

41 0675850 5



ProQuest Number: 10183159

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 10183159

Published by ProQuest LLC (2017). Copyright of the Dissertation is held by the Author.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code  
Microform Edition © ProQuest LLC.

ProQuest LLC.  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106 – 1346

432779.

NOTTINGHAM TRENT  
UNIVERSITY LIBRARY

NOTTINGHAM TRENT UNIVERSITY LIBRARY	

HIGH PERFORMANCE IMAGE  
REGISTRATION USING A  
DISTRIBUTED BLACKBOARD  
ARCHITECTURE

ROGER TAIT

A thesis submitted in partial fulfilment of the requirements  
of Nottingham Trent University for the degree of Doctor  
of Philosophy

April 2007



## **Abstract**

Registration is a method used to geometrically align two images taken from different sensors, viewpoints or instances in time. The images are aligned through a combination of translation, rotation, and scaling. A major drawback of registration is the performance burden associated with resampling and similarity calculation. Such bottlenecks limit registration applications where fast execution times are required. In this research, a novel approach to high performance intensity-based registration is presented. Based on a distributed blackboard architecture and implemented as knowledge sources (KSSs), a framework called iDARBS (imaging Distributed Algorithmic and Rule-based Blackboard System) provides an underlying worker/manager model. Division of intensity data into segments by a Distributor KS followed by allocation to multiple Worker KSs allows concurrent resampling and similarity calculation to be achieved. The supervision of Worker KS activities, the evaluation of computed similarity, and the optimisation of transform parameters that map between segments are performed by a Manager KS. Conveniently, the modular nature of the approach permits different similarity calculation strategies to be added to the iDARBS framework without change. The successful distribution of intensity correlation and mutual information-based similarity metrics for the alignment of 2D and 3D data captured by a range of sensor types is demonstrated. Experimental results show a speedup factor of three combined with an efficiency of 43% was achieved during image registration using eight Worker KSs. During single-modal volume registration using ten Worker KSs, a speedup factor of seven and an efficiency of 67% was accomplished. Finally, a speedup factor of three combined with an efficiency of 50% was achieved during multi-modal volume registration using six Worker KSs. Crucially, the results reported confirm the success of the approach.

## **Acknowledgements**

I would like to extend thanks to my supervisors, Dr Gerald Schaefer and Professor Adrian Hopgood for their guidance, advice, and input regarding this research.

Also I must thank my father Alec, brother Nigel and girlfriend Yalan Li for their encouragement and constant support.

I would like to acknowledge the School of Computing & Informatics technical staff for providing and maintaining the hardware resources used throughout this project. The contributions of Craig Thurlby, Mike Haber, and Mark Howson are especially recognised.

Finally, to friends and colleagues who have not been mentioned, thank you for your ideas and informal discussions.

## Table of contents

<b>ABSTRACT</b> .....	<b>I</b>
<b>ACKNOWLEDGEMENTS</b> .....	<b>II</b>
<b>TABLE OF CONTENTS</b> .....	<b>III</b>
<b>LIST OF FIGURES</b> .....	<b>VI</b>
<b>1 INTRODUCTION</b> .....	<b>1</b>
1.1 IMAGE REGISTRATION .....	1
1.2 AIMS OF RESEARCH .....	6
1.3 SUMMARY .....	9
<b>2 BACKGROUND</b> .....	<b>10</b>
2.1 REGISTRATION ALGORITHM COMPONENTS .....	10
2.1.1 <i>Transform</i> .....	11
2.1.2 <i>Interpolator</i> .....	13
2.1.3 <i>Similarity metric</i> .....	15
2.1.4 <i>Optimiser</i> .....	16
2.2 CLASSIFICATION OF REGISTRATION ALGORITHMS .....	18
2.2.1 <i>Intensity-based registration</i> .....	19
2.2.2 <i>Multi-modal registration</i> .....	21
2.2.3 <i>Multi-resolution registration</i> .....	23
2.2.4 <i>Landmark-based registration</i> .....	25
2.3 PARALLEL PROCESSING .....	27
2.3.1 <i>Related image registration work</i> .....	29
2.4 DISTRIBUTED ARTIFICIAL INTELLIGENCE.....	32
2.4.1 <i>Multi-agent systems</i> .....	33
2.4.2 <i>Blackboard architectures</i> .....	34
2.5 SUMMARY .....	36
<b>3 IDARBS – A DISTRIBUTED IMAGE PROCESSING FRAMEWORK</b> .....	<b>40</b>
3.1 BLACKBOARD SELECTION .....	41
3.1.1 <i>Background on DARBS</i> .....	43
3.1.2 <i>Basic DARBS commands</i> .....	45
3.2 THE IDARBS IMPLEMENTATION .....	46

## Table of contents

---

3.2.1 <i>Modifications and additions to the DARBS architecture</i> .....	48
3.3 PARTITIONING OF INFORMATION ON THE BLACKBOARD .....	52
3.4 KS BEHAVIOUR .....	55
3.4.1 <i>The Distributor KS</i> .....	55
3.4.2 <i>The Worker n KS</i> .....	58
3.4.3 <i>The Manager KS</i> .....	60
3.5 EXPERIMENTAL TESTING .....	62
3.5.1 <i>Aims and setup</i> .....	63
3.5.2 <i>Results and discussion</i> .....	66
3.6 CONCLUSIONS .....	71
3.7 SUMMARY .....	73
<b>4 HIGH PERFORMANCE INTENSITY-BASED IMAGE REGISTRATION .....</b>	<b>75</b>
4.1 TOOLKIT SELECTION .....	76
4.1.1 <i>An intensity-based registration algorithm</i> .....	77
4.2 IMAGE REGISTRATION ON iDARBS .....	81
4.2.1 <i>Information strings and the image registration process</i> .....	82
4.3 KS BEHAVIOUR DURING IMAGE REGISTRATION .....	84
4.3.1 <i>The Distributor KS</i> .....	84
4.3.2 <i>The Worker n KS</i> .....	86
4.3.3 <i>The Manager KS</i> .....	91
4.4 EXPERIMENTAL TESTING .....	97
4.4.1 <i>Results and discussion</i> .....	98
4.4.2 <i>Speedup and efficiencies achieved</i> .....	103
4.5 CONCLUSIONS .....	106
4.6 SUMMARY .....	108
<b>5 SINGLE AND MULTI-MODAL VOLUME REGISTRATION .....</b>	<b>110</b>
5.1 SINGLE-MODAL VOLUME REGISTRATION .....	111
5.2 SINGLE-MODAL VOLUME REGISTRATION ON iDARBS .....	114
5.2.1 <i>Visualisation of registered volumes</i> .....	114
5.3 EXPERIMENTAL TESTING .....	116
5.3.1 <i>Results and discussion</i> .....	117
5.4 MULTI-MODAL VOLUME REGISTRATION .....	119
5.5 MULTI-MODAL VOLUME REGISTRATION ON iDARBS .....	122
5.5.1 <i>Sample generation by the Distributor KS</i> .....	122
5.5.2 <i>Local probability distribution generation by the Worker n KS</i> .....	123
5.5.3 <i>Global probability distribution generation by the Manager KS</i> .....	125

Table of contents

---

5.6 EXPERIMENTAL TESTING.....	126
5.6.1 Results and discussion .....	128
5.7 CONCLUSIONS .....	134
5.8 SUMMARY .....	135
<b>6 CONCLUSIONS, DISCUSSION, AND FUTURE WORK.....</b>	<b>137</b>
6.1 CONCLUSIONS .....	137
6.2 DISCUSSION .....	140
6.2.1 iDARBS – A distributed image processing framework .....	142
6.2.2 High performance intensity-based image registration.....	143
6.2.3 Single and multi-modal volume registration .....	145
6.3 FUTURE WORK.....	148
<b>REFERENCES.....</b>	<b>153</b>
<b>APPENDIX A : LIST OF PUBLICATIONS.....</b>	<b>167</b>
<b>APPENDIX B : VALIDATION OF IDARBS COMPONENTS.....</b>	<b>168</b>
<b>APPENDIX C : TEST IMAGE CAPTURE CONDITIONS .....</b>	<b>170</b>
<b>APPENDIX D : IDARBS TABLES OF RESULTS .....</b>	<b>171</b>
<b>APPENDIX E : IMAGE REGISTRATION RULES.....</b>	<b>174</b>
<b>APPENDIX F : VISUALISATION OF REGISTERED IMAGES .....</b>	<b>179</b>
<b>APPENDIX G : IMAGE REGISTRATION TABLES OF RESULTS .....</b>	<b>182</b>
<b>APPENDIX H : VISUALISATION OF REGISTERED VOLUMES.....</b>	<b>184</b>
<b>APPENDIX I : VOLUME REGISTRATION TABLES OF RESULTS .....</b>	<b>185</b>

## List of figures

Figure 1: Inputs to the similarity metric.....	4
Figure 2: Tightly and loosely-coupled architectures.....	5
Figure 3: Examples of 2D transformations.....	12
Figure 4: Joint probability distributions as a result of rotating the moving image by $0^\circ$ , $2^\circ$ , and $5^\circ$ .....	22
Figure 5: The multi-resolution pyramid scheme.....	24
Figure 6: Example image distribution schemes required for a loosely-coupled architecture. .....	28
Figure 7: An example of the random access pattern of communication.....	31
Figure 8: The components of a blackboard architecture. ....	35
Figure 9: The client/server model employed by DARBS.....	43
Figure 10: The worker/manager model on which iDARBS is based.....	47
Figure 11: The header attached to pixel data.....	49
Figure 12: The simple image viewer attached to iDARBS KSs. ....	51
Figure 13: Blackboard partitions and KS interdependence used to balance communication and processing workloads. ....	53
Figure 14: Example information strings.....	54
Figure 15: The Distributor KS flow diagram.....	56
Figure 16: The variable number of borders depending on a segments position within the selected image. ....	57
Figure 17: The Worker n KS flow diagram. ....	59
Figure 18: The Manager KS flow diagram.....	61
Figure 19: Test images containing screen-printed shampoo bottles.....	65
Figure 20: The sequential and distributed processing speed of mean filtering.....	67
Figure 21: The sequential and distributed processing speed of local histogram equalisation.....	69
Figure 22: The sequential and distributed processing speed of adaptive thresholding.....	71
Figure 23: Information strings for controlling the image registration process.....	83
Figure 24: The updated Worker n KS flow diagram.....	87
Figure 25: The transformation of pixel co-ordinates between fixed and moving segments. .....	89
Figure 26: The combination of transform Jacobian and moving segment gradients. ....	90
Figure 27: The updated Manager KS flow diagram.....	94

**Figure 28: The flow of local derivatives and current transform parameters between iDARBS components. ....96**

**Figure 29: The sequential and distributed processing speed of image registration using mean square error as a similarity metric.....100**

**Figure 30: The sequential and distributed processing speed of image registration using normalised correlation as a similarity metric.....103**

**Figure 31: The speedup and efficiency achieved by distributed image registration using mean square error as a similarity metric.....105**

**Figure 32: The speedup and efficiency achieved by distributed image registration using normalised correlation as a similarity metric.....106**

**Figure 33: A volume rendering using the 3D viewer. ....115**

**Figure 34: The sequential and distributed processing speed of single-modal volume registration using normalised correlation as a similarity metric.....117**

**Figure 35: The speedup and efficiency of single-modal volume registration using normalised correlation as a similarity metric.....118**

**Figure 36: Local joint probability distribution construction.....124**

**Figure 37: Local marginal probability distribution construction. ....125**

**Figure 38: MRI spin-spin relaxation (T2) and proton spin density (PD) volumes. ....127**

**Figure 39: The sequential and distributed processing speed of multi-modal volume registration using mutual information as a similarity metric. ....130**

**Figure 40: The speedup and efficiency of multi-modal volume registration using mutual information as a similarity metric. ....131**

**Figure 41: Local and global joint probability distributions generated by the iDARBS framework. ....132**

**Figure 42: The sequential and distributed time required to perform single iterations of the optimisation cycle using four Worker KSs. ....133**

**Figure 43: Focused KS activation on iDARBS.....149**

**Figure 44: Multi-resolution registration on iDARBS. ....151**

## 1 Introduction

Image registration is an important step in analysis tasks where information is extracted from a combination of sources. For example, in the manufacturing industry there is an increasing need for automated visual inspection in the detection of imperfections [1] [2]. Motivating factors for the adoption of an automated approach include the reduction of expensive labour costs, reproducibility, and the matching of inspection speed with production. Visual inspection is performed by moving samples in front of a camera, where high resolution images are captured. Knowledge is then extracted from ideal and captured images in order that alignment and referential comparison can be made. The ideal represents an image with an acceptable level of quality. Other examples include remote sensing (weather forecasting, the integration of information into geographic information systems, and environmental monitoring [3]) as well as computer vision (target tracking, optical character recognition, and model-base object recognition [4]). Medicine (monitoring tumour growth, treatment verification, and the comparison of patient data with anatomical atlases [5]) is currently the most prominent field of application. The major limitation associated with these applications is the high computational cost associated with image alignment. As a result, they have limited application where fast execution times are required.

### 1.1 Image registration

According to Zitova and Flusser [6] image registration can be classified as either landmark or intensity-based. This is because a universal registration method is impractical due to the wide variety of noise and geometric deformations contained within captured images. The landmark-based registration process consists of four main stages.



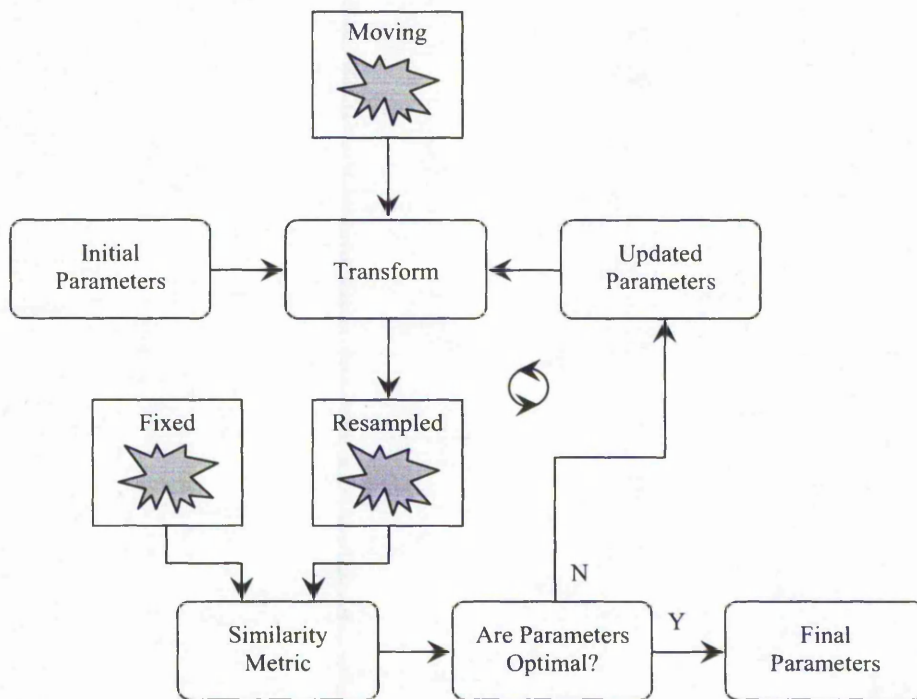
- During the feature detection stage, distinguishing characteristics such as corners, edges, and centres of gravity are manually or automatically identified. Once detected, these features are often called landmarks. The identification of landmarks is performed on both reference (fixed) and sensed (moving) images.
- The optimisation stage [7] controls estimation of transform parameters that geometrically map landmarks from the fixed image to the moving image.
- The feature matching stage is achieved through the use of a similarity metric in which a degree of closeness between corresponding landmarks is calculated.
- On the selection of appropriate transform parameters, pixel values which are mapped into non-integer co-ordinates are interpolated in order to establish their value. This represents the image re-sampling stage [8].

In the more common intensity-based image registration methods, the feature detection stage is omitted [9]. As a consequence, transform parameter optimisation and feature matching are performed using pixel intensities instead of landmarks. Unfortunately, the use of optimisation schemes to explore a search space of allowable transform parameters is particularly susceptible to noise which manifests itself as local optima. Also, because intensity-based methods exploit pixels intensities without any kind of scene analysis, they are more sensitive to image acquisitions conditions than landmark-based approaches. Crucially, miss-alignment can occur during intensity-based registration due to the non-saliency of the images being registered [10]. Smooth areas of the fixed image, for example, can be matched incorrectly with smooth areas of the moving image due to a lack of discriminating features. Unsurprisingly, the growth in computational burden with increasing transformation complexity limits the use of intensity-based methods in real-time applications.

Hybrid registration applications that employ a combination of techniques are increasingly commonplace [11]. With such applications an initial estimate of alignment is made using a landmark-based approach. The landmark-based algorithm allows for the evaluation of large scale transformations at high speed due to the reduced level of computational burden involved. Sub-pixel accuracy is then achieved using an intensity-based approach. As only small corrections are required, the higher burden associated with the evaluation of large numbers of pixels is considered acceptable. In work by Wang and Feng [12], a hybrid approach is employed to accurately and efficiently register protein sequence data. First, wavelet-based registration which fully exploits pixel intensities is used to estimate a global transform. Automatic landmark-based elastic registration is then employed to correct local displacements and enhance registration accuracy. In the method described a hierarchical, from low resolution to high resolution, registration scheme is employed to accelerate both the intensity and landmark-based alignment process.

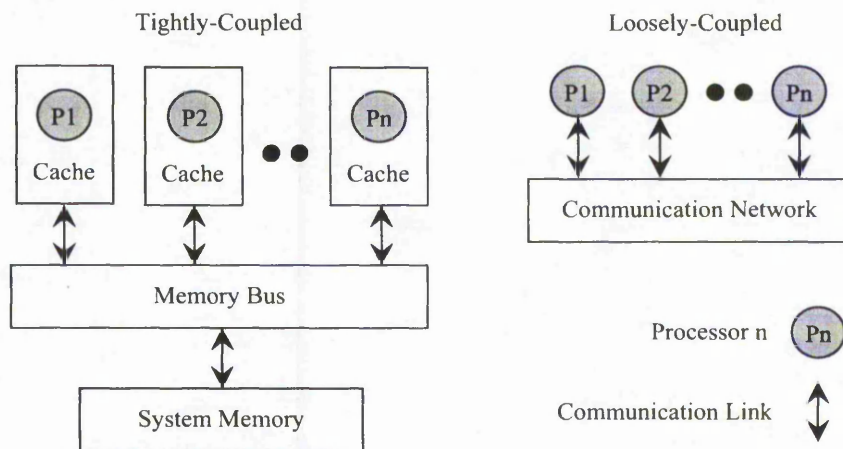
The most important component of an image registration algorithm is the similarity metric used to determine when images are in accurate alignment [13] [14]. Inputs to and output from a basic metric are illustrated in Figure 1. In general, a metric works by examining corresponding pixel values in both fixed and moving images and then formulating a measure of similarity based on the relationship between these intensities. The metric assumes that the relationship changes with variations in the spatial transform used to map between images and a maximum similarity is achieved when the images are in close alignment [15]. Intensity equality which is high when pixels are similar, is one such relationship employed as a similarity metric in single-modal registration where

images are captured using the same sensor type. Total equality, however, is seldom reached due to noise and image acquisition inconsistencies. Therefore additional robustness is achieved by assessing the ratio of intensities and minimising the variance of such ratios. When images are acquired with different sensor types, as in the multi-modal case, an extension of the ratio method which maximises the weighted collection of variances can be employed. Alternatively, a relationship can be formulated by estimating the entropy of corresponding intensity pairs [16]. Where entropy, derived from information theory, measures the amount of information contained within a signal. Although many algorithms have been proposed, similarity calculation remains an intensive task.



**Figure 1: Inputs to the similarity metric. The output is a single value that is used to determine suitability of the optimised transform parameters.**

Unsurprisingly, high image resolutions coupled with complex algorithms are increasing the demand for high speed processing capabilities. The use of parallel computing to overcome the time constraints associated with image processing applications has also grown in popularity [17] [18]. Conveniently, many of the image registration algorithms developed are inherently parallel and therefore well suited to distribution. An important consideration when adopting a parallel processing approach is the architecture of the host system [19]. In a computer constructed of multiple processors with shared-memory, data distribution is not required. These systems are viewed as tightly-coupled architectures. A loosely-coupled architecture, in contrast, consists of multiple computers in different locations. Loosely-coupled architectures require data distribution, communication, and accumulation mechanisms. Importantly, the most effective distribution scheme will depend on the architecture of the host system. Contrasting architectures of host systems are illustrated in Figure 2.



**Figure 2: Tightly and loosely-coupled architectures. Data is fetched either from the main memory system, via a memory bus, or is transferred over a communications network.**

Loosely-coupled architectures have the advantage that the components of an application can reside on different hardware. This means an application can be distributed across a network making it robust and allowing components to fail without bring down the entire application [20]. Loosely-coupled architectures have the disadvantage that each component requires communication and collaboration capabilities which allow them to run as separate processes. Such capabilities represent overheads which reduce the performance of an application. Tightly-coupled architectures, in contrast, allow efficient processing by avoiding data replication and the transfer of information between architecture components. The high cost of hardware required to scale tightly-coupled architectures however is seen as a major disadvantage. It is unlikely, for example, that the hardware employed in a tightly-coupled architecture can be recycled as independent processing units upon retirement [21]. Crucially, a limited body of research into the advantages of image registration on both tightly and loosely-coupled architectures exist. Due to the continued emergence of increasingly complex registration algorithms, there is a growing need to investigate the suitability and performance benefits of more flexible software architectures.

## **1.2 Aims of research**

The purpose of this research is an investigation into high performance intensity-based image registration in a loosely-coupled environment, for the timely alignment of data from such diverse fields as manufacturing [22] and medicine [23]. An intensity-based registration method was selected for distributed implementation because of its increasing use in time critical applications and lack of user intervention, particularly in multi-modal applications where the identification of inherent landmarks is problematic [24]. The investigation considers combining the concepts of image processing and

distributed artificial intelligence to create a flexible, efficient, and robust method of registration that is capable of aligning intensity data produced by a variety of capture devices. Traditionally, registration algorithms have been implemented using single processor architectures which are limited by memory and speed constraints. Such limitations have a negative impact in manufacturing where real-time processing is needed to provide information for the removal of defective products further along the production line [25]. Also in clinical applications where real-time processing allows physicians to monitor the progress of treatment while a patient is present.

The inherent parallelism of a registration algorithm has recently been exploited by Rohlfing and Maurer [26]. In their work, fine-grain parallelism is used to divide an algorithm into low-level components each of which is hosted by a separate processor. Although good for maximising speedup, the fine-grain parallelism employed complicates distribution of the registration algorithm and reduces flexibility of the approach. This is because the basic alignment steps, namely transform optimisation, image re-sampling, and similarity calculation are distributed between all processors. In this research, coarse-grained parallelism is employed to increase flexibility and allows the issues of fine-grained parallelism to be ignored. As a consequence, the basic alignment steps are allocated to individual processors, the most computationally intensive of which being performed concurrently.

According to Nii [27], blackboard architectures provide a co-ordinated and distributed problem solving environment that can be used to combine multiple processing techniques. The performance benefits of a distributed blackboard system have been investigated by Sobczak and Matthews [28]. Results published by their group suggest

that distributed blackboard architectures inherently introduce communication overheads which degrade the performance benefits of parallel implementation. Although maximum parallel performance cannot be achieved by such architectures, a distributed blackboard system was selected for this research because of its flexibility of implementation. Crucially, the modular nature of the blackboard architecture allows the implementation of similarity calculation strategies as specialised components which can be added to the framework without modification. A loosely-coupled architecture was chosen in order to demonstrate the scalability of registration algorithms in a non-specialised distributed processing environment. Alternative architectures, employed in high performance intensity-based image registration research are discussed in Section 2.3.1.

The aims of this research can be summarised as follows:

- Examine the suitability and benefits of high performance intensity-based image registration using a distributed blackboard architecture, for industrial and medical applications through:
  1. The development of a distributed processing environment on which image registration algorithm can be hosted.
  2. The distribution of an image registration algorithm as well as multiple similarity calculation strategies.
  3. Extension of underlying registration functionality to volume and multi-modal datasets.
- Experimental testing to evaluate the performance benefits of the proposed approach. Testing will compare sequential algorithms with their distributed counterparts using off-the-shelf hardware.



- Discuss the advantages and disadvantages of the implemented registration framework as well as identify future work.

### **1.3 Summary**

Parallel processing is increasingly employed in the analysis of high resolutions images using complex algorithms. Image registration, used in the detection of anomalies, is one such example from the fields of manufacturing and medicine. In practical terms, iterative resampling and similarity calculation represents a considerable performance bottleneck that limits the speed of registration algorithms. Understandably, these limitations have been successfully addresses through distribution using specialised multi-processor architectures. The inability to swap algorithm components based on their different strengths, however, restricts flexibility of these implementations. With these limitations in mind, this research investigates the performance benefits of image registration in a loosely-coupled architecture and the flexibility that arises from the resulting coarse-grained approach.



## 2 Background

In Chapter 1, the concept of image registration was established. The basic stages of image alignment were then identified and standard fields of application introduced. Importantly, image resampling and similarity calculation was highlighted as the major performance bottleneck associated with image registration. The aims of this research were also identified.

Approaches to image registration, parallel computing, and distributed artificial intelligence are surveyed in this chapter. Section 2.1 provides detailed descriptions of the components associated with image registration algorithms. In Section 2.2 registration algorithms are classified by application domain as well as computational burden. Section 2.3, in contrast, provides general background information on parallel processing architectures and associated high performance image registration applications. Under the heading of distributed artificial intelligence, Section 2.4 sees both multi-agent systems and blackboard architectures discussed as potential parallel processing environments. A short summary is provided in Section 2.5.

### 2.1 Registration algorithm components

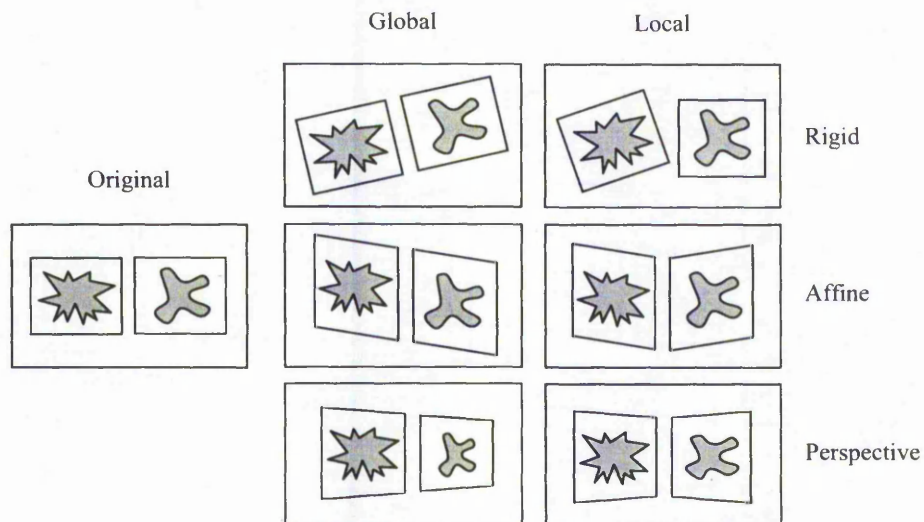
In general, the basic registration algorithm can be divided into four parts. The spatial mapping of intensities throughout the alignment process is achieved with a transform component. An interpolation component is used to evaluate intensities at non-discrete locations. The metric component calculates a measure of alignment accuracy. Optimisation of the similarity measure, using a search space defined by transform parameters, is achieved with an optimisation component. In the following section each component is discussed in more detail.

### 2.1.1 Transform

As already mentioned, the transform component is used to spatially map intensities, from reference (fixed) to sensed (moving) image space. In general, the mapping function should correspond to the assumed geometric deformation between images and to the accuracy of the alignment required [29]. When a model exists for the distortion, such as when the capture device or the geometry of a scene is known, correction based on the inverse of the known deformity can be performed. The Earth's shape combined with a satellite's orbit represents an example of a commonly used model [30]. Depending on the space in which a mapping function is to operate, transforms are normally categorised by their nature and domain. Classification of transform types according to Galbiati [31] is illustrated in Figure 3. Rigid, affine, and perspective movement are described as nature whereas the domain of a transform is considered as being local or global. Global means a transform is applied across an entire image whereas local means the transform is applied within sub-regions. Each transform type is considered a simpler version of the one derived from it. For example, a rigid transform is a simple type of affine transform.

Transform complexity can also be categorised with respect to the degrees of freedom (DOF) it represents. Where DOF, are the set of independent displacements that are specified by a transform. As shown in Figure 3, a transform is considered rigid when only translations and rotations are permitted. When applied to an image, this normally means a total of three DOF. They include two translations, one in each dimension, and a rotation around a single perpendicular axis. When applied to a volume [32], the transform represents six DOF. They include three translations, one in each dimension,

and three rotations one around each axes. Some rigid transforms also include scaling. As a consequence, it is common practice to classify this characteristic as belonging to the rigid transform type. A transform is considered affine when it maps straight lines to straight lines and preserves parallelism between lines. The geometric deformation of shear is also included in this type. Importantly, six DOF are required for registration of two images. 12 DOF, in contrast, are required for the registration of two volume datasets. The perspective transform differs from the affine type in the sense that the parallelism of lines is not preserved.



**Figure 3: Examples of 2D transformations.** If a transform is applied to the entire image it is considered global, when subsections of an image have their own transform defined it is considered local.

Although 12 DOF is sufficient for most registration applications, additional DOF are required for complex non-rigid alignment including perspective projections [33]. At an extreme B-spline free-form deformation, common to medical image analysis, yield transforms with more than 3000 DOF. Transforms of increasing complexity including

rigid, rigid with uniform scaling, rigid with non-uniform scaling, and fully affine are compared by Shekhar and Zagrodsky [34].

When considering the domain of a transform, a global model represents a single set of parameters which are applied to an entire image. A local model, in contrast, can be described as the application of two or more transforms to sub-regions of an image. Where, each transform is sufficiently different that it cannot be accomplished with a single geometric operation. Under these conditions, the individual transforms are considered global but only in a region within the original image. Crucially, local models are normally implemented as an arrangement of patches where different parameters are applied depending on their location within the image. Piecewise linear mapping and piecewise cubic mapping, as described by Goshtasby [35], are two approaches based on interpolation that appear in the literature. More recently, radial basis functions which stem from global transformation methods have been employed in a local mapping context [36]. Because the local continuity of an image is impaired, the use of local transform models is uncommon.

### 2.1.2 Interpolator

In general, when intensities are mapped between images they will be assigned to non-discrete grid locations. To estimate the value of such locations, the interpolation of intensities is used. Interpolation takes place based in a regular grid and ensures that neither holes nor overlaps can occur within a re-sampled image. At an implementation level, interpolation is achieved via convolution of an image with a kernel. Plum *et al.* [37] suggest that the method of interpolation used affects smoothness of the transform parameter search space. As a consequence, the time required to register images

significantly increases. Crucially, a poor choice of an interpolation scheme can result in the appearance of artefacts and local optima. Large changes in similarity are also known to occur when two images are in near optimal alignment and the effects of interpolation are negligible. As interpolation is applied repeatedly during the alignment process, a trade-off between accuracy and efficiency needs to be achieved. Therefore, to reduce computational burden, separable interpolation schemes that allow a 2D kernel to be replaced with a 1D kernel, are commonplace.

It is common for interpolation to be divided into two distinct groups consisting of scene and shape-based methods. With scene-based methods, interpolated values are determined directly from the intensities of an image. In shape-based methods, shape information extracted with the aid of segmentation is used for guiding the interpolation process [38]. A survey of interpolation methods has been published by Lehman *et al.* [39], who compare techniques using spatial and Fourier analysis. Surveyed schemes include nearest neighbour, linear, and B-spline interpolation. For nearest neighbour interpolation, the intensity of a pixel is set to the value of the nearest grid position. Linear interpolation, in contrast, assumes that the intensity of a pixel varies linearly between grid positions. The multiplication of B-spline coefficients, within a predefined neighbourhood that surrounds a pixel, is used during B-spline interpolation. Importantly, artefacts in the re-sampled image are commonplace when nearest neighbour interpolation is employed. Although outperformed by higher-order methods, in terms of search space smoothness and visual appearance, B-spline interpolation offers the best trade-off between accuracy and complexity [40].

### 2.1.3 Similarity metric

By comparing intensities, the metric component quantitatively measures how accurately fixed and moving images are aligned. The selection of a metric component is largely dependent on the type of registration problem to be solved [41] [42]. For example, some metrics possess large capture ranges that are well suited to images misaligned by a large transform. Other metrics, in contrast, are less computationally intensive but require initial transform parameters to be close to optimum. During the alignment process, most metrics sample intensities over an entire image. Some metrics, however, employ a subset of samples drawn from the fixed image. In both cases, similarity is calculated using intensities which fall within the boundary of the moving image. Robust similarity metrics for the registration of dissimilar images are described by Nikou *et al.* [43]. Results obtained from experimental testing show that their metric compares favourably with non-robust techniques. Such metrics are commonplace in the intensity-based registration algorithms described in Section 2.2.1. A similarity metric implementation currently in use is cross correlation and its variants. Basic cross correlation between two translated images can be defined as

$$CC(F, M) = \frac{\sum_{x=1}^I \sum_{y=1}^J F(x, y)M(x-u, y-v)}{\left[ \sum_{x=1}^I \sum_{y=1}^J M^2(x-u, y-v) \right]^{\frac{1}{2}}} \quad 2.1$$

where  $F$  and  $M$  are fixed and moving image intensity functions.  $I$  and  $J$  are the number of rows and columns respectively.  $x$  and  $y$  are discrete grid co-ordinates, while  $u$  and  $v$  are the components of a transform.

On appearance of entropy, other similarity metrics quickly appeared. Although a difficult task, the registration of images captured using different sensor types is commonplace in medical imaging applications. Viola [44] suggests that for two images of differing modality, entropy or mutual information can be used as a measure of similarity. In his research, mutual information is defined as  $MI(A,B)=H(B)-H(B|A)$ . Where  $H(B)$  represents marginal entropy based on a probability distribution constructed from intensities from image  $B$ . While  $H(B|A)$  represents joint entropy based on a probability distribution constructed from intensities from images  $A$  and  $B$  respectively. Mutual information can be described as the amount of information image  $A$  contains about image  $B$ . In theory mutual information is symmetric, however, in practice this is not the case [45]. Implementation aspects of registration algorithms such as interpolation artefacts and robustness to complete miss-registration result in differences when registering image  $A$  to  $B$  and  $B$  to  $A$ . Importantly, the ability to align multi-modal images allows for the comparison of anatomical and functional data that can lead to a diagnosis which would be impossible to gain otherwise. The evaluation of eight mutual information-based similarity metrics, used for the registration of brain scans, is presented by Holden *et al.* [46].

### 2.1.4 Optimiser

To achieve the accurate alignment of images it is necessary to search for transform parameters that yield a high measure of similarity. For this to happen, an optimisation component is employed to explore a search space of allowable parameters [47]. In general, optimisation can be categorised into global and local methods. A single seed point is used to obtain optimum transform parameters in global methods. Local methods, in contrast, employ multiple seed points initialised randomly throughout the

search space. In both cases, an optimum transform is found by searching within the immediate vicinity, stopping only when neighbouring parameters are less favourable. Understandably, both methods can result in transform parameters that correspond to local rather than global optima. As a consequence, local optima are a major cause of failure during the alignment process. To counter this problem, the incorporation of local optimisation strategies within a multi-resolution framework [48] [49] has been suggested. Unsurprisingly, the use of increasingly complex transforms causes the optimisation process to take place in higher dimensional space thus increasing computational burden.

Although successful in many applications, multi-resolution approaches do not always avoid local optima. For this reason, more sophisticated optimisation strategies can be employed. Multi-algorithm techniques, for example, are increasingly used for robust optimisation problems. In work by Bolton *et al.* [50] different optimisation strategies compete in parallel to find a global optimum. Methods employed in their research include a Snyman-Fatti algorithm, a particle swarm algorithm, clustering, and the Bayesian search algorithm. Once converged, results obtained from the different algorithms are used to estimate a global optimum. The use of different optimisation strategies is based upon the observation that no single scheme consistently outperforms all others. In general, the inherent noise in a captured image manifests itself as local optima within the transform parameter search space. This underlying problem is addressed by Zagrodsky *et al.* [51] who apply the well known downhill simplex optimisation strategy, to a search space smoothed using varying degrees of intensity quantisation and hence noise suppression. Results show that improved robustness is achieved during the alignment process.



## 2.2 Classification of registration algorithms

The manner of image acquisition can be used to classify registration algorithms into four main groups.

- In multi-view analysis, where images have been acquired from different viewpoints, the aim of the alignment process is to construct a larger and more comprehensive representation of the scene captured. Work by Wolberg and Zokai [52] represents one such example. In their work a new algorithm is used to register high altitude surveillance images.
- In multi-temporal analysis [53], images of the same scene have been acquired at different times possibly under different conditions. Registration is employed to locate changes which have appeared between consecutive image acquisitions. Landscape planning, where the monitoring of natural features such as river mouths and tidal zones is used to predict land erosion, is a common application domain.
- In multi-modal analysis alignment followed by integration of information obtained from different sources, to gain a more detailed picture, is the goal. In medicine, the combining of Magnetic Resonance Imagery (MRI) and ultrasound data for the monitoring of functional and metabolic activities in the torso is one such application [54].
- In multi-form analysis, images of a scene are aligned with a model. Normally the model is a computer representation such as a digital map [55]. Applications include template matching, where the template represent an ideal, in automated visual inspection.

Understandably, image registration algorithms can also be categorised with respect to application domain and deformation complexity. In this research, classification is organised by means of similarity calculation computational burden.

### 2.2.1 Intensity-based registration

Due to the requirement that all pixels in both images contribute towards similarity calculation, intensity-based registration algorithms have a high computational cost. Also known as image correlation these methods exploit directly the matching of image intensities without any kind of structural analysis. As a consequence, they are sensitive to intensity changes caused by noise, varying degrees of illumination [56] [57], and dissimilar sensor types. Removal of reliance, upon the complex algorithms associated with the feature detection stage, is seen as a major advantage of intensity-based methods. Conveniently, the alignment process can be performed using an entire image or sub-window. One limitation of these methods originates in the geometry of the region being used. Images that differ by a translation are best suited to a rectangular window. When images are affected by more complex deformations, such as shear and scaling, rectangular windows are unsuitable as they are unlikely to encompass all parts of a scene.

Intensity-based registration has been employed in the alignment of x-ray images and biomedical volume data [58] as well as ceramic tiles for the purpose of fault detection [59]. In these applications, cross-correlation of intensities is used as a similarity metric. With such an approach, sub-pixel accuracy of alignment is achieved by interpolating intensities before similarity calculation. Traditionally, cross-correlation has been used to register translated images with only slight rotation and scaling. Today more

sophisticated versions of the algorithm which are capable of handling complex geometric deformations are now commonplace. Crucially, by employing phase correlation in the frequency domain, increased robustness to noise can be obtained via Fourier methods [60]. Phase correlation involves the calculation of a cross-power spectrum for both fixed and moving images. The location of peaks in both spectrums are then identified and aligned. As with all registration algorithms the probability exists that regions within an image, without prominent detail, will be matched incorrectly with other smooth areas due to the lack of discriminating features.

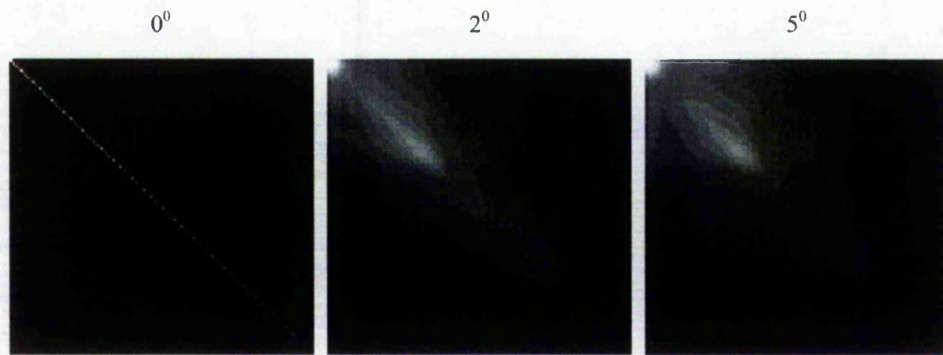
Unfortunately, intensity-based methods are limited to the alignment of images from the same modality. Although popular, many intensity-based algorithms are sensitive to the presence of outliers. These are artefacts (i.e. noise) which can appear in an image and cause biased registration. Using statistics, estimates of mean and covariance can be employed to increase robustness. Robustness is sought by weighting intensities in such a way as to reduce the effects of outliers. In some cases remove them completely. This can be achieved by computing the distance of intensities from a mean. Based on the computed distances, new intensities and a new mean are then determined. Other common statistical approaches to robustness include weighted square error and non-quadratic error. An intensity-based registration algorithm that uses a robust correlation coefficient, as a similarity metric, is introduced by Jeongtae and Fessler [61]. To evaluate the statistical properties of their approach, 2D-to-2D and 2D-to-3D registration of torso phantoms was performed. In each case, experimental results confirm an improvement in robustness to outliers.

### 2.2.2 Multi-modal registration

In multi-modal image registration applications the data to be registered stem from two different capture devices, as opposed to single-modal tasks where images are retrieved using the same sensor type. According to Woods *et al.* [62] the measure of alignment between images of differing modality can be based on the assumption that although different in value, regions of similar intensity in the fixed image will correspond to regions of similar intensity in the moving image. Also, for all pixels in corresponding regions, the ratio of their intensities should vary only slightly. As a consequence, alignment is achieved when the average variance of this ratio is minimised. Crucially, this idea can be realised through the construction of a feature space, also commonly referred to as a joint probability distribution [63]. The joint probability distribution represents a two-dimensional plot that contains combinations of intensities, taken from corresponding co-ordinates in both images. Instead of identifying regions of similar intensity directly within the images, combinations of intensities are analysed using the joint probability distribution.

During the registration process, a variation in alignment between the images causes changes in appearance of the joint probability distribution. When correctly aligned corresponding structures in both images overlap causing a clustering of intensities combinations. Misalignment, in contrast, causes structures in the fixed image to overlap with structures in the moving image which are not their counterpart. This results in the dispersal of intensity combinations within the joint probability distribution. Plum *et al.* [64] have demonstrated the effects of registering an image with itself, the results of which are illustrated in Figure 4. Based on the changing regions within the joint probability distribution, measures of dispersion which guide the registration process

have been proposed and successfully implemented by Studholme *et al.* [65]. Today, entropy which appears low when intensity combinations are dispersed and high when intensity combinations are clustered, is a well recognised and accepted method of similarity calculation.



**Figure 4: Joint probability distributions as a result of rotating the moving image by  $0^\circ$ ,  $2^\circ$ , and  $5^\circ$ . Intensity combinations disperse as the scale of misalignment between images increases.**

Viola and Wells [66] have experimented with a multi-modal registration algorithm in four application domains including medical imagery, video sequence tracking, model-to-scene, and model-to-real world imagery. Their method is based on the formulation of mutual information between a model and an image. Conveniently, no *a priori* knowledge of the relationship between the model and image intensities was required. It was also assumed that when correctly aligned mutual information was maximised. To maximise entropy and increase efficiency stochastic approximation was employed. Because few assumptions about the nature of the imaging process were made, the algorithm could be generalised and applied to a wide variety of modalities. Results obtained by the group indicate that the algorithm is more robust than standard correlation, working well in domains where edge or gradient-descent methods have

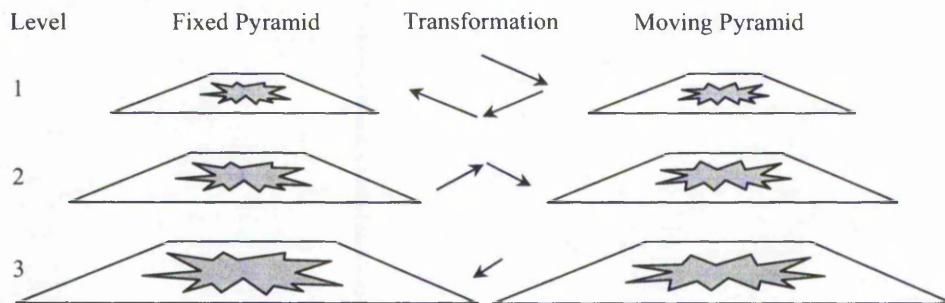
difficulty. As similarity is calculated based on a subset of intensity samples, reduced computational burden is associated with multi-modal registration applications.

### 2.2.3 Multi-resolution registration

To reduce the computational burden of image registration algorithms in general, a multi-resolution scheme can be adopted. Sub-sampling at successive levels by means of interpolation is used to create a coarse-to-fine data structure [67]. Gaussian filtering and intensity averaging have also been used to construct multi-resolution pyramids. Throughout the sub-sampling process, care is taken to maintain geometric consistency at all resolution levels. Once complete, transform parameter optimisation is performed. Because the majority of iterations are performed at the coarsest level, where the transform parameter search space is considerably reduced, a substantial saving in processing time is achieved. Figure 5 illustrates how optimisation proceeds through successive levels until full resolution images are reached. As a consequence, the registration of large scale features is achieved first and only small corrections are required at progressively finer resolutions. Dani and Chaudhuri [68] describe an early implementation in which a summing pyramid is employed, where intensities in coarser levels correspond to the summation of intensities in higher levels.

A hierarchical multi-resolution registration algorithm is presented by Thevenaz *et al.* [69]. Their approach employs a bi-level pyramid which is based on the spline representation of images in conjunction with spline processing. Spline representation is reported by the group as being well suited to the construction of image pyramids and for performing complex geometric transforms at various resolutions. Specifically, the use of a spline model at each resolution level ensures that the pyramid is internally

consistent and allows for easy calculation of exact derivatives. A global 3D transform which can be restricted to rigid-body motion including translation, rotation, and isometric scaling is employed as a deformation model. The optimisation of transform parameters is performed using a variation of the Marquardt-Levenberg algorithm. Results obtained by the group suggest that the pyramid refinement strategy is more robust than single resolution methods, being less likely to become trapped in local optima at coarser levels of resolution.



**Figure 5: The multi-resolution pyramid scheme. As the majority of iterations are performed at the top level, where the search space is small, increased processing speed can be achieved.**

Due to their inherent multi-resolution characteristics, wavelet decomposition for pyramid construction has also been considered in a number of research projects [70] [71]. Importantly, the set of wavelet coefficients, derived for expressing successive degrees of compression, can be employed as the levels in a pyramid. This is achieved by repeatedly filtering the rows and columns of an image using both low and high-pass filters; resulting in the decomposition of an image into four coefficient sets. Turcajova and Kautsky [72] have evaluated the effectiveness of orthogonal and biorthogonal wavelets to register images using an affine transform. Image pyramids, where each level



is one quarter of the size of the previous level due to wavelet compression, were constructed using a fast separable filtering algorithm. The levels consisted of high vertical and horizontal frequency sub-images which capture vertical and horizontal edges respectively. An image containing frequencies in both directions, associated with corners, also formed a level in the pyramids.

### 2.2.4 Landmark-based registration

Landmark-based registration techniques have been employed in the alignment of low-resolution biomedical data [73], the registration of images created by different modalities as well as the alignment of 2D scans with 3D surface models [74]. To be successful, characteristics used as landmarks need to be part of the scene in both images as well as possess high levels of similarity. Features selected as landmarks can be extrinsic, such as visible and accurately detectable artificial markers placed in a scene before data capture. Or intrinsic, for instance salient features contained within an image itself [75]. Intrinsic landmarks commonly used include anatomical structures, centres of mass, and outlying pixels. Logically, a major factor determining the precision of landmark-based registration methods is the accuracy of landmark selection. The number of landmarks used and their relative placement also need to be considered. These underlying problems can be seen as drawbacks which have historically limited the use of landmark-based registration. Such limitations are particularly problematic in multi-modal registration applications where corresponding landmarks are unrecognisable due to differing imaging modalities.

Traditionally, landmark selection has been achieved through manual user intervention. During automated selection, a fixed landmark cloud is determined from salient features



extracted using segmentation. A search is then performed to identify the same features in the moving image [76]. A popular solution to the problem of landmark cloud registration is the iterative closest point algorithm [77]. The algorithm is a descent procedure which seeks to minimise the sum of squared distances between corresponding landmarks. As with all descent-based techniques, the algorithm requires a good initial estimate in order for it to converge. As a consequence, it is assumed that both landmark clouds are approximately aligned and each landmark in the fixed cloud has correspondence with the closest landmark in the moving cloud. It is also assumed that all fixed landmarks have a moving equivalent. A subset from each cloud is employed if there are uneven numbers of landmarks. The goal of the iterative closest point algorithm can be stated more formally using

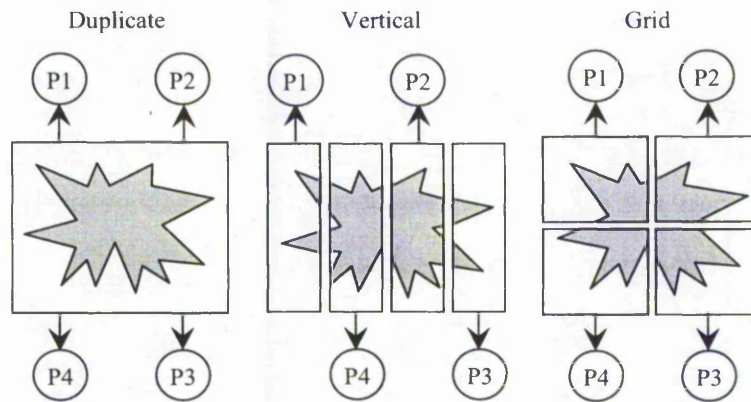
$$\min \sum_{i=1}^N \left\| F_i - (RM_i + T) \right\|^2 \quad 2.2$$

where  $F_i$  is the  $i^{\text{th}}$  landmark of a fixed landmark cloud and  $M_i$  is the  $i^{\text{th}}$  landmark of a moving landmark cloud.  $R$  is a rotation matrix,  $T$  is translation vector, and  $N$  represents the number of landmarks considered. An adaptation called the iterative closest points using invariant features algorithm is introduced by Sharp *et al.* [78]. They investigate the use of Euclidean invariant features during the registration of range images. Contained within the range data, three different invariant features including curvature, moments, and spherical harmonics are identified. The use of these features as landmarks is reported to provide robustness to orthogonal translations and 3D rotations. As the landmarks are extracted directly from sensed data, only invariance to 3D camera motion was required and neither scale nor perspective invariance were considered. Experimental results obtained suggest that the use of invariant features reduces the

probability of being trapped in local optima. Also, under noise-free conditions, convergence of the algorithm occurs in less iterations. Understandably, landmark-based techniques represent the lowest computational burden of the algorithms surveyed. This is because the evaluation of high numbers of pixel co-ordinates, common to intensity-based registration approaches, is avoided.

### **2.3 Parallel processing**

As previously discussed, high image resolutions coupled with complex algorithms are increasing the demand for high speed processing capabilities. In a shared-memory computer, all processors share the same main memory and can work on the same data concurrently. As a consequence, this type of hardware largely eliminates the need for explicit message passing between concurrent tasks [79]. This can be seen as an important advantage over a cluster of independent workstations. Built for multi-processor architectures, multi-threaded programming allows an application to branch into independent and potentially concurrent threads. Multi-threaded applications are suited to multi-processor architectures because the individual threads can run concurrently on all available processors. Support for multi-threaded programming is available with almost all operating system and programming environments. As multi-threaded applications share the same address space, they cause considerably less overheads than the creation of an equivalent number of processes. According to Lewis and Berg [80], although a minimal amount of cost is associated with the creation and handling of multiple threads, the performance gain sought must outweigh all overheads in order for the approach to be useful.



**Figure 6: Example image distribution schemes required for a loosely-coupled architecture. Once an image has been divided into segments, each segment is assigned to a unique processor.**

The processing of an image within a loosely-coupled architecture typically consists of four main steps including image distribution, local processing, data transfer during processing, and segment accumulation [81]. Distribution is the process of dividing an image into segments each of which is then assigned to a unique processor. Figure 6 illustrates how under a duplicate distribution scheme, each processor is sent an exact copy of the original image. Unsurprisingly, this represents the simplest approach. In the vertical distribution scheme shown an image is divided into vertical segments before being assigned to individual processors. Alternatively, a more complex technique can be adopted where an image is divided into a variable sized matrix of segments. Nicolescu and Jonker [82] suggest that pre-processing of an image is sometimes required, in order to make distribution more suitable for the host architecture. Once distributed, each processor applies local processing to the segment allocated to it. When data allocated to other processors are required, it can be transferred by inter-processor communication.

Finally, after application of the algorithm, distributed segments are accumulated into a resulting image.

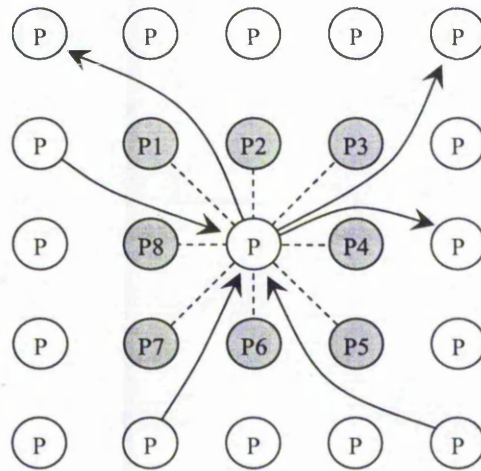
An important consideration is the inter-processor communication required when data allocated to other processors are called for. Communication can be categorised into groups depending on their pattern of access [83]. These patterns of access also represent a strategy for synchronisation between communicating processors. One-to-one access is common in such functionality as subtraction and multiplication, where the intensity of an output pixel maps directly to a corresponding input pixel. Alternatively, one-to-many relationships exist in neighbourhood operators. In such cases, the intensity of an output pixel is based on a function of the input pixel's immediate neighbourhood. When access between input and output pixels is erratic, a global communication pattern is normally required. Flood filling algorithms have this kind of pattern of access. The importance of intensity data stored non-contiguously in memory is highlighted by Seinstra *et al.* [84]. Unsurprisingly, the handling and transmission of non-contiguous data differ from data stored as a single block. In general, data stored randomly in memory cause additional overheads due to the packing of data into a contiguous buffer before transmission.

### 2.3.1 Related image registration work

In the context of parallel processing, intensity-based image registration has been achieved by Warfield *et al.* [85] who introduced a non-rigid algorithm based on the work-pile paradigm. Their goal has been to develop a high performance inter-patient registration algorithm that can be applied without operator intervention, to a database of several hundred scans. In an initial step, each scan is segmented using a statistical classification method. This pre-processing stage is used to identify different tissue types

including skin, white matter, grey matter, bone structure, and background. Once segmented, a transformation which brings these features into alignment is calculated. In the system described, a message passing interface and cluster of symmetric multi-processors execute parallel image re-sampling and similarity calculation operations using multiple threads. Work is dynamically load balanced across a cluster of computers, each containing eight 167 MHz processors and 2048 MB of RAM. Results obtained by the group show that successful registration of  $256 \times 256 \times 52$  brain scans has been achieved in approximately 10 minutes.

Two non-threaded approaches, Single Instruction Multiple Data (SIMD) and Multiple Instruction Multiple Data (MIMD), have been compared by Christensen [86]. His work presents implementation issues and timing analysis for the registration of  $32 \times 32 \times 25$ ,  $64 \times 64 \times 50$ , and  $128 \times 128 \times 100$  volume datasets. During each clock cycle, the SIMD implementation performs calculations in which all processors are performing the same operation. The MIMD implementation, in contrast, breaks an algorithm into independent parts, all of which are solved simultaneously by separate processors. The movement of data in both shared-memory systems is unrestricted and during execution each processor has access to the whole memory. The main performance bottleneck associated with both approaches was reported as scalability of hardware with increasing numbers of processors. Figure 7 shows how, due to the SIMD architecture's mesh interconnections, the need for a router reduces performance when communications are not part of an immediate neighbourhood. Interestingly, the SIMD implementation is reported as being four times slower than its MIMD counterpart. Reduced performance being caused by overheads incurred during serial portions of the algorithm.



**Figure 7: An example of the random access pattern of communication. Such patterns of communication are inefficient when communication is not within the immediate neighbourhood.**

Due to the increased number of dimensions, the registration of volumes is considered a computationally intensive task. Further demands are placed on a registration algorithm when alignment of deformable structures in 3D space is required. Salomon *et al.* [87] introduce deformable registration of volumes which involves optimisation of several thousand parameters and requires several hours processing time on a standard workstation. Based on the simulation of stochastic differential equations and using simulated annealing, a parallel approach that yields processing times compatible with clinical routines is presented. Clinically compatible times are reported by the group as being approximately 30 minutes in duration. Importantly, the implementation employs a hierarchical displacement vector field which is estimated by means of an energy function. The energy function is scaled in relation to the similarity between volumes and is re-evaluated at the end of each transform parameter optimisation cycle. In general, the

algorithm is suited to massively parallel environments and has been successfully applied to the registration of two  $256 \times 256 \times 256$  volumes taking approximately 40 minutes.

Although these projects demonstrate the performance benefits which can be achieved with such architectures, they lack data distribution capabilities. As a consequence, each processor is required to hold complete images during the registration process. This limitation strictly bounds the size of images to the smallest memory size within the architecture. In order to simultaneously overcome the limitations of memory and performance, a data distributed parallel algorithm has been developed by Ino *et al.* [88]. Based on Schnabel's implementation, the algorithm performs multi-modal volume registration using adaptive mesh refinement which requires no user interaction or pre-processing stage. The data distribution scheme employed allows for increased volume size and is achieved by assigning partitioned volume segments to all available processors. Efficiency of the algorithm is improved through the inclusion of load balancing which manages the computational cost associated with each volume segment. Experimental results obtained on a 128 processor cluster show that volumes  $1024 \times 1024 \times 590$  voxels in size can be aligned in minutes rather than hours.

### **2.4 Distributed artificial intelligence**

Artificial intelligence (AI) can be described as a broad subject with branches in philosophy, mathematics, and computer science [89]. Neural networks, genetic algorithms, and knowledge-based systems are just a few techniques which have been realised in algorithm form. Each method is suited to a different class of problem. For example, neural networks are apt for pattern classification tasks whereas genetic algorithms are suited to optimisation problems. In general, automation of both mundane



and expert tasks is the goal of intelligent systems. A mundane task may simply represent a sequence of operations in order to achieve a simple goal. An expert task, in contrast, might be a medical diagnosis which requires specialised skill and knowledge. Initial knowledge-based approaches could perform simple tasks where the answer to a problem consisted of the exploration of a large number of solutions. More focused tasks which require carefully acquired expert knowledge are currently in use. According to Wen and Tao [90], in recent years AI has evolved from small scale research projects into programs that predict weather patterns and control manufacturing processes.

### 2.4.1 Multi-agent systems

Suited to distributed implementation, multi-agent systems offer the possibility of directly representing the individual components of an intelligent system; their autonomous behaviour and interactions with other system components [91]. Specialist behaviour encapsulated within an agent gives it the ability to adapt, interact, and evolve within the environment in which it exists. In order for an agent to adapt, it can receive information through the use of sensors. The agent then makes decisions based upon memory, internal state, and previous experiences. If multiple agents are hosted on independent computers within the same network, communications coupled with a control module are used to achieve interaction and collaboration. According to Ferber [92], synchronous communication requires an agent to suspend communications until all processing tasks are complete. Agents which communicate and process tasks at the same time, in contrast, are operating in an asynchronous environment. At a global level, the propagation of messages to all agents is achieved by means of a broadcast mechanism. With care, limitations in the design of an agent can be overcome through the use of a specific communication protocol.

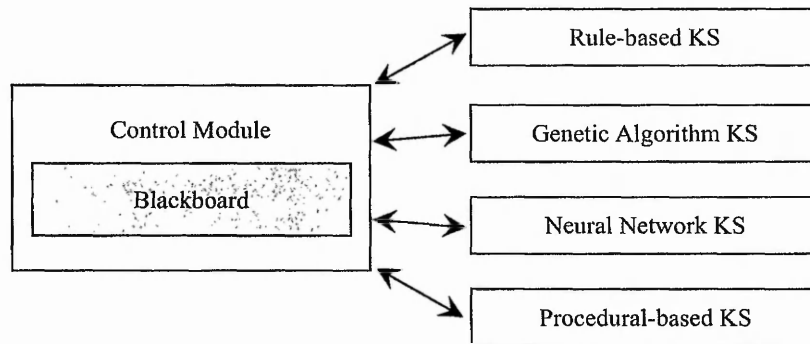


Reactive agents, as described by Harrovet *et al.* [93], are utilised in a parallel processing environment where decisions are made based on simple behaviour. Their multi-agent approach is employed in the detection of homogeneous features in natural objects. Despite the numerous advantages of a parallel implementation and the co-operation of agents in solving a common task, a number of shortcomings persist. Typically, the overheads caused by communication and agent management have a negative impact on performance. Reducing the complexity of an agent by equipping it with the minimum capabilities and increasing network speed through the use of optimised hardware, are simple methods suggested for overcoming some of these limitations. In a different multi-agent system, Lueckenhaus and Eckstein [94] present a three layered architecture for the automatic parallelisation of image analysis tasks. In an initial step, an image processing algorithm is selected for parallelisation. Once selected, execution of the algorithm is carried out concurrently by multiple agents. Management of the overall system is achieved through the collaborative coordination of agent activities.

### 2.4.2 Blackboard architectures

Conveniently, blackboard architectures have emerged as a suitable host for multi-agent implementations [95]. A blackboard architecture is based on the analogy of a group of experts working together to solve a common problem, by writing their ideas onto a shared blackboard. In general, these architectures consist of three distinct components including a blackboard, expert or knowledge source (KS) modules, and a control unit. The blackboard represents an area of shared-memory where KSs can store and retrieve information. As illustrated in Figure 8, KSs can be implemented as rule, genetic algorithm, neural network, or procedural-based modules. Also known as the scheduler,

the control unit monitors changes on the blackboard and is used to determine a focus of attention. The focus of attention can be described as the selection of a specialised KS to be activated or the choice of a solution to pursue. In a distributed blackboard architecture [96], each component can be executed in parallel on separate processors or as separate threads on the same processor. According to Jiang *et al.* [97] division of the blackboard into panels that correspond to levels of analysis, simplifies management of a solution during its evolution.



**Figure 8: The components of a blackboard architecture. A blackboard permits KS modules to be implemented using different processing styles.**

The first blackboard implementation, the Hearsay-II speech understanding system [98], was developed in 1976 at the Carnegie-Mellon University. Hearsay-II was developed as a model that would overcome the limitation of speech-recognition previously encountered in less successful applications. In particular, it was designed as the first system to incorporate context, syntax, semantics, and phonological rules for the recognition of speech. Experience gained with Hearsay-II led to the production of more generalised blackboard architectures. Attempt to GEneralise (AGE) [99] is one such implementation. The goal of AGE was the construction of a collection of building block programs encapsulated as KS, to produce a knowledge engineering library. It was

envisaged that AGE would speedup the building of knowledge-based system by standardising a core selection of AI techniques. Significantly, each building block of the library possessed an intelligent front-end while the packaging of AI methods was designed to remove the need for re-program to suit specific problems. Unsurprisingly, these first blackboard architectures were limited in their flexibility of application.

Based on a blackboard architecture, production rules coupled with object-oriented programming and hybrid knowledge representation schemes have been used to convey engineering and design expertise contained within a knowledge-base [100]. By exploiting specialised processing techniques a user is supplied with standard techniques, load specifications, and professional judgement on liquid retaining structures. Seo and Cho [101], in contrast, employ a blackboard architecture to ensure network security. Detection of network intrusion is made possible, via the blackboard, through the sharing and co-ordination of information between network resources. When the presence of foreign packets is detected, attacker information is send to the firewall in order to prevent future damage in the network. In other research, the shortcomings in modularity that are common to architectures found in autonomous robots have been addressed [102] by Xu and Van Brussel. The availability of system data provided by their blackboard implementation allows new behaviours to be integrated, without the modification of existing modules. The approach thus overcomes the traditionally inflexibility caused by interdependency of system components.

### **2.5 Summary**

Although huge advances in hardware have been made, the registration of image and volume data continues to be limited by memory and speed constraints of conventional

computers. In order to obtain high execution speeds many registration algorithms have been proposed which employ sparse landmark clouds. Typically, landmark-based approaches are employed when the information contained within intensities is less significant than the information contained within the scene itself. Although landmark-based approaches allow the registration of images which are completely different in nature, a common drawback of these methods is that corresponding landmarks can be hard to detect automatically and may change over time. The manual selection of landmarks may also lead to unacceptably long intervention times and poor overall alignment accuracy. As a consequence, it is likely that high-performance registration applications employed in medicine and manufacturing will continue to use intensity-based algorithms hosted in distributed processing environments.

In general, intensity-based registration algorithms are best suited to data that does not contain high levels of prominent detail and where distinguishing information is contained within pixel intensities. A major advantage of intensity-based algorithms is that there is no reliance upon the complex algorithms associated with the feature detection stage in landmark-based registration. Although commonly employed in alignment of data from the same modality, the development of mutual information as a similarity measure has successfully extended intensity-based methods to multi-modal applications. Robustness to noise is seen as a major limitation of intensity-based registration algorithms. Also, large translations, rotations, and scaling are possible but render the algorithm obsolete due to the high computational burden associated with their evaluation. Crucially, while reduced computational expense can be achieved by employing a multi-resolution strategy, this usually means a trade-off between robustness, reliability, and accuracy of alignment.

An important difference between shared and distributed memory architectures is how a registration algorithm is parallelised. For shared-memory computers, the operating system treats all processors as equal and uses the shared environment for processor communication and synchronisation. In such architectures an algorithm is partitioned into sub-algorithmic procedures which are distributed between all available processors. As each processor has access to the whole memory, data movement between processors is straightforward. Due to the need of specialised hardware, the main limitation of these architectures is their inability to easily scale up to large numbers of processors. Distributed memory systems, in contrast, only require methods that allow data to be partitioned and moved between processors when called for. Conveniently, by adopting a coarse-grained parallelism approach the issues of fine-grained parallelism can be ignored. Understandably, the key to parallelising a registration algorithm is to maximise the distribution of workload in relation to introduced overheads. Overheads can be described as the time taken to create slave processes and management of these processes.

Both multi-agent systems and blackboard architectures have been employed as distributed processing networks and can be considered as parallel processing architectures. As blackboard architectures encompass a message passing interface, it is not uncommon for multi-agent systems to be implemented within such environments. As previously discussed, the basic blackboard architecture is constructed from three distinct components including a blackboard, KS modules, and a control unit. In a distributed blackboard implementation each component can be executed in parallel or as separate threads on the same processor. Although maximum parallel performance

cannot be attained by such architectures, due to inherent communication and control overheads, high performance processing capabilities can be achieved. Conveniently, the modular nature of distributed blackboard architectures is well suited to the parallel processing of image and volume data, while at the same time maintaining scalability of implementation.

### 3 iDARBS – A distributed image processing framework

The background of image registration, parallel processing, and distributed artificial intelligence has been discussed in Chapter 2. The basic components of an image registration algorithm were divided into four groups. Registration algorithms were then classified as being either intensity, multi-modal, multi-resolution, or landmark-based. To clarify differences in architecture data distributed, shared-memory, and patterns of access were highlighted in a discussion about parallel processing. High performance intensity-based registration algorithms were also identified and described in detail. From the survey it was clear that the majority of high performance intensity-based registration applications developed [26] [86] [87] [103] [104] [105], employed specialised architectures found predominantly in the research environment. Furthermore, due to the fine-grained parallelism employed, the integration of multiple similarity calculation strategies has been overlooked. As demonstrated by Warfield *et al.* [85] and Ino *et al.* [88], the surveyed literature also makes clear that coarse-grained parallelism can be used to achieve increased performance.

In this chapter, a distributed blackboard architecture is combined with distribution and accumulation mechanisms to form a framework in which image registration algorithms can be hosted. In contrast to surveyed distributed processing architectures, iDARBS (imaging Distributed Algorithmic and Rule-based Blackboard System) is based on a worker/manager model and provides a framework that can reside on a network connected by the TCP/IP communication protocol. Section 3.1 of this chapter discusses the selection of a suitable blackboard architecture. An overview of iDARBS including modifications and addition to the original implementation are given in Section 3.2. The handling of image data and the organisation of control information held on the

blackboard are discussed in Section 3.3. In Section 3.4 the behaviour of iDARBS components for the distribution, processing, and accumulation of image segments is explained in detail. The chapter concludes with experimental testing in Section 3.5, conclusions in Section 3.6, and a short summary in Section 3.7.

Using images obtained from the field of manufacturing the scalable nature of the iDARBS framework is demonstrated. Results presented confirm the ability of iDARBS to host image processing algorithms of varying complexity. Based on comparisons with sequential implementations, the algorithms distributed show that significant speed increases can be achieved. The speedups reported are a strong indication of the suitability of iDARBS for the hosting of image registration algorithms.

### **3.1 Blackboard selection**

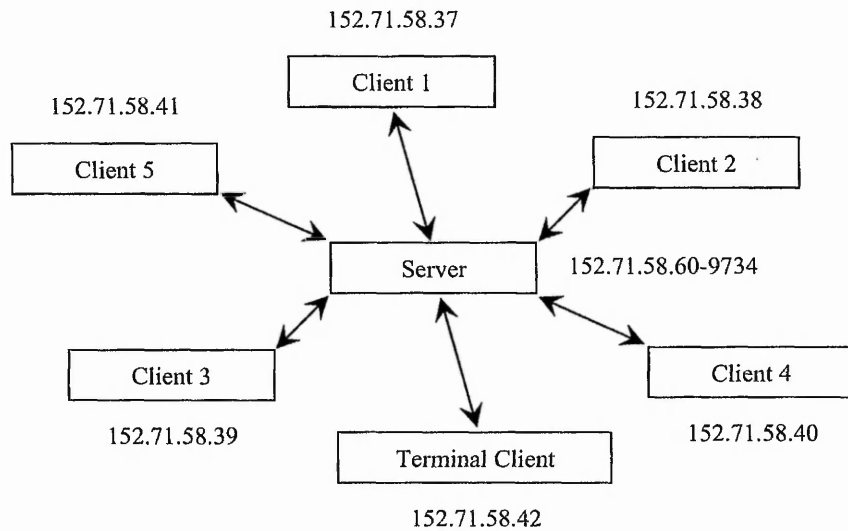
A variety of blackboard architectures were considered at the onset of this project. Originally invented by Barbara Hayes-Roth in 1983, BB1 is a software package that embodies a traditional blackboard architecture [106]. Features developed as part of the BB1 software include uniform control knowledge representation, event-based triggering of operations, and multi-threading capabilities. BB1 is currently open source and freely available from the Knowledge Systems Laboratory website hosted at Stanford University. Unfortunately, as of June 2003, continued development of BB1 was halted and its maintenance by the university given a low priority. As a consequence, BB1 was deemed unsuitable for adoption for this research project. GBBopen [107], in contrast, is an open source blackboard development environment employed in a number of research projects. The GBBopen environment is based on the concepts refined by the UMass generic blackboard architecture and GBB commercial products [108]. At an



implementation level, GBBopen represents an extension of Common Lisp. Although well suited to the requirements of this project, Lisp's limited compatibility with third party image processing libraries, made its selection unsuitable.

Written in C++ and developed in-house at the Nottingham Trent and Open Universities, DARBS (Distributed Algorithmic and Rule-based Blackboard System) [109] is a distributed blackboard architecture based on a client/server model. In DARBS, the server functions as a blackboard while knowledge sources (KSs) are implemented as client modules. The distributed nature of the approach means that both KS and blackboard modules run as separate processes. These independent processes may then reside on any computer in a network connected by the TCP/IP communication protocol. At an implementation level, each KS is required to connect with the blackboard. Once connected, a KS sends commands and waits for a response from the blackboard. Whenever a KS changes the content of the blackboard, a message is broadcast informing that the blackboard has changed. Individual KSs then react to the changes depending on their implemented behaviour. Logical communication and interaction of KSs with the blackboard is achieved through the division of information into partitions.

The client/server model, on which DARBS is based, is illustrated in Figure 9. Each KS requires the IP address and port number of the blackboard in order for it to establish connection. In the example shown, the different IP addresses of the KS modules indicate that they are hosted on independent computers within a network. Conveniently, DARBS contains a terminal client. The terminal client connects to the blackboard in the same way as a KS and represents a powerful debugging tool.



**Figure 9: The client/server model employed by DARBS. Because client modules can be hosted on separate processors, multiple clients can be serviced concurrently.**

Due to its in-house development, the availability of source code meant modifications could be made to suit the need of this research project. Access to developers of the DARBS architecture was also considered of great importance, as they could assist with the understanding of functionality provided by the blackboard implementation. It is believed that this project would have been severely restricted through the adoption of a commercially available blackboard architecture, where cost, source code availability, and copyright issues become limiting factors. For these reasons DARBS was chosen for this research project.

### 3.1.1 Background on DARBS

Proven as an effective framework for multi-agent implementations, the predecessor of DARBS, ARBS (Algorithmic and Rule-based Blackboard System) has been used to

demonstrate co-ordinated problem solving through established techniques. In work by Li *et al.* [110], Shifting Matrix Management has been implemented and compared with two other standard models, Contract Nets and Cooperative Problem-Solving. Inspired by organisational structures, Shifting Matrix Management is a model of agent co-ordination and co-operation that permits temporary lines of authority which reflect the shifting function of a flexible workforce [111]. Comparisons of the models included the control of two autonomous robots. The ARBS Shifting Matrix Management implementation is reported as out-perform the other two models in terms of the number of tasks completed and the task completion rate.

The ARBS architecture discussed has also been successfully employed in the control of a plasma processing unit [112]. In general, the unit is used for depositing coatings on the surface of electronic and mechanical components. Traditional control methods, which rely on well-defined models of manufacturing processes, are ill-suited for the control of plasma processing unit variables. This is because the multiple variables employed are often difficult to model, being interdependent and non-linear in nature. Using crisp and fuzzy rules within a single ARBS KS, control over the depositing process was successfully made possible through the automatic adjustment of pressure, electrical power, and gas flow parameters.

More recently, the DARBS implementation has been employed by Choy *et al.* [113] to investigate the performance of distributed blackboard architectures using a well established test-bed called TileWorld. The test-bed was deemed a suitable application due to it being a distributed problem which is naturally scalable. TileWorld represents a 2D grid environment that contains agents, holes, obstacles, and tiles. Embodied as KSs,

the objective of the agents was to score points whilst moving within the TileWorld environment. Points were scored by picking up tiles and placing them into holes. Significantly, the agents had only a limited view of the TileWorld environment and could not move into a position blocked by an obstacle. Results obtained through testing of the TileWorld implementation, on a single processor, suggest that slow down is approximately exponential as the number of agents increases. Slowing of the blackboard is reportedly caused by time-slicing between processes. Testing also highlighted a small slowdown caused by communications overheads and access contention between agents for blackboard resources [114].

### 3.1.2 Basic DARBS commands

Each DARBS KS represents a structure in which rules and algorithms can be embodied. The basic structure of a rule is where `DARBS_command` is an instruction to be followed by the blackboard, `[pattern]` is an information string to be manipulated, and `[partition]` is the name of the blackboard partition affected by the command. Some simple commands for the addition, query, and removal of information from the blackboard follow:

- `add` appends information to a specified partition. `Okay` is returned to the calling KS on success of the command and `partition changed!` is broadcast to all other KSs.
- `on_partition` is used to check if information is contained within a specific partition. On finding the information `true` is returned otherwise `false`.
- `not_on_partition` checks if information is absent from a specific partition. On finding the information `true` is returned otherwise `false`.

- To remove all information from the blackboard, including partitions, the command `clr_board` is provided. On success, `okay` is returned to the calling KS and `partition changed!` is broadcast to all other KSs.

Commands for the editing of information include `replace` and `replace_multi`.

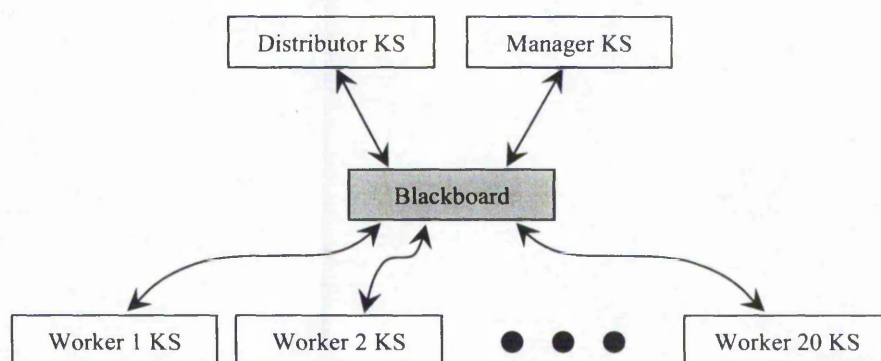
- The command `replace` is used to substitute information in a single partition. Upon success `okay` is returned to the calling KS and `partition changed!` is broadcast to all other KSs. `no match found!` is returned otherwise.
- `replace_multi` substitutes information in multiple partitions using a single instruction. `Okay` is returned to the calling KS on success of the substitutions and `partition changed!` is broadcast to all other KSs. `no match found!` is returned when information cannot be found or `partition not found!` when at least one partition is not found.

In the event that a command fails, a debug message is returned to the calling KS. DARBS also provides standard commands within its source code. For example, `run_algorithm` provides a mechanism for the embedding of shared library algorithms in KS rule files. The result of `run_algorithm` can be returned to a KS if required. `error_running_algorithm!`, in contrast, is printed to a debug window upon failure of the command.

### 3.2 The iDARBS Implementation

iDARBS (imaging DARBS), the underlying framework on which this research is based, consists of Distributor, Worker, and Manager KS types. Initialisation, image selection,

and distribution are performed by the Distributor KS. To simplify design, the selected image is divided vertically and horizontally into segments. Once divided, the edges of a segment are assigned a border. Each segment is then placed on the blackboard in compressed form. When activated Worker KSs retrieve segments from the blackboard, the retrieved segments are decompressed and an image processing operation is performed. Processed segments are then compressed and returned to the blackboard. During the processing of segments, co-ordination of Worker KSs activities is achieved by means of the Manager KS. Finally, processed segments retrieved from the blackboard are decompressed and appended to a resulting image.



**Figure 10: The worker/manager model on which iDARBS is based. Worker KSs perform concurrent processing of image segments while the Manager KS co-ordinates Worker KS activities.**

The worker/manager model employed by the iDARBS framework is shown in Figure 10. The realisation of Distributor, Worker, and Manager KSs components means equal access to segments and image processing operator parameters is possible. As a consequence, the concurrent processing of image segments allows the speed benefits of parallel processing to be realised. Co-ordination of Worker KSs activities, by the

Manager KS, is achieved by means of reactive behaviour and communication via the blackboard. Such behaviour removes the need for a dedicated control module and associated overheads. Understandably, the iDARBS framework allows external functionality to be called by a KS. Alternately, additional functionality can be compiled as part of the iDARBS source code.

### 3.2.1 Modifications and additions to the DARBS architecture

While DARBS provided the distributed client/server model on which iDARBS is based, no image handling capabilities were provided. Some fundamental modifications and additions to DARBS were therefore required.

#### 3.2.1.1 Image storage on the blackboard

The DARBS implementation stores information in the form of Standard Template Library (STL) [115] strings. In general, this type can be thought of as a dynamically resizable list of characters. Unfortunately, the string type causes a considerable communication overhead when processing images. This is because pixel data needs to be formatted into the string type before transmissions and storage upon the blackboard. Pixel data retrieved from the blackboard then needs to be formatted into the raw type before image processing operations can be performed. To remove the need for formatting, the iDARBS blackboard and KSs modules have been modified to handle the raw data type.

Originally, messages handled by DARBS were sent to the blackboard as type char. Before being added to a partition, arriving messages were accumulated in a buffer of type string. As the buffer represented a dynamically resizable list of type char, no loss





identifier associated with the pixel data is added to a partition called *Image container*. This keeps the blackboard consistent with stored pixel data. If an image requires retrieval, the unique identifier contained within the message header is used to search the raw data container. Once an image has been sent to a calling KS, its key is deleted from the raw data container and its unique identifier is removed from the *Image container* partition. Importantly, because no formatting of pixel data into the string type occurs, overheads are significantly reduced.

### 3.2.1.2 Image compression

Compression and decompression of raw data in order to reduce the length of large messages is provided. The zlib library [117] draws upon two well known strategies. Data is first compressed using the LZ77 algorithm and then recompressed with Huffman encoding. Also known as the sliding window algorithm, LZ77 [118] achieves compression by replacing portions of data with references to matching data which has already passed through the encoder. The references consist of a pair of values corresponding to the position of the portion of data in the previously-seen window and the length of the portion of data. Huffman encoding [119], in contrast, is an entropy-based algorithm which employs a code table for encoding source symbols. Realised as a tree, the code table is derived based on the estimated probability of occurrence for each possible source symbol.

To improve upon zlib, a fast adaptive lossless image compression algorithm called SFALIC [120] is also provided. SFALIC uses a simple linear prediction modelling method to achieve high compression speeds. Confirmed in results reported by the author, SFALIC provides increased efficiency for large images, when compared with

non-optimised algorithms such as Huffman coding. Understandably, compression and decompression of image segments using SFALIC is achieved by means of dynamically linked functionality. To overcome the limitations of buffer capacity encountered during the compression of large images, repeated calls to both zlib and SFALIC algorithms are made automatically.

### 3.2.1.3 The image viewer

To view selected images and the results of an image processing operation a simple image viewer has been added to the iDARBS framework using the Qt library [121]. Qt is an open source, platform independent, C++ development kit for the easy and efficient creation of graphical user interfaces. An image viewer is created during initialisation of each KS and can be shown or hidden at any time.

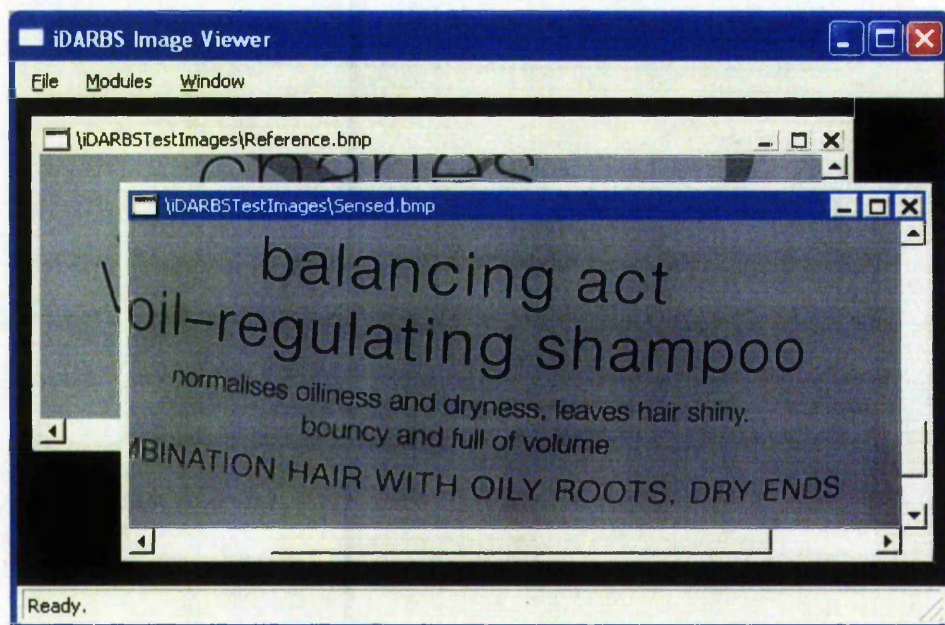


Figure 12: The simple image viewer attached to iDARBS KSs. The viewer allows visual inspection of selected images and the results of an image processing operation.

In Figure 12, a screenshot of the image viewer is provided. Images are loaded into the viewer by means of a standard open file dialog box or by functionality embedded in KS rule files. The image viewer is designed to be a resizable multiple-document-interface with menu bar. Addition of the menu bar allows the viewer to be used as a test-bed in which new functionality can be developed.

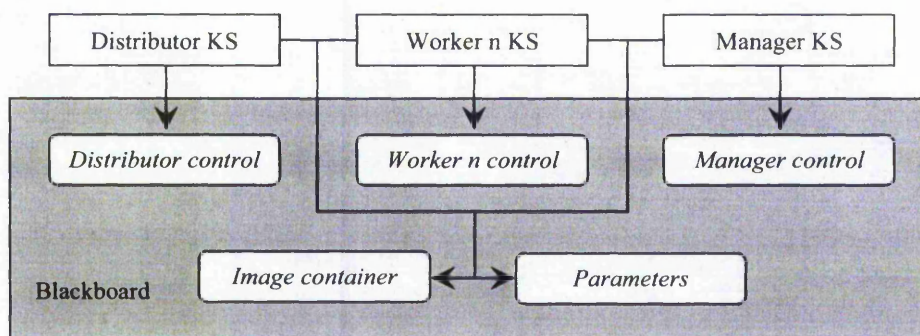
### **3.3 Partitioning of information on the blackboard**

Partitioning of information upon the blackboard aids design of the iDARBS framework by introducing structure to an area of shared-memory. Whenever a partition is changed, all KSs working with information contained within the partition are restarted. As a consequence, the logical organisation of information is used to control the number of partitions from which a KS draws. Due to the exhaustive search required, a drop in performance is expected with a single-partition implementation. Similar inefficiency will be encountered through management and processing of excess partitions. To promote the efficient processing of KS queries and hence increase performance, careful consideration has been given to the format of information stored on the blackboard. In general, short concise names and data have been adopted. Understandably, the use of human readable strings assists in the debugging of errors and permits easy modifications in the future.

As shown in Figure 13, the iDARBS blackboard is divided into the following partitions:

- A *Distributor control* partition controls initialisation of the framework and division of an image into segments.

- The *Worker n control* partitions are used to host processing of segments and allow communication between Manager and Worker KSs.
- Supervision of Worker KSs activities is achieved by means of a *Manager control* partition.
- The *Image container* partition holds the unique identifiers of unprocessed and processed image segments held in the raw data container.
- System variables used by Distributor, Worker, and Manager KSs are maintained in the *Parameters* partition.

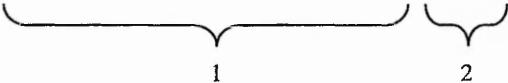


**Figure 13: Blackboard partitions and KS interdependence used to balance communication and processing workloads. By restricting access to partitions, the number of KS restarts is kept to a minimum.**

Logically, the contents of each partition changes as Worker and Manager KSs interact and concurrently process image segments. For example, a conformation flag is added to the *Worker n control* partition by the Worker n KS after initialisation has occurred. Image processing operator parameters are then added to the partition by the Manager KS as it co-ordinates the start of Worker n KS activities. Once a segment has been retrieved, processed, and returned to the blackboard the image processing operator

parameters are removed from the *Worker n control* partition by the Worker n KS. By removing or replacing information, the search of the *Worker n control* partition is kept to an absolute minimum. Importantly, because the current state of the Worker n KS is held on the blackboard, debugging in the event of failure is also made easier. This is because the current information contained within the *Worker n control* partition will indicate where an error has occurred.

```
[ImageSize 1800_1700]
[NumberOfSegments 1_10]
[BorderSize 25]
[LocalHistogramEqualisation 30_30]
```



- 1 Identifying tag.
- 2 List of parameters associated with an image processing operation.

**Figure 14: Example information strings. An information string consists of an identifying tag followed by underscore delimited list of numbers.**

Example information strings are illustrated in Figure 14. As can be seen information is encoded as an identifying tag followed by an underscore delimited list of numbers. Crucially, the length of a string is unrestricted as is the number of components into which a string can be divided. The order of information is also of importance. In anticipation of more complex tasks, information is arranged in frequency of use; the most regularly used appearing first. The ordering of information within a string allows for the fast and efficient query of information by interested KSs.

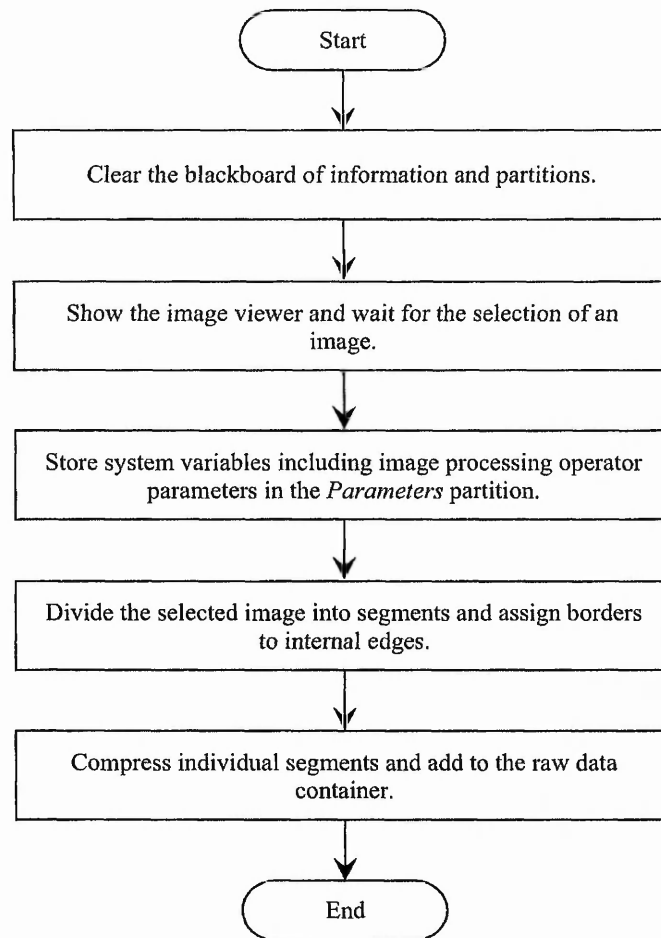


### 3.4 KS Behaviour

As previously stated, iDARBS consists of Distributor, Worker and Manager KS types. Each type is implemented as a rule-based KS. In the following section the behaviour of each KS is explained in detail.

#### 3.4.1 The Distributor KS

Tasks performed by the *Initialise\_Distributor* rule include clearance of all information from the blackboard. A segment list which is locally maintained by the Distributor KS is also initialised in preparation for new image data. On firing of *Select\_Image*, selection of an image is manually performed after the image viewer has been shown. Conveniently, the viewer allows selected images to be previewed before selection for distribution and processing. System variables are then added to the *Parameters* partition. The variables include the size of the selected image, the number of segments into which the selected image is divided, the size of border between segments, and image processing operator parameters. Next, the selected image is divided into segments. The *Store\_Segments* rule causes the resulting segments to be compressed, accumulated in the local segment list, and then sent to the raw data container. On successful storage, the unique identifier given to each segment automatically appears in the *Image container* partition. As the Distributor KS is the only KS connected to the blackboard, no restarting of Worker and Manager KSs occurs. A flow diagram of the Distributor KS, showing the initial setup of the iDARBS framework, is given in Figure 15.

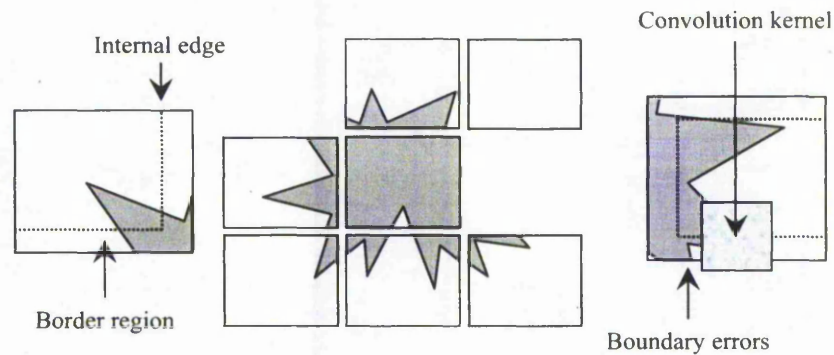


**Figure 15: The Distributor KS flow diagram.** The diagram illustrates the simple behaviour employed to clear information from the blackboard and populate it with system variables and image segments.

#### 3.4.1.1 Distribution of an image

To successfully achieve distribution of a selected image, the Distributor KS employs the number of segments into which the selected image is divided and size of border between segments variables obtained from the *Parameters* partition. By setting the number of segments in the  $x$  axis to one, a horizontal distribution scheme is achieved.

Similarly, by setting the number of segments in the  $y$  axis to one, a vertical distribution scheme is achieved. The selection of both parameters results in the construction of a segment matrix of corresponding size. Division of the selected image into a matrix is used to generate higher number of segments, than can be achieved with either vertical or horizontal schemes alone. If the selected image cannot be evenly distributed, the end segments are used to accommodate discrepancies. Once divided into segments, borders are assigned to internal edges.



**Figure 16: The variable number of borders depending on a segments position within the selected image. Small borders combined with a large image processing operator result in boundary errors within a processed segment.**

Figure 16 shows that the segments generated possess a variable number of borders depending on their position within the selected image. The addition of borders is used to counter inconsistencies caused by a lack of pixels at a segments boundary. Importantly, the use of large borders and high numbers of segments results in increased redundant data moving between Worker KS and blackboard components. Borders which are too small, in contrast, can result in irregularities at the edges of a processed segment depending on the image processing operator employed.



### 3.4.2 The Worker n KS

In iDARBS, multiple Worker KSs are employed to perform computationally intensive image processing tasks. The tasks in question are embedded in the rule files of individual Worker KSs. Division of an image into segments means that the tasks performed by each Worker KS are identical. As a consequence, the rules files of each Worker KS are also identical. To achieve co-ordination of identical KSs, each Worker KS has a unique identifying number. Because the Worker KSs act upon partitions with corresponding numbering, the numbering scheme is also used to identify individual image segments. For example, the Worker n KS adds and removes information from the *Worker n control* partition. The Worker n KS also fetches segment n from the raw data container.

Connection to the blackboard and initialisation of the Worker n KS is performed by the *Initialise\_Worker\_n* rule. The Worker n KS then waits for image processing operator parameters to appear in the *Worker n control* partition. On appearance of the parameters, the *Fetch\_Segment\_n* rule causes segment n to be retrieved and its unique identifier removed from the *Image container* partition. The retrieved segment is decompressed and processed using the image processing operator selected and retrieved parameters. Once processed, segment n is compressed and returned to the raw data container, its unique identifier appearing in the *Image container* partition. These actions form the *Process\_Segment\_n* rule. Finally, the image processing operator parameters in the *Worker n control* partition are removed by the Worker n KS which terminates. This modification to the *Worker n control* partition causes the Manager KS to restart. Figure 17 illustrates the reactive processing of a segment, by the Worker n KS, using a flow diagram.

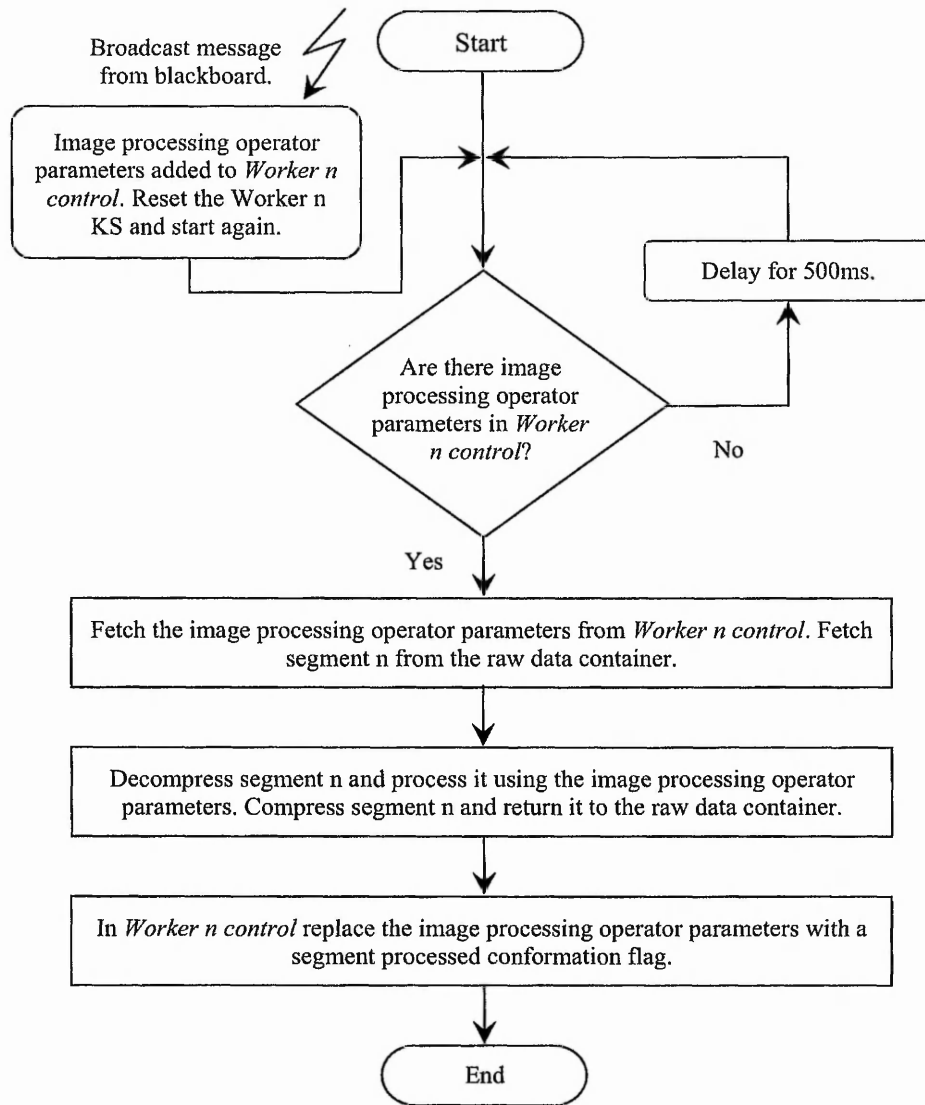
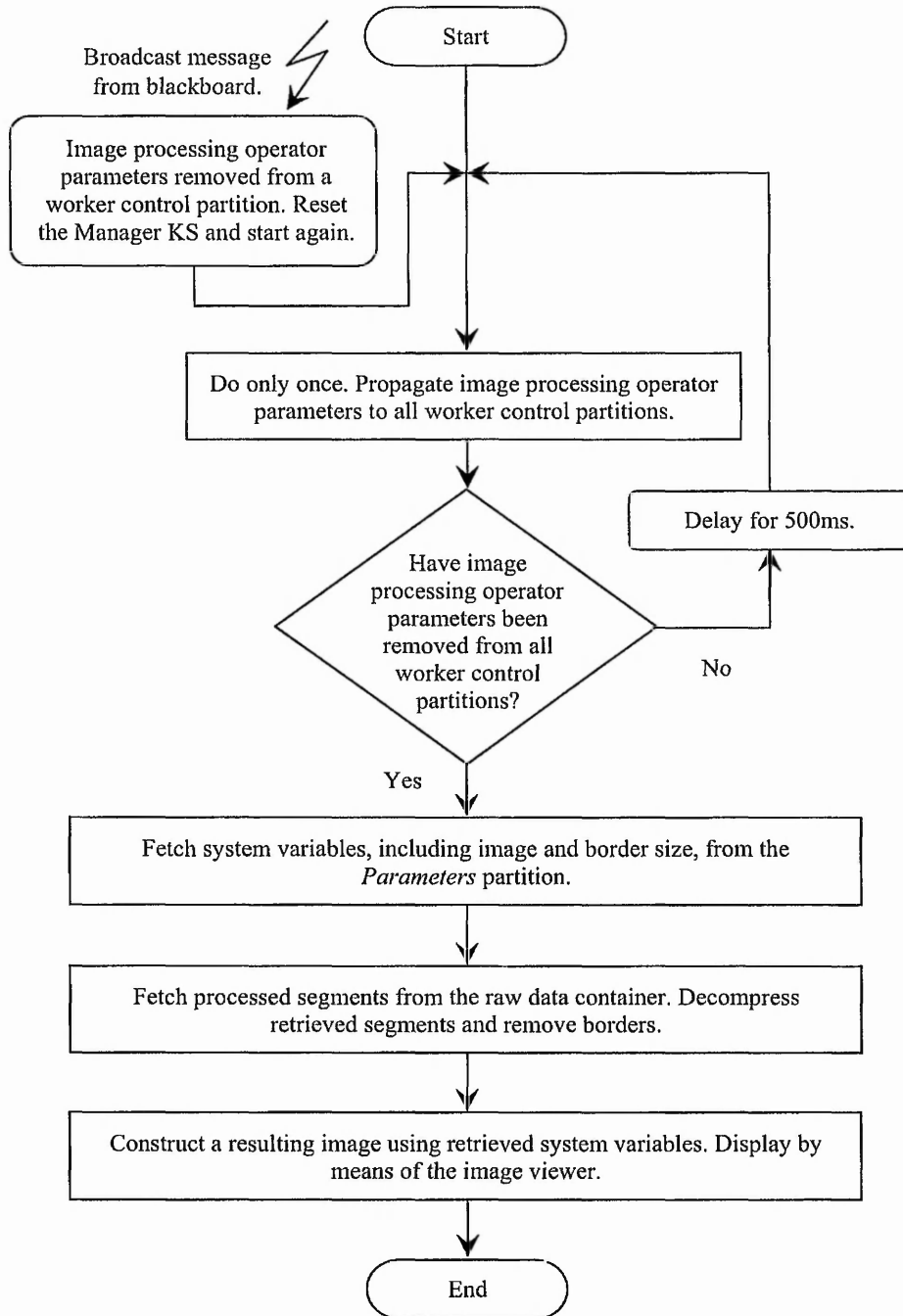


Figure 17: The Worker n KS flow diagram. The diagram illustrates the reactive behaviour employed to fetch and process an image segment.

### 3.4.3 The Manager KS

Tasks performed by the *Initialise\_Manager* rule include clearance of a locally maintained segment list in preparation for new image data. Image processing operator parameters are then propagated to all worker control partitions. This causes all waiting Worker KSs to commence processing and is encapsulated as *Process\_Image*. The Manager KS now waits for image processing operator parameters to be removed from all worker control partitions. Once removed, system variables including selected image and border size are fetched from the *Parameters* partition. Processed segments are then retrieved from the raw data container and their unique identifiers are removed from the *Image container* partition. Each segment retrieved is decompressed and added to a locally maintained segment list. When fired *Reconstruct\_Image* causes a resulting image to be constructed from processed segments.

Inconsistencies at a segments boundary are eradicated when borders are removed during the resulting image construction process. The resulting image is then displayed by means of the image viewer. Importantly, the restarting of Worker KSs, caused by the propagation of image processing operator parameters to worker control partitions, is prevented using a fire once mechanism. The mechanism works by placing a process image conformation flag in the *Manager control* partition, on firing of the *Process\_Image* rule. The *Process\_Image* rule is prevented from firing again until the conformation flag has been removed from the *Manager control* partition. Without this mechanism, both Manager and Worker KSs endlessly restart as they repeatedly modify worker control partitions. Figure 18 illustrates the reactive co-ordination of Worker KSs, by the Manager KS, using a flow diagram.



**Figure 18: The Manager KS flow diagram. The diagram illustrates the reactive behaviour employed to co-ordinate Worker KS activities.**

#### **3.4.3.1 Accumulation of an image**

For a resulting image to be successfully constructed from accumulated segments the Manager KS employs the selected image size and the size of border variables obtained from the *Parameters* partition. Once processed segments have been retrieved and stored in a locally maintained segment list, memory is allocated to a resulting image using the selected image size. For each processed segment in the list, three operations are then performed. First, the position of the segment within the resulting image is calculated. A region of interest excluding borders belonging to the segment is then determined. Finally, the region of interest is copied to the resulting image. Once all segments have been added to the resulting image the locally maintained segment list is cleared of all image data.

### **3.5 Experimental testing**

Modular verification of iDARBS components was sought before speed testing was performed. Addition of the DARBS terminal client to the iDARBS framework resulted in a powerful testing and debugging tool. Once connected, the terminal client was employed to create, manipulate, and destroy partitions using messages manually entered at a command line. During verification, dummy information strings for testing of KS behaviour were sent to and retrieved from the blackboard. The content of individual partitions was also displayed periodically using the terminal client. Due to the client/server implementation, the Distributor, Worker, and Manager KSs were tested separately and then as a whole. Tests were made to ensure that selection of an image could be made, followed by division and storage of segments. The co-ordinated activation of Worker KSs activities together with monitoring of their current state was also evaluated. As was the ability of the Manager KS to retrieve processed segments

and construct a resulting image. Importantly, creation and storage of 1–20 segments was observed as being successful. Details of extensive validation testing are provided in Appendix B.

### 3.5.1 Aims and setup

Speed testing was performed in order to investigate the potential performance increase of image processing algorithms in a distributed processing environment. The results obtained would determine the suitability of a distributed blackboard architecture, for the hosting of image registration algorithms such as those described in Chapter 2. In order for this to be achieved mean filtering, local histogram equalisation, and adaptive thresholding algorithms were selected for initial testing of the iDARBS framework. The algorithms were chosen because they possess patterns of data access which match those employed by registration algorithms. The patterns of data access being one-to-one and many-to-one respectively. The variable kernel size and increasing complexity of each algorithm also allow a range of computational burden to be evaluated. Encapsulated as dynamically linked functionality, the algorithms were embedded in Worker KS rule files.

The setup of experiments was designed to compare the speed of each algorithm, in sequential and distributed processing environments. To achieve this, the effects on performance of adding Worker KSs to the iDARBS framework were investigated. In the sequential environment, an algorithm is hosted and run using a single processor. For the distributed environment, an algorithm is hosted using the iDARBS framework and run on multiple processors. In each experiment, distributed testing represented an ideal case, i.e. one processor for the blackboard, one processor for the Distributor KS, one

processor for the Manager KS, and one processor for each Worker KS. By employing a parallelism granularity which is set to KS level, a fair comparison between sequential and distributed approaches was made. It also permitted the behaviour of individual KSs to concentrate on distribution, processing, and accumulation of image segment; while at the same time allowing the issues of fine-grained parallelism to be ignored.

Obtained from the field of manufacturing, screen-printed shampoo bottles were chosen to provide test images. The bottles, provided by the plastics manufacturer M&H Plastics Ltd, contain defects such as screen leak and missing print. Both of which can be caused by incorrect ink viscosity, material contamination, and wear. In automated visual inspection the choice of resolution determines the smallest size of detectable defect. As a consequence, high resolution images of  $1400 \times 1800$  pixels were captured. For the images to provide an even level of computational burden, illumination conditions were experimented with. Problematic environments were found to be caused by background selection, surface reflectivity, and sample transparency. In such environments, an uneven intensity gradient across captured images occurs. As no rules for the creation of ideal conditions exist, trial and error was employed until acceptable capture conditions were found. The images selected for testing purposes are shown in Figure 19. Details of experiments used to achieve adequate capture conditions are documented in Appendix C.





### 3.5.2 Results and discussion

Timing of each experiment started when the Manager KS propagated image processing operator parameters to all worker control partitions. Timing stopped when the Manager KS found processed segment conformation flags in all worker control partitions. In both cases, the current system time in seconds and milliseconds was printed to a debug window. The time required to process an image was calculated by subtracting the two outputs. Testing was performed three times, results were combined and an average calculated. Crucially, selected images were divided into 1–14 segments and a 25-pixel wide border allocated. Each segment was assigned to an individual Worker KS before being sent to the blackboard. Connected via an Ethernet 100Mbps switch, a network containing approximately 30 computers was used to host the iDARBS framework. Running the Ubuntu operating system, all computers in the network contained AMD Athlon 2GHz processors with 1 gigabyte of random access memory.

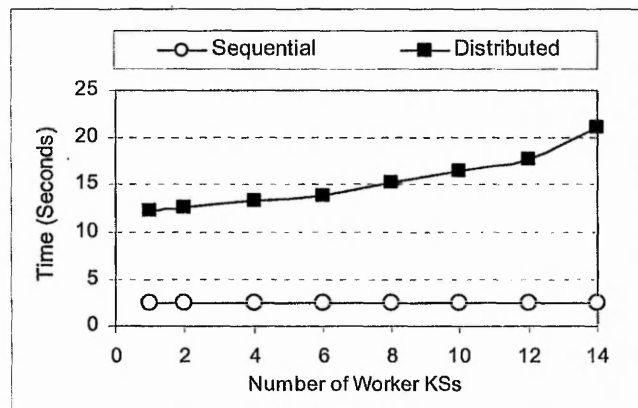
#### 3.5.2.1 Sequential vs distributed mean filtering

Convolution is a mathematical operation based on matrix multiplication of a kernel with an image. This property allows the implementation of image processing operators, whose output is a combination of inputs. Primarily, convolution is used in noise reduction, smoothing, and edge detection filters. The convolution process itself is performed by sliding a kernel over an image. At each pixel location, a new intensity is generated using values within the immediate neighbourhood. In general, the size and shape of a kernel is variable and has a marked effect on the resulting image. Kernels coefficients can be predefined or constructed using intensities drawn from an image. Conveniently, small kernels can be applied multiple times or a large kernel can be employed once to produce a similar effect. Selected for testing purposes, mean filtering

[122] is a computationally simple task whereby the intensity variation between pixels is reduced. During filtering, intensities are replaced with the arithmetic mean of a kernel that surrounds them. Convolution is expressed formally as

$$\sum_{i=1}^M \sum_{j=1}^N I(x-i, y-j)K(i, j) \quad 3.1$$

where  $I$  is an image and  $K$  is a convolution kernel.  $M$  and  $N$  are the number of row and columns. While  $i, j, x$ , and  $y$  are the pixel co-ordinates of the image and kernel respectively. To demonstrate the efficiency of convolution, testing of the distributed mean filtering algorithm was performed using a kernel of  $5 \times 5$  pixels.



**Figure 20: The sequential and distributed processing speed of mean filtering. Increasing numbers of Worker KS are shown.**

The sequential and distributed processing speed of mean filtering, plotted as time against number of Worker KSs, is shown in Figure 20. As expected, distributing the mean filtering algorithm did not yield any performance improvements rather an increase in processing speed was measured. The figure shows that convolution with a small kernel is an efficient task where the speed of distributed processing is outweighed by

KS management and communications overheads. Communication overheads are high due to the Worker KSs polling the blackboard, approximately every 500ms, whenever image segments are not available for processing. For each poll made by the Worker  $n$  KS, the blackboard is required locate and search the *Worker  $n$  control* partition for image processing operator parameters. The blackboard then answers the Worker  $n$  KS regardless as to whether or not the image processing operator parameters were found. Although the contents of the *Worker  $n$  control* partition remain unchanged and the Manager KS is not restarted, polling results in a considerable burden which has a negative impact upon distributed performance.

### 3.5.2.2 Sequential vs distributed local histogram equalisation

Histogram equalisation represents a method that distributes pixel values uniformly throughout the intensity range of an image [123]. During histogram equalisation, contrast is enhanced by altering the intensity histogram of an image so that it corresponds to a desired shape. Often the desired shape is as flat and spread out as possible. This is achieved through the non-linear mapping of values over an entire image, resulting in a regularised distribution of intensities. In more complex conditions, where a global histogram does not capture the regional statistics of an image, local histogram equalisation [124] can be employed. For testing purposes local histogram equalisation was selected, where at each pixel location a kernel representing neighbourhood intensities is constructed. The central pixel is then set based on the equalised histogram of a kernel, large enough in size to capture a comprehensive intensity range. Equalised intensities are calculated using

$$E = k_1[S(x, y) - I_m(x, y)] + k_2I_m(x, y) \quad 3.2$$

where  $k_1$  represent a local gain constant and  $k_2$  represent a local mean constant, both of which have values between zero and one. While  $l_m$  is the local mean intensity of the kernel which is centred at position  $x, y$ . Figure 21 shows results plotted as time against number of Worker KSs, whilst performing local histogram equalisation. In order for the regional statistics of a segment to be captured, a kernel size of  $50 \times 50$  pixels was employed. Although the processing time of a single Worker KS exceeds that of the sequential implementation due to parallelisation overheads, it can be seen that the average execution time reduces from two minutes to approximately 35 seconds when eight Worker KS are employed.

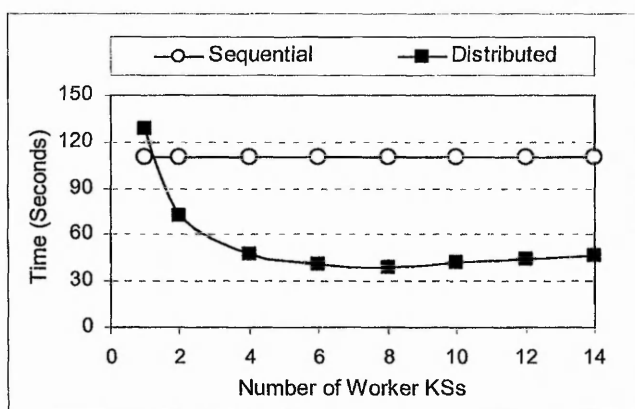


Figure 21: The sequential and distributed processing speed of local histogram equalisation. Increasing numbers of Worker KS are shown.

### 3.5.2.3 Sequential vs distributed adaptive thresholding

Thresholding is a process used to divide an image into foreground and background components by considering a predefined threshold level. In conventional thresholding, all pixel values are compared with a global threshold and set accordingly. This differs from adaptive thresholding [125], where the predefined threshold is dynamically changed over an image. The threshold is changed because it is assumed that smaller

regions contain more uniform conditions than the image as a whole. Importantly, adaptive thresholding is the most computationally expensive algorithm selected for testing purposes. For each pixel, a kernel representing neighbourhood intensities is constructed. An optimum threshold level is then determined, using the between-class variance [126] of a histogram constructed from kernel intensities. Once compared with the threshold, the central pixel value is classified as being either foreground or background in nature. Understandably, the kernel employed need to be of sufficient size as to encompass both foreground and background pixels. The between-class variance employed is defined as

$$\sigma^2_w(t) = P_1(t)\sigma^2_1(t) + P_2(t)\sigma^2_2(t)$$

with

$$P_1(t) = \sum_{g=1}^1 P(g)$$

and

$$P_2(t) = \sum_{g=t+1}^{Maxgrey} P(g)$$

$$\sigma^2_1(t) = \sum_{g=1}^1 [g - \mu_1(t)]^2 P(g) / P_1(t)$$

$$\sigma^2_2(t) = \sum_{g=t+1}^{Maxgrey} [g - \mu_2(t)]^2 P(g) / P_2(t) \quad 3.3$$

$$\mu_1(t) = \sum_{g=1}^1 g \times P(g) / P_1(t)$$

$$\mu_2(t) = \sum_{g=t+1}^{Maxgrey} g \times P(g) / P_2(t)$$

$$P(g) = \frac{1}{mn} \sum_{K(i,j)=g} \frac{K(i,j)}{g}$$

where between-class variance  $\sigma_w^2(t)$  of the kernel represents a weighted sum of foreground  $P_1(t)$  and background  $P_2(t)$  pixel groups as a function of threshold level  $t$ .  $Maxgrey$  is the maximum intensity level while  $P(g)$  is the count of pixel values at level  $g$ .  $\mu_1(t)$ ,  $\mu_2(t)$ ,  $\sigma_1^2(t)$ , and  $\sigma_2^2(t)$  are simply functions of threshold level  $t$ . The value of  $t$  which minimises the between-class variance is selected as the optimum threshold level. For testing of adaptive thresholding a  $7 \times 7$  kernel was deemed of sufficient size. In Figure 22, results plotted as time against number of Worker KSs, show that processing time was reduced from 11 minutes to two minutes when 12 Worker KSs were employed.

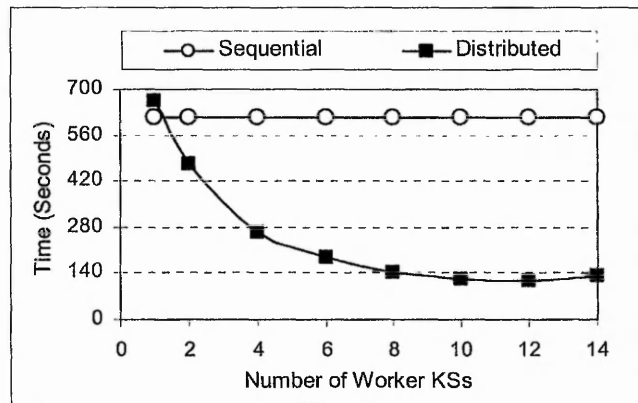


Figure 22: The sequential and distributed processing speed of adaptive thresholding. Increasing numbers of Worker KS are shown.

### 3.6 Conclusions

The results obtained clearly show that the speed of local histogram equalisation and adaptive thresholding hosted on iDARBS is better than sequential versions of the same algorithms. By hosting each Worker KS on an individual processor context switching between processes has been avoided. Crucially, the context switching between multiple

processes on a single processor introduces additional overheads resulting in longer execution times. The processing speeds accomplished during testing are, however, far from ideal. This is mainly due to the overheads caused by the serial processing of KS requests by the blackboard. These overheads are demonstrated in each experiment, by means of the single Worker KS implementation. An additional factor which has been observed to limit processing speed is saturation of the blackboard. Saturation occurs when the speed of requests coming from KSs becomes faster than servicing of request by the blackboard. The serial processing of requests which are arriving at rates resulting in blackboard saturation are demonstrated by the mean filtering experiment.

When comparing results obtained from local histogram equalisation and adaptive thresholding, there are considerable differences in the processing speeds achieved. As previously discussed, the processing speed reached by the iDARBS framework is largely dependent on the overheads of KS management. Because the management of KSs was unchanged for both experiments, the overheads encountered also remained the same. As complexity of the algorithm used for testing grew, the proportion of time lost to overheads became smaller while the time spent process segments increased. Given that the total time required to process an image is distributed between Worker KSs, improve processing speeds are achieved. The adaptive thresholding algorithm also benefits from the fact that Worker KSs are querying the blackboard less frequently. This slowing of requests causes blackboard saturation to occur at a higher number of Worker KSs. As a consequence, the spreading of KS communications alleviates congestion and causes the optimum number of Worker KSs to increase from eight to 12 for local histogram equalisation and adaptive thresholding respectively.

### 3.7 Summary

The aim of this research is an investigation into high performance intensity-based image registration using non-specialised architectures. In order for this to be achieved, an underlying distributed image processing framework called iDARBS was developed and tested. To remove the unnecessary formatting of pixel data into the string type, a limitation imposed by the original DARBS implementation, the iDARBS blackboard and KSs modules were modified to handle raw data. Compression and decompression of raw data in order to reduce transmission overheads was also implemented. For convenience, an image viewer was added to the iDARBS framework and allows visualisation of selected images, images stored on the blackboard, and the results of image processing operations. Partitioning of information upon the blackboard, aids design of the framework by introducing structure to an area of shared-memory. To promote the efficient handling of KS queries, short concise information was adopted. The use of human readable information assisted debugging of the framework and permits modifications in the future.

To perform testing of the iDARBS framework mean filtering, local histogram equalisation, and adaptive thresholding algorithms were implemented. The algorithms were chosen because of their pattern of data access and their increasing computational burden. During mean filtering, intensities were replaced with the arithmetic mean of the kernel that surrounds them. As expected, no improvements in terms of processing speed resulted from distributing the algorithm rather an increase in processing time was measured. For local histogram equalisation, intensities were set based on the equalised histogram of a kernel that surrounded them. Although the processing speed of a single Worker KS was observed to be slower than that of the sequential implementation,



average execution times reduced from two minutes to approximately 35 seconds. Finally, using between-class variance, an optimum threshold level was determined for intensities during adaptive thresholding. Once calculated, each pixel was classified as being either foreground or background in nature. Processing times were reduced from 11 minutes to approximately two minutes. Tables of results are provided in Appendix D.

## 4 High performance intensity-based image registration

In Chapter 3, a scalable distributed image processing framework called iDARBS (imaging Distributed Algorithmic and Rule-based Blackboard System) was introduced. Image distribution and accumulation mechanisms were presented and information strings discussed in detail. Based on a worker/manager model and implemented as knowledge sources (KSs), reactive behaviour was used to control concurrent processing of image segments. Balanced communications and data consistency were achieved through the logical partitioning of information on the blackboard. The resulting approach was shown to work well with a range of computationally intensive tasks, clearly outperforming sequential versions of the same algorithm. Although the results obtained were good, the algorithms employed during testing did not address the limitations of high performance intensity-based image registration. As previously stated, specialised architectures containing shared-memory and multiple processors have been identified as a drawback of these applications. The distribution of a registration algorithm using the iDARBS framework is, however, a step towards addressing this problem.

An approach similar to that employed in Chapter 3 is now taken to distribute similarity calculation for image registration. In Section 4.1 of this chapter, justification is given to the selection of an image processing toolkit. The selection of an intensity-based image registration algorithm followed by its mathematical derivation is then given. Mapping of the selected algorithm to the iDARBS framework, plus development of information strings to control the transform optimisation process are discussed in Section 4.2. In Section 4.3, modifications made to the Distributor, Worker, and Manger KS types are presented. As is functionality for similarity calculation between image segments and the

updating of transform parameters at the end of each optimisation cycle. Experimental testing is provided in Section 4.4, conclusions in Section 4.5, and a short summary in Section 4.6.

Results based on the images discussed in Chapter 3, confirm that the time constraints associated with similarity calculation can be significantly improved when compared with sequential approaches. Furthermore, the implementation of two similarity metrics demonstrates how high performance intensity-based image registration can be achieved using a non-specialised architecture.

#### **4.1 Toolkit selection**

It was deemed unnecessary to independently implement the necessary registration functionality as a number of open source toolkits were readily available. The Visualisation Computational Imaging Science Group (VTK CISG) [127] has developed two open source intensity-based image registration algorithms. Both rigid and non-rigid algorithms are implemented using the open source Visualisation Tool Kit (VTK) [128]. With the rigid algorithm, a global transform is iteratively sought in a multi-resolution fashion, by maximising the mutual information between two images. By restricting the optimisation process, through limiting the degrees of freedom (DOF), the algorithm can be adapted to suit specific registration problems. The non-rigid algorithm, in contrast, employs a free-form deformation model in which local distortions between an image pair are captured. Consisting of a regular grid of B-spline control points, the deformation model deforms an underlying image when moved [129]. Mutual information is again used as a measure of similarity between images. The VTK CISG

toolkit was deemed unsuitable for adoption as distribution of similarity calculation would be restricted to mutual information-based algorithms.

The Insight Segmentation and Registration Toolkit (ITK) [130], has been developed to support the medical imaging community. As a consequence, the toolkit focuses on multiple segmentation and registration algorithms. Motivated by considering registration as a generic problem, the design of algorithms is based on identification of functional components. Where each functional component can be accomplished using a variety of techniques. This flexibility permits algorithms to be created and tailored to a specific problem, using pluggable components that can be easily interchanged. Importantly, the toolkit adopts a pipeline approach based on data objects which are manipulated using filters. At present, the ITK toolkit represents an open source library which has been successfully used in a number of academic and commercial applications [131]. In general, the extensive documentation and archive material provided, offers useful support for the toolkit [132]. The specialist provision of registration algorithms also made it the favoured choice.

##### 4.1.1 An intensity-based registration algorithm

As previously discussed, intensity-based registration is a method used to geometrically align images taken from different sensors, viewpoints or instances in time. Traditionally, correlation has been used as a metric where a maximum similarity between reference (fixed) and sensed (moving) images is searched for [133]. Using such an approach, high levels of alignment accuracy can be achieved by interpolating intensities before evaluating similarity. Although more sophisticated algorithms capable of handling complex geometric deformations now exist [134] [135], correlation-based

registration is primarily used to align translated, rotated, and scaled images. As a consequence, the distribution of correlation-based similarity metrics for the registration of images would provide a foundation on which more complex functionality could be added. The construction of sequential algorithms for comparison purposes would also be relatively straightforward. Like existing correlation-based approaches, the algorithms would be suited to the alignment of images captured using the same sensor type and hence corresponding modality.

When described formally the inputs to a registration algorithm can be defined as the fixed image  $F$ , the moving image  $M$ , and the transform  $T$  used to map pixel co-ordinates between an image pair. Understandably, the goal of the registration process is recovery of a spatial mapping that brings the two images into alignment. To achieve this, the metric  $S(F, M, T)$  is employed to generate a measure of similarity based on how well aligned the transformed moving image is with the fixed image. The measure of similarity produced forms a quantitative criterion which can be optimised in a search space spanned by transform parameters. Importantly, by employing a gradient-descend optimisation technique [136] the metric can be used to produce derivatives of the similarity measure with respect to each transform parameter. The resulting derivatives are used to update the current transform and the process is repeated until an acceptable degree of alignment has been achieved.

Given the fixed image  $F$  and moving image  $M$ , calculation of similarity metric derivatives for updating the set of transform parameters  $P$ , as described by Yoo [137], is defined as

$$\frac{\partial S(P|F, M, T)}{\partial p_i} = \sum_{j=1}^Q \frac{\partial S(P|F, M, T)}{\partial x'_j} \frac{\partial x'_j}{\partial p_i} \quad 4.1$$

where  $Q$  is the number of valid pixels, indexed by  $j$ , between images and  $\partial x'_j / \partial p_i$  is a matrix called the transform Jacobian. Typically, transform  $T$  works upon the set of transform parameters indexed by  $p_i$ . Using the affine transform type, the pixel location  $x$  in the fixed image is mapped to a new position  $x'$  in the moving image using

$$x' = Ax + t \quad 4.2$$

where  $A$  is an  $n \times n$  dimensional matrix and  $t$  represents an  $n \times 1$  dimensional translation vector. The set of parameters  $P$  therefore represent the set of  $n \times n$  coefficients of  $A$  plus the  $n$  components of the translation vector  $t$ . The coefficients maybe as simple as a scaling factor or as complex as the trigonometric terms associated with rotation.

The transform Jacobian  $\partial x'_j / \partial p_i$  identified in Equation 4.1 is used to determine how mapped location  $x'$  moves as a function of variations in the transform parameters. Using  $p_i$  as a parameter from the set of transform parameters  $P$ , the transform Jacobian is defined as

$$\frac{\partial x'}{\partial p_i} = \begin{bmatrix} \frac{\partial x'_1}{\partial p_1} & \frac{\partial x'_1}{\partial p_2} & \dots & \frac{\partial x'_1}{\partial p_i} \\ \frac{\partial x'_2}{\partial p_1} & \frac{\partial x'_2}{\partial p_2} & \dots & \frac{\partial x'_2}{\partial p_i} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial x'_j}{\partial p_1} & \frac{\partial x'_j}{\partial p_2} & \dots & \frac{\partial x'_j}{\partial p_i} \end{bmatrix} \quad 4.3$$

When B-spline interpolation is employed to determine intensities at non-discrete locations, the moving image is represented using B-spline basis functions. Intensities at non-discrete locations are calculated by multiplying the coefficients of the moving image, with a B-spline kernel over the neighbourhood of the mapped pixel position. B-spline interpolation [40] is defined as

$$M(x) = \sum_{k=1}^K C_k \beta^{(2)}(x - x_k) \quad 4.4$$

where  $\beta^{(2)} = \beta^{(2)}(x) \beta^{(2)}(y)$  is a separable convolution kernel,  $C_k$  is a B-spline coefficient calculated from image samples through recursive filtering,  $x_k$  is the co-ordinates of a discrete pixel location, and  $K$  is the number of valid pixels involved. To achieve a compromise between smoothness of search space and computational burden, a third order B-spline kernel can be used. The arguments of the kernel are given as

$$\beta^{(2)}(x) = \begin{cases} \frac{2}{3} - \frac{1}{2}|x|^2(2 - |x|) & 0 \leq x < 1 \\ \frac{1}{6}(2 - |x|)^3 & 1 \leq x < 2 \\ 0 & 2 \leq x \end{cases} \quad 4.5$$

Once computed, derivatives of the similarity measure are used to generate an updated transform. Using  $\partial S(A_{ij})/\partial A_{ij}$  as a derivative of the similarity measure with respect to matrix component  $A_{ij}$ . Transform matrix  $A$  from Equation 4.2 is updated using

$$A'_{ij} = A_{ij} + \frac{\partial S(A_{ij})}{\partial A_{ij}} \lambda \quad 4.6$$

where  $A'_{ij}$  is the updated component and  $\lambda$  is the step length of the optimisation process. The transform vector  $t$ , in contrast, is updated using the expression

$$t = (I - A)C \quad 4.7$$

where  $I$  is an identity matrix,  $A$  is the updated matrix described in Equation 4.6, and  $C$  is a fixed point used for reference throughout the optimisation process.

## 4.2 Image registration on iDARBS

In order for an intensity-based image algorithm to be distributed, both fixed and moving images require division into segments and distribution between processors. As gradient-descent optimisation is to be employed, the metric component will be required to produce derivatives of the similarity measure with respect to each transform parameter. To achieve this, transform parameters have to be propagated to all nodes in the distributed processing network. On receiving the transform parameters, each node is required to compute local derivatives of the similarity measure for the segments allocated to it. The local derivatives computed can then be accumulated and summed into a global derivative. This will allow the transform to be updated based on the similarity of whole images. Convergence testing can then be performed using the newly updated transform parameters. Depending on the success or failure of convergence testing, propagation of updated parameters and hence evaluation of a new transform can occur.

Using the distribution and accumulation mechanisms described in Chapter 3, the iDARBS framework was extended to perform high performance intensity-based image registration. As before the framework consists of Distributor, Worker, and Manager KSSs. Framework initialisation and image selection remain the responsibility of the



Distributor KS, as does the division of selected images vertically and horizontally into segments. Before being compressed and placed on the blackboard the edges of each segment are assigned a border. Upon activation, the Worker KSs retrieve individual segments and compute local derivatives of the similarity measure with respect to each transform parameter. The accumulation and summation of local derivatives is performed by the Manager KS. The Manager KS then updates transform parameters which are propagated to the Worker KSs. Throughout the alignment process, co-ordination of Worker KSs activities is the responsibility of the Manager KS. The calculation of local derivatives and the updating of transform parameters are repeated until predefined thresholds are exceeded. Once exceeded, the Manager KS retrieves fixed and moving images from the blackboard. The moving image is then resampled using the optimised transform parameters.

#### 4.2.1 Information strings and the image registration process

A range of information strings were developed to control firing of KS rules and allow the movement of transform and local derivative parameters between framework components. Example information strings are shown in Figure 23.

- The region of interest string is used to hold the starting co-ordinates and size of a segment without borders. The string is generated by the Distributor KS and placed in all worker control partitions.
- A previous transform string is created by the Distributor KS and used to hold transform parameters from the previous iteration. Updated by the Manager KS, the string is maintained in the *Parameters* partition and used in the calculation of an updated transform.

- Generated and updated by the Manager KS, the current transform string is used for propagation of updated transform parameters to all Worker KSs. The string also co-ordinates the start of Worker KS activities.
- The accumulation of local derivatives is achieved with derivative strings. The creation of a derivative string marks the temporary suspension of a Worker KS's activities.
- The final transform string contains optimum transform parameters. Once created and propagated to all Worker KSs, the string causes the permanent suspension of Worker KS activities.

```
[ROI 0_0_700_900]
[Previous 1.34982342_1.42314742 . . . _11.85115785]
[Current 1.98437218_1.56237292 . . . _16.30191341]
[Derivative -203.68349139_68.62940029 . . . 549]
[Final 1.64514585_0.01234546 . . . _15.79934632]
```

**Figure 23: Information strings for controlling the image registration process. The components of a string include a rotation matrix, a translation vector, and derivatives of the similarity metric.**

The components of the previous, current, and final transform parameter strings include an  $n \times n$  dimensional matrix of coefficients and an  $n \times 1$  dimensional translation vector. Logically, the order in which transform parameters are listed remains the same for all strings in which they appear. As previously stated, the derivative string is constructed from local derivatives of the similarity measure with respect to each transform parameter. Included at the end of the derivative string is the number of valid pixels

transformed between segments. Both the current transform and derivative parameter strings are held in the worker control partitions.

### 4.3 KS behaviour during image registration

Additions and modifications to iDARBS KS types, required to achieve the desired behaviour, are now described in detail. A summarised list of rule files including example contents can be found in Appendix E.

#### 4.3.1 The Distributor KS

Tasks performed by the `Initialise_Distributor` rule include clearance of all information from the blackboard. Segment lists which are locally maintained by the Distributor KS are also initialised in preparation for new image data. The selection of fixed and moving images is performed manually, after the image viewer has been shown, on firing of `Select_Images`. Embedded in `Set_Transform`, a current transform is estimated using the selected images and added to the *Parameters* partition. Optimisation parameters, used in the creation of an updated transform are also added to the *Parameters* partition. Regions of interest are then generated for each segment and added to corresponding worker control partitions. On firing of `Store_Segments`, division of fixed and moving images into segments as well as compression and storage in the raw data container is performed. Understandably, the unique identifiers assigned to each segment are added to the *Image container* partition. As the Distributor KS is the only KS connected to the blackboard at that time, no restarting of either Worker or Manager KSs occurs.

#### **4.3.1.1 Initialisation**

As gradient-descent optimisation is employed, predefined parameters associated with the optimisation process are added to the *Parameters* partition by the Distributor KS. The parameters include an initial step length, a minimum step length, and the maximum number of iterations to be performed. The initial step length is used to initialise the optimisation process. Convergence of optimisation and hence selection of final transform parameters is controlled by the minimum step length. Logically, the maximum number of iterations prevents the optimisation process from entering into an endless loop.

#### **4.3.1.2 Calculation of the initial transform**

For an initial estimate of similarity to be made, the Distributor KS has the ability to calculate an initial centre of rotation and a translation. Employing both fixed and moving images, initial transform parameters are estimated using centres of mass computed from intensity levels. Once calculated, the fixed image centre is set as the rotational centre of the initial transform. The translation component, in contrast, is set as the vector between the fixed and moving image centres of mass. To simplify initialisation, no rotation is specified in the initial transform. In general, the use of centres of mass over geometrical image centres results in a reasonable estimate of the initial transform. This is because the subject of interest is not always in the geometric centre of an image. Understandably, the initial transform is estimated before both images are divided into segments.

### 4.3.2 The Worker $n$ KS

To begin with, connection to the blackboard and initialisation is performed by the `Initialise_Worker_n` rule. The Worker  $n$  KS then waits for the current transform to appear in the *Worker  $n$  control* partition. As soon as the current transform appears, `Fetch_Segments_n` is fired. This rule causes fixed and moving segments to be retrieved from the raw data container followed by removal of their unique identifiers from the *Image container* partition. Once the retrieved segments have been decompressed, the region of interest is fetched from the *Worker  $n$  control* partition. Importantly, `Fetch_Segments_n` represents a fire once mechanism that prevents the Worker  $n$  KS from attempting retrieval of further segments during the optimisation process.

On firing of `Perform_Optimisation_n`, a local derivative and the number of valid pixels transformed between segments are calculated using the current transform. Once calculated, the local derivative and the number of valid pixels are used to replace the current transform in the *Worker  $n$  control* partition. This process is repeated every time a current transform appears in the *Worker  $n$  control* partition. Conveniently, modifications to the *Worker  $n$  control* partition through the addition of current transform parameters, causes the Worker  $n$  KS to restart immediately. As a consequence, the idle time of the Worker  $n$  KS is significantly reduced. Finally, when the final transform appears the Worker  $n$  KS becomes inactive. Figure 24 illustrates the iterative nature of the Worker  $n$  KS engaged in similarity calculation.

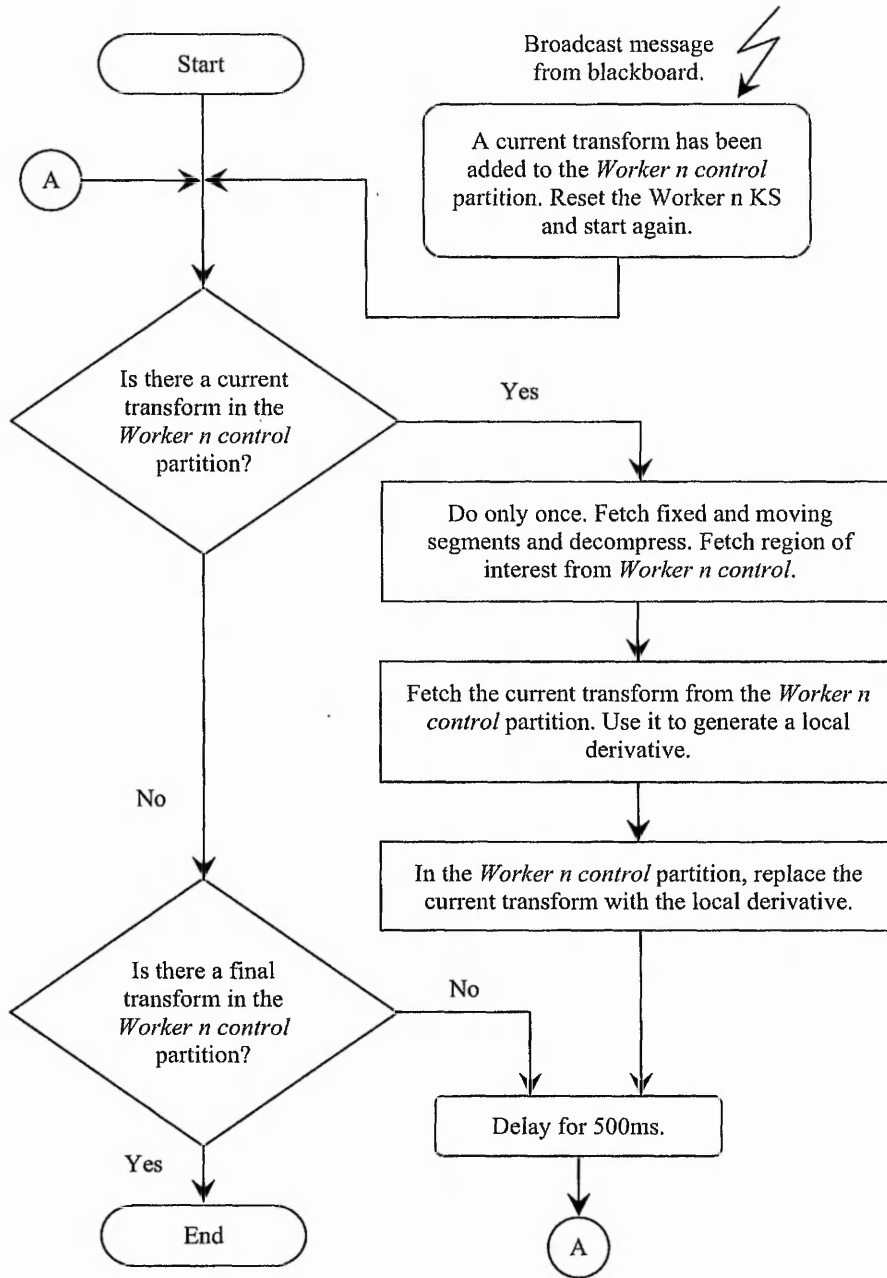
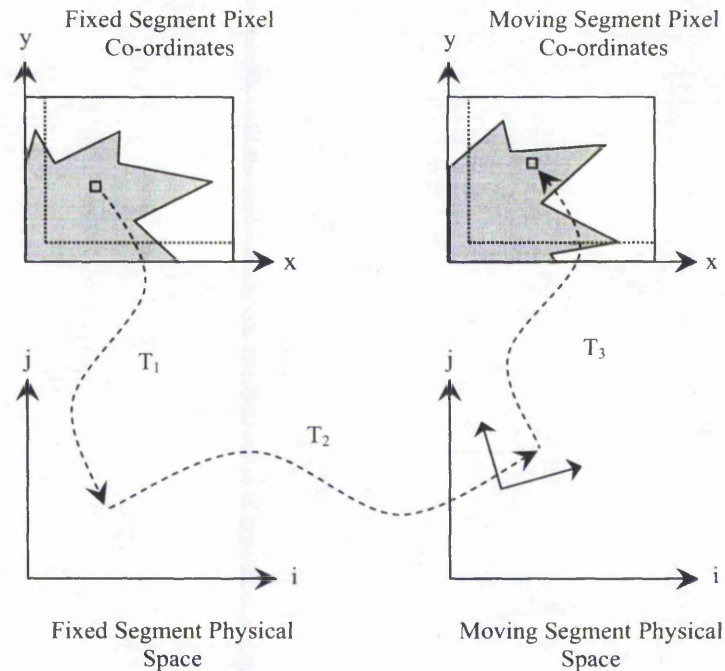


Figure 24: The updated Worker n KS flow diagram. The diagram illustrates the iterative retrieval of transform parameters and the generation of local derivatives.

#### 4.3.2.1 Calculating local derivatives of the similarity measure

In order for similarity between segments to be calculated, the Worker  $n$  KS employs the region of interest and current transform obtained from the *Worker  $n$  control* partition. First, the region of interest is used to identify the fixed segment without its borders. Then, for all pixel co-ordinates contained within the fixed segment region of interest, corresponding moving segment co-ordinates are calculated using the current transform parameters. If the transformation of fixed segment pixel co-ordinates results in a corresponding location that falls inside the moving segment, the number of valid pixels is incremented and a contribution to the local derivative is made. Otherwise the pixel is considered invalid and the next fixed segment pixel co-ordinates are evaluated. On completion, local derivatives of the similarity measure with respect to each transform parameter and the number of valid pixels transformed between segments are calculated by the Worker  $n$  KS.

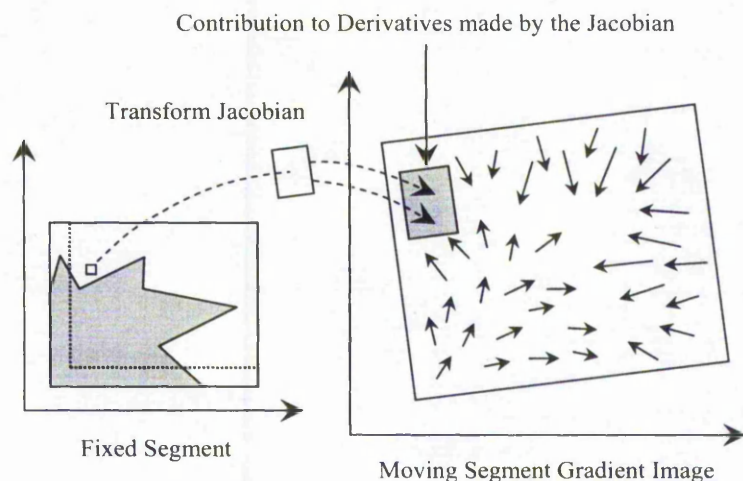
Contributions to a local derivative represent a summation of intensities, from a moving segment gradient image, around the mapped pixel co-ordinates. Using a recursive Gaussian gradient image filter, a gradient image is created from the moving segment. The gradient image represents a vector field in which every vector points in the direction of the nearest edge, an edge being a rapid increase or decrease in neighbouring intensities. Each vector has a magnitude proportional to the second derivative of the intensity in the direction of the vector. Created once after retrieval of the moving segment, the moving segment gradient image is used for all iterations of the optimisation process.



**Figure 25: The transformation of pixel co-ordinates between fixed and moving segments. Pixel co-ordinates are mapped to physical space, then physical space to physical space, and finally physical space to pixel co-ordinates.**

The transformation of fixed segment pixel co-ordinates into moving segment pixel co-ordinates is illustrated in Figure 25. Moving segment pixel co-ordinates are calculated using fixed segment pixel co-ordinates in the form  $x$  and  $y$ . First, fixed segment pixel co-ordinates are mapped into the physical space of the fixed segment  $T_1$  and then into the physical space of the moving segment. This represents the transform to be optimised and is shown as  $T_2$ . In a final step, the physical co-ordinates of the fixed segment pixel in moving segment space are mapped to moving segment pixel co-ordinates  $T_3$ . By employing a scale factor, the mapping of pixel co-ordinates through physical space allows segments of different size to be registered.





**Figure 26: The combination of transform Jacobian and moving segment gradients. The intensities computed are accumulated and form a contribution to the similarity measure derivatives.**

The contributions to a local derivative, at mapped pixel co-ordinates, are evaluated using a transform Jacobian. As shown in Figure 26 the Jacobian is used to determine variation in the mapped co-ordinates, as a function of variation in the transform parameters. Once computed, the spatial variations are combined with vectors from the moving segment gradient image and used to estimate intensities around the mapped location. The estimated values are then summed to form a contribution to the local derivative. When pixel co-ordinates mapped by a transform, correspond to a non-discrete location, B-spline interpolation is used to estimate contributions to the derivative. By recalculating the transform Jacobian at the end of each optimisation cycle variations in the mapped co-ordinates, as a function of the updated transform parameters are determined.

### 4.3.3 The Manager KS

The Manager KS is the most complex of all framework components. On firing of `Initialise_Manager` the current transform, placed in the *Parameters* partition, is retrieved. The current transform is then propagated to all worker control partitions. Next, the Manager KS waits for local derivatives to appear in all worker control partitions. An `Advance_Transform` rule is fired on appearance of the local derivatives, otherwise no action is taken. On firing of `Advance_Transform`, local derivatives are accumulated and a global derivative calculated. Once calculated, the global derivative and previous transform are used to calculate updated transform parameters. Convergence tests that consider optimisation step length and the number of iterations performed are also conducted.

Upon convergence, a final transform is generated. Otherwise the updated transform parameters are set as the current transform. Also, as part of `Advance_Transform`, local derivatives in all worker control partitions are replaced with the newly updated current transform. `Resample_Image` is fired on appearance of the final transform. Once fired, copies of the selected images stored in the raw data container are retrieved and their identifiers are removed from the *Image container* partition. The retrieved images are decompressed and the moving image is resampled using the final transform parameters. Significantly, the restarting of Worker KSs caused by the Manager KS, is prevented using a fire once mechanism. Finally, visual assessment of the registered image is made possible by means of the image viewer. Using a flow diagram, Figure 27 illustrates the iterative updating of current transform parameters by the Manager KS.

Figure 27: (Part A)

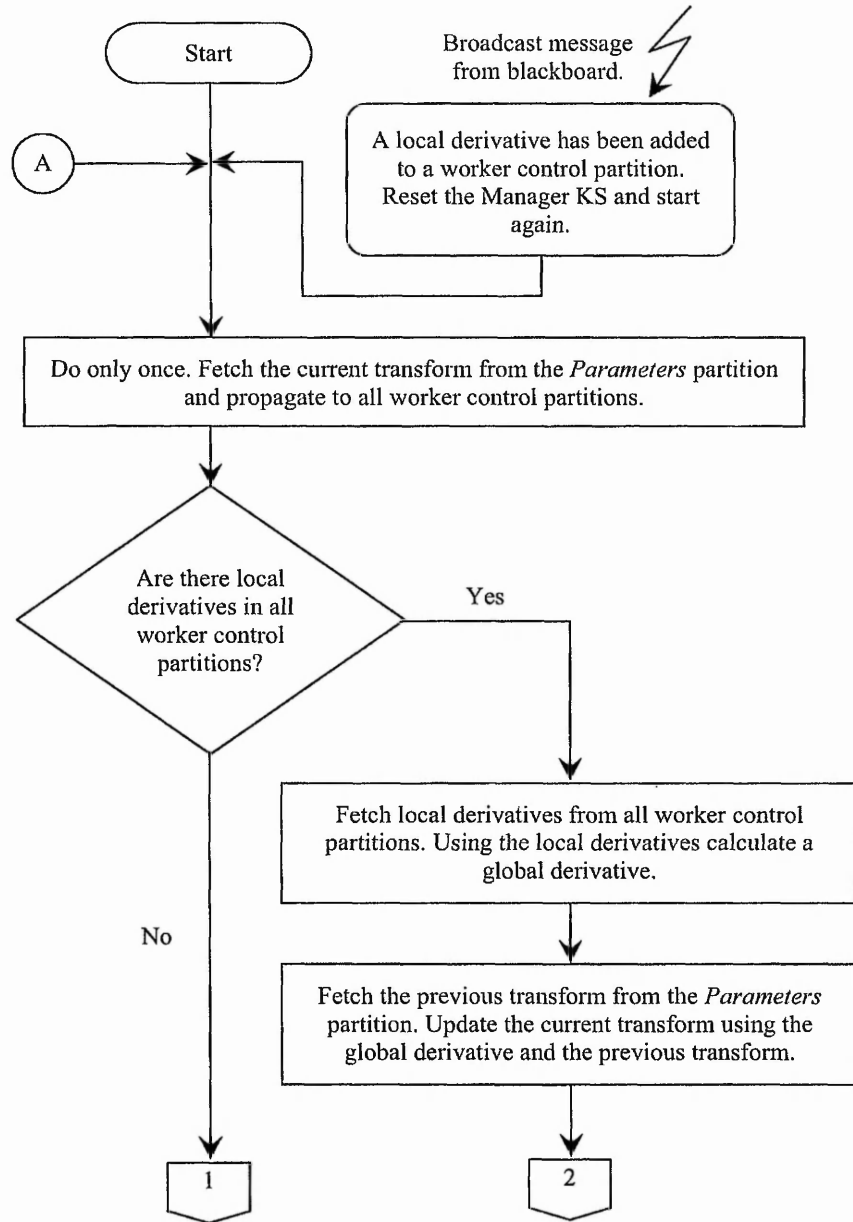


Figure 27: (Part B)

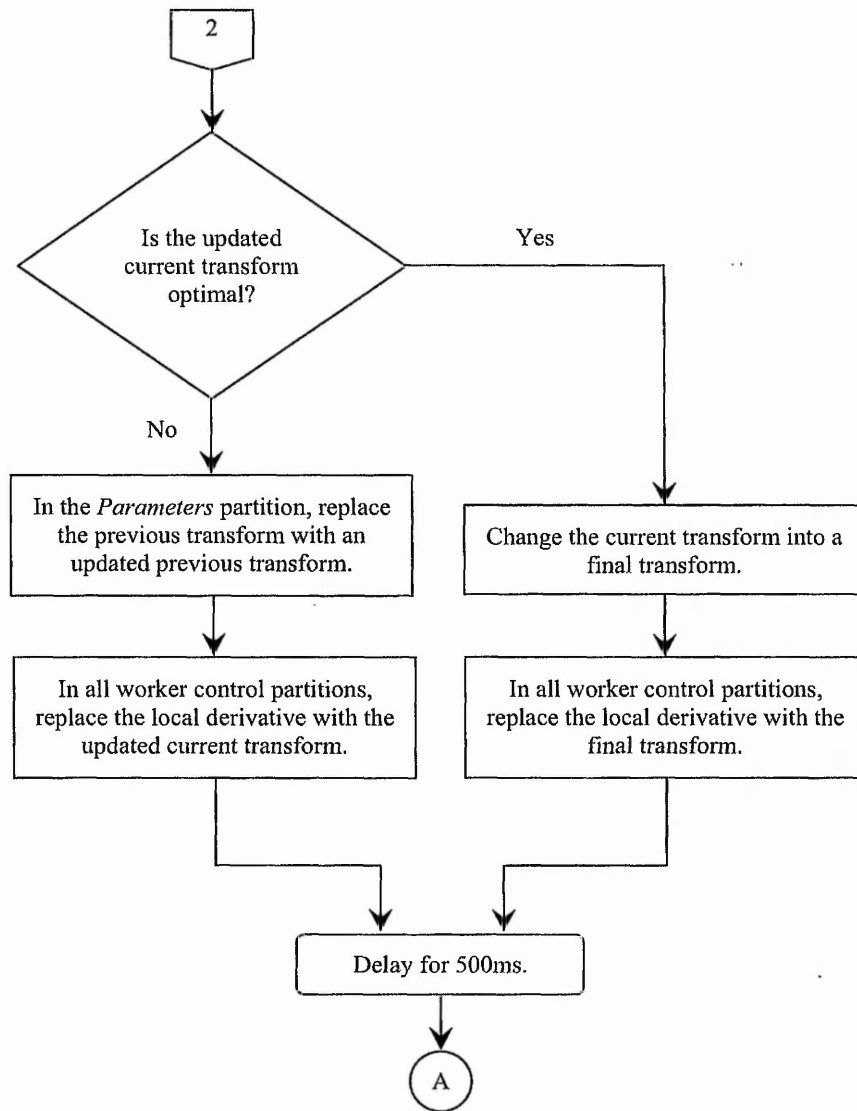


Figure 27: (Part C)

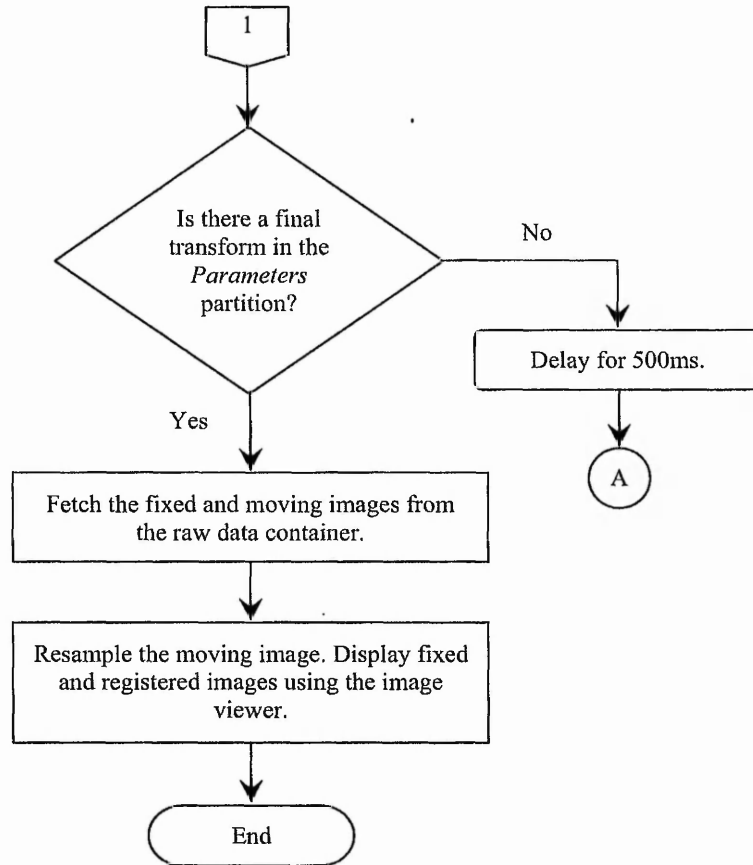


Figure 27: The updated Manager KS flow diagram. The diagram illustrates the iterative updating of transform parameters and their propagation to Worker KSs.

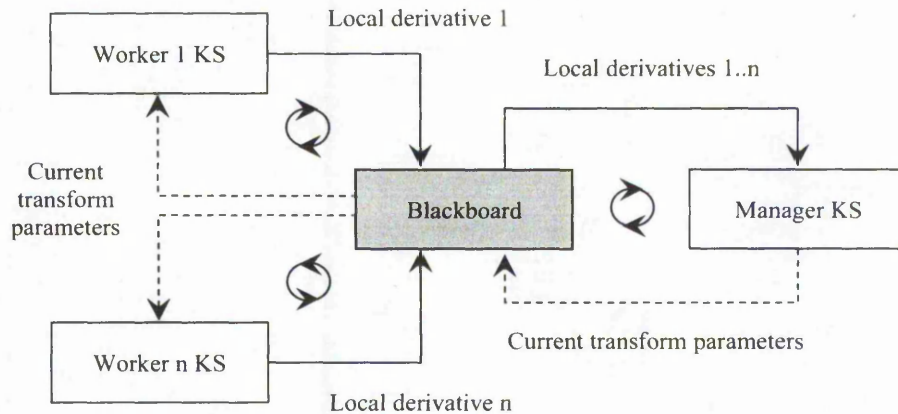
#### 4.3.3.1 Advancing transform parameters

Before the current transform can be updated and optimisation can proceed, a global derivative needs to be calculated. In order for this to be achieved, the Manager KS accumulates local derivatives each of which is then added to a global derivative with respect to each transform parameter. The global derivative is computed by dividing the accumulated derivatives by an accumulated valid pixel total.

To update the current transform, the previous transform held in the *Parameters* partition is retrieved. The global derivative recently accumulated is also employed. The gradient-descent optimisation scheme then advances the current transform in the direction of the global derivative. If the direction of the global derivative abruptly changes, it is assumed that an optimum has been encountered and step length is reduced by a half. After repeated iterations step length is reduced further and the selection of updated transform parameters is restricted to a small area of search space. Once step length becomes smaller than the predefined minimum, the optimisation process is considered as having converged. This allows the precision of the final transform to be specified. In general, large numbers of iterations resulting in long processing times are an indication that the initial step length chosen was too small. The selection of a large step length, in contrast, can result in the optimum transform being missed. If optimisation of the current transform fails to reach the desired precision, the maximum number of iterations is used to halt the optimisation process.

The accumulation of local derivatives by the Manager KS and the propagation of updated transform parameters to Worker KSs are illustrated in Figure 28. Importantly, the replacing of local derivatives with current transform parameters ensures that the local derivatives computed are based on the same transform. Also, to ensure that the current transform is based on local derivatives from the same iteration, local derivatives are replaced with current transform parameters using the `replace_multi` command. The `replace_multi` command represents an atomic instruction that maintains information consistency upon the blackboard. If the Manager and Worker KSs operate with current transform and derivative parameters from different iterations, a corrupt

path through search space will be followed. Similarly, if the Worker and Manager KSs become unsynchronised in the number of iterations performed, a state of deadlock is entered as both Worker and Manager KSs wait for current transform and derivative parameters to appear.



**Figure 28:** The flow of local derivatives and current transform parameters between iDARBS components. Derivatives are accumulated by the Manager KS while transform parameters are propagated to Worker KSs.

#### 4.3.3.2 Visualisation of registered images

Basic visual assessment of the fixed and registered images is achieved using a checkerboard composite that combines alternating segments. Coarse or fine-grained checkerboards can be constructed by changing the number of divisions into which images are separated. The resulting checkerboard composite makes visible the quality of alignment using foreground and background pixels. An image wide assessment can also be computed using the squared difference between corresponding fixed and registered pixels. In the resulting image, misalignment is made visible through the strength of intensities at locations where differences exist. Finally, by assigning the

fixed and registered images weights a transparent overlay can be generated. To construct the overlay, intensities are multiplied by weights and added to values at corresponding pixel locations. With the appropriate selection of weights, detail in both images can be combined into a single output. Alternatively, the detail contained within a single image can be highlighted. Checkerboard, squared difference, and weighted overlay images are provided in Appendix F.

### **4.4 Experimental testing**

Speed tests were conducted in order to demonstrate the performance increase of image registration in sequential and distributed processing environments. To achieve this, the spatial mapping of intensities was performed with an affine transform component. The transform allowed for rotation around an arbitrary centre, followed by translation, and scaling of fixed image pixel co-ordinates. Once transformed, B-spline interpolation was used to evaluate moving image intensities at non-grid locations. The interpolator resulted in intensities whose derivatives were spatially continuous in 2D space. To determine the accuracy of alignment between images, after application of the transform, mean square error and normalised correlation similarity metrics were selected for testing purposes. Optimisation of both similarity metrics, using a search space defined by transform parameters, was achieved with a gradient-descent optimisation scheme. Crucially, the distribution of two similarity metrics demonstrates the flexibility of the iDARBS framework.

The experiments were used to test the effects on performance of adding Worker KSs to the iDARBS framework and were based on those described in Chapter 3. To maintain consistency, the experiments represented an ideal case, i.e. one processor for the



blackboard, one processor for the Distributor KS, one processor for the Manager KS, and one processor for each Worker KS. A sequential algorithm, constructed of the same components was also used as a performance benchmark for comparison. Importantly, the high resolution images described in Chapter 3 were used for evaluation of the registration algorithms. Moving images were generated by resampling with a  $10^\circ$  degrees clockwise rotation about the origin, followed by a positive translation of 13 pixels in the x axis and a positive translation of 17 pixels in the y axis. The origin being the pixel located at the bottom left corner of the image. By artificially creating the moving images, ground truth parameters were known and could be compared with final transform parameters thus providing a numerical assessment of alignment accuracy. The results obtained allow the speed and efficiency achieved by the iDARBS framework to be highlighted.

#### 4.4.1 Results and discussion

Timing of an experiment started when the Manager KS propagated the current transform to all worker control partitions. Timing stopped when the Manager KS placed the final transform in the *Parameters* partition. The time required to register images was calculated by subtracting the start time from the stop time. Speed tests were performed three times, results were then combined and an average calculated. In all cases, selected images were divided by the Distributor KS into 1–14 segments and a 200-pixel wide border allocated. Each resulting segment pairs was assigned to an individual Worker KS before being sent to the blackboard. Logically, the hardware used for testing is that described in Chapter 3.

#### 4.4.1.1 Sequential vs distributed mean square error

Mean square error is a similarity measure computed over all pixels in both fixed and moving images. Significantly, calculation of mean square error is suited to images of the same modality and as a consequence, intensities at corresponding locations need to be similar. The metric is attractive because it is simple to compute and produces a relatively smooth search space. When alignment between images is poor large values are produced by the metric. Small values, in contrast, occur near optimum alignment. The distributed mean square error measure of similarity, computed by the iDARBS framework, is defined as

$$S(F, M, T) = \frac{\sum_{i=1}^R \left[ \sum_{j=1}^{Q_i} (F(x_{ij}) - M(T(x_{ij}, p)))^2 \right]}{\sum_{i=1}^R Q_i} \quad 4.8$$

where  $F$  and  $M$  are fixed and moving segment intensity functions respectively,  $T$  is the spatial transform used to map between segments, and  $x_{ij}$  is the  $j^{\text{th}}$  pixel of segment  $i$  from the fixed image.  $R$  is the number of segments an image is divided into and  $Q_i$  is the number of valid pixels transformed between segments identified by  $i$ . The derivative of the distributed similarity metric with respect to transform parameter  $p$  is computed using

$$\frac{\partial S}{\partial p} = \frac{2 \sum_{i=1}^R \left[ \sum_{j=1}^{Q_i} [F(x_{ij}) - M(T(x_{ij}, p))] \frac{\partial M(T(x_{ij}, p))}{\partial p} \right]}{\sum_{i=1}^R Q_i} \tag{4.9}$$

$$= \frac{2 \sum_{i=1}^R \left[ \sum_{j=1}^{Q_i} [F(x_{ij}) - M(T(x_{ij}, p))] \frac{\partial M(y)}{\partial y} \Big|_{y=T(x_{ij}, p)} \frac{\partial T(x_{ij}, p)}{\partial p} \right]}{\sum_{i=1}^R Q_i}$$

where  $M(T(x_{ij}, p))$  represents an intensity which has been interpolated using the B-spline interpolation scheme described in Equation 4.4. Also, where  $\partial T(x_{ij}, p)/\partial p$  is a transform Jacobian used to estimate variations in the mapped pixel co-ordinates with respect to transform parameter  $p$ , computed using Equation 4.3. During the alignment process, when a pixel location that map outside of the moving segment is encountered, the contribution to the local derivative is discarded.

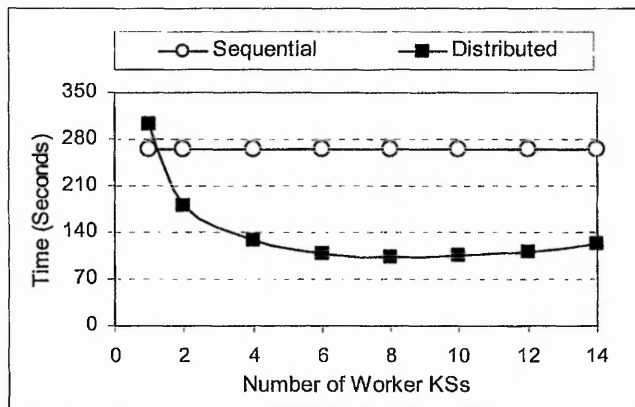


Figure 29: The sequential and distributed processing speed of image registration using mean square error as a similarity metric. Increasing numbers of Worker KS are shown.

Figure 29 shows the speed of sequential and distributed image registration plotted as time against number of Worker KSs, with mean square error as a similarity metric. As

can be seen the average execution time of distributed registration was reduced from four minutes and 30 seconds to approximately one minute and 45 seconds when ten Worker KSs were employed. For each test, the distributed metric converged after 19 iterations with transform parameters which matched those computed by the sequential algorithm.

#### 4.4.1.2 Sequential vs distributed normalised correlation

Normalised correlation is also suited to images of the same modality. The similarity metric works by computing the pixel-wise cross-correlation of the images to be registered. Once calculated, the cross-correlation is normalised by the square root of the autocorrelation of each image. Appealing properties of the metric include insensitivity to noise and the production of a search space containing sharp peaks and well defined troughs. The accurate alignment of images, results in values near one being produced by the metric. Misalignment, in contrast, produces values of less than one. The distributed normalised correlation measure of similarity, computed by the iDARBS framework, is defined as

$$S(F, M, T) = -1 \frac{\sum_{i=1}^R \left[ \sum_{j=1}^Q (F(x_{ij}) - M(T(x_{ij}, p))) \right]}{\sqrt{\sum_{i=1}^R \left[ \sum_{j=1}^Q F(x_{ij})^2 \sum_{j=1}^Q M(T(x_{ij}, p))^2 \right]}} \quad 4.10$$

where  $F$ ,  $M$ ,  $T$ ,  $R$ , and  $Q$  are the intensity functions, spatial transform, number of segment, and number of valid pixels previously defined. The derivative of the distributed normalised correlation similarity metric with respect to transform parameter  $p$  is computed using

$$\begin{aligned} \frac{\partial S}{\partial p} &= \frac{\sum_{i=1}^R \left[ \sum_{j=1}^{Q_i} \left( F(x_{ij}) \frac{\partial M(T(x_{ij}, p))}{\partial p} \right) - b \sum_{j=1}^{Q_i} \left( M(T(x_{ij}, p)) \frac{\partial M(T(x_{ij}, p))}{\partial p} \right) \right]}{a} \\ &= \frac{\sum_{i=1}^R \left[ \sum_{j=1}^{Q_i} \left( F(x_{ij}) \frac{\partial M(T(x_{ij}, p))}{\partial p} \right) - b \sum_{j=1}^{Q_i} \left( M(T(x_{ij}, p)) \frac{\partial M(y)}{\partial y} \Big|_{y=T(x_{ij}, p)} \frac{\partial T(x_{ij}, p)}{\partial p} \right) \right]}{a} \end{aligned}$$

with

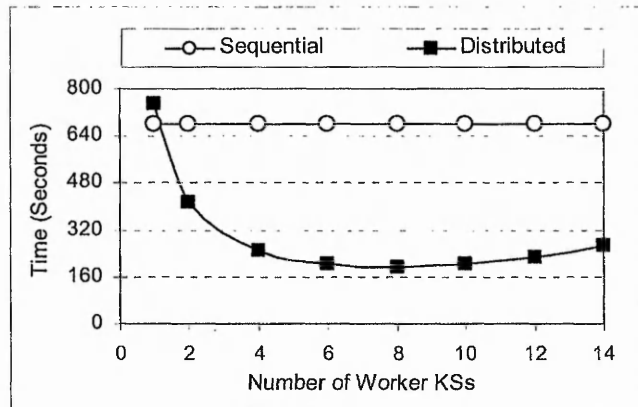
$$a = \sqrt{\sum_{i=1}^R \left[ \sum_{j=1}^{Q_i} F^2(x_{ij}) \sum_{j=1}^{Q_i} M^2(T(x_{ij}, p)) \right]} \quad 4.11$$

and

$$b = \frac{\sum_{i=1}^R \left[ \sum_{j=1}^{Q_i} F(x_{ij}) M(T(x_{ij}, p)) \right]}{\sum_{i=1}^R \left[ \sum_{j=1}^{Q_i} M^2(T(x_{ij}, p)) \right]}$$

where  $M(T(x_{ij}, p))$  and  $\partial T(x_{ij}, p)/\partial p$  are discrete intensities interpolated using B-spline interpolation and transform Jacobian previously defined. Again mapped pixels which lie outside of a moving segment do not contribute to the similarity measure.

Figure 30 shows results plotted as time against number of Worker KSs, obtained whilst performing image registration with normalised correlation as a similarity metric. The figure shows that processing time of the distributed algorithm was reduced from 12 minutes to approximately three minutes when eight Worker KSs were employed. For each test, the distributed metric was observed to converge after 31 iterations with transform parameters which matched those computed by the sequential algorithm. The increase in the number of iterations performed, when compared with mean square error, being caused by the different path through transform parameter search space followed.



**Figure 30: The sequential and distributed processing speed of image registration using normalised correlation as a similarity metric. Increasing numbers of Worker KS are shown.**

#### 4.4.2 Speedup and efficiencies achieved

The overall processing time of a registration algorithm will depend on initial alignment accuracy and the size of the images being registered. As a consequence, the time required to register two images can be seen as a function of the number of processing nodes employed. In general, the increase in processing speed due to parallelisation of an algorithm with multiple processors is governed by Amdahl's law [138] [139]. The law states that the distribution of an algorithm can be analysed in terms of sequential and parallel portions. The law also suggests that the maximum or ideal speed increase achievable through distribution of an algorithm with  $N$  processing nodes is an  $N$  times speedup. The maximum speedup, however, is always constrained by the sequential portion of the algorithm. In order to evaluate the speed increase achieved by the two distributed similarity metrics, speedup is calculated as

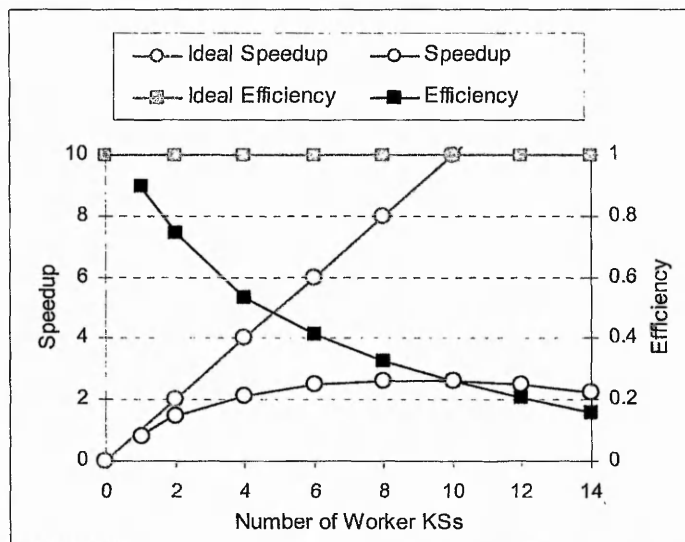
$$Speedup = \frac{t_s + t_p}{t_s + t_p / N} = \frac{1}{t_s + t_p / N} \quad 4.12$$

where  $t_p$  is the parallelisable portion of the algorithm and  $t_s$  is the sequential portion. During the alignment process,  $t_p$  is the time spent by the Worker KSs computing local derivatives.  $t_s$ , in contrast, represents the accumulation of local derivatives by the Manager KS, followed by the updating of transform parameters, and their propagation to Worker KSs. The amount of speedup achieved by each Worker KS is called efficiency [140] and is calculated using

$$Efficiency = \frac{Speedup}{N} \quad 4.13$$

where *Speedup* is the speed improvement calculated using Equation 4.12 and  $N$  is the number of processing nodes or Worker KSs previously defined. Both speedup and efficiency should be calculated using averaged registration times.

Speedup and efficiency, plotted against number of Worker KSs, achieved during distributed image registration with mean square error as a similarity metric are shown in Figure 31. Ideal speedup and efficiency rates as suggested by Amdahl's law are also shown. As can be seen a peak speedup factor of 2.5 and an efficiency of 25% was achieved by ten Worker KSs. The results obtained show that as the numbers of Worker KSs grow speedup steadily increases to a peak, a deterioration in performance then occurs. Understandably, the initial speed increase is due to the small number of Worker KSs involved in similarity calculation. The deterioration in performance, in contrast, is associated with the overheads of managing increasing numbers of Worker KSs.



**Figure 31: The speedup and efficiency achieved by distributed image registration using mean square error as a similarity metric. Increasing numbers of Worker KSs are shown.**

Speedup and efficiency, plotted against number of Worker KSs, achieved during distributed image registration with normalised correlation as a similarity metric are shown in Figure 32. The ideal speedup and efficiency rates as suggested by Amdahl's law are again shown. As can be seen a peak speedup factor of 3.5 and an efficiency of 43% was achieved by eight Worker KSs. The results obtained show that efficiency steadily deteriorates as the number of Worker KSs increases. This is an indication that the Worker KSs are only partly utilised during the alignment process. A cause of under-utilisation is the wait by Worker KSs, until local derivatives have been accumulated and updated transform parameters have been calculated by the Manager KS. As discussed in Section 3.5.2, additional inefficiencies are also introduced through the polling of the blackboard approximately every 500ms by idle Worker KSs. Both the forced wait and



blackboard polling are amplified as the image segments get smaller and the number of Worker KSs increases.

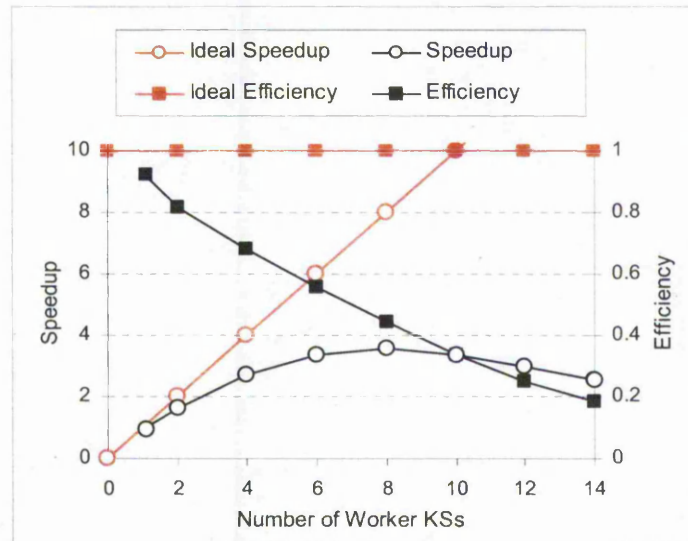


Figure 32: The speedup and efficiency achieved by distributed image registration using normalised correlation as a similarity metric. Increasing numbers of Worker KSs are shown.

## 4.5 Conclusions

Using the iDARBS framework, distributed image registration with two different similarity metrics has been demonstrated. The experimental results obtained are a clear indication that high performance intensity-based image registration can be achieved using non-specialised architectures. In general, the computational burden of a sequential similarity metric  $C_s$  which is required to transform all pixels within the fixed image to moving image co-ordinates, is defined as

$$C_s = C_g + N_i N_p (C_m + C_j) \quad 4.14$$

where  $C_g$  represents the complexity of computing moving image gradients,  $N_i$  is the number of iterations performed on convergence of optimisation, and  $N_p$  is the number of pixels considered valid.  $C_m$  represents the complexity of computing the contribution of one pixel to the metric derivatives and  $C_j$  represents the complexity of computing the transform Jacobian during a single iteration of the optimisation process. The computational burden of a similarly metric distributed using the iDARBS framework  $C_d$  in contrast, is defined as

$$C_d = \frac{C_g}{N_s} + N_i \frac{N_p}{N_s} \left( \frac{C_m}{N_s} + \frac{C_j}{N_s} \right) \quad 4.15$$

where  $C_g$ ,  $N_i$ ,  $N_p$ ,  $C_m$ , and  $C_j$  are as previously stated and  $N_s$  is the number of segments an image is divided into. When comparing the burden associated with sequential and distributed similarity calculation, as defined in Equations 4.14 and 4.15, it is clear that the saving in processing time becomes increasingly significant as the numbers of image segments grow. This has been confirmed by the experimental results obtained from distributed mean square error and normalised correlation similarity metric testing. Because  $C_m$  depends on the number of image dimensions and  $C_j$  is influenced by the number of transform parameters, it is also obvious that processing time will be reduced during the alignment of volume datasets.

Speedup which reaches a maximum and slowly drops away is a general trend observed for distributed image registration using the iDARBS framework. Understandably, as the numbers of Worker KSs grow the numbers of queries made to the blackboard also increase. Crucially, each time a query is made by either Worker or Manager KSs the blackboard is interrupted. If an interrupting query cannot be serviced immediately it is

added to a queue. As a consequence, a peak speedup is reached when the overheads caused by interrupting queries begin to counteract the benefits of parallel implementation. An overhead also identified as reducing speedup is the formatting of transform and derivative parameters into strings. Although the overheads are constant as both transform and derivative parameters remain the same length, the overheads incurred are compounded by the iterative nature of the alignment process. Finally, the high number of iterations and hence larger cumulative overhead, associated with the normalised correlation metric causes the speedup it achieves to drop at a faster rate than the mean square error metric.

#### **4.6 Summary**

To achieve high performance image registration, the iDARBS framework described in Chapter 3 has been extended. Initial transform parameters were estimated by the Distributor KS using centres of mass computed from intensity levels. The Distributor KS then divided selected images into segments which were placed on the blackboard. Once triggered, Worker KSs retrieved segments and computed local derivatives with respect to each transform parameter. A contribution to a local derivative being the summation of intensities, from a gradient image, around mapped pixel co-ordinates. The accumulation of local derivatives was performed by the Manager KS. During the updating of transform parameters, once step length through search space had become smaller than a predefined minimum, the optimisation process was considered as having converged. On failure of convergence, updated transform parameters were propagated to all Worker KS and the process was repeated. Successful convergence, in contrast, resulted in the generation of final transform parameters and the resampling of the moving image.

For testing of the extended iDARBS framework, an intensity-based image registration algorithm was constructed. The algorithm was chosen as it represented a foundation on which more complex functionality could be added. Components of the algorithm included an affine transform, B-spline interpolation as well as mean square error and normalised correlation similarity metrics. The similarity metrics were chosen in order to demonstrate the flexibility of iDARBS, in handling different registration algorithm components. Computed over all pixels in both images, the mean square error metric was simple to compute and produces a relatively smooth search space. During testing, the distributed metric achieved a maximum speedup factor of 2.5 and an efficiency of 25%. Also computed over all pixels in both images, the distributed normalised correlation metric produced a maximum speedup factor of 3.4 and an efficiency of 43%. Tables of experimental results are provided in Appendix G.

## 5 Single and multi-modal volume registration

The distribution of an intensity-based image registration algorithm was introduced in Chapter 4. Functionality for distributed similarity calculation was presented and modifications to knowledge source (KS) behaviour were discussed in detail. Parallelisation of the algorithm was achieved through the distribution of transform parameters and accumulation of locally computed derivatives, where both sets of parameters were encoded as short concise information strings. Using the resulting approach, the performance bottleneck associated with image resampling and similarity calculation was shown to be successfully alleviated. Although the results obtained demonstrate that non-specialised architectures can be employed, they do not fully address the limitations of restricted flexibility imposed by fine-grained parallelism. The distribution of single and multi-modal volume data would provide greater justification for the iDARBS (imaging Distributed Algorithmic and Rule-based Blackboard System) implementation by demonstrating a large degree of flexibility.

In this chapter, the iDARBS framework and distributed registration algorithm introduced in Chapter 4 are extended to single and multi-modal volume data. Section 5.1 provides a short introduction to volume registration and identifies the components involved in 3D registration. In Section 5.2, modifications to Distributor, Worker, and Manager KSs are outlined. Experimental testing of distributed single-modal volume registration is discussed in detail in Section 5.3. The concept of mutual information as a similarity measure is introduced in Section 5.4. In Section 5.5, functionality developed for distributed similarity calculation using mutual information is outlined. Experimental testing of distributed multi-modal volume registration is presented in Section 5.6, conclusions in Section 5.7, and a short summary in Section 5.8.

Results obtained confirm that the time constraints associated with volume registration can be significantly improved when compared with sequential approaches. The implementation of multi-modal volume registration also successfully demonstrates flexibility of the coarse-grained parallelism employed.

### **5.1 Single-modal volume registration**

The ability to visualise hidden structures in detail using a 3D image or volume has become a valuable resource in both medicine and manufacturing. Typically, capture devices generate an image where pixels represent values within a regular grid in 2D space. A volume, in contrast, is formed by stacking together multiple image slices. As a consequence, the intensities within a volume correspond to small areas commonly referred to as voxels. Unlike pixels, the voxels represent values within a regular grid in 3D space. Due to advances in technology it is common for data to be acquired directly as a volume, an example of this is the well known Visible Human Project® [141]. Understandably, as with image registration the alignment of volumes makes possible the combining of different structural and functional information for diagnosis and planning purposes.

As registration applications shift from alignment of images to the processing of 3D data, volume registration has become a well recognised method [142] [143]. Although almost identical in implementation terms to correlation-based image registration, correlation-based volume registration is considerably more computationally intensive. This is because the data involved in similarity calculation grows by a degree of freedom. For example, the interpolation of intensities at non-discrete locations has an extra dimension

as does the transformation between reference (fixed) and sensed (moving) intensity co-ordinates. The registration of volumes was considered a logical choice for implementation as the long processing times would allow the speed increases of a distributed processing architecture to be demonstrated. Using normalised correlation as a similarity metric, the algorithm would be designed for the alignment of data captured using the same sensor type and hence same modality.

The inputs to a volume registration algorithm can be described as the fixed and moving data as well as a transform used to map between voxel co-ordinates. The goal of the registration process is the recovery of a spatial mapping that brings the two volumes into alignment. Conveniently, the transform component can be implemented as a Quaternion [144] which expresses the rotational relationship between two vectors. The Quaternion makes possible the retrieval of one vector by operating with the other. To achieve this, the orientation of the first vector in relation to a second vector is described using a versor. The change in magnitude between the two vectors, in contrast, is encapsulated as a tensor. Since the versor represent an orientation change it provides a convenient representation of rotations in 3D space. When the versor is coupled with a translation vector, a rotation and translation in 3D space can be achieved. Using the versor type, the voxel location  $x$  in the fixed volume is mapped to the new position  $x'$  in the moving volume using

$$x' = V * x + t \quad 5.1$$

where  $V$  is a versor,  $*$  stands for the multiplication of the versor with a vector, and  $t$  is a translation vector. In general, the versor consists of three components including direction, angle, and norm. The direction of a versor is taken as being parallel to the axis

around which the first vector is rotated, in order to be mapped to the second vector, and orthogonal to the plane defined by the two vectors. The angle, in contrast, is the measure of rotation between the two vectors for which the versor is a Quaternion. The norm of the versor is defined as a function of its rotation angle.

As the behaviour of versor and vector components are different, traditional gradient-descent optimisation schemes cannot be employed. This is because the versor addition operation does not correspond to the notion of accumulation employed in vector addition [145]. Optimisation is therefore performed with a versor rigid 3D transform optimiser component. Rotational updates are made to the current versor through addition with a versor that represents the change in angle determined by the optimiser. When described formally, at the end of each optimisation cycle updates to versor  $V$ , from Equation 5.1, are computed using

$$V' = dV + V \quad 5.2$$

where  $V'$  is the updated versor to be used in the next optimisation cycle,  $dV$  is the variation determined by the optimiser, and  $+$  stands for the addition of versors. The derivative  $dV$  is computed using

$$dV_i = \left[ \frac{\partial S(V_i)}{\partial V_i} \right] \lambda \quad 5.3$$

where  $\partial S(V_i)/\partial V_i$  is a derivative of the similarity measure with respect to versor component  $V_i$ , weighted by step length  $\lambda$ . Importantly, the derivative represents how much similarity  $S$  changes with variations in  $V$ . The transform vector  $t$ , from Equation 5.1, is updated using



$$t = C - V' * C \quad 5.4$$

where  $C$  is a fixed point used for reference throughout the optimisation process,  $V'$  is the updated versor described in Equation 5.2, and  $*$  stands for the multiplication of the versor with a vector.

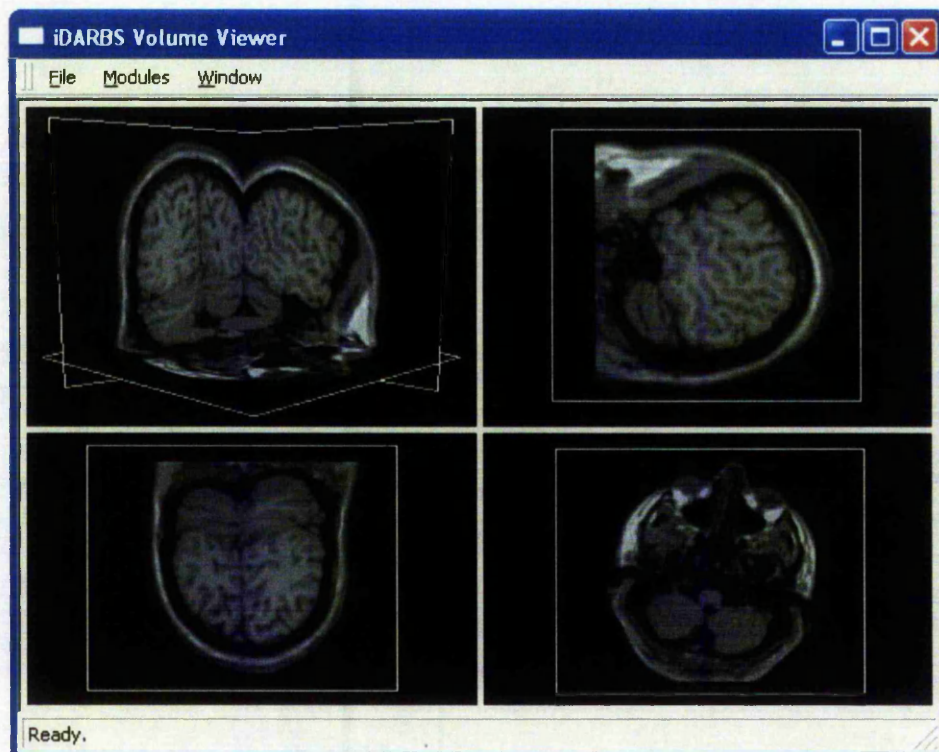
## 5.2 Single-modal volume registration on iDARBS

To perform intensity-based registration of volumes, the underlying behaviour of Distributor, Worker, and Manager KSs was modified. The mapping of intensities between fixed and moving segments by Worker KSs is achieved using a versor rigid 3D transform component. The transform represents a 3D rotation and translation, where the rotation is specified by a (unit) Quaternion and the translation is implemented as a vector. B-spline interpolation is also employed by Worker KSs to evaluate moving volume intensities at non-grid co-ordinates. Understandably, the intensities produced by the interpolator have derivatives which are spatially continuous in 3D space. Once similarity between volumes has been computed, versor rigid 3D transform optimisation is employed by the Manager KS to search for new transform parameters. An adaptation of gradient-descent optimisation, the optimiser combines the current rotation with the computed gradient to produce a new rotation versor. Translation parameters are, however, simply updated in vector space.

### 5.2.1 Visualisation of registered volumes

For selected volumes to be rendered, a 3D viewer was constructed. The viewer consists of a single perspective and three orthographic projections in the  $x$ ,  $y$ , and  $z$  planes. A volume loaded into the viewer can be displayed as either a complete object or as slices.

When viewed as a complete object, only the surface of a structure is visible. When viewed as slices, the internal detail of a structure is presented. For convenience, a dialog box allows movement between slices within a selected volume. In both cases the scene viewed can be rotated, translated, and zoomed using simple mouse interactions.



**Figure 33: A volume rendering using the 3D viewer. Perspective and orthographic projections of a human head are shown.**

Figure 33 shows a volume rendering using the 3D viewer. By extracting individual slices from fixed, moving, and registered volumes, visual assessment of alignment accuracy can be achieved using checkerboard, squared difference, and weighted overlay images. Volume overlay, where each dataset is assigned a different colour channel is also provided and given as an example in Appendix H.

### 5.3 Experimental testing

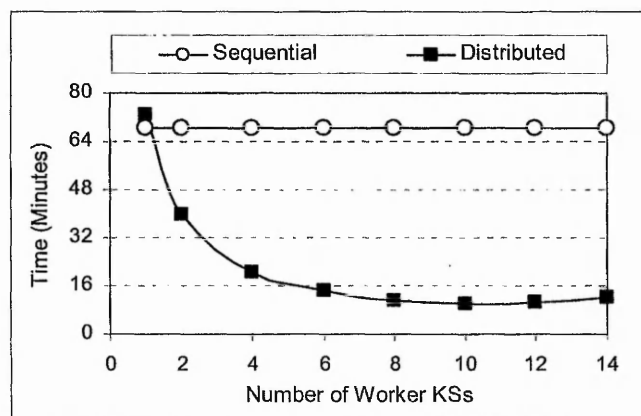
Many capture devices are employed in the field of medicine. Magnetic Resonance Imaging (MRI) is a diagnostic imaging modality which has the ability to derive contrast from a number of physical parameters [146]. An MRI scanner consists of a large magnet, microwave transmitter, and microwave antenna. During a scan, the patient is placed within a high intensity magnetic field. This causes the magnetic moments of hydrogen atoms within the body to align in the direction of the magnetic field. Low-level radio waves are then transmitted through the body causing the magnetic moments to resonate and emit microwaves. Microwaves emitted by the body are recorded using the microwave antenna. The signal recorded is filtered, amplified, and reconstructed into a cross-sectional image. Importantly, the capture of different anatomical structures is made possible through the selection of a pulse sequence and is achieved by varying the magnetic field emitted by the scanner. Conveniently, MRI volumes specifically designed for the testing of new registration algorithms can be freely obtained [147] [148] [149] and hence were deemed suitable for testing purposes.

The pulse sequence, spin-lattice relaxation (T1), is an MRI imaging modality used to highlight grey matter contained within the body. For testing of the distributed algorithm, two T1 volumes were obtained from the McConnell Brain Imaging Centre [150]. The McConnell Brain Imaging Centre website is an open access simulated brain database that contains realistic MRI volumes produced by an MRI simulator. Selection of three acquisition parameters allows realistic MRI volumes of the brain to be acquired. The parameters include modality, slice thickness, and noise content. Both volumes obtained are based on an anatomically normal brain, have a slice thickness of 1mm, and noise

content of 3%. Both volumes are also  $181 \times 217 \times 181$  voxels in size. The moving volume represents a dataset which has been rotated 10 degrees clockwise about the origin and translated 15 voxels in the  $x$  axis. The origin being the voxel located at the bottom left corner of the volume.

### 5.3.1 Results and discussion

During testing, the timing of each experiment started when the Manager KS propagated transform parameters to all worker control partitions. Timing stopped when the Manager KS placed final transform parameters in the *Parameters* partition. The time required to register volumes was calculated by subtracting the start time from the stop. Testing was performed three times, results were then combined and an average calculated. In all cases, selected volumes were divided by the Distributor KS into 1–14 segments and a 20-voxel wide border was allocated.



**Figure 34:** The sequential and distributed processing speed of single-modal volume registration using normalised correlation as a similarity metric. Increasing numbers of Worker KSs are shown.

Figure 34 shows results plotted as time against number of Worker KSs, obtained whilst performing single-modal volume registration with normalised correlation as a similarity metric. The figure shows how processing time of the distributed algorithm was reduced from 68 minutes to approximately ten minutes when ten Worker KSs were employed. The distributed algorithm was observed to converge after 54 iterations with transform parameters which matched those computed by the sequential algorithm.

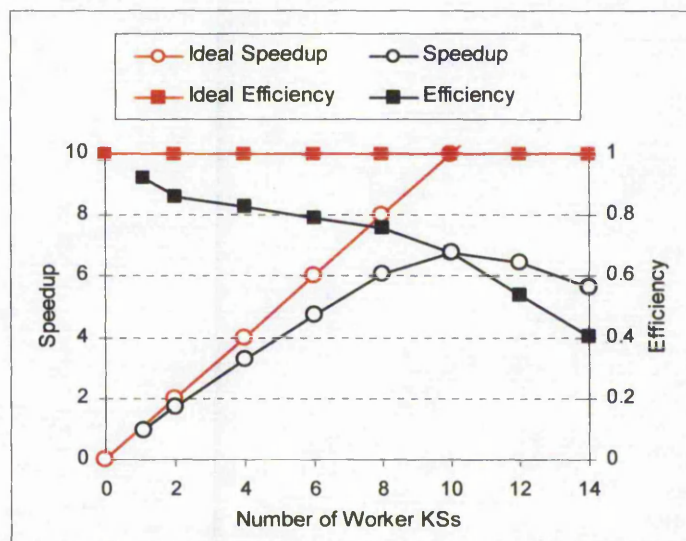


Figure 35: The speedup and efficiency of single-modal volume registration using normalised correlation as a similarity metric. Increasing numbers of Worker KSs are shown.

In Figure 35 speedup and efficiency, as outlined in Section 4.4.2, of distributed single-modal volume registration with normalised correlation as a similarity metric are shown. As can be seen a peak speedup factor of seven and an efficiency of 67% was achieved by ten Worker KSs. This is roughly twice the speedup and an increase in efficiency of approximately 25% when compared with the peak performance of distributed image



registration. Both the increase in speedup and efficiency are a clear indication that Worker KSs are better utilised during volume registration than during image registration. The improved performance can be attributed to the fact that the proportion of time a Worker KS spends idle has become smaller, while the proportion of time spent processing intensities has increased. The increased utilisation also reduces the number of times Worker KSs poll the blackboard and introduce unnecessary communications overheads. Conveniently, only simple modifications to the Distributor, Worker, and Manager KSs were required. As a consequence, Worker KSs continue to generate local derivatives while the Manager KS updates transform parameters and supervises activities.

#### **5.4 Multi-modal volume registration**

In multi-modal registration, the data to be aligned stem from two different capture devices. This is in contrast with single-modal registration where data are retrieved using the same sensor type. Importantly, the registration of volumes from differing modalities with traditional correlation-based similarity metrics is inadvisable and unreliable. It has, however, been extensively shown [151] [152] that metrics based on the evaluation of mutual information are well suited to such problems. Derived from information theory, the concept of mutual information has been proposed in a number of different forms. The most common being that mutual information is a qualitative measure of how much information can be obtained about a random variable from the knowledge of another random variable [153]. The main advantage of employing mutual information is that the type of dependency between two variables does not have to be specified and as a result complex mappings can be modelled. In a registration context, as no assumptions about

the nature of the capture device need to be made, an algorithm can be generalised and applied to data from a variety of modalities.

Given the fixed volume  $F$ , moving volume  $M$ , and transform  $T$  the calculation of similarity  $S$  using mutual information, as described by Mattes *et al.* [154], is defined as

$$S(F, M, T) = \sum_{l=1}^{MB} \sum_{k=1}^{FB} jpd(l, k) \log \frac{jpd(l, k)}{mmpd(l) fmpd(k)} \quad 5.5$$

where  $jpd$  is a joint probability distribution extracted from both volumes using a set of samples and approximated as a histogram of intensities.  $fmpd$  and  $mmpd$  are marginal probability distributions extracted from fixed and moving volumes respectively. While  $k$  and  $l$  are the indices of fixed and moving histogram bins, the totals of which are defined as  $FB$  and  $MB$ .

Each entry in the joint probability distribution denotes the number of times an intensity in the fixed volume coincides with intensities in the moving volume. The marginal probability distributions, in contrast, are found by summing the number of times intensities appear in their respective volumes. In general, it is common for contributions to a probability distribution to be smoothed using a probability density function such as Parzen windowing [155]. Parzen windowing is a simple convolution scheme that places a kernel over the bin into which a contribution is made and updates the histogram using corresponding kernel coefficients. Once generated, both joint and marginal probability distributions are normalised and an estimation of similarity calculated. In this form, mutual information is a measure of the distance between the joint and marginal

distribution of intensities. Mutual information is therefore considered a measure of dependence which is maximal when the two volumes are accurately aligned.

Formulation of the joint probability distribution  $jpd$  is defined as

$$jpd(l, k) = \alpha \sum_{i=1}^Q \beta^{(3)} \left( k - \frac{F(x_i) - f^{\wedge}}{\Delta b_f} \right) \beta^{(3)} \left( l - \frac{M(T(x_i, P)) - m^{\wedge}}{\Delta b_m} \right) \quad 5.6$$

where  $\alpha$  is a normalisation factor based on multiplying bin size by the number of valid samples.  $Q$  is the number of valid samples, indexed by  $x_i$ , transformed between volumes.  $f^{\wedge}$  and  $m^{\wedge}$  are the minimum intensities levels of the fixed and moving volumes respectively.  $\Delta b_f$  and  $\Delta b_m$  represent the intensity ranges covered by the fixed and moving histogram bins. While  $\beta^{(3)}$  is a cubic B-spline Parzen window. The fixed marginal probability distribution  $fmpd$  is defined as

$$fmpd(k) = \alpha \sum_{i=1}^Q \beta^{(0)} \left( k - \frac{F(x_i) - f^{\wedge}}{\Delta b_f} \right) \quad 5.7$$

where  $\alpha$  is a normalisation factor computed by dividing the value of each histogram bin by the total of all histogram bin values and  $\beta^{(0)}$  is a zero order B-spline Parzen window.

The moving marginal probability distribution  $mmpd$  is defined as

$$mmpd(l) = \sum_{k=1}^{FB} jpd(l, k) \quad 5.8$$

where  $l$ ,  $FB$ , and  $jpd$  are the index, total number of fixed histogram bins, and joint probability distribution previously defined.



## 5.5 Multi-modal volume registration on iDARBS

Using the method outlined in Section 5.4, the iDARBS framework was extended to perform volume registration with mutual information as a similarity metric. Framework initialisation and volume selection remain the responsibility of the Distributor KS. In order for optimum transform parameters to be computed, the Distributor KS generates random samples before dividing selected volumes into segments. Upon activation, each Worker KS retrieves volume segments and a list of corresponding samples from the blackboard. Each sample is then evaluated and local probability distributions generated. The accumulation of local contributions and construction of global probability distributions is performed by the Manager KS. Once the global probability distributions have been constructed, derivatives of the similarity measure with respect to each transform parameter are extracted. Updated transform parameters are then calculated and propagated to the Worker KSs. The updating of transform parameters is repeated until predefined thresholds are exceeded.

### 5.5.1 Sample generation by the Distributor KS

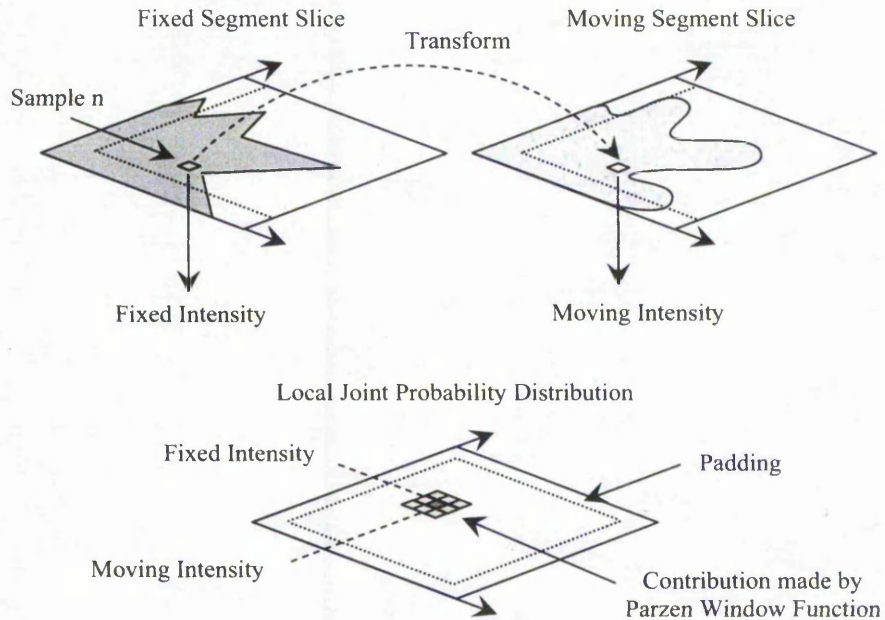
To ensure that the histograms on which probability distributions are based are consistent, minimum and maximum intensity levels as well as bin size are computed by the Distributor KS. The minimum and maximum intensity levels are extracted from selected volumes before division into segments. Bin size, in contrast, is computed as the difference between minimum and maximum intensity levels divided by the number of histogram bins employed. Padding is added to the bin size and removes the effects of boundary conditions introduced during smoothing of local probability distributions. As a consequence, the increased bin size results in histograms which are shifted to one side. To counter shifting, all sampled intensities are incremented by the padding amount.

This stops minimum values from entering invalid regions of a local histogram. Once calculated, minimum and maximum intensity levels as well as bin size are placed in the *Parameters* partition by the Distributor KS.

Before the selected volumes are divided into segments, random samples in the form of co-ordinates are also generated by the Distributor KS. As soon as the volumes have been divided each sample is mapped to a corresponding segment. The co-ordinates of each sample are then updated to the co-ordinate system of the assigned segment. The mapping of co-ordinates to individual segments results in lists of samples. Once generated, each sample list is assigned to a Worker KS and placed in its corresponding worker control partition.

### 5.5.2 Local probability distribution generation by the Worker $n$ KS

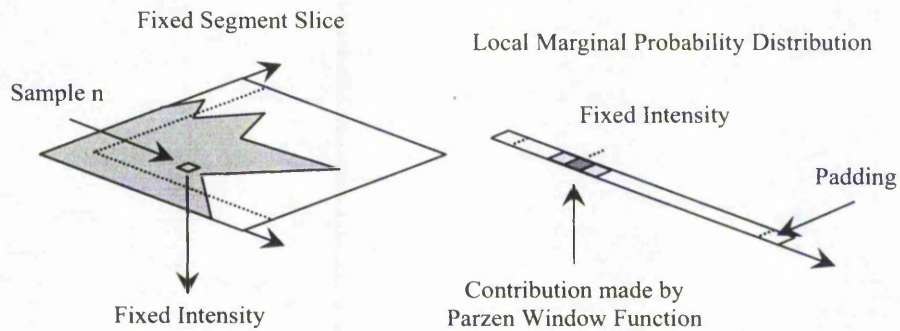
In order for similarity between segments to be calculated, new functionality has been added to the Worker  $n$  KS. First a sample list and corresponding segments are retrieved from the blackboard. The retrieved samples are then used to generate local probability distributions which are held by the Worker  $n$  KS. Once generated, the local probability distributions are placed in the *Worker  $n$  control* partition. To maintain consistency, the local probability distributions are constructed using the minimum and maximum intensity levels as well as bin size found in the *Parameters* partition. By clearing the local probability distributions at the end of each optimisation cycle, the integrity of histogram data is also preserved. Conveniently, the use of sparse histograms during the transferral of local probability distributions results in the quantity of data placed on the blackboard being significantly reduced.



**Figure 36: Local joint probability distribution construction. Fixed and moving segment intensities are used to determine into which histogram bins a contribution is made by a Parzen window function.**

To generate the local joint probability distribution, for each sample in the fixed segment, corresponding moving segment co-ordinates are computed using the current transform parameters. If transformation of the fixed sample co-ordinates results in a location that falls inside the moving segment, the number of valid samples is incremented and a contribution to the joint probability distribution is made. Otherwise the sample is considered invalid and the next sample is evaluated. As shown in Figure 36, an appropriate histogram bin is determined using fixed and moving intensities. Significantly, both intensities are scaled to fit within the minimum and maximum range

then incremented by the padding amount. Once a contribution has been made, the histogram is smoothed using a B-spline Parzen window function.



**Figure 37: Local marginal probability distribution construction. Intensities from a single volume are used to determine into which histogram bins contributions are made by the Parzen window function.**

Figure 37 shows how contributions to a local marginal probability distribution are determined using the intensity of a sample. As the marginal probability distribution represents a 1D histogram, a single volume is used in its generation. Conveniently, this makes generation of the fixed marginal probability distribution independent of transform parameters. As with the joint probability distribution, intensities are scaled to fit within the minimum and maximum range then incremented by the padding amount. The histogram is smoothed using a B-spline Parzen window function once a contribution has been made.

### 5.5.3 Global probability distribution generation by the Manager KS

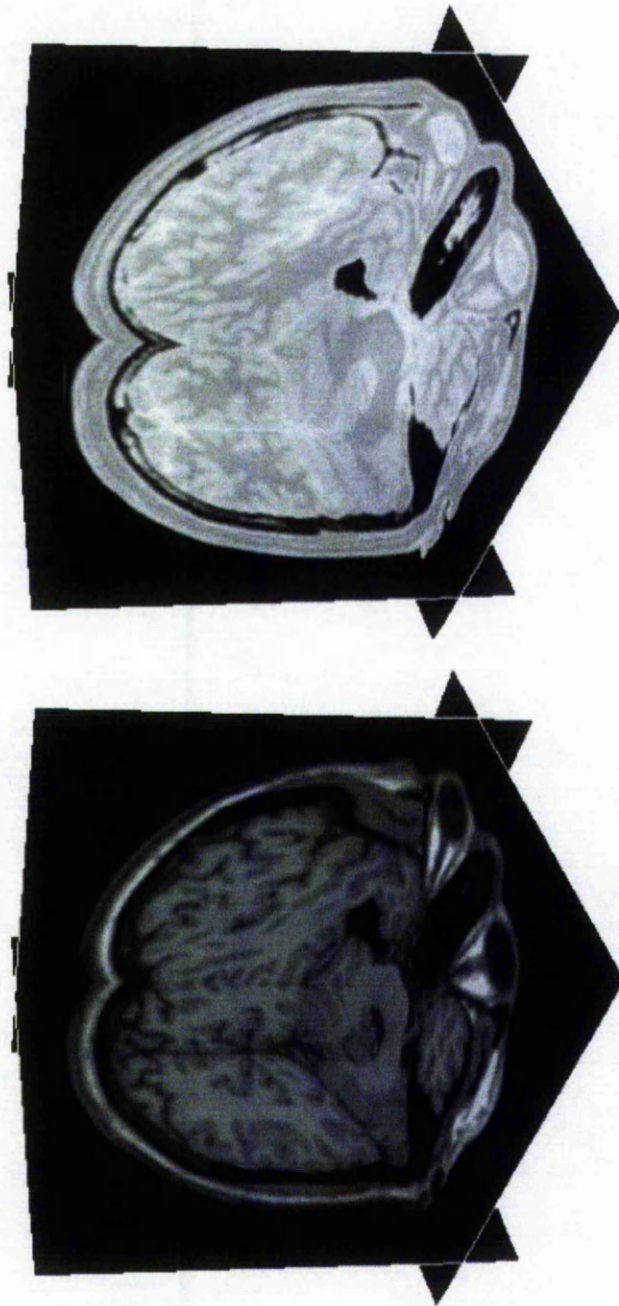
Global probability distributions are constructed by the Manager KS once initialised, using minimum and maximum intensities levels as well as bin size fetched from the

*Parameters* partition. Contributions to the global probability distributions are then made through the accumulation of local histograms generated by the Worker KSs. Once all contributions have been made, both joint and marginal global probability distributions are normalised. As set out in the mutual information definition, the joint probability distribution is normalised using a factor calculated by multiplying bin size by the number of valid samples. Normalisation of the marginal probability distributions, in contrast, is achieved by dividing the value of each histogram bin by the total of all bin values. To preserve the integrity of histogram data the global probability distributions are cleared at the end of each optimisation cycle.

## 5.6 Experimental testing

Two parameters determine the accuracy of statistics generated by the joint and marginal probability distributions. The first being the number of histogram bins used for probability distribution construction and the second being the number of spatial samples drawn upon. Conveniently, these two parameters have been investigated by a number of researchers [156] [157]. High numbers of histogram bins have been shown to result in continuous distributions that reduce signal to noise ratio and accurately describe sample contributions. Small histogram bin numbers, in contrast, limit resolution and reduce the quality of separation between clustered intensity combinations. Understandably, the number of samples drawn upon is dependent on the content of a volume. As a guide, highly detailed volumes require approximately 20% of their voxels to be used as samples. Only 1% of voxels, in contrast, are to be used when volumes contain low detail. To balance computational burden with alignment accuracy, 625 histogram bins and 100000 samples were used for testing purposes.





**Figure 38: MRI spin-spin relaxation (T2) and proton spin density (PD) volumes. Both volumes were used in testing of the distributed mutual information metric.**

Spin-spin relaxation (T2) and proton spin density (PD) are two pulse sequences used in the creation of MRI datasets. These imaging modalities are used to highlight white matter and cerebrospinal fluid contained within the body. Differences in structure highlighted by these modalities can be correlated with age and sex. A range of conditions including heart and vascular disease, cancer, joint and musculoskeletal degeneration can also be identified. For testing of the distributed mutual information metric, T2 and PD volumes were obtained from the McConnell Brain Imaging Centre website [150]. Both volumes obtained were based on an anatomically normal brain, have a slice thickness of 1mm, and noise content of 3%. Both volumes are also  $181 \times 217 \times 181$  voxels in size. The moving volume represents a dataset which has been rotated 10 degrees clockwise about the origin and translated 15 voxels in the  $x$  axis. Screenshots of the fixed T2 and moving PD volumes selected for testing purposes are provided in Figure 38.

### 5.6.1 Results and discussion

Mutual information is a measure of similarity calculated using samples extracted from fixed and moving volumes respectively. Mutual information as a metric is computationally attractive as only a subset of intensities requires evaluation. Conveniently, similarity calculation is suited to the alignment of data captured using different sensor types and hence dissimilar modalities. When alignment between volumes is poor, large values are produced by the metric. Small values, in contrast, occur near optimal alignment. The distributed mutual information measure of similarity, computed by the iDARBS framework, is defined as

$$S(F, M, T) = \sum_{i=1}^R \left[ \sum_{l=1}^{MB} \sum_{k=1}^{FB} jpd(i, l, k) \log \frac{jpd(i, l, k)}{mmpd(i, l) fmpd(i, k)} \right] \quad 5.9$$

where  $F$  is the fixed volume,  $M$  is the moving volume, and  $T$  is a spatial transform.  $R$  is the number of segments a volume is divided into, each segment having the index  $i$ .  $MB$  is the total number of moving volume histogram bins each bin having the index  $l$  and  $k$  is the index of a fixed volume histogram bin, the total of which is given as  $FB$ .  $jpd$ ,  $fmpd$ , and  $mmpd$  are the joint, fixed, and moving probability distributions described in Equations 5.6, 5.7, and 5.8 respectively. The derivative of the distributed similarity metric with respects to transform parameter  $p$  is computed using

$$\frac{\partial S}{\partial p} = \frac{1}{\Delta b_m \sum_{i=1}^R Q_i} \sum_{i=1}^R \sum_{j=1}^{Q_i} \left[ \begin{array}{c} \sum_{l=1}^{MB} \sum_{k=1}^{FB} \beta^{(0)} \left( k - \frac{F(x_{ij}) - f^{\wedge}}{\Delta b_f} \right) \\ \beta^{(3)} \left( l - \frac{M(T(x_{ij}, p)) - m^{\wedge}}{\Delta b_m} \right) \\ \left( -\frac{\partial M(T(x_{ij}, p))}{\partial p} \right)^T \end{array} \right] \quad 5.10$$

$$= \frac{1}{\Delta b_m \sum_{i=1}^R Q_i} \sum_{i=1}^R \sum_{j=1}^{Q_i} \left[ \begin{array}{c} \sum_{l=1}^{MB} \sum_{k=1}^{FB} \beta^{(0)} \left( k - \frac{F(x_{ij}) - f^{\wedge}}{\Delta b_f} \right) \\ \frac{\partial \beta^{(3)}(u)}{\partial u} \Big|_{u=l - \frac{M(T(x_{ij}, p)) - m^{\wedge}}{\Delta b_m}} \\ \left( -\frac{\partial M(t)}{\partial t} \Big|_{t=T(x_{ij}, p)} \right)^T \frac{\partial T(x_{ij}, p)}{\partial p} \end{array} \right]$$

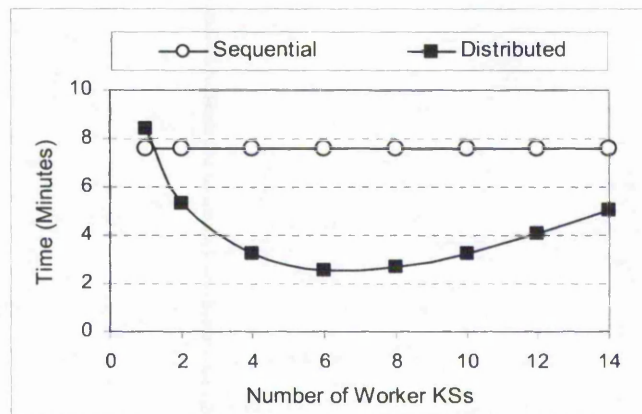
where  $x_{ij}$  is the  $j^{\text{th}}$  sample assigned to segment  $i$ , the total of which is defined as  $Q_i$ .  $f^{\wedge}$ ,  $m^{\wedge}$ ,  $\Delta b_f$ , and  $\Delta b_m$  are the minimum intensity levels and intensity ranges covered by the fixed and moving histogram bins.  $\beta^{(0)}$  and  $\beta^{(3)}$  are the zero order and cubic B-spline



Parzen windows described in Equations 5.6 and 5.7 respectively. Crucially, the gradient  $\partial M(t)/\partial t$  is computed using a cubic B-spline convolution scheme that employs a derivative operator. This is simply the derivative of the B-spline kernel in the respective dimensions of the volume and is defined as

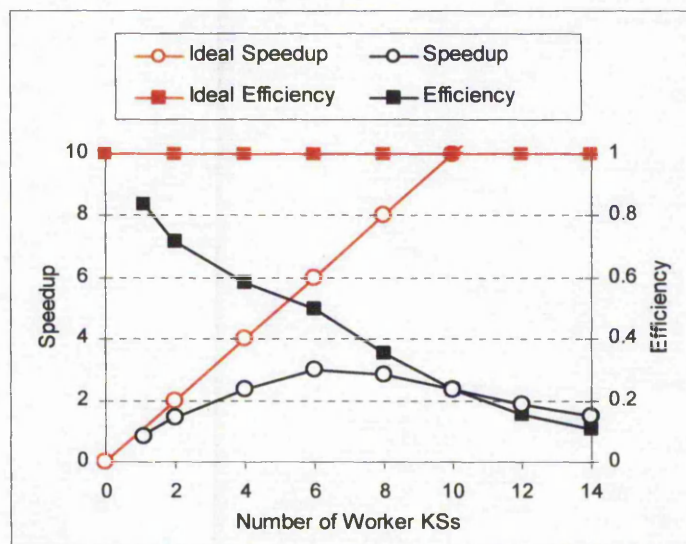
$$\frac{\partial M(x)}{\partial x} = \sum_{k=1}^K C_k \left( \frac{d\beta^{(3)}(u)}{du} \Big|_{u=x-x_k} \beta^{(3)}(y-y_k) \beta^{(3)}(z-z_k) \right) \quad 5.11$$

where  $\partial M(y)/\partial y$  and  $\partial M(z)/\partial z$  have similar definitions and the cubic B-spline Parzen window kernel  $\beta^{(3)} = \beta^{(3)}(x) \beta^{(3)}(y) \beta^{(3)}(z)$  is separable.  $C_k$  is a B-spline coefficient calculated from volume samples through recursive filtering,  $x$  is the co-ordinates of a discrete voxel location, and  $K$  is the number of voxels identified by  $k$  contained within the kernel.



**Figure 39: The sequential and distributed processing speed of multi-modal volume registration using mutual information as a similarity metric. Increasing numbers of Worker KSs are shown.**

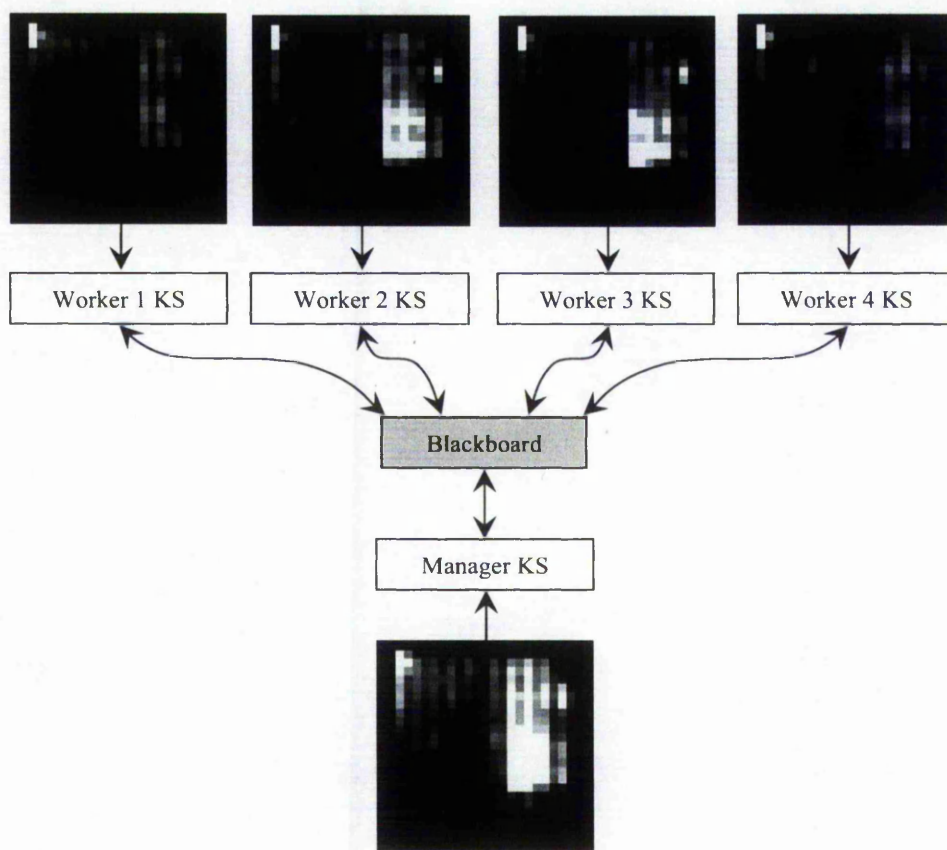
Figure 39 shows results plotted as time against number of Worker KSs, obtained whilst performing distributed multi-modal volume registration with mutual information as a similarity metric. The figure shows how processing time was reduced from eight minutes to approximately three minutes when six Worker KSs were employed. The distributed algorithm was observed to converge after 26 iterations with transform parameters which matched those computed by the sequential algorithm.



**Figure 40: The speedup and efficiency of multi-modal volume registration using mutual information as a similarity metric. Increasing numbers of Worker KSs are shown.**

The speedup and efficiency of distributed multi-modal volume registration with mutual information as a similarity metric are shown in Figure 40. As can be seen a peak speedup factor of three and an efficiency of 50% was achieved by six Worker KSs. Although the speed gains and efficiency are smaller than those achieved by distributed single-modal volume registration, a significant saving in processing time has been demonstrated. In general, the reduced speedup is due to the small set of samples

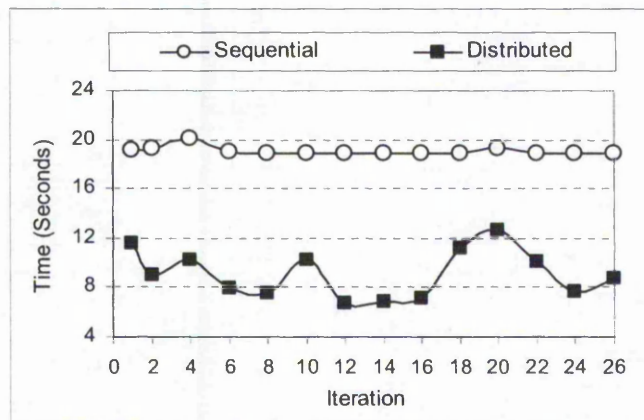
employed as well as the increased complexity of local and global probability distribution generation. When compared with single-modal volume registration, the number of Worker KS used to achieve a maximum speedup has reduced. Also, the speedup maintained after a maximum has been reached deteriorates at a faster rate. Both reductions in performance can be attributed to the increased data flowing between framework components.



**Figure 41: Local and global joint probability distributions generated by the iDARBS framework. Local histograms are hosted by the Worker KSs while a global histogram is maintained by the Manager KS.**



In Figure 41 local and global joint probability distributions generated by the iDARBS framework are shown. In general, the transferral of local probability distributions as sparse histograms causes overheads which vary with changes in alignment accuracy. For example, when segments are in close alignment intensity combinations within the local probability distributions cluster. This results in a reduced number of valid histogram bins and leads to small local probability distributions that are quick to generate and efficiently passed between Worker and Manager KS components. Large misalignment, in contrast, causes the dispersal of intensity combinations and the increase in valid histogram bins together with their associated overheads. The overheads described are also influenced by the partitioning scheme employed to divide a volume into segments. For example, the local probability distribution hosted by Worker 1 KS is generated from segments that correspond to the top of a volume. In these segments, the majority of voxels represent background intensities and hence have similar values. As a consequence, the number of valid histogram bins is significantly reduced, as are overheads.



**Figure 42:** The sequential and distributed time required to perform single iterations of the optimisation cycle using four Worker KSs. The current iteration performed is shown.

The time required to perform single iterations of the multi-modal volume registration optimisation cycle are shown in Figure 42. As can be seen the sequential algorithm maintains an almost constant speed while the distributed algorithm, using four Worker KSs, fluctuates. The fluctuations are a direct result of changing message lengths due to the clustering and dispersal of intensity combinations.

## 5.7 Conclusions

With some simple extensions to the iDARBS framework, distributed single and multi-modal volume registration has been implemented. This is a clear indication of the flexibility which can be achieved by a coarse-grained architecture such as iDARBS. Results obtained from distributed single-modal registration testing show a significant increase in speedup and efficiency, when compared with the distributed image registration approach presented in Chapter 4, can be achieved. The increases demonstrate how Worker KSs are better utilised during distributed volume registration than during distributed image registration. The improved performance observed is a direct result of the increased proportion of time Worker KSs spend processing intensities rather than handling overheads. Although the speed gains of distributed multi-modal volume registration are less than those achieved by the single-modal implementation, a significant saving in processing time has been demonstrated. The reduced speedup is due to the small number of samples used for similarity calculation and the complexity of probability distribution generation.

Crucially, the set of samples used in the calculation of mutual information need to be generated before division of the fixed volume into segments. This ensured that the

samples produced are the same for both sequential and distributed algorithms. It also allows equivalent paths through transform parameter search space to be followed and a fair comparison to be made. In general, the concurrent generation of local probability distributions described makes speedup and scalability of the algorithm possible. The use of sparse histograms during transferral of local probability distributions, however, causes fluctuating overheads which are related to the scale of misalignment between segments. This is in contrast with the single-modal approach, where the generation of local derivatives results in overheads which remain constant throughout the alignment process. Logically, the use of small probability distributions reduced overheads and results in improved speedup and efficiency of the iDARBS multi-modal registration implementation. A trade off between accuracy of alignment and the number of histogram bins employed, however, needs to be made.

### 5.8 Summary

Using MRI volumes of an anatomically normal human head, the distributed registration of single-modal data was successfully demonstrated. The distributed calculation of similarity which has grown by a degree of freedom was achieved using the same worker/manager model described in Chapter 4. Reactive control of Worker KS activities was accomplished using information strings that contain transform and similarity metric derivative parameters. Significantly, the spatial mapping of intensities was made possible with a versor rigid 3D transform component that consists of a rotation and translation in 3D space. Optimisation of transform parameters was achieved with a versor rigid 3D transform optimisation component which combines the current rotation with a computed gradient. Translation parameters, in contrast, were simply updated in 3D vector space. Once registered, volumes were visually assessed as

a complete object and as slices in perspective and orthographic projections using a simple 3D viewer. Testing of the distributed metric resulted in a reduction of processing time from 68 minutes to approximately ten minutes.

Additions to the iDARBS framework were made in order that distributed multi-modal volume registration could be performed. Functionality added to the Distributor KS included the estimation of minimum and maximum intensity levels, the calculation of histogram bin size, and the generation of random samples. To generate local probability distributions, used in the calculation of similarity, Worker KSs determined contributions based on sample intensity levels. Contributions successfully added to a local probability distribution were then smoothed using a B-spline Parzen window convolution scheme. Functionality added to the Manager KS allowed for the construction of global probability distributions using locally generated histograms. The extraction of similarity measure derivatives and the updating of transform parameters were also performed by the Manager KS. Importantly, testing of the distributed mutual information metric using MRI volumes of differing modality demonstrated a reduction in processing time from eight minutes to approximately three minutes. Tables of experimental results are provided in Appendix I.

## **6 Conclusions, discussion, and future work**

In Chapter 5, distributed single and multi-modal volume registration was introduced. Contributions made by this thesis are now reviewed.

In Section 6.1, conclusions are given. Modifications to the underlying blackboard architecture, suitability of the worker/manager model employed, and load balancing of knowledge sources (KSs) are discussed in Section 6.2. Section 6.2.1, in contrast, identifies the pros and cons of the iDARBS (imaging Distributed Algorithmic and Rule-based Blackboard System) framework, as a parallel image processing platform. The advantages of distributed image registration, using mean square error and normalised correlation as similarity metrics, are established in Section 6.2.2 as are the disadvantages. In Section 6.2.3 the strengths and weaknesses of distributed single and multi-modal volume registration are presented. The chapter ends with suggestions for future work in Section 6.3.

### **6.1 Conclusions**

Image registration is an important step in industrial and medical analysis tasks where information is extracted from a combination of sources. The surveyed literature demonstrates that the most successful algorithms employ intensity-based correlation as a measure of similarity [5] [6] [15]. Although a variety of similarity metrics have been developed, in practice they represent a considerable computational burden during the alignment process. The main reason for this is the high cost associated with multiple evaluations of a complex transform and the interpolation of non-discrete intensity coordinates. The researched literature also makes clear that concurrent similarity calculation can be achieved and provides better processing speeds than non-parallel



approaches [85] [88]. In general, the large speed increases reported are difficult to obtain and only specialised hardware is capable of maintaining such speedups when scaled. As a consequence, the applications developed to address the problem of slow registration speeds are restricted to high-cost specialised architectures found predominantly in the research environment.

While various approaches to distribution have been employed, it is fine-grained parallelism that achieves the best results [26] [86] [87] [103] [104] [105]. These methods are based on the low level decomposition of an algorithm within a tightly-coupled architecture. Such algorithms are difficult to implement and minimise computational expense by eliminating the exchange of data between processors. In this research, a novel agent-based approach to high performance intensity-based image registration is presented for the first time. Based on a distributed blackboard architecture and implemented as KSs, the iDARBS framework provides an underlying worker/manager model. As described in Chapter 3, the division of intensity data by a Distributor KS and the allocation of segments to multiple Worker KSs permits concurrent processing capabilities. Importantly, the supervision of Worker KS activities and the construction of a resulting image are performed by a Manager KS. Co-ordination is achieved through a combination of information strings and reactive behaviour. The approach has been shown to improve the processing speed of computationally intensive image processing tasks, clearly outperforming sequential versions of the same algorithm.

To achieve high performance intensity-based image registration, similarity calculation functionality was added to the iDARBS framework. Parallelisation is accomplished

through the distribution of transform parameters and the accumulation of locally computed derivatives outlined in Chapter 4. Using this technique, the alignment of images using mean square error and normalised correlation similarity metrics has been demonstrated. Both metrics provide a clear indication that high performance intensity-based image registration can be achieved using non-specialised architectures. The comparison between sequential and distributed algorithms provided in Section 4.4 show that processing time can be reduced as the number of Worker KSs is increased. An increasing speedup which reaches a maximum and then slowly drops away is a general trend observed for both metrics. The initial speedup is caused by the small number of Worker KSs requiring servicing by the blackboard. A maximum speedup is reached when distribution is counteracted by the queuing of requests for access to the blackboard. The queuing of requests results in Worker KSs that are not fully utilised during the registration process.

The introduction of a 3D transform component and optimisation scheme described in Chapter 5 allows for the registration of volume data. A significant increase in speedup, when compared with distributed image registration, is achieved by the resulting implementation. The improved performance is the direct result of the increased proportion of time spent processing intensities rather than handling overheads. The modifications to KS behaviour and addition of functionality presented in Section 5.5, permits evaluation of mutual information and subsequent alignment of data captured using different sensor types. Parallelisation of similarity calculation is achieved through the distribution of transform parameters and the accumulation of locally computed probability distributions. Significant savings in processing time have been demonstrated while at the same time the fluctuating overheads caused by transferral of sparse

histograms have been highlighted. In conclusion, although the speedups achieved by the iDARBS framework are smaller than those reached by tightly-coupled architectures. The coarse-grained approach employed is flexible and easily extended to accommodate a variety of alignment strategies.

## 6.2 Discussion

For high performance intensity-based registration to be achieved, modifications to the underlying blackboard architecture DARBS (Distributed Algorithmic and Rule-based Blackboard System) have been made. The implementation of a raw data container, to remove the formatting of intensity data into the string type, is one such example. Transmission overheads and storage limitations have also been addressed through the introduction of lossless compression algorithms [117] [120]. As with any compression scheme the ratio of compression reached is, however, directly related to the content of an image. Understandably, modifications to the blackboard architecture were also made when faults were discovered during testing. One fault occurred during the transmission of intensity data between framework components. The flaw was found to be caused by the handling of super long messages by KS and blackboard modules. Once identified, the fault was successfully fixed using a greedy algorithm. Modifications made to the underlying threading of blackboard processes means the iDARBS framework can be hosted on a variety of Linux platforms.

In general, it can be argued that the Distributor KS is not needed and the behaviour it possesses can be implemented as part of the Manager KS. This would simplify the distributed registration applications and removing the need to accumulate processed segments at the end of the optimisation cycle. Such a strategy was not employed using

iDARBS for the sake of simplicity and to explicitly separate initialisation of the framework from the processing of segments. The primary function of the Distributor KS is division of selected images into segments and estimation of initial transform parameters. If required, these responsibilities can be transferred to the Manager KS through the addition of extra rules. The re-distribution of Distributor KS behaviour would, however, cause the Manager KS to become increasingly complex. As a consequence, overheads would also increase due to the evaluation of additional rules throughout the optimisation process.

An important consideration highlighted by testing is that the trigger mechanism, designed to co-ordinate Worker KS activities, represents a performance bottleneck in communications to and from the blackboard. Once triggered, Worker KSs try to obtain segments from the blackboard simultaneously causing an initial overloading of communications. A second overload occurs when the Worker KSs complete their assigned tasks and try to return processed segments to the blackboard. This synchronisation occurs because the Worker KSs are working in a first-come first-served fashion. Logically, the creation of a schedule prior to the processing of segments represents a static load-balancing strategy. Such a schedule would be suitable in circumstances where the time required for Worker KSs to process their respective segments is approximately the same. If however, Worker KSs are required to perform tasks that take different lengths of processing time, a dynamic load balancing approach [158] would be more appropriate. The goal of any such scheme would be to reduce the idle time of Worker KSs.

### 6.2.1 iDARBS – A distributed image processing framework

Conveniently, the underlying architecture on which the iDARBS framework is based does not have a control module and hence true opportunism and independence can be achieved. In contrast with other multi-agent image processing applications [93] [94], the use of reactive behaviour reduces the processing time of distributed image segments. For example, the autonomous behaviour described in Section 3.4 minimises overheads by permitting only KS-to-blackboard communications. The simplicity of Worker KS implementation also increases efficiency and makes behavioural modification a relatively straightforward task. This is because externally developed functionality can be embedded within Worker KS rule files or called by means of dynamically linked libraries. Importantly, whenever communication cannot be avoided, the multi-threaded nature of the iDARBS framework allows concurrent servicing of KS requests. As no specialised hardware is required, iDARBS can reside on a network connected by TCP/IP communications. This allows the framework to be easily scaled when compared with tightly-coupled architectures.

The parallelisation of an algorithm using the iDARBS framework is limited by the number of segments an image can be divided into. Also, depending on the partitioning scheme employed, the quantity of redundant data passed between framework components can increase significantly. For instance, partitioning in either horizontal or vertical directions results in segments which contain between one and two borders. Partitioning in both horizontal and vertical directions, in contrast, results in segments that contain between two and four borders depending on their position within an image. The restarting of KSs outlined in Section 3.3 is also considered as being inefficient. Restarts occur whenever the contents of a partition changes and is designed to ensure

that all KS possess up-to-date information. The cumulative restarting of KSs, however, causes the iDARBS framework to slow down as the number of Worker KSs increases. Logically, restarts are only advantageous during the distribution of control information by the Manager KS as it results in the immediate action of Worker KSs. To reduce unnecessary restarting, the number of partitions a KS uses needs to be kept to a minimum.

### 6.2.2 High performance intensity-based image registration

By employing an intensity-based algorithm, the complicated segmentation of features fundamental to landmark-based image registration has been avoided. As a consequence, the algorithm distributed in Chapter 4 does not require user intervention. At an implementation level, the transferral of transform and derivative parameters as short concise strings minimises communication overheads. The method is advantageous because the messages generated remain approximately the same length for all iterations of the optimisation process. Unlike other high performance intensity-based image registration applications [85] [88], the decoupling of algorithm components allows transforms of any type to be incorporated. This explicit separation of components also permits different similarity calculation and transform parameter optimisation strategies to be employed, with only minor modifications to the existing framework. As described in Section 4.4, this has been demonstrated through the distribution of mean square error and normalised correlation similarity metrics. Conveniently, the accumulation and summation of local derivatives used to parallelise the registration algorithm does not restrict scalability, unlike the size of an image.

In general, the size of border assigned to segments is directly related to the scale of misalignment between images. Large borders, required to handle significant misalignment, increase the quantity of redundant data passed between framework components. Small borders, in contrast, can result in the introduction of null pixel values and a corrupt path through transform parameter search space being followed. Although a high level of accuracy is achieved, the use of double precision numbers increases the size of messages that are passed between framework components. The summation of derivative parameters described in Section 4.3.3.1 can also result in the introduction of rounding errors. After multiple evaluations, the path followed through transform parameter search space deviates to such an extent that the number of iterations performed to reach convergence changes. As a result, an unfair comparison between sequential and distributed algorithms is made. To ensure that the same path is followed, derivative parameters in both sequential and distributed algorithms are rounded to six decimal places at the end of each optimisation cycle.

During distributed registration, transform and derivative parameters are placed in worker control partitions. As a consequence, each Worker KS focuses its attention towards an associated blackboard partition thus simplifying the search for required data. The simple KS behaviour outlined in Section 4.3 keeps information stored on the blackboard at an absolute minimum and removes the need for partition wide searches. Conveniently, the Manager KS can be used to generate a log of transform parameters in the *Parameters* partition. As the alignment of two images can last for several minutes, visual assessment of the *Parameters* partition using the DARBS terminal client makes real-time monitoring of the transform parameter optimisation process possible. Different from other high performance intensity-based image registration applications

[26] [86] [105], the storage of transform and derivative parameters means similarity calculation can be restarted in the event of failure. Such redundancy can be achieved by saving segments locally to file once they have been retrieved from the blackboard. Failed KSs are then restarted with transform parameters found in the *Parameters* partition.

The accumulation and summation of local derivatives on which the distributed algorithms are based restricts the framework to rigid body registration. In order for non-rigid alignment [129] [156] to be adopted, modifications to both Worker and Manager KS behaviour are required. For example, each Worker KS would be required to calculate local transform parameters for the image segment assigned to it. The Manager KS, in contrast, would be required to co-ordinate the start of Worker KS activities and accumulate final transform parameters at the end of the registration process. In general, the partitioning of images into segments also restricts the distribution of similarity calculation to metrics that employ one-to-one data access. As illustrated in Section 4.3.2.1, under the present implementation one-to-many access is only possible when required data is within the immediate neighbourhood of transformed intensity co-ordinates. To host statistical metrics that employ erratic patterns of data access, duplicate images need to be held by each Worker KS. This increases redundant data and reduces performance of the iDARBS framework.

### 6.2.3 Single and multi-modal volume registration

The generalisation of iDARBS from images to volumes proved to be straightforward and only minor modifications were required. The flexibility accomplished is mainly due to the decoupling of algorithm components previously mentioned. In order for an affine



transform in 3D space to be accommodated, the Quaternion transform type as described in Section 5.1 has been adopted. Because the Quaternion consists of fewer parameters than a matrix representation, the overheads of passing transform and derivatives parameters between framework components are reduced. Conveniently, the complexity of the transform parameters search space and hence the sequential portion of the algorithm is also minimised. This is in contrast with other high performance intensity-based registration applications [87] [103] [104], where matrix representations of transforms are employed. Significantly, the additional burden associated with processing volume intensities is placed on the resampling and similarity calculation stages, both of which represent distributed portion of the algorithm. Unsurprisingly, the effects of compression to alleviate memory constraints are greater during volume registration than during image registration.

The processing times presented in Section 5.3 are a direct result of the increased intensity data associated with volume images. Understandably, processing time is not significantly shortened through the compression of volume segments alone. Although a fast lossless scheme has been employed, the speed gains achieved through the transmission of compressed segments are lost during the compression process itself. Depending on the available memory of computers within the host network, the size of a volume is also limited. For example, memory requirement is doubled through the construction of gradient volumes from which contributions to the local derivatives are taken. The effects on memory are only compounded when a multi resolution scheme is adopted. As with distributed image registration, the use of traditional intensity-based similarity metrics restricts alignment of volumes of the same modality. To overcome this, more sophisticated metrics that evaluate mutual information are need.

Mutual information computed directly from volume intensities has the advantage of being a process which operates on a subset of samples. Conveniently, the random generation of samples and their assignment to segments described in Section 5.5.1 results in subsets of varying sizes. The evaluation of similarity using variable sized subsets assists load balancing by unevenly distributing the computational burden between Worker KSs. The random position of samples also causes the computational burden of individual Worker KSs to change during the alignment process. This is because the translation and rotation used to evaluate moving segment co-ordinates, cause different numbers of samples to become valid and invalid during different iterations of the optimisation cycle. Importantly, as demonstrated, the joint probability distribution used for similarity evaluation can be constructed from multiple local histograms. Although more time consuming to compute, the multi-modal similarity calculation described is as scalable as the single-modal implementation. The summation of local probability distributions employed is also simpler than the methods of global histogram construction [26] [86] [88] described by other researchers.

Adequate for the estimate of initial transform parameters between volumes of the same modality, centres of mass are less reliable in the mutual information-based approach. The reason for this is that the intensities they exploit no longer represent the same structures within the scene. As a consequence, more complex and time consuming methods of initial estimation are required. Unsurprisingly, the effects on message length caused by variations in alignment discussed in Section 5.6.1 are significant. These effects are magnified through the construction of local probability distributions using double precision numbers. Although the use of a single precision format is plausible, the

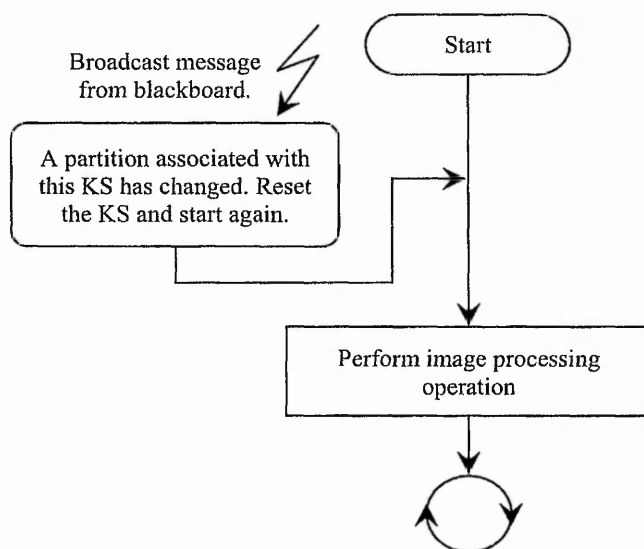
effects of smoothing by the Parzen window will be compromised and alignment accuracy will be reduced. Under the current implementation, it is necessary to visit all bins in a probability distribution in order for a sparse histogram to be generated. This is seen as inefficient. For instance, when voxel combinations cluster the number of valid bins is reduced resulting in the unnecessary parsing of invalid histogram bins at the end of each optimisation cycle.

### **6.3 Future work**

In general, the research presented in this thesis represents a significant step towards high performance intensity-based registration of single and multi-modal data using non-specialised architectures. Despite the success of iDARBS, as demonstrated by the results obtained, it is clear that the work can be extended in a number of ways. Areas in which the framework can be improved include Worker and Manager KS efficiency, robustness of the transform parameter optimisation process, increased processing speed through the use of a multi-resolution strategy, and the distribution of deformable registration algorithms.

A major cause of inefficiency has been identified as the wait caused by the Manager KS as it accumulates local derivatives and updates transform parameters. To increase efficiency, once derivatives have been computed, each Worker KS should add its contribution to a global derivative stored on the blackboard. As a consequence, the Manager KS would be required to fetch and not accumulate the global derivative. By transferring the accumulation of local derivatives from a sequential portion of the algorithm to a distributed portion, Worker KS waiting time will be reduced. Because data consistency needs to be maintained, this step will require mutually exclusive access

to the global derivative for both Worker and Manager KSs. Mutually exclusive access can be made possible with use of existing DARBS command [114]. Although the time required by the Manager KS to accumulate local derivatives during single-modal registration is small, the concurrent construction of global joint probability distribution for multi-modal registration would prove beneficial.



**Figure 43: Focused KS activation on iDARBS. Both Worker and Manager KS only evaluate rules once an associated partition has changed and the KS has restarted.**

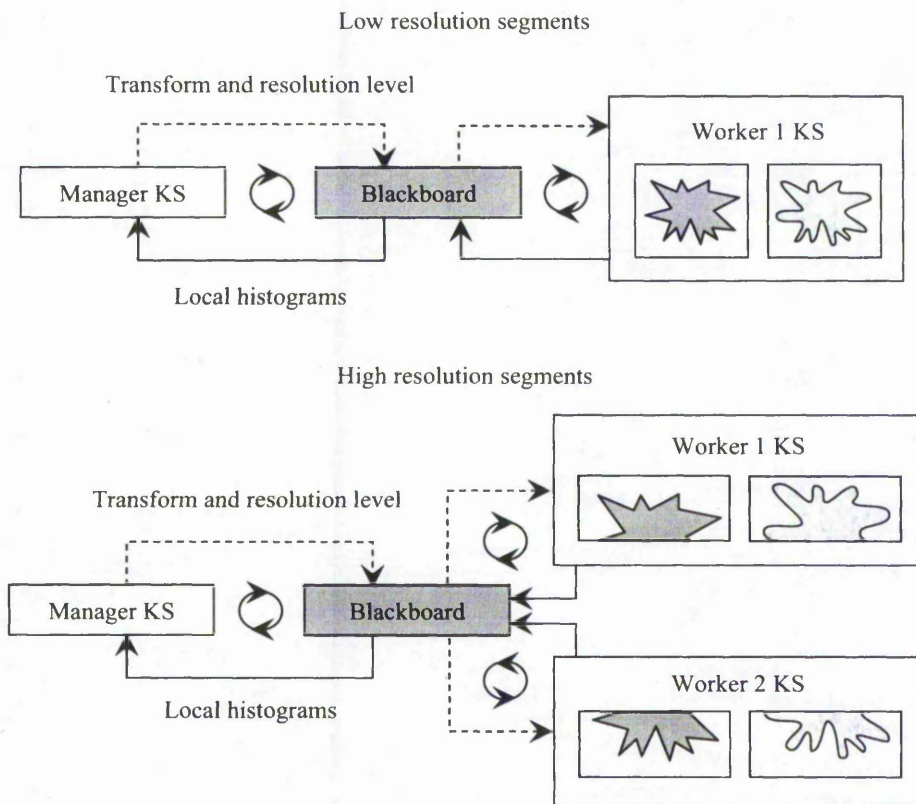
Efficiency can also be improved through the use of event-based triggering of actions and the focused activation of individual KSs. The idea being to achieve reduced polling of the blackboard by idle KSs and hence minimise communication overheads within the iDARBS framework as a whole. In order to maintain the existing implementation, such a scheme could be achieved through small modifications to the Worker and Manager KS types. The simple modification shown in Figure 43 would allow individual KSs to

re-evaluate their rules, only when the contents of their associated partitions have changed and not after each complete rule evaluation as in the current implementation.

Conveniently, the transforms used during both image and volume registration are not restricted to a single optimisation scheme. Therefore to increase robustness of the alignment process, the updating of transform parameters should be performed concurrently using multiple optimisation techniques [159] [160]. This can be achieved, for example, by specialised KSs that employ conjugate-gradient, quasi-Newton, and least-square methods. As the underlying blackboard architecture on which iDARBS is based is suited to the mixing of processing styles, genetic algorithm and neural network-based strategies may also be employed. Once evaluated, a voting strategy or averaging could be used to select transform parameters for propagation to Worker KSs. Because the optimisation of transform parameters using iDARBS is performed sequentially, the strategies employed need not be suited to parallel implementation. As a consequence, the use of multiple optimisation strategies may result in shorter paths through the transform parameter search space thus resulting in increased processing speeds.

Under the current iDARBS implementation, registration does not make use of a multi-resolution strategy [69]. It can therefore be argued that the framework is incomplete. Such a technique was not employed in order to preserve the simplicity of KS implementation and maximise speedup. The adoption of a multi-resolution scheme could, however, be achieved without the addition of specialised KSs. Logically, the number of resolution levels would be set by the Distributor KS during initialisation of the framework. The current resolution level would be updated by the Manager KS at the end of each optimisation cycle and propagated to Worker KSs with transform

parameter. By employing the Worker KSs to down-sample retrieved segment, the transmission of multi-resolution data can be avoided. Also, as illustrated in Figure 44, to improve efficiency, single or multiple Worker KSs could be used at different resolution levels depending on the size of resampled segments. Although additional functionality would be required by the Worker KSs, the Distributor and Manager KSs would remain largely unchanged.



**Figure 44: Multi-resolution registration on iDARBS. The number of Worker KSs would be set based on the size of resampled segments.**

Finally, deformable registration is based on the assumption that an evenly spaced mesh of control points can be placed over fixed and moving images. A transform is then used

to estimate the displacement necessary to map between corresponding control point locations. In order for registration to be performed, each control point plus underlying intensities are transformed and compared with their counterpart until an acceptable level of similarity is achieved. Such algorithms can be successfully hosted upon the iDARBS framework by assigning subsets of control points to each Worker KS. Conveniently, the use of free-form deformations [161] [162] would permit the independent movement of control points assigned to Worker KSs. A spline-based representation, where each control point has a global influence resulting in the movement of neighbouring control points, could also be implemented. Because control points are distributed, Worker KS-to-Worker KS communications would be required. Crucially, the resulting implementation would address the computational burden associated with the estimation of transform parameters for locally deformed images.

## References

- [1] Newman, T.S. and Jain, A.K. A Survey of Automated Visual Inspection, *Computer Vision and Image Understanding*, 1995, 61, 231-262.
- [2] Bayro-Corrochano, E. Review of Automated Visual Inspection 1983 to 1993, *Intelligent Robots and Computer Vision*, 1993, 2055, 128-158.
- [3] Le Moigne, J. Morisette, J. Cole-Rhodes, A. Johnson, K. Stone, H. Zavorin, I. Eastman, R. and Netanyahu, N. Registration of Multiple Sensor Earth Science Data, *Proceedings of the Earth Science Technology Conference*, Palo Alto, USA, 2004.
- [4] Zhengwei, Y. and Cohen, F.S. Image Registration and Object Recognition Using Affine Invariants and Convex Hulls, *IEEE Transactions on Image Processing*, 1999, 8(7), 934-946.
- [5] Maintz, J.B.A. and Viergever, A. A Survey of Medical Image Registration, *Medical Image Analysis*, 1998, 2, 1-36.
- [6] Zitova, B. and Flusser, J. Image Registration Methods: A Survey, *Image and Vision Computing*, 2003, 21, 977-1000.
- [7] Jenkinson, M. and Smith, S. A Global Optimisation Method for Robust Affine Registration of Brain Images, *Medical Image Analysis*, 2001, 5, 143-156.
- [8] Grevera, G.J. and Udupa, J.K. An Objective Comparison of 3D Image Interpolation Methods, *IEEE Transactions on Medical Imaging*, 1998, 17, 642-652.
- [9] Goshtasby, A.A. 2-D and 3-D Image Registration for Medical, Remote Sensing, and Industrial Applications, *John Wiley*, USA, 2005.
- [10] Mäkelä, T. Clarysse, P. Sipilä, O. Pauna, N. Pham, Q.C. Katila, T. and Magnin, I.E. A Review of Cardiac Image Registration Methods, *IEEE Transactions on Medical Imaging*, 2002, 21(9), 1011-1021.
- [11] Zhang, Z. and Blum, R.S. A Hybrid Image Registration Technique for a Digital camera Image Fusion Application, *Information Fusion*, 2001, 2(2), 135-149.
- [12] Wang, X. and Feng, D.D. Hybrid Registration for Two-Dimensional Gel Protein Images, *Proceedings of the 3rd Asia-Pacific Bioinformatics Conference*, Singapore, Malaysia, 2005.



## References

---

- [13] Penney, G.P. Weese, J. Little, J.A. Desmedt, P. Hill, D.L.G. and Hawkes, D.J. A Comparison of Similarity Measures for Use in 2D-3D Medical Image Registration, *IEEE Transactions on Medical Imaging*, 1998, 17, 586-595.
- [14] Hill, D.L.G. and Hawkes, D.J. Voxel Similarity Measures for Automated Image Registration, *Proceedings of the SPIE Conference on Visualisation in Biomedical Computing*, 1994, 216, 205-216.
- [15] Brown, L.G. A Survey of Image Registration Techniques, *ACM Computing Surveys*, 1992, 325-376.
- [16] Shannon, C.E. A Mathematical Theory of Communication (Parts 1 and 2), *Bell System Technical Journal*, 1948, 27, 379-423 and 623-656.
- [17] Jamieson, L.H. Delp, E.J. Wang, C. Li, J. and Weil, F.J. A Software Environment for Parallel Computer Vision, *Computer: IEEE Computer Society Press*, 1992, 25, 73-77.
- [18] Seinstra, F.J. and Koelma, D. The Lazy Programmer's Approach to Building a Parallel Image Processing Library, *Proceedings of the 15<sup>th</sup> International Parallel & Distributed Processing Symposium (IPDPS'01)*, San Francisco, USA, 2001, 23-27.
- [19] Koelma, D. and Sips, H.J. A Software Architecture for Parallel Image Processing, *Proceedings of the 3<sup>rd</sup> Annual Conference of the Advanced School for Computing and Imaging*, Netherlands, Holland, 1997, 34-40.
- [20] Tannenbaum, A.S. Distributed Operating Systems, *Prentice Hall*, USA, 1995.
- [21] Grama, A. Karypis, G. Kumar, V. and Gupta, A. Introduction to Parallel Computing: Design and Analysis of Algorithms, *Addison Wesley*, USA, 2003.
- [22] Moganti, M. Ercal, F. Dagli, C. and Tsunekawa, S. Automatic PCB Inspection Algorithms: A Survey, *Computer Vision and Image Understanding*, 1996, 63, 287-313.
- [23] Stacy, M. Hanson, D. Camp, J. and Robb, R.A. High Performance Computing in Biomedical Imaging Research, *Parallel Computing*, 1998, 24, 1287-1321.
- [24] Shekhar, C. Govindu, V. and Chellapa, R. Multi-sensor Image Registration by Feature Consensus, *Pattern Recognition*, 1999, 32, 39-52.
- [25] Petkovic, T. Krapac, J. Loncaric, S. and Sercer, M. Automated Visual Inspection of Plastic Products, *Proceedings of the 11<sup>th</sup> International Electrotechnical and Computer Science Conference*, Zadar, Croatia, 2002, 283-286.

## References

---

- [26] Rohlfing, T. and Maurer, C.R. Non-rigid Image Registration in Shared-memory Multi-processor Environments with Application to Brains, Breasts and Bees, *IEEE Transactions on Information Technology in Biomedicine*, 2003, 7, 16-25.
- [27] Nii, H.P. Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures, *AI Magazine*, 1986, 7, 38-53.
- [28] Sobczak, R.S. and Matthews, M.M. A Massively Parallel Expert System Architecture for Chemical Structure Analysis, *Proceedings of the 5<sup>th</sup> Distributed Memory Computing Conference*, 1990, 11-17.
- [29] Van Den Elsen, P.A. Pol, E.J.D. and Viergever, M.A. Medical Image Matching-A Review with Classification, *IEEE Engineering in Medicine and Biology*, 1993, 12, 26-39.
- [30] Shin, D. Pollard, J.K. and Muller, J.P. Accurate Geometric Correction of ATSR Images, *IEEE Transactions on Geoscience and Remote Sensing*, 1997, 35, 997-1006.
- [31] Galbiati, L.J. Machine Vision and Digital Image Processing Fundamentals, *Prentice Hall*, USA, 1990.
- [32] Firl, E.A. Wesarg, S. and Dold, C. Fast CT/PET Registration Based on Partial Volume Matching, *International Congress Series*, 2004, 1268, 31-36.
- [33] Ruiz-Alzola, J. Westin, C.F. Warfield, S.K. Alberola, C. Maier, S. and Kikinis, R. Non-rigid Registration of 3D Tensor Medical Data, *Medical Image Analysis*, 2002, 6, 143-161.
- [34] Shekhar, R. and Zagrodsky, V. Mutual Information-based Rigid and Nonrigid Registration of Ultrasound Volumes, *IEEE Transactions on Medical Imaging*, 2002, 21, 9-22.
- [35] Goshtasby, A. Image Registration by Local Approximation Methods, *Image and Vision Computing*, 1988, 6, 255-261.
- [36] Fornefett, M. Rohr, K. and Stiehl, H.S. Radial Basis Functions with Compact Support for Elastic Registration of Medical Images, *Image and Vision Computing*, 2001, 19, 87-96.
- [37] Pluim, J.P.W. Maintz, J.B.A. and Viergever, M.A. Interpolation Artefacts in Mutual Information-based Image Registration, *Computer Vision and Image Understanding*, 2000, 77, 211-232.

## References

---

- [38] Lucchese, L. Doretto, G. and Cortelazzo, G.M. A Frequency Domain Technique for Range Data Registration, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002, 24, 1468-1484.
- [39] Lehmann, T.M. Gonner, C. and Spitzer, K. Survey: Interpolation Methods in Medical Image Processing, *IEEE Transactions on Medical Imaging*, 1999, 18, 1049-1075.
- [40] Lehmann, T.M. Gonner, C. and Spitzer, K. Addendum: B-spline Interpolation in Medical Image Processing, *IEEE Transactions on Medical Imaging*, 2001, 20(7), 660-665.
- [41] Roche, A. Malandain, G. Ayache, N. and Prima, S. Towards a Better Comprehension of Similarity Measures used in Medical Image Registration, *Lecture Notes in Computer Science*, 1999, 1679, 555-566.
- [42] Zhang, J. and Rangarajan, A. Affine Image Registration Using a New Information Metric, *IEEE Transactions on Computer Vision and Pattern Recognition*, 2004, 848-855.
- [43] Nikou, C. Heitz, F. and Armspach, J. Robust Voxel Similarity Metrics for the Registration of Dissimilar Single and Multi-modal Images, *Pattern Recognition*, 1999, 32, 1351-1368.
- [44] Viola, P. Alignment by Maximization of Mutual Information, PhD Thesis, Artificial Intelligence Laboratory MIT, 1995.
- [45] Pluim, J.P.W. Mutual Information Based Registration of Medical Images, PhD Thesis, Utrecht University, 2001.
- [46] Holden, M. Hill, D.L.G. Denton, E.R.E. Jarosz, J.M. Cox, T.C.S. Rohlfing, T. Goodey, J. and Hawkes, D.J. Voxel Similarity Measures for 3-D Serial MR Brain Image Registration, *IEEE Transactions on Medical Imaging*, 2000, 19, 94-102.
- [47] Jenkinson, M. Bannister, P. Brady, M. and Smith, S. Improved Optimization for the Robust and Accurate Linear Registration and Motion Correction of Brain Images, *NeuroImage*, 2002, 17, 825-841.
- [48] Lau, Y.H. Braun, M. and Hutton, B.F. Non-rigid Image Registration Using A Median-filtered Coarse-to-fine Displacement Field and A Symmetric Correlation Ratio, *Physics in Medicine and Biology*, 2001, 46, 1297-1319.
- [49] Chalermwat, P. and El-Ghazawi, T. Multi-resolution Image Registration Using Genetics, *Proceedings of 1999 International Conference on Image Processing (ICIP'99)*, Kobe, Japan, 1999, 2, 452-456.

## References

---

- [50] Bolton, H.P.L. Groenwold, A.A. and Snyman, J.A. The Application of a Unified Bayesian Stopping Criterion in Competing Parallel Algorithms for Global Optimization, *Computers and Mathematics with Application*, 2004, 48, 549-560.
- [51] Zagrodsky, V. Shekhar, R. and Cornhill, F. Multi-function Extension of Simplex Optimization Method for Mutual Information-based Registration of Ultrasound Volumes, *Medical Imaging 2001: Image Processing*, 2001, 508-515.
- [52] Wolberg, G. and Zokai, S. Image Registration for Perspective Deformation Recovery, *Proceedings of the SPIE Conference on Automatic Target Recognition*, Florida, USA, 2000.
- [53] Dai, X. and Khorram, S. Development of a Feature-based Approach to Automated Image Registration for Multi-temporal and Multi-sensor Remotely Sensed Imagery, *Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS'97)*, Singapore, Malaysia, 1997, 243-245.
- [54] Pagoulatos, N. Haynor, D.R. and Kim, Y. Image-based Registration of Ultrasound and Magnetic Resonance Images: A Preliminary Study, *Proceedings of SPIE Medical Imaging: Image Processing*, Washington, USA, 2000, 156-164.
- [55] Stockman, G. Kopstein, S. and Benett, S. Matching Images to Models for Registration and Object Detection via Clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1982, 4, 229-241.
- [56] Batchelor, B.G. Illumination and Image Acquisition Techniques for Industrial Vision Systems, *Industrial Applications of Image Analysis*, 1984, 97-118.
- [57] Freeling, R. The Significance of Lighting in Industrial Inspection Tasks, *Proceedings of 1985 International Conference on Robotics and Automation*, Missouri, USA, 1985, 458-460.
- [58] Russakoff, D.B. Rohlfing, T. and Maurer, C.R. Fast Intensity-based 2D-3D Image Registration of Clinical Data Using Light Fields, *Proceedings of the 9<sup>th</sup> IEEE International Conference on Computer Vision*, Nice, France, 2003, 416-423.
- [59] Costa, C.E. and Petrou, M. Automatic Registration of Ceramic Tiles for the Purpose of Fault Detection, *Machine Vision and Applications*, 2000, 11, 225-230.
- [60] Castro, E.D. and Morandi, C. Registration of Translated and Rotated Images Using Finite Fourier Transforms, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1987, 9, 700-703.

## References

---

- [61] Jeongtae, K. and Fessler, J.A. Intensity-based Image Registration using Robust Correlation Coefficients, *IEEE Transactions on Medical Imaging*, 2004, 23, 1430-1444.
- [62] Woods, R.P. Mazziotta, J.C. and Cherry, S.R. MRI-PET Registration with an Automated Algorithm, *Journal of Computer Assisted Tomography*, 1993, 17, 536-546.
- [63] Hill, D.L.G. Hawkes, D.J. Harrison, N.A. and Ru, C.F. A Strategy for Automated Multi-modal Image Registration Incorporating Anatomical Knowledge and Imager Characteristics, *Information Processing in Medical Imaging*, 1993, 687, 182-196.
- [64] Pluim, J.P. Maintz, J.B.A. and Viergever, M.A. Mutual Information-based Registration of Medical Images: A Survey, *IEEE Transactions on Medical Imaging*, 2003, 22, 986-1004.
- [65] Studholme, C. Hill, D.L.G. and Hawkes, D.J. An Overlap Invariant Entropy Measure of 3D Medical Image Alignment, *Pattern Recognition*, 1999, 32, 71-86.
- [66] Viola, P. and Wells, W.M. Alignment by Maximization of Mutual Information, *International Journal of Computer Vision*, 1997, 24, 137-154.
- [67] Kumar, R. Sawhney, H.S. Asmuth, J.C. Pope, A. and Hsu, S. Registration of Video to Geo-referenced Imagery, *Proceedings of the 1998 International Conference on Pattern Recognition*, Brisbane, Australia, 1998, 1393-1399.
- [68] Dani, P. and Chaudhuri, S. Automated Assembling of Images: Image Montage Preparation, *Pattern Recognition*, 1995, 28, 431-445.
- [69] Thevenaz, P. Ruttimann, U.E. and Unser, M. A Pyramid Approach to Subpixel Registration Based on Intensity, *IEEE Transactions on Image Processing*, 1998, 7, 27-41.
- [70] Chalermwat, P. El-Ghazawi, T. and Le Moigne, J. Wavelet-based Image Registration on Parallel Computers, *Supercomputing 97*, 1997, 1-9.
- [71] Zavorin, I. and Le Moigne, J. Use of Multi-resolution Wavelet Feature Pyramids for Automatic Registration of Multi-sensor Imagery, *IEEE Transactions on Image Processing*, 2005, 14(6), 770-782.
- [72] Turcajova, R. and Kautsky, J. A Hierarchical Multi-resolution Technique for Image Registration, *Proceedings of SPIE Mathematical Imaging: Wavelet Applications in Signal and Image Processing*, Colorado, USA, 1996.

## References

---

- [73] Temkin, B. Vaidyanath, S. and Acosta, E. A High Accuracy, Landmark-based, Sub-pixel Level Image Registration Method, *International Congress Series*, 2005, 1281, 254-259.
- [74] Betke, M. Hong, H. and Ko, J.P. Automatic 3D Registration of Lung Surfaces in Computed Tomography Scans, *Proceedings of the 4<sup>th</sup> International Conference on Medical Image Computing and Computer-assisted Intervention*, Utrecht, Netherlands, 2001, 725-733.
- [75] Betke, M. Hong, H. Thomas, D. Prince, C. and Ko, J.P. Landmark Detection in the Chest and Registration of Lung Surfaces with an Application to Nodule Registration, *Medical Image Analysis*, 2003, 7, 265-281.
- [76] Banerjee, S. Mukherjee, D.P. and Majumdar, D.D. Point Landmarks for Registration of CT and MR Images, *Pattern Recognition Letters*, 1995, 16, 1033-1042.
- [77] Arun, K.S. Huang, T.S. and Bistein, S.D. Least-squares Fitting of Two 3-D Point Sets, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 1987, 9, 698-700.
- [78] Sharp, G.C. Lee, S.W. and Wehe, D.K. ICP Registration Using Invariant Features, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002, 24, 90-102.
- [79] Lubeck, O.M. Supercomputer Performance: The Theory, Practice, and Results, *Advances in Computers*, 1988, 27, 309-362.
- [80] Lewis, B. and Berg, D.J. Threads Primer: A Guide to Multi-threaded Programming, *Prentice Hall*, USA, 1996.
- [81] Taniguchi, R. Makiyama, Y. Tsuruta, N. Yonemoto, S. and Arita, D. Software Platform for Parallel Image Processing and Computer Vision, *Proceedings of SPIE Parallel and Distributed Methods For Image Processing*, California, USA, 1997, 2-10.
- [82] Nicolescu, C. and Jonker, P. Parallel Low-level Image Processing on a Distributed Memory System, *Proceedings of the 15<sup>th</sup> Workshop on Parallel and Distributed Processing*, Orlando, USA, 2000, 226-233.
- [83] Seinstra, F.J. Koelma, D. and Geusebroek, J.M. A Software Architecture for User Transparent Parallel Image Processing, *Parallel Computing*, 2002, 28, 967-993.
- [84] Seinstra, F.J. Koelma, D. Geusebroek, J.M. Verster, F.C. and Smeulders, A.W.M. Efficient Applications in User Transparent Parallel Image Processing,

## References

---

- Proceedings of the 16<sup>th</sup> International Parallel & Distributed Processing Symposium*, Florida, USA, 2002, 113-130.
- [85] Warfield, S.K. Jolesz, F. and Kikinis, R.A. High Performance Approach to the Registration of Medical Imaging Data, *Parallel Computing*, 1998, 24, 1345-1368.
- [86] Christensen, G.E. MIMD vs SIMD Parallel Processing: A Case Study in 3D Medical Image Registration, *Parallel Computing*, 1998, 24, 1369-1383.
- [87] Salomon, M. Heitz, F. Perrin, G.R. and Armspach, J.P. A Massively Parallel Approach to Deformable Matching of 3D Medical Images via Stochastic Differential Equations, *Parallel Computing*, 2005, 31, 45-71.
- [88] Ino, F. Ooyama, K. and Hagihara, K. A Data Distributed Parallel Algorithm for Non-rigid Image Registration, *Parallel Computing*, 2005, 31, 19-43.
- [89] Hopgood, A.A. The State of Artificial Intelligence, *Advances in Computers*, 2005, 65, 1-75.
- [90] Wen, Z. and Tao, Y. Building a Rule-based Machine Vision System for Defect Inspection on Apple Sorting and Packing Lines, *Expert Systems with Applications*, 1999, 16, 307-313.
- [91] Murch, R. and Johnson, J. Intelligent Software Agent, *Prentice Hall*, USA, 1998.
- [92] Ferber, J. Multi-agent Systems: An Introduction to Distributed Artificial Intelligence, *Addison Wesley*, London, 1999.
- [93] Harrovet, F. Ballet, P. Rodin, V. and Tisseau, J. ORIS: Multi-agent Approach for Image Processing, *Proceedings of SPIE Parallel and Distributed Methods for Image Processing*, San Diego, USA, 1998, 3452, 57-68.
- [94] Lueckenhaus, M. and Eckstein, W. Multi-agent Based System for Parallel Image Processing, *Proceedings of SPIE Parallel and Distributed Methods for Image Processing*, San Diego, USA, 1997, 3166, 21-30.
- [95] Jufeng, W. Hancheng, X. Xiang, L. and Jingping, Y. Multi-agent Based Distributed Control System for an Intelligent Robot, *Proceedings of the IEEE International Conference on Services Computing (SCC'04)*, 2004, 633-637.
- [96] McManus, J.W. A Concurrent Distributed System for Aircraft Tactical Decision Generation, *Proceedings of the 9<sup>th</sup> Digital Avionics Systems Conference*, New York, USA, 1990, 505-512.

## References

---

- [97] Jiang, Y.C. Xia, Z.Y. Zhong, Y.P. and Zhang, S.Y. An Adaptive Adjusting Mechanism for Agent Distributed Blackboard Architectures, *Microprocessors and Microsystems*, 2005, 29, 9-20.
- [98] Erman, L.D. Hayes-Roth, F. Lesser, V.R. and Reddy, D.R. The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty, *ACM Computing Surveys*, 1980, 12, 213-253.
- [99] Englemore, R. and Morgan, T. Blackboard Systems, *Addison-Wesley Publishers Ltd*, Wokingham, 1988.
- [100] Chau, K.W. and Albermani, F. Hybrid Knowledge Representation in a Blackboard KBS for Liquid Retaining Structure Design, *Engineering Applications of Artificial Intelligence*, 2004, 17, 11-18.
- [101] Seo, H.S. and Cho, T.H. An Application of Blackboard Architecture for the Coordination among Security Systems, *Simulation Modelling and Theory*, 2003, 11, 269-284.
- [102] Xu, H. and Van Brussel, H. A Behaviour-based Blackboard Architecture for Reactive and Efficient Task Execution of an Autonomous Robot, *Robotics and Autonomous Systems*, 1997, 22, 115-132.
- [103] Hastings, S. Kurc, T. Langella, S. Catalyurek, U. Pan, T. and Saltz, J. Image Processing for the Grid: A Toolkit for Building Grid-enabled Image Processing Applications, *Proceeding of the 3<sup>rd</sup> IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2003, 36-43.
- [104] Ourselin, S. Stefanescu, R. and Pennec, X. Robust Registration of Multi-modal Images: Towards Real-time Clinical Applications, *Proceedings of the 5<sup>th</sup> International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI'02)*, 2002, 140-147.
- [105] Rohlfing, T. Maurer, J.C.R. Hill, D.L.G. Hartkens, W.T. Hall, A. Truwit, C.L. Liu, H. Martin, A.J. and Shahidi, R. Intra-operative Brain Deformation Using Non-rigid Image Registration on a Shared-memory Multiprocessor Computer, *Proceedings of the 16<sup>th</sup> International Congress and Exhibition of Computer Assisted Radiology and Surgery (CARS'02)*, 2002, 150-155.
- [106] The Knowledge Systems Laboratory at Stanford, *The BBI Blackboard Control Architecture*, Available at: <http://ksl.stanford.edu/projects/AIS/> [Accessed 16<sup>th</sup> June 2004].
- [107] The GBBopen Project, *GBBopen*, Available at: <http://gbbopen.org/> [Accessed 19<sup>th</sup> June 2004].



## References

---

- [108] Corkill, D.D. Gallagher, K.Q. and Murray, K.E. GBB: A Generic Blackboard Development System, *Proceeding of the National Conference on Artificial Intelligence*, Philadelphia, USA, 1986, 1008-1014.
- [109] Nolle, L. Wong, K.C.P. and Hopgood, A.A. DARBS: A Distributed Blackboard System, *Research and Development in Intelligent Systems XVIII*, 2001, 161-170.
- [110] Li, G. Weller, M.J. and Hopgood, A.A. Shifting Matrix Management: A Model for Multi-agent Cooperation, *Engineering Applications of Artificial Intelligence*, 2003, 16, 191-201.
- [111] Smith, R.G. and Davis, R. Frameworks for Cooperation in Distributed Problem Solving, *IEEE Transactions on Systems Man and Cybernetics*, 1981, 11, 61-70.
- [112] Hopgood, A.A. Phillips, H.J. Picton, P.D. and Raithwaite, N.S. Fuzzy Logic in a Blackboard System for Controlling Plasma Deposition Processes, *Artificial Intelligence in Engineering*, 1997, 12, 253-260.
- [113] Choy, K.W. Hopgood, A.A. Nolle, L. and O'Neill, B.C. Performance of a Multi-agent Simulation on a Distributed Blackboard System, *International Journal of Simulation Systems, Science & Technology*, 2005, 6(9), 57-72.
- [114] Choy, K.W. Blackboard Architecture for Intelligent Embedded Systems, PhD Thesis, Nottingham Trent University, 2005.
- [115] Deitel, H.M. and Deitel, P.J. C++ How to Program: Second Edition, *Prentice Hall*, USA, 1998.
- [116] Hughes, C. and Hughes, T. Object-oriented Multithreading Using C++, *Wiley*, 1997.
- [117] The zlib Compression Library, *zlib*, Available at: <<http://www.zlib.net/>> [Accessed 1<sup>st</sup> April 2005].
- [118] Ziv, J. Lempel, A. A Universal Algorithm for Sequential Data Compression, *IEEE Transactions on Information Theory*, 1977, 23(3), 337-343.
- [119] Huffman, D.A. A Method for the Construction of Minimum-redundancy Codes, *Proceedings of the Institute of Radio Engineers (IRE'52)*, 1952, 1098-1102.
- [120] Starosolski, R. Simple Fast and Adaptive Lossless Image Compression Algorithm, Available at: <<http://sun.iinf.polsl.gliwice.pl/~rstaros/sfalic-/index.-html>> [Accessed 1<sup>st</sup> April 2005].
- [121] Trolltech, *Qt*, Available at: <<http://www.trolltech.com/products/index.html>> [Accessed 10<sup>th</sup> November 2003].

## References

---

- [122] Davies, E.R. *Machine Vision: Theory, Algorithms, Practicalities*, Academic Press, 1990.
- [123] Gonzalez, R.C. and Woods, R.E. *Digital Image Processing: Second Edition*, Prentice Hall, USA, 2002.
- [124] Dale-Jones, R. and Tjahjadi, T. A study and Modification of the Local Histogram Equalisation Algorithm, *Pattern Recognition*, 1993, 26, 1373-1381.
- [125] Umbaugh, S. *Computer Vision and Image Processing*, Prentice Hall, USA, 1998.
- [126] Otsu, N. A Threshold Selection Method from Gray-level Histograms, *IEEE Transactions on Systems Man and Cybernetics*, 1979, 62-66.
- [127] Hartkens, T. Rueckert, D. Schnabel, J.A. Hawkesand, D.J. and Hill, D.L.G. VTK CISG Registration Toolkit: An Open Source Software Package for Affine and Non-rigid Registration of Single and Multi-modal 3D Images, *Proceedings of Bildverarbeitung in der Medizin*, Lipzig, Germany, 2002, 409-412.
- [128] The Computational Imaging Science Group Radiological Sciences Kings College London, *VTK CISG Registration Toolkit*, Available at: <<http://www-ipg.ums.ac.uk/cisg/vtk-software/>> [Accessed 20<sup>th</sup> August 2004].
- [129] Rueckert, D. Sonoda, L.I. Hayes, C. Hill, D.L.G. Leach, M.O. and Hawkes, D.J. Non-rigid Registration Using Free-form Deformations: Application to Breast MR Images, *IEEE Transactions on Medical Imaging*, 1999, 18, 712-721.
- [130] The National Library of Medicine, *Insight Segmentation and Registration Toolkit*, Available at: <<http://www.itk.org/>> [Accessed 25<sup>th</sup> August 2004].
- [131] Hanssen, N. Von Rymon-Lipinski, B. Jansen, T. Lievin, M. and Keeve, E. Integrating the Insight Toolkit ITK into a Medical Software Framework, *Proceedings of Computer Assisted Radiology and Surgery (CARS 02)*, Paris, France, 2002.
- [132] Ibanez, L. Schroeder, W. Ng, L. Cates, J. and the Insight Software Consortium. The ITK Software Guide: Second Edition, *Kitware Inc*, Available at: <<http://www.itk.org/>> [Accessed 25<sup>th</sup> August 2004].
- [133] Pratt, W.K. Correlation Techniques of Image Registration, *IEEE Transactions on Aerospace and Electronic Systems*, 1974, 10, 353-358.
- [134] Foroosh, H. Zerubia, J.B. and Berthod, M. Extension of Phase Correlation to Sub-pixel Registration, *IEEE Transactions on Image Processing*, 2002, 11, 188-200.

## References

---

- [135] Kaneko, S. Satoh, Y. and Igarashi, S. Using Selective Correlation Coefficient for Robust Image Registration, *Pattern Recognition*, 2003, 36, 1165-1173.
- [136] Nocedal, J. and Wright, S.J. Numerical Optimization, *Springer-Verlag*, USA, 1999.
- [137] Yoo, T.S. Insight into Images: Principles and Practices for Segmentation, Registration, and Image Analysis, *A K Peters*, USA, 2004.
- [138] Amdahl, G.M. Validity of the Single Processor Approach to Achieving Large-scale Computing Capabilities, *Proceedings of the AFIPS Conference*, 1967, 30, 483-485.
- [139] Gustafson, J.L. The Scaled-sized Model: A Revision of Amdahl's Law, *Proceedings of the 3<sup>rd</sup> International Conference on Supercomputing*, 1988, 130-133.
- [140] Xavier, C. and Iyengar, S.S. Introduction to Parallel Algorithms, *John Wiley & Sons*, USA, 1998.
- [141] The National Library of Medicine, *The Visible Human Project*, Available at: [http://www.nlm.nih.gov/research/visible/visible\\_human.html](http://www.nlm.nih.gov/research/visible/visible_human.html) [Accessed 15<sup>th</sup> September 2004].
- [142] Ding, L. Goshtasby, A. and Satter, M. Volume Image Registration by Template Matching, *Image and Vision Computing*, 2001, 19, 821-832.
- [143] Periaswamy, S. and Farid, H. Medical image Registration with Partial Data, *Medical Image Analysis*, 2006, 10, 452-464.
- [144] Smith, H.J. Quaternions for the Masses, Available at: <http://www.geocities.com/hjsmithh/Quatdoc/Qindex.html> [Accessed 5<sup>th</sup> January 2006].
- [145] Conway, J.H. and Smith, D.A. On Quaternions and Octonions: Their Geometry, Arithmetic, and Symmetry, *A.K. Peters Ltd*, 2003.
- [146] Stark, D.D. and Bradley, W.G. Magnetic Resonance Imaging: Third Edition, *Mosby Publishing*, St Louis, 1999.
- [147] The Stanford Volume Data Archive, Available at: <http://graphics.stanford.edu/data/voldata/> [Accessed 24<sup>th</sup> September 2004].
- [148] The Volume Dataset Portal, Available at: <http://www.custard.org/~andrew-visualization/datasets/> [Accessed 20<sup>th</sup> September 2004].

## References

---

- [149] The Volume Library, Available at: <<http://www9.informatik.uni-erlangen.de/External/vollib/>> [Accessed 21<sup>st</sup> September 2004].
- [150] The McConnell Brain Imaging Centre, *BrainWeb: Simulated Brain Database*, Available at: <<http://www.bic.mni.mcgill.ca/brainweb>> [Accessed 10<sup>th</sup> November 2005].
- [151] Hajnal, J.V. Hill, D.L.G. and Hawkes, D.J. *Medical Image Registration*, CRC Press, Boca Raton, 2001.
- [152] Maes, F. Collignon, A. Vandermeulen, D. Marchal, G. and Suetens, P. Multi-modality Image Registration by Maximization of Mutual Information, *IEEE Transactions on Medical Imaging*, 1997, 16, 187-198.
- [153] Zhu, Y.M. Volume Image Registration by Cross-entropy Optimization, *IEEE Transactions on Medical Imaging*, 2002, 21, 174-180.
- [154] Mattes, D. Haynor, D.R. Vesselle, H. Lewellen, T.K. and Eubank, W. Non-rigid Multi-modality Image Registration, *Medical Imaging 2001: Image Processing*, 2001, 1609-1620.
- [155] Pham, D.L. Prince, J.L. and Dagher, A.P. Estimation of Joint Probability Density Functions in Magnetic Resonance Imaging, *Proceedings of The Ninth Workshop on Image and Multidimensional Signal Processing (IMDSP)*, 1996, 148-149.
- [156] Mattes, D. Haynor, R.D. Vesselle, H. Lewellen, T.K. and Eubank, W. PET-CT Image Registration in the Chest Using Free-form Deformations, *IEEE Transactions on Medical Imaging*, 2003, 22, 120-128.
- [157] Wells, W. Viola, P. Atsumi, H. Nakajima, S. and Kikinis, R. Multi-modal Volume Registration by Maximization of Mutual Information, *Medical Image Analysis*, 1996, 1, 35-51.
- [158] Iqbal, S. and Carey, G.F. Performance of Dynamic Load Balancing Algorithms with Variable Number of Processors, *Journal of Parallel Distributed Computing*, 2005, 65, 934-948.
- [159] Zagrodsky, V. Shekhar, R. and Cornhill, F. Multi-function Extension of Simplex Optimization Method for Mutual Information-based Registration of Ultrasound Volumes, *Medical Imaging 2001: Image Processing*, 2001, 508-515.
- [160] Bolton, H.P.J. Schutte, J.F. and Groenwold, A.A. Multiple Parallel Local Searches in Global Optimization, *Lecture Notes in Computer Science*, 2000, 1908, 88-95.

## References

---

- [161] Rohlfing, T. Maurer, C.R. Bluemke, J. and Jacobs, M. Volume-preserving Non-rigid Registration of MR Breast Images Using Free-form Deformations with an Incompressibility Constraint, *IEEE Transactions on Medical Imaging*, 2003, 22, 730-741.
  
- [162] Loeckx, D. Maes, F. Vandermeulen, D. and Suetens, P. Non-rigid Image Registration Using Free-form Deformations with a Local Rigidity Constraint, *Lecture Notes in Computer Science*, 2004, 3216, 639-646.

## Appendix A: List of publications

Tait, R.J. Schaefer, G. and Hopgood, A.A. Registration and Fusion of Multi-sensor Data Using Multiple Agents, *Proceedings of SPIE: Opto-Ireland: Imaging and Vision*, Dublin, Ireland, 104-112, 2005.

Tait, R.J. Schaefer, G. Hopgood, A.A. and Nolle, L. Defect Detection Using a Distributed Blackboard Architecture, *Proceedings of the 19<sup>th</sup> European Simulation Multi-conference*, Riga, Latvia, 283-287, 2005.

Schaefer, G. Tait, R.J. Howell, K. and Hopgood, A.A. Automated Overlay of Visual and Thermal Images for the Assessment of Morphea Patients, *Thermology International*, 15(4), 157, 2005.

Tait, R.J. Schaefer, G. Howell, K. Hopgood, A.A. Woo, P. and Harper, J. Automated Overlay of Visual and Thermal Medical Images, *Proceedings of the 18<sup>th</sup> International EURASIP Conference (BIOSIGNAL 2006)*, Brno, Czech Republic, 260-262, 2006.

Tait, R.J. Schaefer, G. Hopgood, A.A. and Nolle, L. Automated Visual Inspection Using a Distributed Blackboard Architecture, *International Journal of Simulation Systems, Science and Technology*, 7(3), 12-21, 2006.

Tait, R.J. Schaefer, G. and Hopgood, A.A. iDARBS - A Distributed Blackboard System for Image Processing, *Proceedings of 13<sup>th</sup> International Conference on Systems, Signals and Image Processing (IWSSIP 2006)*, Budapest, Hungary, 431-434, 2006.

Tait, R.J. Schaefer, G. and Hopgood, A.A. Towards High Performance Image Registration Using Intelligent Agents, *Proceedings of 13<sup>th</sup> International Conference on Systems, Signals and Image Processing (IWSSIP 2006)*, Budapest, Hungary, 435-438, 2006.

Tait, R.J. Schaefer, G. Hopgood, A.A. and Zhu, S.Y. Efficient 3-D Medical Image Registration Using a Distributed Blackboard Architecture, *Proceedings of 28<sup>th</sup> Annual International Conference IEEE Engineering in Medicine and Biology Society (EMBC06)*, New York, USA, 3045-3049, 2006.

Schaefer, G. Tait, R.J. and Zhu, S.Y. Overlay of Thermal and Visual Medical Images Using Skin Detection and Image Registration, *Proceedings of 28<sup>th</sup> Annual International Conference IEEE Engineering in Medicine and Biology Society (EMBC06)*, New York, USA, 965-967, 2006.

Tait, R.J. Schaefer, G. and Hopgood, A.A. High Performance Medical Image Registration Using a Distributed Blackboard Architecture, *First IEEE Symposium on Computational Intelligence in Image and Signal Processing (CIISP07)*, Hawaii, USA, 252-257, 2007.

## Appendix B: Validation of iDARBS components

The Distributor KS was the first component validated. Tests were made to ensure that selection of an image could be made, followed by division and storage of segments. To perform these tests the blackboard module was started and the Distributor KS connected. The Distributor KS was observed to successfully clear the blackboard and placed an initial parameters string in the *Parameters* partition. The open file dialog box then appeared. Upon loading of an image, the image selection dialog box appeared as intended. Once selected the image size was added to the parameters string and a compressed segment created. Unfortunately, during sending of the compressed segment to the raw data container a problem was encountered. The size of the string sent between the Distributor KS and blackboard components was found to exceed the size of the incoming data buffer. To prevent failure of the Distributor KS in the event of strings that are larger in size than the buffer, a greedy algorithm which feeds a dynamically resizing buffer was been employed. With this modification made to all KSs and the blackboard module the compresses segment was successfully added to the raw data container and its identifier added to the *Unprocessed* partition. Creation and storage in the raw data container of 1–20 segments was observed as being successful.

Next to be tested was the Manager KS. The co-ordinated activation of Worker *n* KS activities together with monitoring of its current state, are areas in which testing was performed. To simulate the Worker *n* KS, a stub KS capable of printing messages to a debug window was created and connected to the blackboard. The activation of the stub KS, in the form of a message appearing in the debug window, was successfully achieved. Using the terminal client, a visual check on the *Worker n control* partition confirmed the presents of a start string. Subsequent successful activation of 1–20 stub KSs, also in the form debug window messages and control partition content checks, was also achieved. Additional functionality was then added to the stub KS, so that when activated a stop string would be added to the *Worker n control* partition. A debug message was also added to the Manager KS to indicate that it had successfully found the stop strings. Once test conditions had been reset, the Manager KS was again connected to the blackboard module and its actions evaluated. The Manager KS has been demonstrated to work correctly under a number of permutations, including the absence of the stop string in the *Worker n control* partition.

The ability of the Manager KS to retrieve processed segments and construct a resulting image was evaluated. In order to carry out testing, the blackboard module was started and a single compressed segment was added to the raw data container using the Distributor KS. Commands were then manually sent to the blackboard, by means of the terminal client, that added a stop string to a Worker KS control partition. On appearance of the stop string the single compressed segment was retrieved from the raw data container, decompressed and copied to the resulting image. Display of the resulting image occurred as expected. Using 1–20 segments, test conditions were reset and the Manager KS actions evaluated. During construction of the resulting image from multiple segments an error was found to occur. Debugging highlighted the problem as being caused by the incorrect calculation of position, of segments being copied to the resulting image. The error was promptly fixed allowing successfully creation and display of a resulting image. No resulting image was to be constructed in the absence of segments this behaviour was intended and was observed during subsequent testing.

During validation of Distributor and Manager KSs, Worker KSs were replaced with stubs. Upon testing of the Worker *n* KS implementation, an image segment was added to the *Unprocessed* partition. Using the terminal client, a start string was added to the *Worker n control* partition. Retrieval and decompression of the image segment, verified visually by means of the image viewer, occurred successfully. Functionality for the return of a processed segment was then added to the Worker *n* KS. Test conditions were replicated and a start string was added to the *Worker n control* partition. Successful compression and storage in the *Processed* partition of a segment occurred. Confirmation that the Worker *n* KS had placed the stop string in the *Worker n control* partition was established using the terminal client. Crucially, no erroneous behaviour was observed during testing of multiple Worker KS working concurrently as well as single and multiple Worker KS working concurrently combined with both Distributor and Manager KSs.

Initial debugging of the iDARBS framework exposed incompatibilities between the thread libraries installed as default with different operating systems. LinuxThreads and Native POSIX Threads Library for Linux (NPTL) are the libraries in question. It was discovered that whenever a thread was generated by the blackboard to service a KS, the thread failed to terminate resulting in deadlock. This fundamental problem was caused by an update of the computer network to a newer operating system. Importantly, DARBS was originally developed using Red Hat 9, the default thread library being NPTL. The newer operating system Ubuntu, had as default the LinuxThreads library. To swap from LinuxThreads to NPTL and hence remove the cause of deadlock, NPTL support was enabled and LinuxThreads support disabled in the Linux kernel. The glibc library was also recompiled with NPTL support. Although NPTL is designed as a replacement for LinuxThreads, depending on the kernel version being used LinuxThreads remains the default thread library for many operating systems.

The LinuxThreads are no longer under development and have been replaced with the Native POSIX Threads Library for Linux (NPTL). Although it is included with most distributions NPTL may not be the default thread library. Fortunately, NPTL threading is supplied as the default library with at least one kernel image. However, even with correct threading support the blackboard has been found to crash. For example, whilst employing 2 Worker KSs and after approximately 50 iterations sometimes a single client module will fail. Similarly when 4 Worker KSs are employed and after approximately 25 iterations a single client module will failed. These failures were found to be caused by the blackboard module. Using a system monitor the behaviour encountered was identified as being caused by a memory leak. Because this behaviour had not been experienced with either Red Hat 9 or Debian Sarge operating systems a corresponding set of glibc packages were installed. To solve the memory leak problem gcc-3.2, g++-3.2, and cpp-3.2 packages were installed.



## Appendix C: Test image capture conditions

During image capture, backscatter produced by the reflection of light from the background was found to be a major cause of problems. Positioning of the light source at angles of forty five and ninety degrees from straight ahead was therefore experimented with. An un-textured non-reflective sheet of card was used to minimise reflection of light back into the cameras field of view. Coloured card was also used at first as it was thought the use of colour would simplify the segmentation process. Unfortunately, due to the transparent nature of the bottle segmentation of edges became less accurate. Experience gained through testing has show that the use of black card best enhances the contrast between a bottle and its background.

For initial testing, light consisting of a desk lamp was used to illuminate the sample. The light source was positioned behind and above the camera in order to stop light shining directly into the cameras field of view. Under these illumination conditions the top of the bottle became very bright with a gradual reduction in intensity down the bottles surface. Dark areas and shadowing appeared around the bottom of the bottle. An addition bright reflective area appeared down the centre line of the bottles curved surface. Attempts were made to move the light source into a number of different positions and distances however they all resulted in the same effect. A second desk lamp was then introduced. Both light sources were positioned behind and either side of the camera. This configuration removed highlighting of the bottle which was encountered with the single light source. Unfortunately, two bright reflective areas appeared long the bottles curved surface. In all cases, care was taken not to introduce shadows caused by positioning the light source close to and behind the camera.

Since the bottles were semi-transparent, additional complications of illumination quickly became apparent. Traditionally, transparent objects have been illuminated from the top, sides and rear. This is because front lighting tended to pass directly through some material and does not reflect sufficient light back, in order to form a high quality image. Side lighting was used in an attempt to counter this problem. Unfortunately, due to its curved nature, this resulted in the dark print on the bottles reverse side becoming visible. The addition of backlighting further increased the visibility of unwanted print.

**Appendix D: iDARBS tables of results**

Mean filtering				
Worker KSs	Start time (s)	End time (s)	Difference (s)	Average (s)
<b>Sequential</b>	82.22	84.652	2.432	2.457
	202.22	204.575	2.355	
	123.343	125.927	2.584	
<b>X1</b>	74.101	86.601	12.5	12.2
	83.28	95.48	12.2	
	21.159	33.059	11.9	
<b>X2</b>	17.21	29.65	12.44	12.53
	48.38	61.381	13.001	
	41.268	53.417	12.149	
<b>X4</b>	87.503	100.794	13.291	13.311
	37.296	50.667	13.371	
	39.294	52.565	13.271	
<b>X6</b>	42.422	56.234	13.812	13.9
	42.493	56.59	14.097	
	201.067	214.858	13.791	
<b>X8</b>	68.24	83.461	15.221	15.3
	59.34	74.331	14.991	
	67.921	83.609	15.688	
<b>X10</b>	61.747	77.746	15.999	16.43
	66.28	82.931	16.651	
	25.613	42.253	16.64	
<b>X12</b>	93.439	111.362	17.923	17.8
	56.932	75.234	18.302	
	79.32	96.495	17.175	
<b>X14</b>	19.927	41.027	21.1	21.107
	23.745	45.146	21.401	
	23.936	44.756	20.82	
<b>X16</b>	13.23	36.211	22.981	23.343
	24.51	47.211	22.701	
	85.78	110.127	24.347	
<b>X18</b>	423.42	449.34	25.92	25.8
	12.944	37.945	25.001	
	76.112	102.591	26.479	
<b>X20</b>	49.299	77.407	28.108	28.3
	26.105	54.015	27.91	
	23.443	52.325	28.882	

Local histogram equalisation				
Worker KSs	Start time (s)	End time (s)	Difference (s)	Average (s)
<b>Sequential</b>	713.27	822.662	109.392	110.401
	428.367	545.597	117.23	
	674.618	779.199	104.581	
<b>X1</b>	591.258	718.558	127.3	128.304
	492.688	623.288	130.6	
	411.98	538.992	127.012	
<b>X2</b>	142.779	217.119	74.34	72.348
	242.979	317.209	74.23	
	137.437	205.911	68.474	
<b>X4</b>	43.124	88.688	45.564	47.376
	502.696	548.795	46.099	
	474.565	525.03	50.465	
<b>X6</b>	877.157	917.588	40.431	41.072
	492.905	532.105	39.2	
	583.989	627.574	43.585	
<b>X8</b>	529.717	572.108	42.391	39.221
	561.159	602.26	41.101	
	573.229	607.4	34.171	
<b>X10</b>	960.852	1001.087	40.235	41.678
	616.053	653.395	37.342	
	264.326	311.783	47.457	
<b>X12</b>	653.983	698.971	44.988	44.639
	630.575	672.876	42.301	
	729.132	775.76	46.628	
<b>X14</b>	687.56	731.799	44.239	45.893
	708.461	755.752	47.291	
	180.981	227.13	46.149	
<b>X16</b>	374.493	421.835	47.342	46.363
	274.323	321.24	46.917	
	473.727	518.557	44.83	
<b>X18</b>	346.364	396.365	50.001	50.004
	853.661	902.644	48.983	
	245.027	296.055	51.028	
<b>X20</b>	342.816	396.269	53.453	53.371
	453.243	507.475	54.232	
	678.43	730.858	52.428	

Adaptive thresholding				
Worker KSs	Start time (s)	End time (s)	Difference (s)	Average (s)
<b>Sequential</b>	7622.862	8246.595	623.733	625.481
	8547.595	9172.855	625.26	
	179.799	807.249	627.45	
<b>X1</b>	9518.758	10183.74	664.982	666.342
	228.683	896.604	667.921	
	10932.598	11598.721	666.123	
<b>X2</b>	219.117	686.41	467.293	469.546
	203.917	668.819	464.902	
	3912.105	4388.548	476.443	
<b>X4</b>	684.889	946.371	261.482	263.024
	5785.543	6043.655	258.112	
	234.395	503.873	269.478	
<b>X6</b>	598.987	783.22	184.233	186.704
	805.132	993.423	188.291	
	524.727	712.315	187.588	
<b>X8</b>	108.202	249.868	141.666	142.996
	426.342	569.934	143.592	
	923.671	1067.401	143.73	
<b>X10</b>	1087.01	1219.441	132.431	133.234
	495.354	628.363	133.009	
	613.348	747.61	134.262	
<b>X12</b>	671.698	796.352	124.654	126.223
	776.172	904.608	128.436	
	876.592	1002.171	125.579	
<b>X14</b>	319.921	463.142	143.221	142.451
	743.235	889.465	146.23	
	213.237	351.139	137.902	
<b>X16</b>	842.421	989.72	147.299	148.028
	241.521	391.25	149.729	
	857.578	1004.634	147.056	
<b>X18</b>	935.236	1090.065	154.829	153.261
	424.209	580.601	156.392	
	1055.226	1203.788	148.562	
<b>X20</b>	289.499	449.42	159.921	159.892
	972.556	1135.449	162.893	
	443.23	600.092	156.862	

## Appendix E: Image registration rules

### Distributor KS

- Initialise\_Distributor
  - Clears the blackboard of all data.
  - Performs initial setting up of variables.
- Select\_Images
  - Displays an image viewer.
  - Waits for selection of fixed and moving images.
  - Adds initial parameters to the blackboard.
- Set\_Transform
  - Calculates fixed and moving images centres using moments.
  - Adds the vector between the centres to the blackboard as an initial transform.
- Store\_Segments
  - Divides the fixed and moving images into segments.
  - Compressed image segments.
  - Sends compressed segments to the blackboard for storage.

### Worker n KS

- Initialise\_Worker\_n
  - Fetches initial parameters from the blackboard.
  - Performs initial setting up of variables.
- Wait\_Worker\_n
  - Causes the Worker n KS to repeatedly loop.
- Fetch\_Segment\_n
  - Fires only once on appearance of the first current parameters information string.
  - Retrieves fixed and moving image segments from the blackboard.
  - Decompressed the fetched segments.
- Perform\_Optimisation\_n
  - Fires on appearance of the current parameters information string.
  - Extracts parameters from the current parameters information string.
  - Generates a local derivative using extracted transform parameters.
  - Replaces the current parameters information string with a derivative information string.

### **Manager KS**

- **Initialise\_Manager**
  - Fetches initial parameters from the blackboard.
  - Performs initial setting up of variables.
  - Propagates the current parameters information string to all Worker KSs.
- **Wait\_Manager**
  - Causes the Manager KS to repeatedly loop.
- **Advance\_Transform**
  - Retrieves local derivatives and valid pixel numbers for all worker control partitions.
  - Sums retrieved local derivatives to form a global derivative.
  - Sums retrieved valid pixel numbers to form a global number of pixels.
  - Updates the current transform parameters using the global derivative and number of valid pixels.
  - Propagates the updated transform parameters to all Worker KSs.
- **Resample\_Image**
  - Fires on appearance of stop information strings.
  - Retrieves processed image segments from the blackboard.
  - Decompressed the fetched segments.
  - Removes borders from segments.
  - Constructs a resulting images and displays by means of an image viewer.

```
/*
Employed by the Distributor KS.
Adds initial optimisation parameters to the blackboard.
*/
RULE Initalise_Distributor

IF
[
  [not_on_partition [Initalised] [DistributorControl]]
]

THEN
[
  [add [Optimisation 1.00 0.001 200] [Parameters]]
  AND
  [run_algorithm [IP_Distributor.so Initalise[NIL] sAlgorithmResult]]
  AND
  [add [Initalised] [DistributorControl]]
]

BECAUSE
  [Distributor not initalised]
END
```

```
/*
Employed by the Distributor KS.
Using centres of mass an initial centre of rotation and translation is estimated.
*/
RULE Set_Transform

IF
[
  [not_on_partition [Transform] [DistributorControl]]
]

THEN
[
  [run_algorithm [IP_Distributor.so CalculateInitalTransform[NIL]
sAlgorithmResult]]
  AND
  [add [InitalTransform ~sAlgorithmResult] [Parameters]]
  AND
  [add [Transform] [DistributorControl]]
]

BECAUSE
  [Transform not set]
END
```

## Appendix E: Image registration rules

---

```
/*
Employed by the Worker n KS.
Used to generate local derivatives of the similarity metric with respect to each transform
parameter.
*/
RULE Perform_Optimisation_1

IF
[
  [on_partition [Current ?trans ?grad] [Worker1Control]]
]

THEN
[
  [run_algorithm [IP_Worker.so GenerateLocalDerivatives[~trans
~grad] sAlgorithmResult]]
  AND
  [replace [Current ==] [Worker1Control] [Derivative
~sAlgorithmResult]]
]

BECAUSE
  [Optimisation not performed]
END

/*
Employed by the Manager KS.
Used to propagate the initial transform parameters to all worker control partitions and
acts as a trigger mechanism to commence Worker KS activities.
*/
RULE Initalise_Manager

IF
[
  [not_on_partition [Initalised] [ManagerControl]]
  AND
  [on_partition [InitalTransform ?initalTransform] [Parameters]]
]

THEN
[
  [add [Current ~initalTransform 0.00_0.00] [Parameters]]
  AND
  [add [Current ~initalTransform 0.00_0.00] [Worker1Control]]
  AND
  [add [Current ~initalTransform 0.00_0.00] [Worker2Control]]
  AND
  [add [Initalised] [ManagerControl]]
]

BECAUSE
  [Manager not initalised]
END
```



```

/*
Employed by the Manager KS.
Used to accumulate local derivatives, compute global derivatives, and update the
current transform parameters.
*/
RULE Advance_Parameters

IF
[
  [on_partition [Current ?trans ?grad] [Parameters]]
  AND
  [on_partition [Derivative ?grad1 ?noPixels1] [Worker1Control]]
  AND
  [on_partition [Derivative ?grad2 ?noPixels2] [Worker2Control]]
]

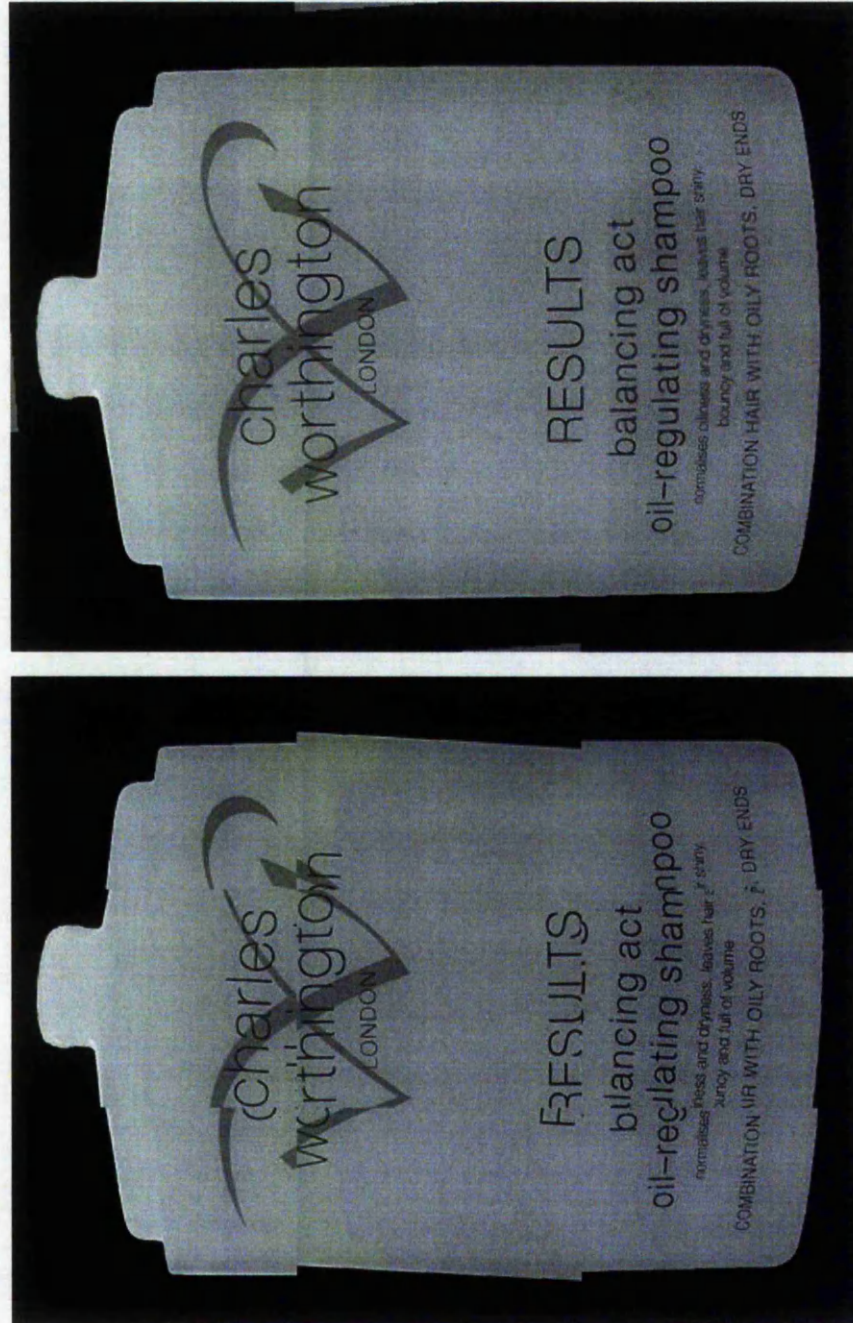
THEN
[
  [run_algorithm [IP_Manager.so SumPixels[~noPixels1 ~noPixels1]
sumedPixels]]
  AND
  [run_algorithm [IP_Manager.so SumDerivatives[~grad1 ~grad1
~sumedPixels]
sumedGradient]]
  AND
  [run_algorithm [IP_Manager.so AdvanceParameters[~trans
~sumedGradient] sAlgorithmResult]]
  AND
  [replace [Current ~trans ~grad] [Parameters] [~sAlgorithmResult]]
  AND
  [replace_multi [Derivative ~grad1 ~noPixels1] [Worker1Control]
[~sAlgorithmResult] [Derivative ~grad2 ~noPixels2] [Worker2Control]
[~sAlgorithmResult] ]
]

BECAUSE
  [Parameters not advanced]
END

```

Simple Double-To-String, String-To-Double, Long-To-String, and String-To-Long methods form the basic components for creation and interpretation of information strings processed by the rule files. With an accuracy of 18 decimal places, STL string stream operators are used to convert double precision and long integer numbers into strings and back again when required. By appending numbers to an underscore delimited string, information strings containing transform parameters and derivatives are created. A Transform-To-String method is used to convert a transform into an information string which is tagged as being previous, current, or final depending on its required purpose. The conversion of derivatives into a string is performed by a Derivative-To-String method. Unsurprisingly, the parsing of information strings into components and conversion into double precision type is performed by String-To-Transform and String-To-Derivative methods.

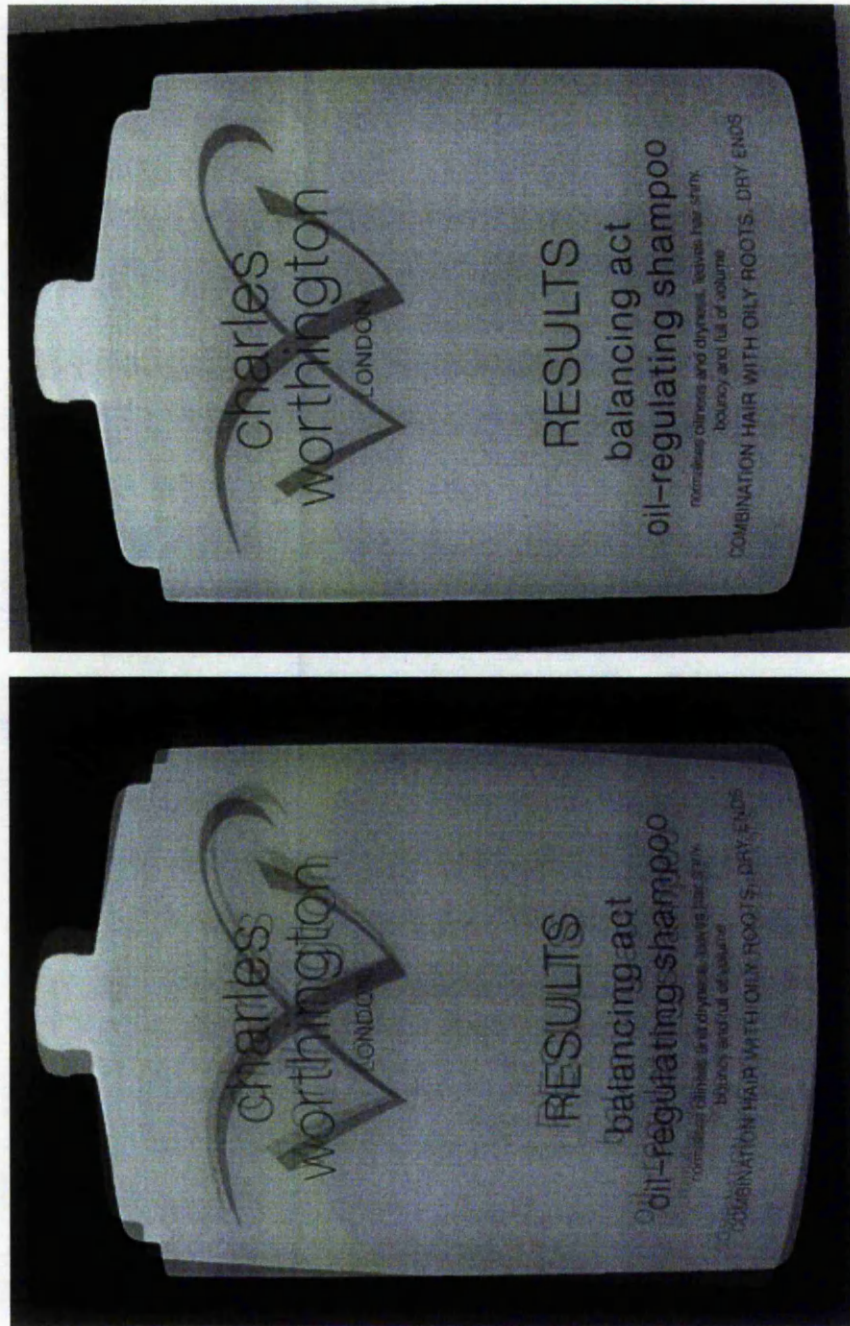
## Appendix F: Visualisation of registered images



Unregistered and registered checkerboard composite images.







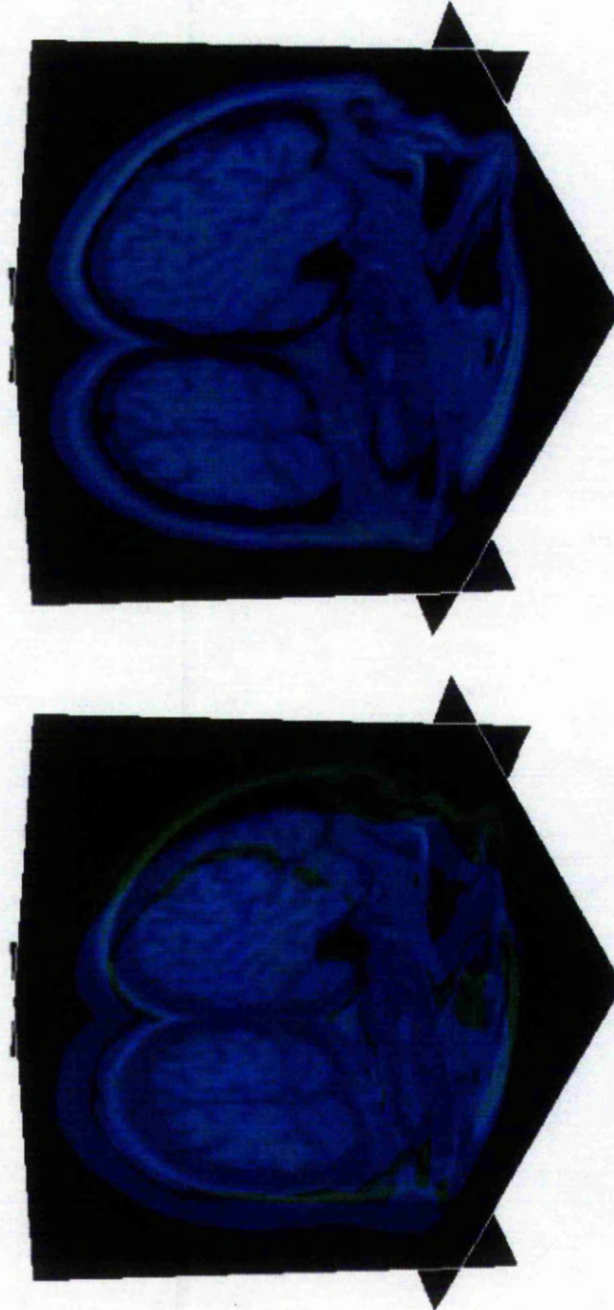
Unregistered and registered weighted overlay images.

**Appendix G: Image registration tables of results**

Image registration with mean square error as a similarity metric				
Worker KSS	Start time (s)	End time (s)	Difference (s)	Average (s)
<b>Sequential</b>	283.12	550.231	267.111	268.193
	7855.2	8119.434	264.234	
	225.566	498.8	273.234	
<b>X1</b>	983.11	1281.304	298.194	302.021
	1806.847	2113.481	306.634	
	2385.96	2687.195	301.235	
<b>X2</b>	265.613	445.53	179.917	179.1703
	186.89	365.678	178.788	
	570.56	749.366	178.806	
<b>X4</b>	157.945	285.559	127.614	127.001
	531.76	657.87	126.11	
	4881.495	5008.774	127.279	
<b>X6</b>	497.495	606.718	109.223	108.308
	7962.56	8070.48	107.92	
	8367.48	8475.261	107.781	
<b>X8</b>	867.849	970.84	102.991	103.277
	388.278	491.97	103.692	
	1939.606	2042.754	103.148	
<b>X10</b>	573.456	677.557	104.101	103.927
	1093.388	1197.269	103.881	
	337.34	441.139	103.799	
<b>X12</b>	365.599	474.031	108.432	108.912
	2902.753	3010.653	107.9	
	329.545	439.949	110.404	
<b>X14</b>	800.56	920.55	119.99	121.927
	223.42	344.181	120.761	
	785.585	910.615	125.03	
<b>X16</b>	132.33	274.586	142.256	142.225
	670.693	816.025	145.332	
	211.88	350.967	139.087	
<b>X18</b>	30.382	185.503	155.121	152.671
	617.18	769.071	151.891	
	177.679	328.68	151.001	
<b>X20</b>	3160.873	3327.072	166.199	164.163
	3749.704	3913.691	163.987	
	4316.134	4478.437	162.303	

Image registration with normalised correlation as a similarity metric				
Worker KSs	Start time (s)	End time (s)	Difference (s)	Average (s)
<b>Sequential</b>	59623.961	60307.682	683.721	681.426
	1122.148	1802.379	680.231	
	3029.318	3709.644	680.326	
<b>X1</b>	2661.744	3410.856	749.112	749.291
	4823.23	5571.95	748.72	
	3452.34	4202.381	750.041	
<b>X2</b>	543.654	958.636	414.982	416.323
	503.74	921.63	417.89	
	285.334	701.431	416.097	
<b>X4</b>	69837.398	70085.58	248.182	249.713
	625.752	875.268	249.516	
	222.828	474.269	251.441	
<b>X6</b>	4069.266	4273.139	203.873	204.822
	632.767	836.769	204.002	
	263.111	469.702	206.591	
<b>X8</b>	117.215	309.416	192.201	192.016
	715.437	909.417	193.98	
	296.691	486.558	189.867	
<b>X10</b>	539.239	743.53	204.291	203.89
	243.983	446.894	202.911	
	483.63	688.098	204.468	
<b>X12</b>	7533.14	7758.44	225.3	227.231
	268.386	496.617	228.231	
	8949.933	9178.095	228.162	
<b>X14</b>	6099.41	6365.611	266.201	267.412
	682.304	950.694	268.39	
	309.43	577.075	267.645	
<b>X16</b>	132.33	441.142	308.812	310.162
	670.693	982.723	312.03	
	211.88	521.524	309.644	
<b>X18</b>	112.483	472.873	360.39	363.812
	2750.942	3114.861	363.919	
	350.82	717.947	367.127	
<b>X20</b>	7979.976	8370.096	390.12	392.989
	8717.383	9107.368	389.985	
	399.542	798.404	398.862	

## Appendix H: Visualisation of registered volumes



Unregistered and registered coloured volume renderings.



**Appendix I: Volume registration tables of results**

Volume registration with mean square error as a similarity metric				
Worker KSs	Start time (s)	End time (s)	Difference (s)	Average (s)
<b>Sequential</b>	6528.669	8198.155	1669.486	1646.682
	88836.91	90472.212	1635.302	
	1091.72	2726.98	1635.26	
<b>X1</b>	1510.4	3261.729	1751.329	1745.501
	4093.43	5839.67	1746.24	
	3032.366	4771.3	1738.934	
<b>X2</b>	79922.539	80900.905	978.366	991.940
	1704.614	2705.139	1000.525	
	3688.261	4685.191	996.93	
<b>X4</b>	1052.711	1619.716	567.005	552.259
	117.352	660.779	543.427	
	342.993	889.34	546.347	
<b>X6</b>	393.785	789.452	395.667	397.734
	185.992	586.99	400.998	
	2989.58	3386.118	396.538	
<b>X8</b>	4778.543	5114.389	335.846	337.211
	133.402	469.925	336.523	
	7001.956	7341.22	339.264	
<b>X10</b>	448.553	757.409	308.856	305.483
	182.965	485.23	302.265	
	2981.32	3286.64	305.32	
<b>X12</b>	3741.45	4049.955	308.505	308.786
	538.48	853.645	315.165	
	4829.27	5131.96	302.69	
<b>X14</b>	4911.32	5217.91	306.59	313.507
	384.345	702.077	317.732	
	7038.094	7354.294	316.2	
<b>X16</b>	5914.179	6236.87	322.691	327.749
	6719.518	7045.78	326.262	
	540.186	874.48	334.294	
<b>X18</b>	634.501	980.72	346.219	349.233
	460.882	824.344	363.462	
	327.11	665.13	338.02	
<b>X20</b>	2816.638	3178.4	361.762	359.455
	3756.595	4112.686	356.091	
	629.24	989.754	360.514	



Volume registration with normalised correlation as a similarity metric				
Worker KSs	Start time (s)	End time (s)	Difference (s)	Average (s)
<b>Sequential</b>	28651.71	32734.53	4082.82	4098.23
	68117.82	72209.85	4092.03	
	2910.19	7030.03	4119.84	
<b>X1</b>	1321.229	5687.149	4365.92	4357.8
	39274.76	43634.68	4359.92	
	3921.3	8268.86	4347.56	
<b>X2</b>	38019.092	40402.01	2382.918	2383.38
	2721.281	5101.282	2380.001	
	4912.914	7300.135	2387.221	
<b>X4</b>	19645.291	20876.182	1230.891	1240.26
	7972.66	9211.67	1239.01	
	3324.23	4575.109	1250.879	
<b>X6</b>	8322.819	9180.211	857.392	865.26
	8829.832	9702.842	873.01	
	7885.219	8750.597	865.378	
<b>X8</b>	1151.98	1822.362	670.382	677.46
	9046.081	9720.901	674.82	
	700183.925	700871.103	687.178	
<b>X10</b>	5784.81	6394.011	609.201	604.98
	1284.83	1884.821	599.991	
	295.931	901.679	605.748	
<b>X12</b>	4092.011	4727.931	635.92	637.32
	294.944	935.044	640.1	
	128.812	764.752	635.94	
<b>X14</b>	123.23	848.059	724.829	726.72
	238.129	964.051	725.922	
	201.934	931.343	729.409	
<b>X16</b>	50392.839	51215.84	823.001	823.2
	59023.29	59841.039	817.749	
	2933.454	3762.304	828.85	
<b>X18</b>	8643.015	9596.143	953.128	950.58
	9038.72	9987.642	948.922	
	2384.23	3333.92	949.69	
<b>X20</b>	309238.23	310418.211	1179.981	1186.92
	49922.33	51109.758	1187.428	
	3883.29	5076.641	1193.351	

Volume registration with mutual information as a similarity metric				
Worker KSs	Start time (s)	End time (s)	Difference (s)	Average (s)
<b>Sequential</b>	356.804	815.527	458.723	455.46
	382.829	834.84	452.011	
	288.921	744.567	455.646	
<b>X1</b>	235.5	740.482	504.982	506.004
	839.992	1346.114	506.122	
	329.991	836.899	506.908	
<b>X2</b>	464.442	783.435	318.993	318.72
	583.821	899.688	315.867	
	440.392	761.692	321.3	
<b>X4</b>	239.76	433.991	194.231	194.82
	823.277	1018.797	195.52	
	322.864	517.573	194.709	
<b>X6</b>	104.198	257.068	152.87	153.12
	2019.283	2173.674	154.391	
	2993.291	3145.39	152.099	
<b>X8</b>	80960.535	81122.778	162.243	161.46
	782.982	945.092	162.11	
	271.119	431.146	160.027	
<b>X10</b>	2702.187	2895.019	192.832	194.4
	3822.232	4017.432	195.2	
	182.492	377.66	195.168	
<b>X12</b>	544.342	789.662	245.32	244.44
	728.928	978.929	250.001	
	2664.38	2902.379	237.999	
<b>X14</b>	298.91	597.78	298.87	300.06
	293.921	595.142	301.221	
	998.239	1298.328	300.089	
<b>X16</b>	285.829	673.053	387.224	385.8
	742.934	1133.181	390.247	
	296.38	676.309	379.929	
<b>X18</b>	739.938	1160.119	420.181	419.28
	493.303	910.778	417.475	
	389.293	809.477	420.184	
<b>X20</b>	382.928	840.76	457.832	460.86
	769.452	1231.208	461.756	
	493.829	956.821	462.992	