directives [28] and OCCA, a portable programming language, relies on OpenCL and CUDA to execute kernel code [71].

OpenCL allows the user to program a variety of devices but comes at the extra cost of having the user create a command queue, and establish a relationship between the *host* (the hardware that will call the kernel) and the *device* (the hardware that will execute the kernel). A wrapper can easily facilitate writing multiple projects in OpenCL; further discussion on OpenCL can be found in [28]. Nvidia's CUDA hides the host-device relationship and provides an easier environment in which to program GPGPU's, at the cost of limiting programming to NVidia's devices. Due to its portability, OpenCL is employed as the GPGPU API for this work.

Understanding the memory hierarchy and the threading model is fundamental for developing high performance GPGPU software. Computation on the GPGPU is performed on a predefined N_0 , $N_0 \times M_0$, or $N_0 \times M_0 \times P_0$ dimensional grid of computing units, where $M_0, N_0, P_0 \in \mathbb{N}$. Having options in grid dimensions provides different ways to index the compute units. Each unit of the grid is referred to as a *work-item* (*threads* in CUDA). Each work-item is provided with a small amount of private memory. Work-items are grouped together to form *work-groups* (*blocks* in CUDA), and each work-group is paired with additional memory, defined as *local memory* (*shared memory* in CUDA). Local memory can be accessed by any of the work-group's work-items.

Two branches of memory that are accessible to any type of computational unit are *constant memory* and *global memory*. Constant memory is designed for data whose values stay constant throughout the kernel, and global memory is designed to be visible to all compute units on the device. Data stored as a general buffer and transferred between the host and device must first reside in global or constant memory. A third type of memory is *image memory* (*texture memory* in CUDA), in which accessing image data is done via an API call that enables hardware optimizations when accessing spatially local data. Texture memory is a part of global memory. Figure (4.7) illustrates the GPGPU memory model. Further discussion on the GPGPU's memory hierarchy can be found in [28, 72].

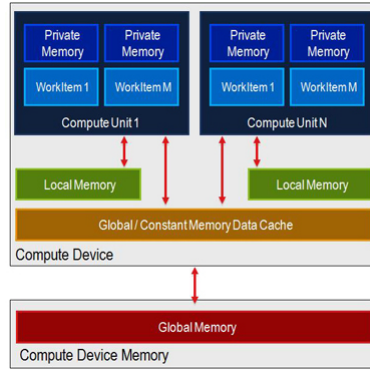


Figure 4.7 : OpenCL memory hierarchy [28]

4.3 Block Matching on the GPGPU

For each reference block, the block matching algorithm takes a candidate target block from a search window and evaluates the similarity according to a provided metric. Given a reference $n \times m \times z$ block with a $(m + 2 \times p_0) \times (n + 2 \times p_0) \times (z + 2 \times p_0)$

search window, $(2 \times p_0 + 1)^3$ metrics must be computed, where p_0 denotes the maximum displacement in a spatial component. Thus for M voxels, the block matching algorithm requires $M \times (2 \times p_0 + 1)^3$ metric evaluations, whereas employing the MMBM algorithm requires twice as many. The fact that the block match algorithm requires the same set of instructions for different data makes it a perfect candidate for implementation on the GPGPU. Following the implementations in [38, 42, 73], the fine grain parallelization occurs when computing the block match metric. The general outline of the block matching procedure is as follows:

Algorithm 1 Block Matching on the GPGPU

1: **procedure** BLOCK MATCHING

Set: Reference Block Size, Search Window Radius, Work-group Size, Number of Groups

1. Assign a reference block to each work-group
 2. Load reference block to shared memory
 3. Each thread in work-group computes a metric for a potential displacement.
 4. Each work-group applies a reduction to recover the global minimum
-

At the end of step three of the block match algorithm, the metric values for each candidate target block are distributed throughout the memory of various work-items. In order to recover the optimal displacement, a reduction step is necessary. As an example of a reduction operator, consider the task of adding values in different work-items. In order to compute the sum, work-items must be able to transfer their values to other threads. This task is accomplished by making use of local memory. Threads store their values in local memory and iteratively employ half of the work-items to add values from local memory. In the case of block matching, rather than adding

values, work-items simply compare values and seek the optimal displacement for the reference block. Figure (4.8) illustrates the reduction operator.

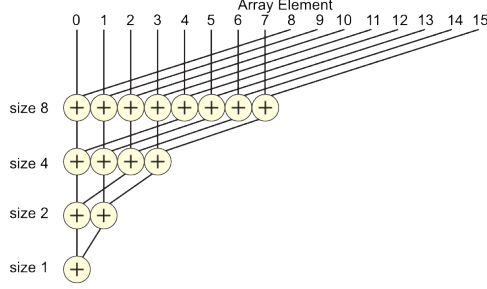


Figure 4.8 : Example of reduction operator: Adding values in distributed work-items (threads) [74].

The block matching algorithms contains the following parameters: the size of the **reference block**, **search window radius**, and the choice of **metric**. Each parameter will contribute to the computational workload of the algorithm; however, the size of the search window will play the greatest role in computational expense. As such, the task of choosing a reference block and search window is not trivial. In the benchmark images (Table 5.1) displacements can range from an average of 6 mm to 15.16 mm. Having a priori information provides bounds to which we can set our window. During a blind registration, in which one has no prior knowledge of the magnitudes of displacements, selecting an appropriately sized search window is more challenging.

Choosing the appropriate block matching metric is dependent on image content [75]. Adopting the same MMBM algorithm that MILO uses, Radial MILO employs the Zero Mean Normalized Cross-Correlation (ZMNCC) metric [16]. This allows for the consideration of substantial changes in voxel intensity between images pairs [31]. The term in the denominator of the metric, \hat{F} , normalizes such that $\hat{F} \in [-1, 1]$,

$$\hat{F}(R, T) = \frac{\sum_{z=1}^n \sum_{v=1}^m \sum_{u=1}^z (R(\hat{u}, \hat{v}, \hat{z}) - \bar{R}) (T(\hat{u}, \hat{v}, \hat{z}) - \bar{T})}{\left(\sum_{\hat{z}=1}^n \sum_{\hat{v}=1}^m \sum_{\hat{u}=1}^z (R(\hat{u}, \hat{v}, \hat{z}) - \bar{R})^2 \sum_{\hat{v}=1}^m \sum_{\hat{z}=1}^n \sum_{\hat{u}=1}^z (T(\hat{u}, \hat{v}, \hat{z}) - \bar{T})^2 \right)^{\frac{1}{2}}}$$

Employing the ZMNCC metric is used to pose the following optimization problem for each voxel displacement:

$$\arg \min_{d \in B_p} 1 - |\hat{F}(R(\mathbf{x}_i), T(\mathbf{x}_i + \mathbf{d}))|. \quad (4.1)$$

A perfect direct relation between the reference and target block, $\hat{F} = 1$, is expected when the reference block and the candidate target block are positively correlated. A perfect decreasing relation, $\hat{F} = -1$, is expected when the reference block and candidate target block are negatively correlated. By posing such an optimization problem, Equation (4.1), the block matching algorithm seeks to find the target reference block with the highest absolute correlation [76]. The choice of block matching parameters are discussed in the results chapter.

4.4 The $2^{\hat{d}}$ data structure class

Radial MILO employs the $2^{\hat{d}}$ tree data structure in order to adaptively conform to the MMBM estimates. Employing an object oriented programming paradigm, the $2^{\hat{d}}$ tree data structure can easily be encapsulated as an object with a root node as a member variable. The root node corresponds to the parent of all future nodes. A node can be modeled as a **struct** and can encapsulate all necessary information about the box it represents. Features such as searching and determining centers of leaf nodes can be implemented as methods. As an example of the contents of a node see Figure (4.9).

Algorithm 2 demonstrates how to build a $2^{\hat{d}}$ tree data structure starting from a root node with arbitrary point capacity.

```

1 typedef struct node_t{
3     float  xmin ,xmax ,ymin ,ymax ,zmin ,zmax ;
        list<voxel> voxels;  //list of voxels in the node
5     list<voxelNode*> children; //list of pointers to the children
7 }node_s;

```

Figure 4.9 : Example of octree node and member variables. A C++ list is used to encapsulate the voxel displacements and children nodes

Algorithm 2 Build $2^{\hat{d}}$ tree

1: **procedure** BUILD $2^{\hat{d}}$ TREE

1. Insert voxel to root node
 2. If root has no children nodes and list of voxels is not filled to capacity,
insert voxel into list of voxels and return
 3. If node contains children find the child node that encapsulates the point,
if list of voxels is not full, insert voxels and return
 4. If child node is full, split node and redistribute voxels
into the children nodes
 5. Go to step 3
-

4.5 Radial Functions

Radial MILO uses radial functions instead of splines to create a parameterization. This meshless approach removes the constraint of needing to have data distributed over a uniform mesh. The combination of radial functions and a $2^{\hat{d}}$ tree allow for a parameterization that conforms to displacement estimates provided by the MMBM algorithm. Recalling that radial functions are a class of functions, there are three main properties that should be considered when selecting a radial function [23]:

- **Locality:** How each basis function influences the whole transformation.
- **Solvability:** As with the associated B-spline matrix, the matrix associated with the radial functions must also be full rank to ensure unique solutions.
- **Computational Efficiency:** Forming an associated linear system for volumetric images becomes computationally demanding for large numbers of voxels. Radial functions with compact support provide an advantage in the sense that only points within the radius of compact support need to be evaluated. Functions with global support lead to fully dense matrices.

4.5.1 Overview of Globally Supported Radial Functions

A variety of radial functions with global support have been proposed for image registration (Table 4.1) [23].

Global support leads to each data point influencing parameterizations based on these radial functions. Overall each of these functions require evaluations of transcendental functions (the logarithm, the square root function at $\mu = 0.5$ for the multiquadratics, and the exponential function), and due to their global support, parameterizations based on these functions lead to dense linear algebra.

Examples of Radial Functions	
Thin plate splines	$R_{TPS}(r) = \begin{cases} r^{4-d} \ln r & : 4 - d \in 2\mathbb{N} \\ r^{4-d} & : \text{else} \end{cases}$
Multiquadratics	$R_M(r) = (r^2 + c^2)^\mu, \mu \in \mathbb{R}^+$
Inverse Multiquadratics	$R_{IM}(r) = (r^2 + c^2)^{-\mu}, \mu \in \mathbb{R}^+$
Gaussians	$R_G(r) = e^{-\frac{r^2}{2\sigma^2}}, \sigma \in \mathbb{R}$

Table 4.1 : Examples of radial functions with global support; r denotes the euclidean distance from the center of the radial function to the point being evaluated [23].

4.5.2 A Class of Compactly Supported Functions: Wendland's Functions

Although there are a variety of compactly supported radial functions, a popular class of functions for elastic image registration are the ω -functions of Wendland [23, 63, 77],

$$\omega_{\hat{d}, \hat{k}}(r) = I^{\hat{k}} (1 - r)_+^{\ell}. \quad (4.2)$$

The Wendland functions are defined by the dimension of the approximating space, \hat{d} , and the desired order, \hat{k} . The term $(1 - r)_+$ is the truncated function defined as:

$$\varphi_{\ell}(r) = (1 - r)_+^{\ell} = \begin{cases} (1 - r)^{\ell} & : r \in [0, 1] \\ 0 & : \text{else.} \end{cases} \quad (4.3)$$

By providing a radius of compact support of one; varying support can be created by normalizing the input data. ℓ is defined as $\ell = \lfloor \frac{\hat{d}}{2} \rfloor + \hat{k} + 1$, and higher order polynomials are constructed by applying the integral operator,

$$I\varphi = \int_r^{\infty} t\varphi(t)dt. \quad (4.4)$$

The following are examples of Wendland functions for $\hat{d} = 3$ and $\hat{k} = 0, 1, 2$:

$$\begin{aligned}\omega_{3,0}(r) &= (1-r)_+^2 \\ \omega_{3,1}(r) &= (1-r)_+^4 (4r+1) \\ \omega_{3,2}(r) &= (1-r)_+^6 (35r^2 + 18r + 3) .\end{aligned}$$

4.5.3 Discussion

Using radial functions eliminates the need to have data distributed throughout a uniform mesh. Parameterizations based on radial functions with global support employ each data point at the cost of dense linear algebra and much more computation when compared to compactly supported functions. Aside from having to choose a support radius, the Wendland functions are among the most appealing since their region of compact support allows for a paring with a $2^{\hat{d}}$ data structure and swift evaluations. Radial MILO uses Wendland functions because of their compact support and computational efficiency.

4.6 Filtering MMBM Data via ℓ_1 -minimization

Recalling the initial ℓ_1 problem,

$$\min_{p, \alpha} \|p\|_1 \text{ s.t. } A\alpha - d - p = 0.$$

The MILO algorithm addressed solving the ℓ_1 filtering problem by transforming the problem into the LASSO/Basis Pursuit De-Noising model (3.12) and employing existing numerical software, Your ALgorithms for L1 (YALL1) [78]. YALL1 is a MATLAB package designed for solving a variety of ℓ_1 minimization problems subject to an underdetermined linear system.

As an improvement to the traditional approach, Radial MILO uses an alternative optimization scheme, Block Coordinate Descent, that takes advantage of the ℓ_1 filtering formulation. Unlike the MILO algorithm, Radial MILO employs a parameterization modeling the trajectory of voxels as quadratic functions in time. For each voxel, \mathbf{x}_k , at time, t_z , a linear equation can be made using the displacement, $d_j^k(t_z)$, for each spatial dimension j , at time t_z ,

$$A_{k,:} = \left(\sum_i^N \alpha_i^j B_i(\mathbf{x}_k) \right) t_z^2 + \left(\sum_i^N \beta_i^j B_i(\mathbf{x}_k) \right) t_z = d_j^k(t_z), \quad (4.5)$$

define the associated linear system, $A\gamma_j = d_j^k(t_z)$.

4.6.1 Block Coordinate Descent

Block coordinate descent solves a relaxed version of problem (3.14) (written here with the Radial MILO parameterization, spatial subscripts dropped for compactness) without the need of large matrix factorization,

$$\min_{p,\gamma} \|p\|_1 + \frac{1}{\lambda} \|A\gamma - d - p\|_2. \quad (4.6)$$

Block coordinate descent is a popular method for minimizing a real-valued function of several variables, and a discussion on theoretical properties can be found in [79]. Variables are partitioned into blocks and at each iteration the function is minimized with respect to one of the blocks. In the case of the ℓ_1 filtering problem (4.6) the blocks are designated as p and γ . Other blocks may be chosen, but this is the most natural way to partition the unknown variables. Minimizing with respect to γ is simply done via a least squares solve,

$$\gamma \leftarrow (A^T A)^{-1} A^T (d + p). \quad (4.7)$$

Since $\|p\|_1$ is not differentiable at zero, a minimizer can be computed via subdifferential calculus [80]. Explicitly, the solution is given by $p_i^+ = S_\lambda((A\gamma - d)_i)$. Where S_λ is defined as:

$$S_\lambda(a) = \begin{cases} a - \lambda & : a > \lambda \\ 0 & : |a| \leq \lambda \\ a + \lambda & : a < -\lambda. \end{cases} \quad (4.8)$$

Alternating between iterates results in our algorithm:

Algorithm 3 Block Coordinate Descent

1: **procedure** BLOCK COORDINATE DESCENT

while Not Covered

$\gamma \leftarrow (A^T A)^{-1} A^T (d + p)$

$p_i \leftarrow S_\lambda((A\gamma - d)_i)$

4.6.2 Filtering Algorithm via ℓ_1 -minimization

By employing the parameterization, Equation (4.5), and the ℓ_1 filtering model (4.6), spatially inaccurate data is identified by the by the non-zero elements in p for a suitable λ . A weakness of employing the ℓ_1 filtering model is that it is difficult to determine what the weight parameter, λ , should be to detect the target number of spatially inaccurate data. Another issue is being determining the amount of outliers in block match estimates. Determining a target number of spatially inaccurate data remains a heuristic. Numerical experiments demonstrate that for the benchmark images assuming 20% spatially inaccurate data provided the best results. By employing block coordinate descent (Algorithm 3), I present an iterative approach (Algorithm 4) to finding the target sparsity.

Algorithm 4 Filtering via ℓ_1 Minimization

1: **procedure** FILTERING VIA ℓ_1

1. Compute Cholesky decomposition: $R^T R = A^T A$
2. Set desired sparsity threshold
3. Set $\gamma, p \leftarrow \hat{0}$
4. For $\lambda = \lambda_{max} : d\lambda : \lambda_{min}$

- 4.1. **while** Not Converged:

- 4.1.1 $x \leftarrow R^{-1} R^{-T} A^T (d + p)$

- 4.1.2 $p_i \leftarrow S_\lambda ((A\gamma - d)_i)$

- 4.2 If p has desired sparsity, then return

4.6.3 Moving Least Squares

Once outliers have been identified and removed, the full displacement field can then be computed by either interpolating the estimates or by local approximations. As demonstrated in [16], the parameterization only acts as a "fit." Thus by recovering a full displacement from the filtered data higher spatial accuracy can be achieved. Like the original MILO algorithm, Radial MILO employs moving MLS to interpolate the displacement field. For each spatial component, the displacement is computed from the reference image to the target image by employing moving least squares to recover the coefficients for an affine function:

$$f(\mathbf{x}) = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} \quad (4.9)$$

In order to determine the full trajectory of a single voxel a similar approach can be made by recovering the coefficients of the following quadratic,

$$\mathbf{x}_j(t; x_0) = a_t^{(j)} t^2 + b_t^{(j)} t + x_0^{(j)}. \quad (4.10)$$

Formulation 4.10 describes the trajectory of a single voxel, \mathbf{x} , at its starting location, x_0 , in the spatial component, j . Radial MILO's MLS employs the nearest 15 estimates to compute local displacements, numerical results have demonstrated that this yielded the strongest results. A truncated Gaussian is used as a weight function, and the radius of compact support is determined by the furthest point used in the estimation.

4.7 Radial MILO Algorithm

Building on the previous sections, the full Radial MILO algorithm is provided below:

Algorithm 5 Radial MILO

1: **procedure** RADIAL MILO

1. For each target image compute MMBM estimates
 2. Insert MMBM estimates into an $2^{\hat{d}}$ tree, the centers of the non-empty leaves will serve as the centers of the radial functions
 3. Construct a parameterization by employing radial functions
 4. Solve the ℓ_1 filtering problem (4.6) for a desired sparsity via algorithm 4
 5. Recover the full displacement field using Moving Least Squares
-

Chapter 5

Numerical Experiments

Spatial accuracy was assessed using ten sets of 4DCT images from the DIR-LAB dataset. The images were acquired as part of the radiotherapy treatment of thoracic malignancies at the University of Texas M.D. Anderson Cancer Center in Houston, TX. The images are publicly available online from the DIR-Lab website at www.dir-lab.com [32]. The DIR-Lab provides 75 landmarks propagated through five images capturing the expiratory phase, and 300 landmarks propagated between "extreme phases," maximum inhalation and maximum exhalation. Radial MILO was used to register across the five expiratory phase images. Spatial accuracy was determined by comparing the MILO solution to the set of landmarks.

The numerical experiments in this thesis were conducted using a Macbook Pro with a 2.5GHz Intel Core i7 processor, 16 GB of memory, and a Nvidia GeForce GT 750M GPGPU. The compute grid was set to have 512 work-groups, and each work-group was assigned 256 work-items. These parameters were chosen in order to take advantage of the employed GPGPU architecture [27, 28].

5.1 Image Description

Table 5.1 corresponds to the 4DCT image sets provided by the DIR-Lab. Image dimensions are measured in number of voxels, and voxels are measured in millimeters. The "Num Features" column reports the quantity of landmarks propagated through

maximum inhalation and exhalation. The mean (and standard deviation) of image landmarks are shown in the "Displacement" column. The "Repeats" column is shown as the Nm/Nobs, where Nobs is the number of observers and Nm is the number of landmark measurements the observer reported. The "Observer Error" column shows the average observer error (standard derivation) with the respect to the primary observer.

Label	Dimension	Voxels (mm)	Num of Features	Displacement (mm)	Repeats	Observer Error (mm)
4DCT1	$256 \times 256 \times 94$	$0.97 \times 0.97 \times 2.5$	1280	4.01 (2.91)	200/3	0.85 (1.24)
4DCT2	$256 \times 256 \times 112$	$1.16 \times 1.16 \times 2.5$	1487	4.65 (4.09)	200/3	0.70 (0.99)
4DCT3	$256 \times 256 \times 104$	$1.15 \times 1.15 \times 2.5$	1561	6.73 (4.21)	200/3	0.77 (1.01)
4DCT4	$256 \times 256 \times 99$	$1.13 \times 1.13 \times 2.5$	1166	9.42 (4.81)	200/3	1.13 (1.27)
4DCT5	$256 \times 256 \times 106$	$1.10 \times 1.10 \times 2.5$	1268	7.10 (5.14)	200/3	0.92 (1.16)
4DCT6	$512 \times 512 \times 128$	$0.97 \times 0.97 \times 2.5$	419	11.10 (6.98)	150/3	0.97 (1.38)
4DCT7	$512 \times 512 \times 136$	$0.97 \times 0.97 \times 2.5$	398	11.59 (7.87)	150/3	0.81 (1.32)
4DCT8	$512 \times 512 \times 128$	$0.97 \times 0.97 \times 2.5$	476	15.16 (9.11)	150/3	1.03 (2.19)
4DCT9	$512 \times 512 \times 128$	$0.97 \times 0.97 \times 2.5$	342	7.82 (3.99)	150/3	0.75 (1.09)
4DCT10	$512 \times 512 \times 120$	$0.97 \times 0.97 \times 2.5$	435	7.63 (6.54)	150/3	0.86 (1.45)

Table 5.1 : 4DCT image properties

5.2 Block Matching Parameters

In order to gain insight on the overall effect of choosing an **initial number of reference voxels**, **search window radius**, and **number of basis functions**, I ran numerical experiments across all image sets, and provide the results for image sets 1 and 6. These numerical experiments demonstrated that initiating the MMBM algorithm with 65,024 voxels provided strong results, and initiating with more voxels did not significantly improve the results. Similar experiments were carried out for

the rest the images and it was demonstrated that reference block sizes of $7 \times 7 \times 3$ and $11 \times 11 \times 3$ provided strong results for image sets 1-5 and image sets 6-10, respectively. There is a small variance between the number of basis functions used in the registration since the number of basis functions is determined by the octree's ability to spatially partition data.

Radial functions were assigned a compact support radius of 50 and 75 voxels, and search window radii were set to 10 and 15 voxels, for image sets 1 and 6, respectively. The ℓ_1 filtering algorithm was set to assume 20% of MMBM data was spatially inaccurate. The parameters held fixed were chosen based on additional numerical experiments.

Reference Block	M	M*	Num. Basis Fun	Avg. (std) Error
$5 \times 5 \times 3$	33280	13241	205	0.85 (0.97)
$5 \times 5 \times 3$	33280	13241	500	0.82 (0.99)
$5 \times 5 \times 3$	33280	13241	1001	0.83 (1.02)
$5 \times 5 \times 3$	65024	26124	200	0.81 (0.96)
$5 \times 5 \times 3$	65024	26124	501	0.76 (0.95)
$5 \times 5 \times 3$	65024	26124	1002	0.81 (0.92)
$7 \times 7 \times 3$	32512	17321	201	0.82 (1.05)
$7 \times 7 \times 3$	32512	17321	505	0.76 (0.93)
$7 \times 7 \times 3$	32512	17321	1001	0.82 (1.00)
$7 \times 7 \times 3$	65024	35213	203	0.76 (0.80)
$7 \times 7 \times 3$	65024	35213	503	0.75 (0.80)
$7 \times 7 \times 3$	65024	35213	1002	0.81 (0.88)
$7 \times 7 \times 3$	153600	54310	501	0.75 (0.93)
$9 \times 9 \times 5$	32512	21932	200	0.78 (0.99)
$9 \times 9 \times 5$	32512	21932	503	0.75 (0.97)
$9 \times 9 \times 5$	32512	21932	1000	0.80 (0.97)
$9 \times 9 \times 5$	65024	43925	201	0.72 (0.97)
$9 \times 9 \times 5$	65024	43925	501	0.78 (0.97)
$9 \times 9 \times 5$	65024	43925	1003	0.72 (0.89)
$9 \times 9 \times 5$	97536	65431	202	0.76 (0.99)
$9 \times 9 \times 5$	97536	65431	501	0.77 (0.95)
$9 \times 9 \times 5$	97536	65431	1002	0.73 (0.90)

Table 5.2 : Comparison of varying reference block sizes, and number of basis functions, for image set 1. M is the initial number of voxels, and M* denotes estimates returned by the MMBM algorithm, Num. Basis Fun are the number of basis functions used.

Reference Block	M	M*	Num. Basis Fun	Avg. (std) Error
$5 \times 5 \times 3$	32512	5351	200	2.8 (2.20)
$5 \times 5 \times 3$	32512	5351	500	3.03 (2.67)
$5 \times 5 \times 3$	32512	5351	1002	3.25 (2.92)
$5 \times 5 \times 3$	65024	11421	202	3.07 (2.85)
$5 \times 5 \times 3$	65024	11421	504	2.84 (2.45)
$5 \times 5 \times 3$	65024	11421	1003	2.7 (2.53)
$9 \times 9 \times 5$	32512	11321	204	1.21 (1.36)
$9 \times 9 \times 5$	32512	11321	504	1.13 (1.15)
$9 \times 9 \times 5$	32512	11321	1003	1.15 (1.34)
$9 \times 9 \times 5$	65024	21951	203	1.18 (1.49)
$9 \times 9 \times 5$	65024	21951	503	1.06 (1.20)
$9 \times 9 \times 5$	65024	21951	1002	1.05 (1.34)
$9 \times 9 \times 5$	97536	33003	202	1.17 (1.43)
$9 \times 9 \times 5$	97536	33003	503	1.07 (1.14)
$9 \times 9 \times 5$	97536	33003	1002	1.06 (1.13)
$11 \times 11 \times 3$	32512	12314	202	1.07 (1.21)
$11 \times 11 \times 3$	32512	12314	500	1.05 (1.17)
$11 \times 11 \times 3$	32512	12314	1001	1.08 (1.18)
$11 \times 11 \times 3$	65024	24205	200	1.03 (0.95)
$11 \times 11 \times 3$	65024	24205	502	1.00 (1.04)
$11 \times 11 \times 3$	65024	24205	1000	1.01 (0.95)
$11 \times 11 \times 3$	97536	36254	201	1.03 (1.12)
$11 \times 11 \times 3$	97536	36254	500	1.01 (1.03)
$11 \times 11 \times 3$	97536	36254	1003	1.05 (1.01)

Table 5.3 : Comparison of varying reference block sizes, and number of basis functions, for image set 6. M is the initial number of voxels, and M* denotes estimates returned by the MMBM algorithm, Num. Basis Fun are the number of basis functions used.

To gain insight on compute time of the MMBM algorithm, two plots measuring compute time versus varying search window radius are presented. The first set of experiments employs a reference block size of $7 \times 7 \times 3$ voxels and the second set employs a reference block of $11 \times 11 \times 3$ voxels. Both experiments are set with 79,360 initial reference voxels.

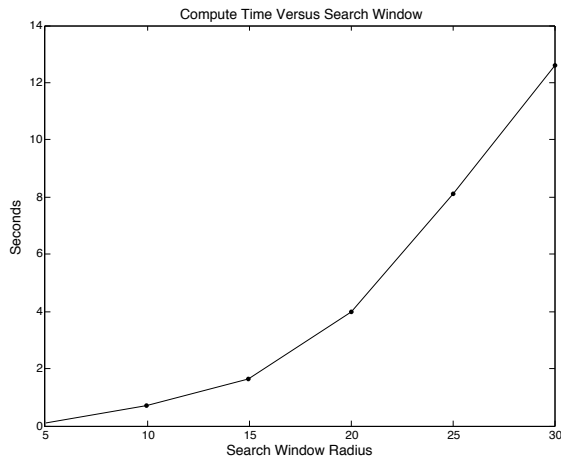


Figure 5.1 : MMBM algorithm timing results: 79,360 block matches, $7 \times 7 \times 3$ reference block.

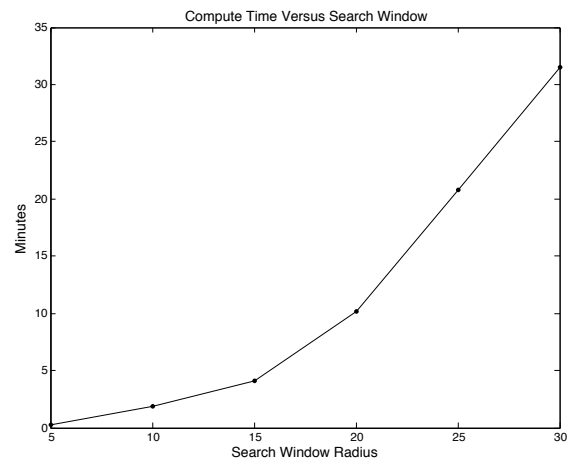


Figure 5.2 : MMBM algorithm timing results: 79,360 block matches, $11 \times 11 \times 3$ reference block.

Window Radius	Compute Time
5	6 sec
10	42 sec
15	99 sec (1.65 min)
20	239 sec (3.98 min)
25	288 sec (4.8 min)
30	756 sec (12.6 min)

Window Radius	Compute Time
5	16 sec
10	113 sec
15	245 sec (4.08 min)
20	609 sec (10.15 min)
25	1251 sec (20.85 min)
30	1887 sec (31.45 min)

Table 5.4 : Compute time for varying window radius for a fixed reference block size: $7 \times 7 \times 3$ (left), $11 \times 11 \times 3$ (right). Values are rounded up to nearest second.

5.3 Registration Results

Radial MILO was used to register five target images over the course of the expiratory phase and followed the implementation described in Chapter 4. Each registration was initiated with 79,360 reference voxels. Images sets 1-5 employed roughly 500 quintic Wendland functions with a support radius of 50 voxels and roughly 1000 basis functions with support radius of 75 voxels for cases 6-10. A small variance on the number of basis functions is made on account of the octree’s spatial partitioning. The selection of the parameters were determined by numerous numerical experiments.

Although the computational workload of the MMBM algorithm is cumbersome, employing a GPGPU can provide a reasonable compute time [16, 35]. Currently the ℓ_1 filtering algorithm is implemented serially via MATLAB. The rest of the algorithm is implemented in C++. Computing the full displacement field via MLS was completed with the aid of the Armadillo Linear Algebra Library [81]. Table 5.5 breaks down the different components of the Radial MILO algorithm for the registration of five images. The MLS column provides the duration of computing the whole displacement field from exhale to inhale.

Case	Window Radius	MMBM	ℓ_1 Solve	MLS (exhale/inhale - Displacement Field)	Total Compute Time
1	10	210 (sec) (3.5 min)	296 (s)	70 (s)	576 (sec) (9.6 min)
2	10	210 (sec) (3.5 min)	226 (s)	90(s)	526 (sec) (8.76 min)
3	10	210 (sec) (3.5 min)	286 (s)	78 (s)	574 (sec) (9.56 min)
4	15	495 (sec) (8.25 min)	253 (s)	72 (s)	820 (sec) (13.67 min)
5	15	495 (sec) (8.25 min)	274 (s)	76 (s)	845 (sec) (14.08 min)
6	15	1225 (sec) (20.4 min)	652 (s)	130 (s)	2007 (sec) (33.45 min)
7	20	3045 (sec) (50.75 min)	501 (s)	140 (s)	3686 (sec) (61.43 min)
8	20	3045 (sec) (50.75 min)	401 (s)	125 (s)	3571 (sec) (59.5 min)
9	15	1225 (sec) (20.4 min)	510 (s)	127 (s)	1862 (sec) (31.03 min)
10	15	1225 (sec) (20.4 min)	535 (s)	115 (s)	1875 (sec) (31.25 min)

Table 5.5 : Compute time of the Radial MILO components, rounded to the nearest second

5.3.1 Spatial Accuracy: Expiratory Phases

Table 5.6 presents the average error and standard deviation of the Radial MILO algorithm against the 75 propagated landmarks in the expiratory phase.

Case	Avg. Error (Std.): T10	Avg. Error (Std.): T20	Avg. Error (Std.): T30	Avg. Error (Std.): T40	Avg. Error (Std.): T50
1	0.25 (0.78)	0.62 (0.94)	0.72 (0.80)	0.76 (0.94)	0.77 (0.94)
2	0.54 (0.93)	0.63 (0.93)	0.68 (0.77)	0.60 (0.77)	0.68 (0.89)
3	0.76 (1.02)	0.88 (1.00)	0.87 (1.05)	0.87 (0.99)	0.83 (0.82)
4	0.82 (1.01)	1.05 (1.01)	1.12 (1.09)	1.12 (1.20)	1.18 (1.12)
5	1.00 (1.24)	1.07 (1.23)	1.04 (1.21)	1.03 (1.12)	1.25 (1.55)
6	0.76 (1.14)	1.31 (1.17)	1.21 (1.54)	1.13 (1.22)	0.97 (1.15)
7	0.74 (0.96)	1.20 (1.30)	1.22 (1.23)	0.95 (1.45)	1.06 (1.22)
8	0.96 (1.53)	1.13 (1.25)	1.14 (1.35)	1.12 (1.22)	1.15 (1.16)
9	0.87 (1.10)	0.74 (0.94)	0.99 (0.94)	1.22 (1.21)	1.12 (1.09)
10	0.88 (1.32)	1.20 (1.42)	1.24 (1.51)	0.97 (1.01)	0.82 (1.01)

Table 5.6 : Spatial Accuracy on 75 landmarks, all results are given in mm

5.3.2 Spatial Accuracy: Maximum Inhalation and Exhalation

In addition to providing image registration benchmark images, the DIR-Lab includes a comparison of spatial accuracies of registration algorithms, registering from maximum inhalation to maximum exhalation across at least 300 landmarks. Their comparison includes LFC, which is a similar image registration algorithm to Radial MILO [35]. Since no published comparison between LFC and the traditional MILO algorithm is available, the traditional MILO algorithm was implemented and used to register from maximum inhalation to maximum exhalation. Numerical results demonstrate that Radial MILO is able to achieve competitive results when compared to other block matching based algorithms.

Algorithm	Case 1	Case 2	Case 3	Case 4	Case 5
Radial MILO** (No ℓ_1)	1.10 (1.34)	0.90 (1.00)	0.98 (1.10)	2.19 (3.63)	2.40 (3.12)
Radial MILO**	0.75 (0.91)	0.73 (0.92)	0.86 (1.10)	1.29 (1.15)	1.23 (1.40)
MILO**[16]	0.75 (0.94)	0.75 (1.00)	0.966 (1.14)	1.31 (1.21)	1.42 (1.77)
LFC[35]	0.85 (1.00)	0.74 (0.99)	0.93 (1.07)	1.33 (1.51)	1.14 (1.25)
MLS[32]	1.58 (1.30)	1.47 (1.12)	2.27 (1.40)	2.50 (1.68)	2.55 (1.92)
CCLG[82]	1.02 (1.03)	1.29 (1.22)			2.50 (1.91)
COF[82]	1.17 (1.07)	1.37 (1.27)			2.57 (1.85)
LCI[82]	1.21 (1.21)	1.44 (1.69)			3.01 (2.85)
LII [82]	1.40 (1.27)	1.49 (1.35)			3.59 (2.83)
PF [83]	1.11 (1.09)	1.04 (1.15)	1.36 (1.20)	2.51 (2.49)	1.84 (1.74)
EPF [83]	1.10 (1.09)	1.00 (1.15)	1.32 (1.21)	2.42 (2.48)	1.82 (1.87)
AF [83]	1.15 (1.11)	1.05 (1.19)	1.39 (1.22)	2.34 (2.19)	1.81 (1.83)
DF [83]	1.19 (1.13)	1.16 (1.23)	1.48 (1.21)	2.59 (2.48)	1.91 (1.77)
ADF [83]	1.11 (1.09)	1.02 (1.14)	1.35 (1.20)	2.27 (2.09)	1.80 (1.80)
IC [83]	1.24 (1.30)	1.28 (1.62)	1.42 (1.22)	3.27 (4.09)	1.67 (1.57)
CPP [15]	1.07 (1.10)	0.99 (1.12)	1.23 (1.32)	1.51 (1.58)	1.95 (2.02)
4DLTM [15]	0.97 (1.02)	0.86 (1.08)	1.01 (1.17)	1.40 (1.57)	1.67 (1.79)
ALK [84]	0.89 (1.00)	0.83 (1.02)	1.08 (1.15)	1.45 (1.53)	1.55 (1.73)
cTVLI** [85]	0.78 (0.92)	0.78 (0.92)	0.93 (1.09)	1.24 (1.30)	1.22 (1.43)
cEPE**[86]	0.80 (0.92)	0.77 (0.92)	0.92 (1.10)	1.22 (1.24)	1.21 (1.47)
KDR** [87]	1.03 (0.48)	1.00 (0.46)	1.14 (0.61)	1.37 (0.97)	1.38 (1.19)
NLR** [88]	0.77 (0.90)	0.78 (0.90)	0.93 (1.06)	1.27 (1.26)	1.11 (1.46)
LMP**[89]	0.74 (0.90)	0.78 (0.90)	0.91 (1.05)	1.24 (1.25)	1.17 (1.48)
SGM3D** [90]	0.76 (0.92)	0.72 (0.87)	0.94 (1.07)	1.24 (1.26)	1.15 (1.42)
NGFa** [91]	0.78 (0.89)	0.79 (0.90)	0.93 (1.05)	1.27 (1.27)	1.07 (1.46)
NGFb**[91]	0.76 (0.89)	0.80 (0.88)	0.96 (1.07)	1.33 (1.29)	1.18 (1.45)

Table 5.7 : Comparison chart 1 for inhale/exhale registration, values denote average error (standard deviation error) all values are given in millimeters. ** Denotes analysis over 300 point-pair sets.

Algorithm	Case 6	Case 7	Case 8	Case 9	Case 10
Radial MILO** (No ℓ_1)	1.08 (1.40)	1.40 (1.27)	2.23 (3.56)	1.04 (1.09)	1.13 (2.17)
Radial MILO**	0.92 (1.02)	1.02 (1.21)	1.21 (1.72)	0.94 (1.12)	0.95 (1.42)
MILO**[16]	1.00 (1.20)	1.10 (1.21)	1.32 (1.52)	1.00 (1.05)	1.04 (1.2)
LFC[35]	1.04 (1.05)	1.03 (1.01)	1.11 (1.18)	1.04 (1.00)	1.05 (1.10)
CPP [15]	1.94 (1.72)	1.79 (1.46)	1.96 (2.33)	1.33 (1.17)	1.84 (1.90)
4DLTM [15]	1.58 (1.65)	1.46 (1.29)	1.77 (2.12)	1.19 (1.12)	1.59 (1.87)
ALK [84]	1.52 (1.28)	1.29 (1.22)	1.75 (2.40)	1.22 (1.07)	1.47 (1.68)
cTVL1** [85]	0.94 (0.99)	1.01 (0.96)	1.11 (1.28)	0.98 (1.00)	0.94 (1.03)
cEPE** [86]	0.90 (1.00)	0.98 (1.01)	1.16 (1.45)	1.00 (0.97)	0.99 (1.28)
NLR** [88]	0.91 (1.00)	0.86 (0.98)	1.03 (1.19)	0.97 (0.94)	0.87 (0.87)
LMP** [89]	0.90 (1.00)	0.87 (0.97)	1.04 (1.18)	0.98 (0.96)	0.89 (0.99)
SGMD3D** [90]	0.90 (0.98)	0.89 (0.95)	1.13 (1.40)	0.91 (0.93)	0.83 (0.92)
NGFa** [91]	0.90 (0.99)	0.85 (0.98)	1.03 (1.23)	0.94 (0.93)	0.83 (0.97)
NGFb** [91]	1.03 (1.04)	0.92 (0.93)	1.13 (1.15)	1.00 (0.96)	0.91 (0.99)

Table 5.8 : Comparison chart 2 for inhale/exhale registration, values denote average error (standard deviation error) all values are given in millimeters. ** Denotes analysis over 300 point-pair sets.

5.4 Review of DIR-LAB Reported Algorithms

Tables 5.7 and 5.8 present a comparison of over twenty different registration algorithms used to register the 4DCT images in Table 5.1. As such this thesis provides a brief description of each and categorizes them by their common techniques. Further details can be found in the articles referenced in Tables 5.7 and 5.8.

5.4.1 Searching Based Methods

Searching Based Methods aim to find a displacement vector for a single voxel via an exhaustive search of the target image. The Least Median of Squares (LFC) algorithm proposed in [35] is an example of such a method. LFC can be thought of as prelude to the traditional MILO algorithm encompassing the general three steps, see Chapter 4. The LFC algorithm was designed for pairs of images with the assumption that the motion of a single voxel can be described through a compressible flow model. An exhaustive search via a block matching algorithm is conducted in order to find an initial set of estimates to fit the model. As demonstrated in this thesis, the block matching algorithm does not guarantee spatially accurate displacements, therefore LFC employs an additional filtering technique. Once the estimates have been filtered the complete displacement field is recovered via MLS [35]. Although LFC employs a physics based block matching metric, Radial MILO provides estimates that are more spatially accurate with respect to the landmarks provided by the DIR-LAB. This could be due to inaccuracies with the LFC filtering technique.

Semi-Global Stereo Matching (SGM3D) is another example of a searching method, in which the optimal displacement for a single voxel is found by searching neighborhoods along potential lines emanating from the voxel [90]. Though conceptually simple, the approach is one of the top algorithms in the DIR-LAB dataset and its

structure is well suited for GPGPU programming. The SGM3D algorithm outperforms the Radial MILO algorithm, potentially because the search algorithm is more robust than block matching. A potential direction for improving Radial MILO would be to substitute the searching algorithm used by SGM3D.

5.4.2 Optical Flow Based Methods

Recalling Chapter 2, Optical Flow Methods calculate the motion between two image frames for every voxel position [18, 19, 20, 21]. The following algorithms from the comparison chart can be classified as optical flow methods: Combined Compressible Local Global (CCLG), Local Compressible Local (LCI), Compressible Optical Flow (COF), Local Incompressible flow (LII) [82], Advanced Lukas-Kanade Optical Flow (ALK) [84], and combined Gradient End Point Error (cEPE) [86]. Under the assumption that voxel intensity stays constant throughout time, $I(x(t), t) = \text{const}$, the classic optical flow equation is derived to be,

$$\frac{dI}{dt} = I_t + \nabla I \cdot v = 0, \quad (5.1)$$

where v denotes the velocity field optical flow algorithms aim to recover. Under the assumption of non-constant voxel intensity over time, the equation is derived to be,

$$I_t + \nabla I_t \cdot v + I \operatorname{div}(v) = 0. \quad (5.2)$$

Various articles discuss these derivations, [17, 82, 84, 86]. The problem is underdetermined as is, since there is only one equation and velocity has three components. Each of the previously mentioned methods apply regularization techniques in order to ensure the problem is well posed. Algorithms CCLG and LCI assume a non-constant voxel intensity while the remaining algorithms assume a constant velocity. As can be observed in the listed results, assuming that voxel intensity does not change over

frames does not necessarily lead to poor results, e.g. ALK and cEPE. In comparison to Radial MILO, ALK and cEPE can overall provide more spatially accurate results. Although the methods are all based on optical flow, they each employ different regularization strategies thus hinting that the regularization is critical to their performance.

5.4.3 Demons Based Methods

This subsection discusses the variants of the Demons image registration algorithms: Evolved Passive Force (EPF), Active Force (AF), Double Force (DF), Adjusted Double Force (ADF) and Inverse Consistent (IC) as evaluated by Gu et. al. [83], as well as the Knowledge Driven Registration algorithm (KDR) proposed by Muezing et. al. [87]. The original Demons algorithm applies the displacement vector,

$$d\mathbf{r}^{(n+1)} = \frac{(I_R^{(n)} - I_T^{(0)})\nabla I_T^{(0)}}{(I_R^{(n)} - I_T^{(0)})^2 + (\nabla I_T^{(0)})^2}, \quad (5.3)$$

onto every voxel of the reference image. In the described formulation, $I_T^{(0)}$ is the unmodified target image, and $I_R^{(n)}$ is the reference image at the n^{th} iteration. The displacement vector is applied and updated until convergence. The original formulation is commonly known as Passive Force (PF) on account of its derivation [83, 92, 93]. The variants of the Demons modify the displacement vector but the general idea is the same, [83, 87]. The fact that this algorithm acts on individual voxels makes it well suited for implementation on the GPGPU.

Building on the framework of the Demons algorithm, Muezing et. al. draws from statistical learning community to introduce a variation of the original Demons algorithm, KDR. Recalling that an image is a discretely sampled signal, it is necessary to smooth the image in order to compute derivative information. By varying the

smoothing kernel and comparing the quality of the registration, the algorithm can be "trained" to determine an appropriate smoothing kernel.

The original Demons algorithm and its variants (EPF, AF, DF, ADF, IC, and KDR) have not been reported to achieve the same spatial accuracy as Radial MILO. The assumption of the Demons algorithm is that the displacement vector is reasonably small or local [83], which is easily violated in image set 8 of the test cases in Table 5.1.

5.4.4 Trajectory Based Methods

Trajectory Based Methods aim to recover the potential trajectory that a voxel may have taken across various images. Castillo et. al. present the 4D Local Trajectory Modeling (4DLTM) algorithm. Which parameterizes voxel trajectories and uses compressible flow to describe the path that a voxel may have taken. As such, the problem reduces to finding a set of coefficients for the trajectory model [15]. The 4DLTM algorithm is compared against the Component Phase to Phase (CPP) approach which connects displacements of voxels in 4D images through linear paths.

The advantage of the 4DLTM algorithm over the CPP algorithm is the employment of a global model that describes the path throughout the images, since connecting piecewise would propagate errors. The drawback of the approach is that it leads to a non-convex formulation yielding the possibility of getting stuck in local minima when a gradient based scheme is used. Moreover the proposed algorithms do not achieve a comparable spatial accuracy to Radial MILO.

5.4.5 Variational Methods

The remaining algorithms can be classified as Variational Methods. Variational methods employ calculus of inequalities to minimize a functional. It should be noted that some of the methods described in the optical flow classification can also be grouped as variational methods. Thus the purpose of this section is to describe methods that do not fall into the optical flow category.

Examples of variational methods presented in the DIR-LAB comparison include methods that aim to align images based on notable structures. For example Ruhaak et. al. [88]’s algorithm, NLR, simply aligns based on structure, while Polzin et. al.’s algorithm, LMP [89] demonstrates that methods can be improved by paring the algorithms with automatic landmark detection. Konig et. at. [91] demonstrates that variational methods that aim to align based on structure can be ported onto GPGPU’s through his algorithm, LMP. This alignment approach has resulted in the most spatially accurate registration algorithms on the charts, and an overall higher spatial accuracy than Radial MILO.

A variational method that is notably different from previously mentioned methods is Hermann et. al.’s [85] registration algorithm based on Total Variation and the L_1 norm, cTV- L_1 . An introduction to $TV-L_1$ based optical flow can be found in [94]. The aim of the formulation is to minimize the total variation of the velocity field subject to a relatively sparse residual. The algorithm acts on independent voxels and is well suited for the GPGPU. The drawback of the proposed method is the need for smoothing the velocity field in order to compute the gradient, which may be a source for error. Overall cTV- L_1 is a competitive image registration algorithm, however Radial MILO generally achieves a higher spatial accuracy.

5.5 Visual Results

Since no image processing thesis would be complete without images, I provide 2D visualizations of the Radial MILO algorithm. Figure 5.3 demonstrates a quadtree adaptively partitioning the image into boxes. The centers of the occupied boxes then become the centers of the radial basis functions. Figure 5.4 denotes the target image for the reference image. Figure 5.5 overlays a sampled displacement field with unfiltered estimates (red) and filtered estimates (yellow). Figure 5.6 demonstrates a denser displacement field from the reference image T00 to T50 of case 1.

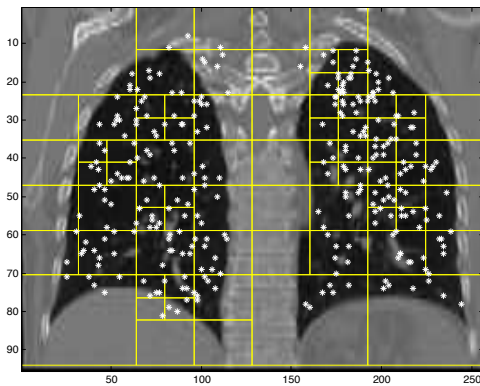


Figure 5.3 : Quadtree conforming to the data



Figure 5.4 : Case 1: Target Image

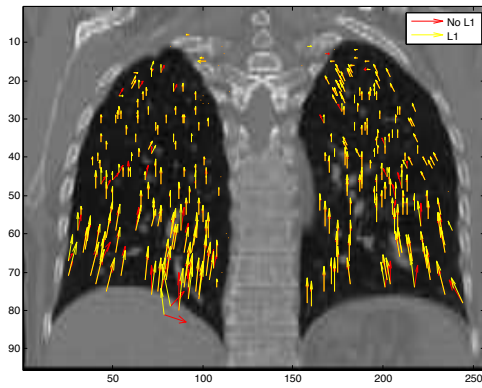


Figure 5.5 : Comparison of filtering estimates through ℓ_1 vs not filtering

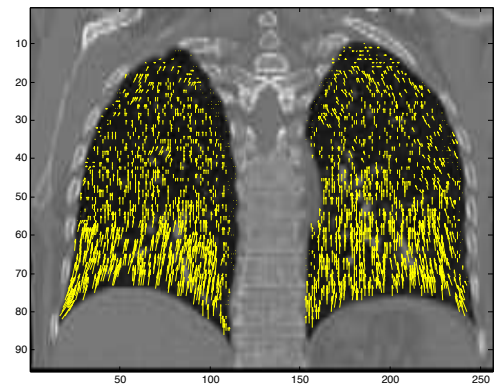


Figure 5.6 : A dense displacement field for case 1, mapping voxels from image T00 to T50

Chapter 6

Conclusion

This thesis presents Radial MILO, a 4D image registration algorithm inspired by the MILO algorithm. Building on the MILO framework, Radial MILO is also composed of three components: (1) an initial set of estimates are computed using an MMBM algorithm, (2) ℓ_1 optimization techniques are used in conjunction with a spatial parameterization to detect potentially spatially inaccurate data, (3) once erroneous estimates are removed, a full displacement field is computed through local approximations.

In MILO, uniform B-splines are used to create a parameterization; a requirement of splines is that data must be within the compact support of each function. Such a requirement is not guaranteed with MMBM data and has often lead to rank deficient linear systems in practice. Additionally the ℓ_1 optimization problem was traditionally solved by employing QR factorization.

As an improvement to the MILO algorithm, Radial MILO employs an adaptive parameterization based on radial basis functions and a 2^d data structure. It also uses a block coordinate descent algorithm, relieving the need for the additional storage associated with QR factorization. By modeling voxel displacements as a quadratic function in time, the parameterization is also able to incorporate multiple images. In order to validate the accuracy of the algorithm, results are compared to landmarks in Thoracic 4DCT images provided by the DIR Lab (<http://www.dir-lab.com/ReferenceData.html>). Although Radial MILO does contain various param-

eters, numerical results demonstrate that it can out-perform, or perform as competitively as, existing algorithms based on block matching (LFC, MILO).

The components of Radial MILO were implemented in C++, with the exception of the ℓ_1 optimization problem (4.6), which was solved using MATLAB. Aside from the block matching, most of the computational time is spent solving the ℓ_1 filtering problem. Thus a future research direction could be investigating a parallelizable algorithm to solve the ℓ_1 filtering problem. The compute time of the block matching algorithm can also be greatly reduced by employing a higher-end GPGPU.

Using a $2^{\hat{d}}$ tree data structure is an attempt to create a parameterization based on the locations of the MMBM data. As an alternative, an area of interest could be trying to conform a mesh to the segmented image. Another possible direction could be to explore interpolation as opposed to moving least squares. Although moving least squares does serve as a good approximate to interpolation, an interesting direction might be to investigate if exact interpolation yields better results.

Bibliography

- [1] R. A. Schowengerdt, *Remote Sensing, Third Edition: Models and Methods for Image Processing*. Orlando, FL, USA: Academic Press, Inc., 2006.
- [2] S. Dawn, V. Saxena, and B. Sharma, “Remote sensing image registration techniques: A survey,” in *Proceedings of the 4th International Conference on Image and Signal Processing, ICISP’10*, (Berlin, Heidelberg), pp. 103–112, Springer-Verlag, 2010.
- [3] M. Zuliani, *Computational Methods for Automatic Image Registration*. PhD thesis, Dec 2006.
- [4] L. Zheng and R. S. Blum, *Multi-sensor image fusion and its applications*. Signal processing and communications, Boca Raton, FL: Taylor & Francis, 2005.
- [5] S. Dawn, V. Saxena, and B. Sharma, “Remote sensing image registration techniques: A survey,” in *Image and Signal Processing* (A. Elmoataz, O. Lezoray, F. Nouboud, D. Mamass, and J. Meunier, eds.), vol. 6134 of *Lecture Notes in Computer Science*, pp. 103–112, Springer Berlin Heidelberg, 2010.
- [6] I. N. Bankman, ed., *Handbook of Medical Imaging*. Orlando, FL, USA: Academic Press, Inc., 2000.
- [7] T. S. Yoo and M. J. Ackerman, “Open source software for medical image processing and visualization,” *Commun. ACM*, vol. 48, pp. 55–59, Feb. 2005.

- [8] J. Weese, “Geometric and physical modelling in medical image processing: Methods, applications and examples,” in *Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling*, SPM '06, (New York, NY, USA), pp. 73–73, ACM, 2006.
- [9] M. Schellmann, J. Vörding, S. Gorlatch, and D. Meiländer, “Cost-effective medical image reconstruction: From clusters to graphics processing units,” in *Proceedings of the 5th Conference on Computing Frontiers*, CF '08, (New York, NY, USA), pp. 283–292, ACM, 2008.
- [10] H. Jacinto, R. Kéchichian, M. Desvignes, R. Prost, and S. Valette, “A web interface for 3d visualization and interactive segmentation of medical images,” in *Proceedings of the 17th International Conference on 3D Web Technology*, Web3D '12, (New York, NY, USA), pp. 51–58, ACM, 2012.
- [11] B. Zitova and J. Flusser, “Image registration methods: a survey,” *Image and Vision Computing*, vol. 21, no. 11, pp. 977 – 1000, 2003.
- [12] L. G. Brown, “A survey of image registration techniques,” *ACM Comput. Surv.*, vol. 24, pp. 325–376, Dec. 1992.
- [13] V. Mani and D. rivazhagan, “Survey of medical image registration,” *Journal of Biomedical Engineering and Technology*, vol. 1, no. 2, pp. 8–25, 2013.
- [14] A. Sotiras, C. Davatzikos, and N. Paragios, “Deformable medical image registration: A survey,” *Medical Imaging, IEEE Transactions on*, vol. 32, pp. 1153–1190, July 2013.
- [15] E. Castillo, R. Castillo, J. Martinez, M. Shenoy, and T. Guerrero, “Four-dimensional deformable image registration using trajectory modeling,” *Physics*

- in Medicine and Biology*, vol. 55, no. 1, p. 305, 2010.
- [16] D. F. Edward Castillo, Richard Castillo and T. Guerrero, “Computing Global Minimizers to a Constrained B-Spline Image Registration Problem from Optimal ℓ_1 Perturbations to Block Match Data,” *Medical physics*, vol. 41, no. 4, p. 041904, 2014.
 - [17] B. K. P. Horn and B. G. Schunck, “Determining optical flow,” *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.
 - [18] O. Fluck, C. Vetter, W. Wein, A. Kamen, B. Preim, and R. Westermann, “A survey of medical image registration on graphics hardware,” *Comput. Methods Prog. Biomed.*, vol. 104, pp. e45–e57, Dec. 2011.
 - [19] E. Castillo, *Optical Flow Methods for the Registration of Compressible Flow Images and Images Containing Large Voxel Displacements or Artifacts*. PhD thesis, William Marsh Rice University, 2008.
 - [20] T. Pock, M. Urschler, C. Zach, R. Beichel, and H. Bischof, “A duality based algorithm for tv-l1 optical-flow image registration,” in *Proceedings of the 10th International Conference on Medical Image Computing and Computer-assisted Intervention*, MICCAI’07, (Berlin, Heidelberg), pp. 511–518, Springer-Verlag, 2007.
 - [21] M. Lefébure and L. D. Cohen, “Image registration, optical flow and local rigidity,” *J. Math. Imaging Vis.*, vol. 14, pp. 131–147, Mar. 2001.
 - [22] R. Szeliski and J. Coughlan, “Spline-based image registration,” *Int. J. Comput. Vision*, vol. 22, pp. 199–218, Mar. 1997.

- [23] X. Yang, Z. Xue, X. Liu, and D. Xiong, “Topology preservation evaluation of compact-support radial basis functions for image registration,” *Pattern Recognition Letters*, vol. 32, no. 8, pp. 1162 – 1177, 2011.
- [24] T. Schiwietz, J. Georgii, and R. Westermann, “Interactive model-based image registration,” in *Proceedings of Vision, Modeling and Visualization 2007*, pp. 213–221, 2007.
- [25] K. Ø. Noe, K. Tanderup, J. C. Lindegaard, C. Grau, and T. S. Sørensen, “GPU accelerated viscous-fluid deformable registration for radiotherapy,” *Studies in health technology and informatics*, vol. 132, pp. 327–332, 2008.
- [26] A. Eklund, P. Dufort, D. Forsberg, and S. LaConte, “Medical image processing on the GPU - past, present and future,” *Medical Image Analysis*, vol. 17, pp. 1073–1094, 2013.
- [27] D. B. Kirk and W.-m. W. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1st ed., 2010.
- [28] B. Gaster, L. Howes, D. R. Kaeli, P. Mistry, and D. Schaa, *Heterogeneous Computing with OpenCL*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1st ed., 2011.
- [29] K. Murphy, B. van Ginneken, J. Reinhardt, S. Kabus, K. Ding, X. Deng, K. Cao, K. Du, G. Christensen, V. Garcia, T. Vercauteren, N. Ayache, O. Commowick, G. Malandain, B. Glocker, N. Paragios, N. Navab, V. Gorbunova, J. Sporring, M. de Bruijne, X. Han, M. Heinrich, J. Schnabel, M. Jenkinson, C. Lorenz,

- M. Modat, J. McClelland, S. Ourselin, S. Muenzing, M. Viergever, D. De Nigris, D. Collins, T. Arbel, M. Peroni, R. Li, G. Sharp, A. Schmidt-Richberg, J. Ehrhardt, R. Werner, D. Smeets, D. Loeckx, G. Song, N. Tustison, B. Avants, J. Gee, M. Staring, S. Klein, B. Stoel, M. Urschler, M. Werlberger, J. Vandemeulebroucke, S. Rit, D. Sarrut, and J. P. W. Pluim, “Evaluation of registration methods on thoracic ct: The empire10 challenge,” *Medical Imaging, IEEE Transactions on*, vol. 30, pp. 1901–1920, Nov 2011.
- [30] J. Vandemeulebroucke, D. Sarrut, and P. Clarysse, “The popi-model, a point-validated pixel-based breathing thorax model,” *Proceedings of the 15th International Conference on the Use of Computers in Radiation Therapy (ICCR '07)*, 2007.
- [31] R. Castillo, E. Castillo, D. Fuentes, M. Ahmad, M. L. Abbie Wood, and T. Guerrero, “A reference dataset for deformable image registration spatial accuracy evaluation using the copdgene study archive,” *Physics in Medicine and Biology*, vol. 58, pp. 2861–2877, 2013.
- [32] R. Castillo, E. Castillo, R. Guerra, T. M. Valen E. Johnson, A. K. Garg, and T. Guerrero, “A framework for evaluation of deformable image registration spatial accuracy using large landmark point sets,” *Physics in Medicine and Biology*, vol. 54, p. 1849, 2009.
- [33] G. Song, N. J. Tustison, B. B. Avants, and J. C. Gee, “Lung ct image registration using diffeomorphic transformation models,” in *Medical Image Analysis for the Clinic: A Grand Challenge*, pp. 23–32, 2010.
- [34] B. B. Avants, N. Tustison, and G. Song, “Advanced normalization tools (ants),”

2009.

- [35] E. Castillo, R. Castillo, B. White, J. Rojo, and T. Guerrero, “Least median of squares filtering of locally optimal point matches for compressible flow image registration,” *Physics in Medicine and Biology*, vol. 57, no. 15, p. 4827, 2012.
- [36] S. Chan, E. Panchanathan, “Review of block matching based motion estimation algorithms for video compression,” *Electrical and Computer Engineering, 1993. Canadian Conference*, vol. 1, pp. 151–154, 1993.
- [37] D. Thomas, “A study on block matching algorithms and gradient based method for motion estimation in video compression,” in *Advances in Digital Image Processing and Information Technology* (D. Nagamalai, E. Renault, and M. Dhanuskodi, eds.), vol. 205 of *Communications in Computer and Information Science*, pp. 136–145, Springer Berlin Heidelberg, 2011.
- [38] E. Monteiro, M. Maule, F. Sampaio, C. Diniz, B. Zatt, and S. Bampi, “Real-time block matching motion estimation onto gpgpu,” in *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pp. 1693–1696, Sept 2012.
- [39] V. Garcia, O. Commowick, and G. Malandain, “A Robust and Efficient Block Matching Framework for Non Linear Registration of Thoracic CT Images,” in *Grand Challenges in Medical Image Analysis (MICCAI workshop)*, (Beijing, China, China), pp. 1–10, 2010.
- [40] S. Ourselin, A. Roche, S. Prima, and N. Ayache, “Block matching a general framework to improve robustness of rigid registration of medical images,” in *Medical Image Computing and Computer Assisted Intervention MICCAI 2000* (S. Delp,

- A. DiGoia, and B. Jaramaz, eds.), vol. 1935 of *Lecture Notes in Computer Science*, pp. 557–566, Springer Berlin Heidelberg, 2000.
- [41] A. Rodriguez, C. Fernandez-Lozano, J. Dorado, and J. Rabuñal, “Two-dimensional gel electrophoresis image registration using block-matching techniques and deformation models,” *Analytical Biochemistry*, vol. 454, no. 1, pp. 53–59, 2014.
- [42] Y. Liu, A. Kot, F. Drakopoulos, C. Yao, A. Fedorov, A. Enquobahrie, O. Clatz, and N. P. Chrisochoides, “An itk implementation of a physics-based non-rigid registration method for brain deformation in image-guided neurosurgery,” *Frontiers in Neuroinformatics*, vol. 8, no. 33, 2014.
- [43] J. R. Edward Castillo Richard Castillo, Benjamin White, “Least median of squares filtering of locally optimal point matches for compressible flow image registration,” *Physics In Medicine and Biology*, vol. 57, pp. 4827–4833, Aug 2012.
- [44] M. Unser, “Splines: A perfect fit for signal and image processing,” *IEEE Signal Processing Magazine*, vol. 16, pp. 22–38, November 1999.
- [45] M. Embree, *Lecture Notes: Numerical Analysis 1*. 2010.
- [46] J. A. Cottrell, T. J. Hugues, and Y. Bazileves, *Isogeometric Analysis Toward Unification of CAD and FEA*. 2010.
- [47] T. J. Jacobson, “Optimized knot placement for b-splines in deformable image registration,” *Medical physics*, vol. 38, p. 3, August 2011.

- [48] R. Arcangéli, M. Cruz Lopez de Silanes, and J. J. Torrens, *Multidimensional Minimizing Splines: Theory and Applications*. Dordrecht: Springer, 2004.
- [49] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York: Springer, 2nd ed., 2006.
- [50] A. Bruhn, J. Weickert, and C. Schnörr, “Lucas/kanade meets horn/schunck: Combining local and global optic flow methods,” *International Journal of Computer Vision*, vol. 61, pp. 211–231, 2005.
- [51] Z. Xie and G. Farin, “Image registration using hierarchical b-splines,” *Visualization and Computer Graphics, IEEE Transactions on*, vol. 10, pp. 85–94, Jan 2004.
- [52] R. G. Baraniuk, “Compressive sensing,” *Lecture Notes in IEEE Signal Processing Magazine*, vol. 24, pp. 118–120, Jul. 2007.
- [53] W. M. Candes E., “An introduction to compressive sampling,” *IEEE Signal Processing Magazine*, vol. 346, no. 9-10, pp. 589 – 592, 2008.
- [54] Y. Zhang, “On theory of compressive sensing via ℓ_1 -minimization: Simple derivations and extensions,” 2008.
- [55] R. Tibshirari, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistics Society*, vol. 58, pp. 267–288, 1996.
- [56] A. Drozdek, *Data Structures and Algorithms in C++*. Pacific Grove, CA, USA: Brooks/Cole Publishing Co., 2nd ed., 2000.
- [57] M. Kelly and A. Breslow, “Quadtree construction on the gpu: a hybrid cpu-gpu approach,” tech. rep., Swathmore College.

- [58] H. Wendland, *Scattered Data Approximation*. Cambridge University Press, 2004. Cambridge Books Online.
- [59] G. Rambally and R. Rambally, “Octrees and their applications in image processing,” in *Southeastcon '90. Proceedings., IEEE*, pp. 1116–1120 vol.3, Apr 1990.
- [60] E. Haber, S. Heldmann, and J. Modersitzki, “An octree method for parametric image registration,” *SIAM J. Scientific Computing*, vol. 29, no. 5, pp. 2008–2023, 2007.
- [61] R. S. Sampath, *A Parallel Geometric Multigrid Method For Finite Elements on Octree Meshes Applied to Elastic Image Registration*. PhD thesis, William Marsh Rice University, September 2008.
- [62] E. Haber, S. Heldmann, and J. Modersitzki, “Adaptive mesh refinement for non-parametric image registration,” *SIAM Journal on Scientific Computing*, vol. 30, no. 6, pp. 3012–3027, 2008.
- [63] M. D. Buhmann and M. D. Buhmann, *Radial Basis Functions*. New York, NY, USA: Cambridge University Press, 2003.
- [64] R. Caverotto and A. D. Rossi, “Landmark-based registration using a local radial basis function transformation,” *Journal of Numerical Analysis, Industrial and Applied Mathematics*, vol. 5, no. 3-4, pp. 141–152, 2010.
- [65] B. Merry, J. Gain, and P. Marais, “Moving least-squares reconstruction of large models with gpus,” *Visualization and Computer Graphics, IEEE Transactions on*, vol. 20, pp. 249–261, Feb 2014.

- [66] A. Masood, A. Siddiqui, and M. Saleem, “A radial basis function for registration of local features in images,” in *Advances in Image and Video Technology* (D. Mery and L. Rueda, eds.), vol. 4872 of *Lecture Notes in Computer Science*, pp. 651–663, Springer Berlin Heidelberg, 2007.
- [67] L. He, Z. Peng, B. Everding, X. Wang, C. Y. Han, K. L. Weiss, and W. G. Wee, “Review: A comparative study of deformable contour methods on medical image segmentation,” *Image Vision Comput.*, vol. 26, pp. 141–163, Feb. 2008.
- [68] B. N. Li, C. K. Chui, S. Chang, and S. H. Ong, “Integrating spatial fuzzy clustering with level set methods for automated medical image segmentation,” *Comput. Biol. Med.*, vol. 41, pp. 1–10, Jan. 2011.
- [69] M. Rastgarpour, J. Shanbehzadeh, and H. Soltanian-Zadeh, “A hybrid method based on fuzzy clustering and local region-based level set for segmentation of inhomogeneous medical images,” *J. Med. Syst.*, vol. 38, pp. 1–15, Aug. 2014.
- [70] N. Sharma and L. M. Aggarwal, “Automated medical image segmentation techniques,” *Journal of medical physics / Association of Medical Physicists of India*, vol. 35, pp. 3–14, Jan. 2010.
- [71] D. S. Medina, A. St.-Cyr, and T. Warburton, “OCCA: A unified approach to multi-threading languages,” *CoRR*, vol. abs/1403.0968, 2014.
- [72] J. Thompson and K. Schlachter, “An introduction to the open cl programming model,” 2012.
- [73] F. Massanes, M. Cadennes, and J. G. Brankov, “Compute-unified device architecture implementation of a block-matching algorithm for multiple graphical

- processing unit cards,” *Journal of Electronic Imaging*, vol. 20, no. 3, pp. 033004–033004–10, 2011.
- [74] “Reduction kernel description.” https://www.cs.uaf.edu/2012/fall/cs441/lecture/11_29_reduction.html. Accessed: 2015-01-27.
- [75] N. Roma, J. Santos-Victor, and J. Tomé, “A comparative analysis of cross-correlation matching algorithms using a pyramidal resolution approach,” 2002.
- [76] G. Casella and R. Berger, *Statistical inference*. Duxbury Press Belmont, Calif, 1990.
- [77] M. Fornefett, K. Rohr, and H. Stiehl, “Radial basis functions with compact support for elastic registration of medical images,” *Image and Vision Computing*, vol. 19, no. 12, pp. 87 – 96, 2001.
- [78] J. Yang and Y. Zhang, “Alternating direction algorithms for l1-problems in compressive sensing,” *SIAM Journal on Scientific Computing*, vol. 33, pp. 250–278, Jan. 2011.
- [79] P. Tseng, “Convergence of a block coordinate descent method for nondifferentiable minimization,” *J. Optim. Theory Appl.*, vol. 109, pp. 475–494, June 2001.
- [80] B. Stephen, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Found. Trends Mach. Learn.*, vol. 3, pp. 1–122, Jan. 2011.
- [81] C. Sanderson, “Armadillo: An open source c++ linear algebra library for fast prototyping and computationally intensive experiments,” tech. rep., NICTA, Australia, October 2010.

- [82] E. Castillo, R. Castillo, Y. Zhang, and T. Guerrero, “Compressible image registration for thoracic computed tomography images,” *Journal of Medical and Biological Engineering*, vol. 29, pp. 222–233, 2009.
- [83] X. Gu, H. Pan, Y. Liang, R. Castillo, D. Yang, D. Choi, E. Castillo, A. Majumdar, T. Guerrero, and S. B. Jiang, “Implementation and evaluation of various demons deformable image registration algorithms on a gpu,” *Physics in Medicine and Biology*, vol. 55, no. 1, p. 207, 2010.
- [84] C. Hoog Antink, T. Singh, P. Singla, and M. Podgorsak, “Evaluation of advanced lukas?kanade optical flow on thoracic 4d-ct,” *Journal of Clinical Monitoring and Computing*, vol. 27, no. 4, pp. 433–441, 2013.
- [85] S. Hermann and R. Werner, “Tv-l1-based 3d medical image registration with the census cost function,” in *Image and Video Technology* (R. Klette, M. Rivera, and S. Satoh, eds.), vol. 8333 of *Lecture Notes in Computer Science*, pp. 149–161, Springer Berlin Heidelberg, 2014.
- [86] S. Hermann and R. Werner, “High accuracy optical flow for 3d medical image registration using the census cost function,” in *Image and Video Technology* (R. Klette, M. Rivera, and S. Satoh, eds.), vol. 8333 of *Lecture Notes in Computer Science*, pp. 23–35, Springer Berlin Heidelberg, 2014.
- [87] S. E. Muezing, B. Van Ginneken, and J. P. Pluim, “Knowledge driven regularization of the deformable field for pde based non-rigid registration algorithms,” *Medical Image Analysis for the Clinic - A Grand Challenge*, 2010.
- [88] J. Ruhaak, S. Heldmann, T. Kipshagen, and B. Fischer, “Highly accurate fast lung ct registration,” 2013.

- [89] B. Fischer and J. Modersitzki, “Combining landmark and intensity driven registrations,” *PAMM*, vol. 3, no. 1, pp. 32–35, 2003.
- [90] S. Hermann, “Evaluation of scan-line optimization for 3d medical image registration,” in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pp. 3073–3080, June 2014.
- [91] L. Konig and J. Ruhaak, “A fast and accurate parallel algorithm for non-linear image registration using normalized gradient fields,” in *Biomedical Imaging (ISBI), 2014 IEEE 11th International Symposium on*, pp. 580–583, April 2014.
- [92] J.-P. Thirion, “Image matching as a diffusion process: an analogy with maxwell’s demons,” *Medical Image Analysis*, vol. 2, no. 3, pp. 243 – 260, 1998.
- [93] J.-P. Thirion, “Fast Non-Rigid Matching of 3D Medical Images,” Research Report RR-2547, 1995.
- [94] J. Sánchez Pérez, E. Meinhardt-Llopis, and G. Facciolo, “TV-L1 Optical Flow Estimation,” *Image Processing On Line*, vol. 3, pp. 137–150, 2013.