AUTOMATED VISUAL INSPECTION USING A DISTRIBUTED BLACKBOARD ARCHITECTURE

Roger J. Tait, Gerald Schaefer, Adrian A. Hopgood, and Lars Nolle
School of Computing and Informatics
Nottingham Trent University
Clifton Lane, Nottingham, NG11 8NS, UK
e-mail: roger.tait@students.ntu.ac.uk, {gerald.schaefer,adrian.hopgood,lars.nolle}@ntu.ac.uk

Abstract. The framework of an automated vision system for monitoring quality control is presented where detection of various forms of defects is achieved by combining distributed artificial intelligence and image processing. A distributed blackboard architecture manages the processing of image data via an area of shared memory where the current understanding of the problem evolves. Registration into a common co-ordinate system and segmentation of reference and sensed images is performed in a multi-resolution fashion by intelligent agents that work in parallel and communicate with each other by means of the blackboard. Pixel-level fusion is then performed on registered images in order to exploit complementary and redundant data, allowing identification of suspected defects.

Keywords: visual inspection, defect detection, blackboard system, parallel image processing, image registration.

1. INTRODUCTION

In manufacturing industries there is an increasing need for automated detection and characterisation of defects. The motivating factors for the adoption of an automated inspection approach include the reduction of expensive labour costs, reproducibility, and the matching of high-speed inspection with high-speed production. Identification of functional and cosmetic defects in finished products has been achieved using a number of techniques; a general overview of automated visual inspection is provided by (Newman and Jain 1995). The processing techniques described can be grouped into referential comparison (Lee 1978), non-referential modelling (Wen and Tao 1999), and hybrid inspection (Bayro-Corrochano 1993).

A variety of approaches in the Printed Circuit Board (PCB) production environment are described by (Moganti *et al.* 1996). One such technique (Gokturk *et al*. 1999) uses the differentiating characteristics of a design, stored as a library, to compare and detect defects in a captured image. The detection of defects in the production of moulded plastic products which employs a Fourier descriptor and differential gradient operator to classify imperfections as being either shape or surface anomalies is described by (Petkovic *et al*. 2002). Despite the advantages of speed, consistency, and accuracy of an automated system over manual inspection, often the objective of these approaches is to ensure electrical connectivity, check form, and classify quality using an inspection technique that consists of a sequentially defined set of steps.

In many referential and hybrid inspection techniques, image registration (Zitova and Flusser 2003) is used to geometrically align two images taken from different sensors, viewpoints or instances in time. Both reference (fixed) and sensed (moving) images are aligned through a combination of scaling, translation, and rotation. The basic registration process consists of four stages: feature detection, transform optimisation, feature matching, and image re-sampling. Distinguishing characteristics are identified during the feature-detection stage. The transform-optimisation stage controls estimation of transform parameters that geometrically align fixed and moving images. Feature matching is achieved through the use of similarity measures (Penney *et al*. 1998) in which a degree of likeness between corresponding image features is calculated. On selection of an appropriate transform, pixel values that are mapped into non-integer coordinates are interpolated to establish their value. This represents the image re-sampling stage. In contrast, intensity-based algorithms omit the feature-detection stage.

The use of distributed computing in order to overcome time constraints associated with many image processing problems, such as registration, is growing in popularity. High image resolutions coupled with complex algorithms have increased the demand for high-speed processing capabilities. Conveniently, many algorithms used for the processing of images are inherently parallel and are therefore well suited to a distributed architecture (Taniguchi *et al*. 1997). In general, the parallel processing of an image consists of image distribution, local processing, data transfer during processing, and sub-image accumulation. An important consideration when adopting a parallel processing approach is the architecture of the host system. In a scheme based on a single machine with multiple processors, data distribution is not required. This system can be viewed as a tightly-coupled architecture. In contrast, a loosely-coupled scheme consisting of multiple computers in different locations will require its own distribution, communication, and accumulation mechanisms. Tools for the development of platform-independent parallel image processing applications have been reported by (Nicolescu and Jonker 2000).

The framework presented in this paper is designed to monitor the quality of a sample and to be capable of providing information for the removal of a defective product further along a production line. We have adopted a distributed blackboard architecture that supports multiple agents, each of which performs an independent task. The blackboards distributed nature makes available the performance benefits of parallel processing and allows for the possibility of integrating different inspection techniques. Tests show that the framework described realises referential comparison with substantial speedups when compared with a non-distributed implementation.

2. THE BLACKBOARD ARCHITECTURE

DARBS the Distributed Algorithmic and Rule-based Blackboard System (Nolle *et al.* 2001), is a distributed blackboard architecture based on a client/server model. The server functions as the blackboard while agents, known as knowledge sources (KSs) are implemented as client modules. The distributed nature of the implementation means both blackboard and client modules are running as separate processes. These independent processes may then reside on any PC in a network connected by TCP/IP communication. DARBS was developed at the Open and Nottingham Trent Universities. The availability of its source code has made possible the necessary modifications for this research. DARBS and its predecessor software have been employed in a variety of applications (Hopgood *et al.* 1997).

A blackboard system is a multi-agent system in which the agents, or KSs, can only communicate via the blackboard. During execution, the blackboard is used to host the evolution of a solution to a problem. A KS in contrast is a structure in which rules and algorithms can be embodied. Reading from and writing to the blackboard is implemented as standard KS functionality and provides the mechanism for communication between KSs. When updates to the solution of a problem are made, broadcast messages are used to inform all relevant KSs. A KS's behaviour guides it on a course of action in response to these messages. Storage on the blackboard of data processed by the framework ensures equal access for all active KSs. The modular implementation allows for the addition of extra KSs with specialised behaviour as required.

Blackboard modules with specialised behaviour, developed for the purposes of this research, include a Distributor, several Workers and a Manager KS as shown in figure 1. For a KS to be part of the framework it must first establish connection with the blackboard over the network. Framework initialisation and image selection are performed by the Distributor KS. The Distributor KS then splits an image into segments before placing them on the blackboard. Worker KSs take segments from the blackboard, register them and process them according to their implemented behaviour. The Manager KS coordinates all Worker KSs activities. A resulting image is then constructed from processed segments.

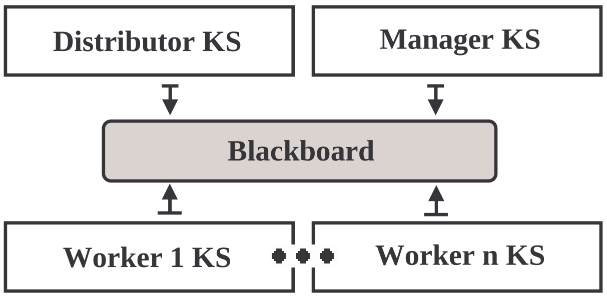

Figure 1: Blackboard modules

2.1 Blackboard Partitions

As the current state of a problem is stored on the blackboard, partitioning is used to balance communication and processing workloads. The partitioning of data aids the design of the framework by introducing structure to the blackboard. Due to the exhaustive search required, a drop in performance can be expected with a single-partition implementation. Similar inefficiency can be expected when a KS requests information through management and processing of excess partitions (Choy *et al.* 2004). To combat these problems the chosen partition implementation allows interaction between KSs in a logical and efficient manner. Also to simplify the creation of KS rules, the number of partitions with which a KS works is kept to a minimum.

In figure 2 we show KS access to blackboard partitions.

- The *Distributor control* partition controls division of an image into segments.
- The *Worker control* partitions are used to manage processing of segments.
- The *Manager control* partition is used for supervision of Worker KSs.
- The *Parameters* partition is used for maintenance of variables used by all KSs.
- The *Fixed*, *Moving*, and *Processed* partitions hold segments of their respective types.

2.2 Data Transmission between Components

Segment data is transmitted to and from the blackboard by the KSs. Transmission data is divided into several parts. The first component corresponds to an image identification number while the next values represent the image size. A minus symbol marks the end of attributes and the start of pixel data. The image format adopted for the inspection framework is that of the ITK (NLM 2004) image type which is implemented as an

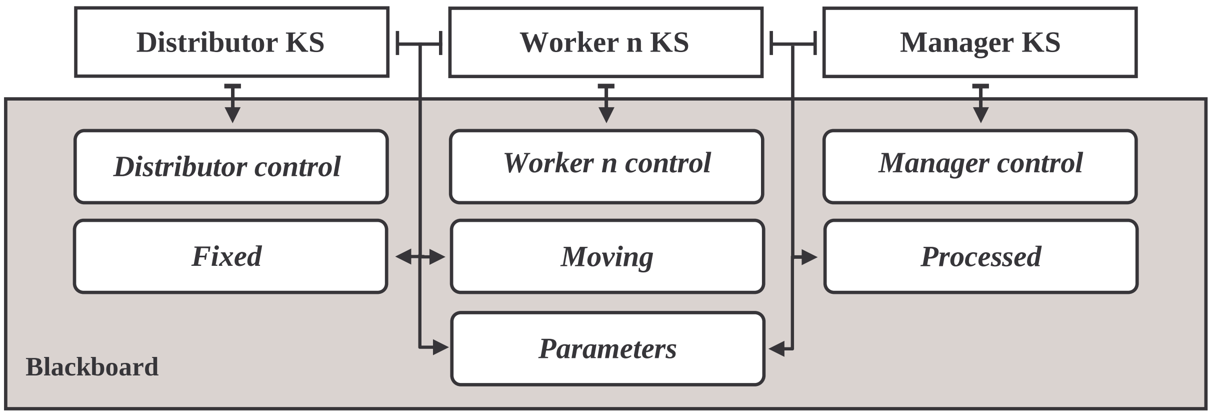

Figure 2: Blackboard partitions

unsigned char array. As the intensity levels in a segment can cover the full range of ASCII characters, care is taken to prevent disruption of a transmission due to escape characters.

2.3 Distribution of Image Data

Because the most effective distribution scheme depends on the framework's architecture, several representations have been evaluated. Finally, a distribution scheme was chosen whereby full resolution images are divided into a variable number of segments, each containing approximately the same number of pixels. As illustrated in figure 3, an image is split up into 2 to 10 segments. This maximises the possibility of detail appearing in all segments and evenly distributes the workloads between processors. The distribution scheme also benefits from the fact that no inter-processor communication is required. Duplication methods were not considered due to their transmission overheads.

3. KS IMPLEMENTATION

By changing the structure of a KS's rules and through provision of additional functionality, the behaviour of a KS can be altered. The ITK toolkit was used to provide KSs with registration and segmentation functionality through the embedding in rule files of shared library algorithms. Three simple image processing modules that interface ITK with the framework have been created; they encapsulate registration, segmentation, and fusion functionality.

An algorithm that is capable of aligning images produced by a range of modalities forms the basis of the registration module. The algorithm can be tailored, via the blackboard, to a specific problem with dynamically selectable components. The components consist of transform, interpolation, metric, and optimiser types. Both translation-only and affine transform types are available to perform a spatial mapping between points in fixed and moving segments. In order to evaluate pixel values at non-grid positions, bi-linear and nearest neighbour interpolation schemes are provided. Either normalised cross correlation or mutual information metrics can be used to measure the match between segments after a transformation has been applied. A gradient-descent optimiser is used to search iteratively for the transform that best satisfies the chosen metric.

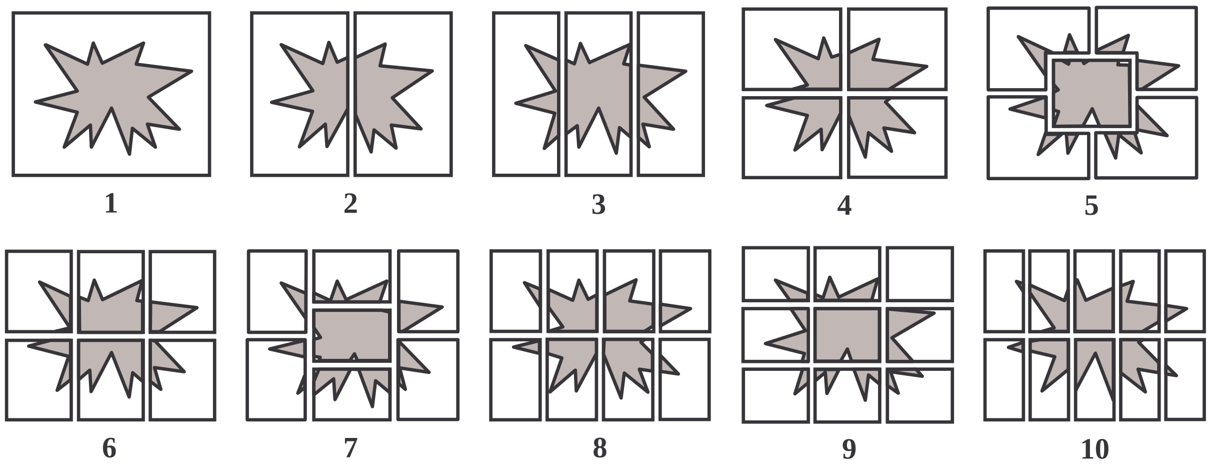

Figure 3: Image segments

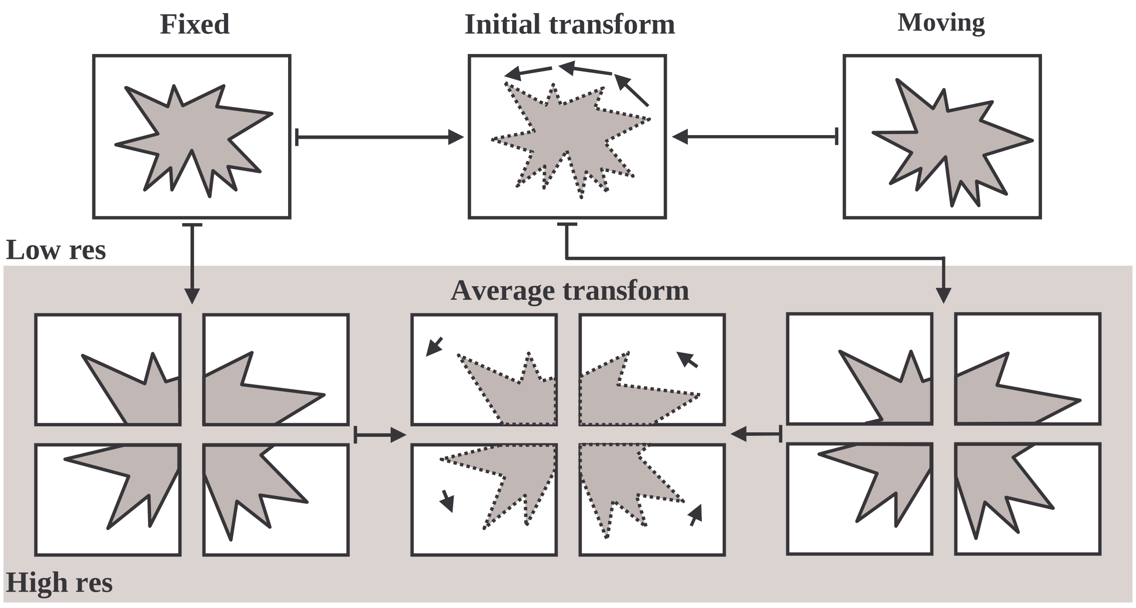

Figure 4: Multi-resolution scheme

Global and automatic threshold-level selection functionality to separate an object from its background, through maximising the between-class variance (Otsu 1979), is provided by the segmentation module. Morphological operators for noise removal are also part of the segmentation module. The fusion module implements an exclusive-or (XOR) operation which is employed by KSs to identify immediate differences between images and hence potential defects.

An overview of the multi-resolution scheme employed to increase speed and accuracy of the framework is given in figure 4. Sub-sampling is used to create a low-resolution image. As most of the iterations are attempted at the low-resolution level where the number of pixels is small, considerable savings in terms of computational time are achieved. Optimisation then continues to the full-resolution segments. As a consequence, registration of large scale features is achieved first and only small-scale corrections are required at full resolution. Importantly, by starting at a low-resolution level, local minima within the search space are usually avoided.

The multi-resolution approach is achieved in the following way:

- Images are loaded into the framework by means of the Distributor KS. Both fixed and moving images are then down-sampled and placed in their respective partitions.
- A single Worker KS retrieves the low-resolution images from the blackboard and passes them to the registration module. The registration module calculates a transform which best aligns the two images. Once calculated, the Worker KS places the transform in the *Parameters* partition.
- The Distributor KS clears the *Fixed* and *Moving* partitions of all data. The full-resolution fixed image is divided according to the current distribution scheme and segments are placed in the *Fixed* partition.
- The Manager KS acknowledges appearance of the transform on the blackboard.
- The transform is fetched by the Distributor KS and used to resample the full-resolution moving image. The moving image is then divided using the same distribution scheme and resulting segments are placed in the *Moving* partition.
- Individual Worker KSs retrieve segments from the *Fixed* and *Moving* partition and store them locally. Each Worker KS calculates a transform using the registration module and places it in the *Parameters* partition.
- Once transforms have been calculated for all segments, they are fetched from the *Parameters* partition by the Manager KS. The transforms are averaged and the average is placed in the *Parameters* partition.
- The Worker KSs retrieve the average transform and use it to resample their locally stored moving segment. Segmentation and fusion according to a Worker KS's implemented behaviour is then performed. Processed segments are placed in the *Processed* partition.
- Finally, the Manager KS fetches all processed segments from the blackboard and constructs the resulting image.

3.1 The Distributor KS

The Distributor KS consists of the following six rules:

- *Initalise_Framework*
- *Set_Parameters*
- *Select_Images*
- *Resize_Images*

- *Down_Sample*
- *Distribute_Segments*

Tasks performed by the *Initalise_Framework* rule include the placement of global data on the blackboard. The data include the size of borders between adjacent segments and the number of segments into which an image is to be divided. Initial registration parameters stored on the blackboard are set by means of *Set_Parameters*.

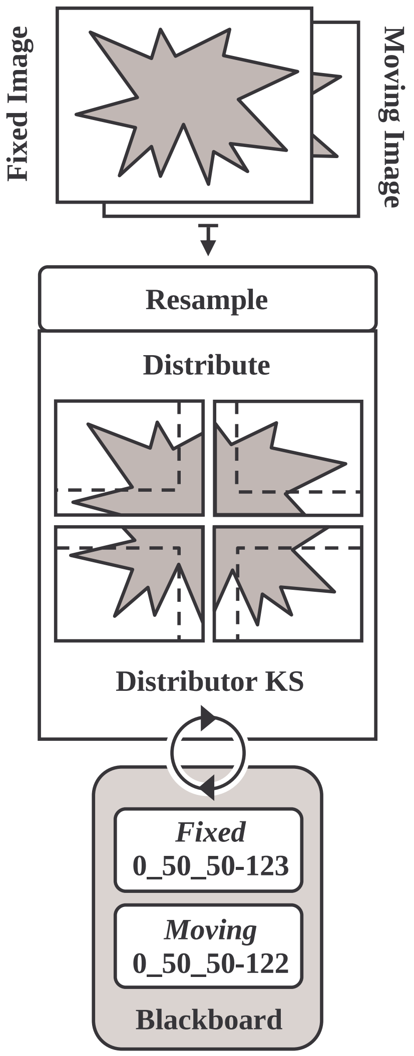

Figure 5: Distributor KS

Fixed and moving images are loaded into the framework on firing of the *Select_Images* rule. *Resize_Images* causes both fixed and moving images to be resized, thus ensuring segments created from the two images will be of the same size. If required, pixels from the bottom and right-hand sides of an image are removed. Resizing also simplifies the production of a resulting image. Images are down-sampled whenever the *Down_Sample* rule is fired. Likewise, division of images into segments and the sending to their respective partitions is performed by *Distribute_Segments*. Importantly, before transmission, an identification number is assigned to each down-sampled image and full-resolution segment.

In figure 5 we show the dividing and storing, in *Fixed* and *Moving* partitions, of an image by the Distributor KS. Since the number of segments is variable, an optimum can be found. An increase in computational overhead, however, will occur when a large number of segments are allocated. Similarly, the objective of parallel implementation will be defeated when the number of segments is small. The figure also shows that only edges that face neighbouring segments have a border. The border is designed to remove non-pixel values that enter at the edges of a segment due to translation and rotation during registration. It also removes the inconsistencies caused by convolution algorithms that require each pixel's neighbourhood. Although the size of border is variable, the setting of a wide border will cause a decrease in efficiency as additional redundant data will be accrued and processed by the framework.

3.2 The Worker KS

The Worker KS comprises five rule files:

- *Initalise_Worker*
- *Wait_Worker*
- *Fetch_Segments*
- *Calculate_Transform*
- *Process_Segments*

Connection to the blackboard and initialisation of a Worker KS is performed by *Initalise_Worker*. Firing of the *Wait_Worker* rule causes the Worker KS to enter a loop, where it waits for the changes in data on the blackboard. Changes that allow a Worker to break free of the loop include appearance of down-sampled images, full-resolution segments, and transform data. Initial changes are caused by the Manager KS. This trigger mechanism ensures that all active Worker KSs commence processing in an ordered fashion.

Fetch_Segments represents a generic rule that is used to retrieve both down-sampled images and full-resolution segments. To ensure each worker obtains a different segment, a Currently-Waiting-to-be-Processed (CWP) identification variable is maintained by the blackboard. Whenever the rule is fired both fixed and moving segments with identification numbers that match the CWP variable are fetched and stored locally. The CWP is then incremented and returned to the blackboard.

The *Calculate_Transform* rule results in a transform being placed on the blackboard. The transform is calculated by the passing of fixed and moving segments to the registration module. In contrast, a transform is fetched from the blackboard and locally stored segments are resampled whenever *Process_Segments* is fired. Currently, segmentation and fusion functionality also form part of this rule and occur when a transformed segment is passed to either of the respective image processing modules. Once processed, segments are returned to the blackboard.

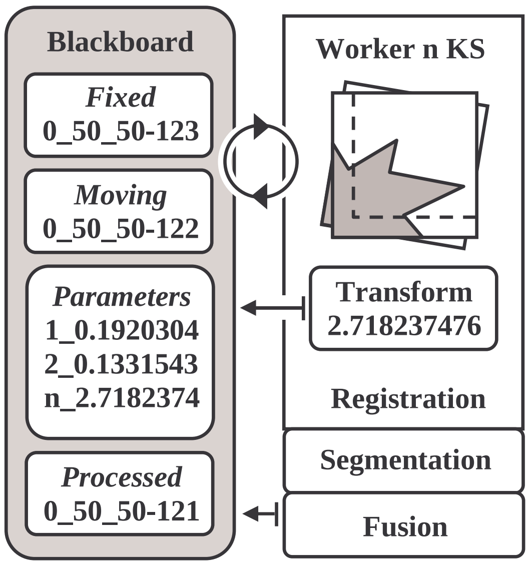

Figure 6: Worker KS

Figure 6 shows the transform calculation between segments by a Worker KS. Processing segments in a first-come first-served fashion allows a fixed number of Worker KSs to process a fixed number of segments. Alternatively a variable number of segments can be processed by a variable number of Worker KSs. This flexibility of processing means that, when an error is encountered, a graceful degradation of the framework can occur. Importantly, memory usage of the blackboard is kept to a minimum through removal of segments once they have been retrieved from *Fixed* and *Moving* partitions.

3.3 The Manager KS

The Manager KS consists of four rule files, including:

- *Initalise_Manager*
- *Wait_Manager*
- *Fetch_Transforms*
- *Accumulate_Image*

The Manager KS is the simplest of all framework components. *Initalise_Manager* causes the Manager KS to be initialised with parameters retrieved from the blackboard. Waiting for changes to appear on the blackboard is caused by the *Wait_Manager* rule firing continuously. Whenever the required number of transforms or processed segments appears on the blackboard the Manager KS breaks free of its perpetual loop. When fired, *Fetch_Transforms* retrieves all transforms from the blackboard. If transforms are missing, no action is taken, otherwise the transforms are averaged and the average transform placed on the blackboard. Finally, processed segments are fetched from the blackboard by the *Accumulate_Image* rule. The segments are added to a list which is locally maintained by the Manager KS.

The total number of segments and border size variables are retrieved from the blackboard as soon as all segments have been gathered. Each segment is then taken from the locally maintained list. Its borders are removed and the segment is inserted into a resulting image. The constructed resulting image is automatically displayed by means of an image viewer. To maintain consistency, the locally maintained segment list is cleared of all data whenever the Manager KS is started and after construction of a resulting image.

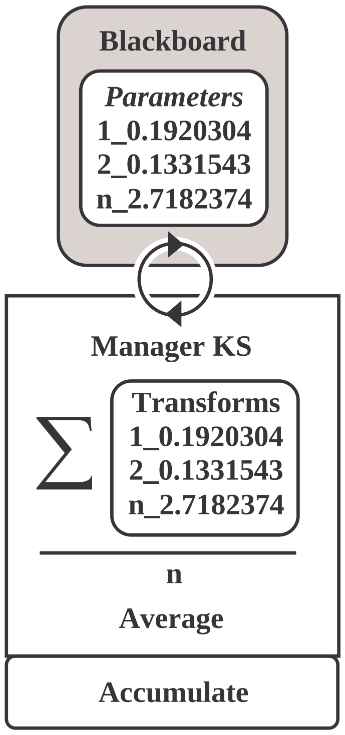

Figure 7: Manager KS

The Manager KS, calculating an average of the transforms stored in the *Parameters* partition is shown in figure 7. Again, graceful degradation of the framework in the event of an error can be achieved. The output from the Manager KS represents a globally transformed moving image, segmented and fused according to implemented Worker KS behaviour. This is in contrast to previous work undertaken with the blackboard (Tait *et al.* 2005) where optimisation was performed independently on each segment resulting in areas of localised registration.

4. EXPERIMENTAL RESULTS

Image subtraction behaviour was added to the Worker KSs in order to evaluate the initial detection performance of the framework and registration precision. PCB images of approximately 1000x500 pixels were chosen as test samples. The fixed image represents a sample with an acceptable and verified quality of manufacture. In contrast, the moving image is a sample containing a variety of defects. These include a spur and open circuit, both of which can be caused by dirt on a blank board or by air bubbles from electrolysis. Once selected, PCB images were divided by the Distributor KS into segments and a 10-pixel wide border was assigned. Subtraction before processing revealed that an unknown translation and rotation existed between fixed and moving images.

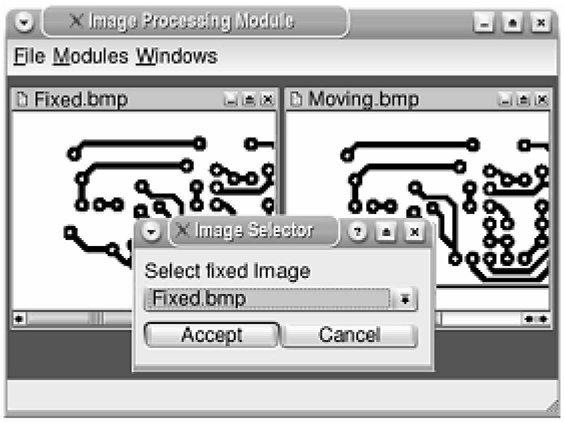

Figure 8: Image viewer

In figure 8 the Distributor KS in image selection mode is shown. The Qt library (Trolltech 2004) was chosen for the creation of the viewer because of its compatibility with the DARBS components and its ease of integration with 3rd party libraries. Implementation of the viewer means image and segment data can be made available in a visual format, during any stage of processing.

4.1 Registration

Spatial alignment into a common co-ordinate system of both fixed and moving segments was performed before a referential comparison could be made. Accuracy of registration is wholly dependant upon the selection of appropriate transform, interpolation, metric, and optimisation components. Registration parameters were selected based on *a priori* knowledge gained through trial and error. The following components produced the best results:

- Affine transformation, which allows for translation, rotation, and scaling of segments.
- Linear interpolation in order to allow pixel intensities to vary continuously.
- Cross correlation metric in order to perform a pixel-wise association between segments.
- A regular step gradient descent optimiser because of its compatibility with the other components.

In order to create a robust algorithm, *a priori* knowledge was also used to select initial parameters for the transform and metric components. To provide a consistent set of parameters the identity transform was set as the initial affine transform. Similarly, pixel intensities, for segments not including their borders, were set as the search space in which the normalised cross correlation metric would operate.

4.2 Segmentation

Once registered, threshold levels were automatically determined for fixed and registered segments. In order to segment the conductor from the insulation material both segments were globally thresholded.

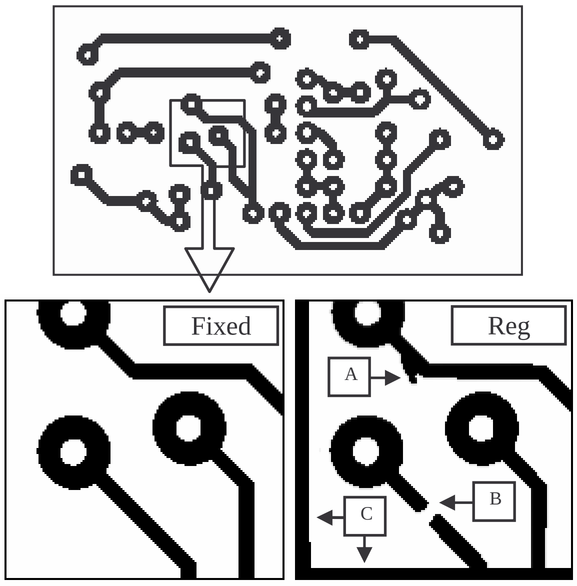

Figure 9: Segmented PCB segments

As shown in figure 9 both spur (A) and open circuit (B) defects have been segmented and are clearly visible in the registered segment. Additionally, translation and rotation caused by the registration process have introduced non-pixel locations which are visible at the bottom and left-hand sides (C). All extraneous pixels at the segment's borders are removed by the Manager KS when it constructs the resulting image.

4.3 Fusion

For the purposes of testing, subtraction was implemented as an exclusive-or (XOR) operation between fixed and registered segments. Regardless of the sub-pixel accuracy achieved by the registration algorithm, differences appeared after subtraction as phantoms.

Figure 10 shows that these phantoms appear along the contours of the conductive material (D,E). As the segments are of a relatively high resolution, phantoms which appear are generally smaller in size than potential defects. In order to achieve better segmentation, morphological opening was applied to the difference segment. Expansion and contraction of the segment caused by the opening operator resulted in the removal of phantoms. Importantly, to conserve small defects, the morphological structuring element consisted of a single pixel. As a consequence the opening operator can be used to eliminate both large and small phantoms through changes in size and shape of the structuring element.

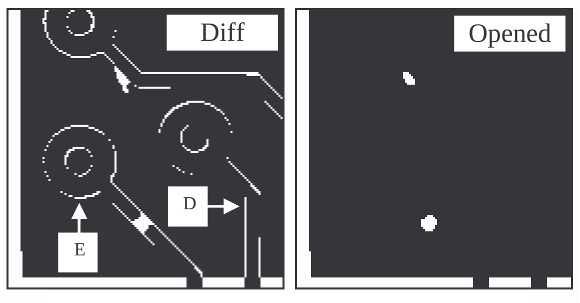

Figure 10: Fused PCB segments

4.4 Performance

To extend upon previous work (Tait *et al.* 2005) and to provide quantitative evidence of the frameworks performance advantages, a sequential algorithm provided by the ITK toolkit was updated with the same components and used as a performance benchmark for comparison. Testing of the framework was carried out in a computer lab with personal computers networked together by an Ethernet 100Mbps switch. All computers in the network contained AMD Athlon 1.67GHz processors with 224 megabytes of random access memory and were running the Debian Sarge Linux operating system. During all tests the number of segments was equal to the number of Worker KSs. The algorithm was applied to four images and the average processing time calculated.

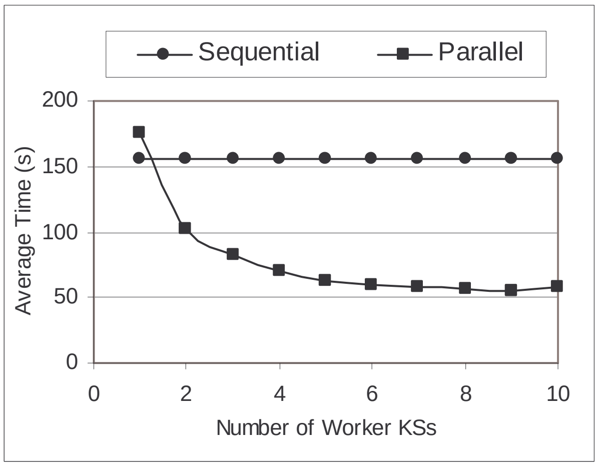

Figure 11: Sequential vs parallel

The sequential execution time and average parallel speedup achieved during registration, segmentation, and fusion of PCB images is shown in figure 11. It can be seen that the distribution of image data approximately reduces execution time from 2.5 mins to 1 min when eight Worker KSs are employed. Thus, the framework realizes large-scale registration with substantial speedups. This maximum efficiency slowly diminishes as the number of processors increases due to communication overheads and a lack of load balancing.

4.5 Discussion

An initial overloading of communications is caused by Worker KSs that try to obtain segments from the blackboard when first triggered. A second overload occurs when Worker KSs have finished processing and try to return segments to the *Processed* partition. This synchronisation occurs when Worker KSs process in a first-come first-served fashion, particularly in situations when the number of segments is high. Creation of a schedule (Ventura and Kim 2003), prior to commencement of the Worker KSs, represents a static load-balancing approach that will be adopted in a future implementation. Similarly, compression of transmission data will cut communication times. The goal of these improvements will be to better distribute communications and reduce the idle time of Worker KSs.

In general, the main cause of failure of the framework to identify single and multiple defects, as well as the introduction of superfluous artefacts into an image, can be attributed to inadequate illumination. Additional testing in suspected problematic environments including surface reflection, background selection, and sample transparency has therefore been performed. As no rules for the creation of ideal conditions exist, in each case trial and error was employed until an acceptable environment was found. With professional photographic and lighting equipment it should be possible to create adequate conditions for industrial application.

5. CONCLUSIONS

An automated visual inspection framework implemented on a distributed blackboard architecture has been introduced. As the choice of resolution determines the smallest size of detectable defect, high resolutions have been used to prevent loss of detail during the processing of a difference image. The difference image is produced by subtracting fixed and moving images after registration into a common co-ordinate system. Preliminary testing of the intensity-based registration algorithm showed that, to obtain reasonable processing speeds, images needed to be small or down-sampled in size. Large high-resolution images were found to take more than a minute to register. In contrast, smaller images required only a few seconds. To address the problems of resolution and speed, the blackboard architecture provides a distributed approach. The blackboard allows an image to be divided into a number of smaller segments whose processing is distributed between multiple agents, known as knowledge sources or KSs. Segments are processed concurrently so that high sensitivity and increased speeds can be maintained. It is intended that accurate detection of smaller defects will be achieved through experimentation with greater resolutions.

REFERENCES

Bayro-Corrochano, E. 1993. “Review of automated visual inspection 1983 to 1993 Part II: Approaches to intelligent systems.” *SPIE: Intelligent Robots and Computer Vision*, 159-172.

Choy, K.W. Hopgood, A.A. Nolle, L. and O'Neill, B.C. 2004. “Implementation of a Tileworld Testbed on a Distributed Blackboard System.” *Proceedings of The 18th European Simulation Multiconference*, 129-135.

Gokturk, S. Akarun, L. and Bozma, I. 1999. “Automated Inspection of Printed Circuit Boards Using a Novel Approach”. *IEEE Workshop on Nonlinear Signal and Image Processing*.

Hopgood, A.A. Phillips, H.J. Picton, P.D. and Raithwaite, N.S. 1997. “Fuzzy Logic in a Blackboard System for Controlling Plasma Deposition Processes”. *Artificial Intelligence in Engineering*, (12), 253-260.

Lee, D.T. 1978. “A Computerized Automatic Inspection System for Complex Printed Thick Film Patterns.” *SPIE: Applications of Electronic Imaging Systems*, 143, 172-177.

Moganti, M. Ercal, F. Dagli, C.H. and Tsunekawa, S. 1996. “Automatic PCB Inspection Algorithms: A Survey.” *Computer Vision and Image Understanding*, (63), 287-313.

Newman, T.S. and Jain, A.K. 1995. “A Survey of Automated Visual Inspection.” *Computer Vision and Image Understanding*. 61(2), 231-262.

Nicolescu, C. and Jonker, P. 2000. “Parallel Low-level Image Processing on a Distributed Memory System.” *In the Proceedings of the 15th Workshop on Parallel and Distributed Processing*, 226-233.

NLM Insight Segmentation and Registration Toolkit, http://www.itk.org, 2004.

Nolle, L. Wong, K.C.P. and Hopgood, A.A. 2001. “DARBS: A Distributed Blackboard System.” *Research and Development in Intelligent Systems*, XVIII, 161-170.

Otsu, N. 1979. “A Threshold Selection Method from Gray-Level Histograms.” *IEEE Transactions on Systems Man and Cybernetics*, 62-66.

Penney, G.P. Weese, J. Little, J.A. Desmedt, P. Hill, D.L.G. and Hawkes, D.J. A. 1998. “Comparison of Similarity Measures for Use in 2D-3D Medical Image Registration.” *IEEE Transactions on Medical Imaging*, 17, 586-595.

Petkovic, T. Krapac, J. Loncaric, S. and Sercer, M. 2002. “Automated visual inspection of plastic products.” *Proceedings of ERK 2002*, 283-286.

Tait, R. Schaefer, G. Hopgood, A.A. and Nolle, L. 2005. “Defect Detection Using a Distributed Blackboard Architecture.” *Proceedings of The 19th European Simulation Multiconference*, 283-287.

Taniguchi, R. Makiyama, Y. Tsuruta, N. Yonemoto, S. and Arita, D. 1997. “Software Platform for Parallel Image Processing and Computer Vision.” *Proceedings of SPIE Parallel and Distributed Methods For Image Processing*, 2-10.

Trolltech Qt, http://www.trolltech.com, 2004.

Wen, Z. and Tao, Y. 1999. “Building a Rule-based Machine Vision System for Defect Inspection on Apple Sorting and Packing Lines.” *Expert Systems with Applications*, (16), 307-313.

Zitova, B. and Flusser, J. 2003. “Image Registration Methods: A Survey”. *Image and Vision Computing*, 21, 977-1000.

Ventura, J.A. and Kim, D. 2003. “Parallel Machine Scheduling with Earliness-tardiness Penalties and Additional Resource Constraints”. *Computers and Operations Research Archive*, 30, 1945-1958.

Roger Tait received an honours degree in Computer Science in 2003. Working as a Software Engineer, he completed a placement year with the Forschungs-zentrum Karlsruhe - Institut für Angewandte Informatik (IAI). Currently he is a PhD student at Nottingham Trent University where he is carrying out research into image processing and artificial intelligence for use in non-destructive evaluation.

Gerald Schaefer gained his PhD in Computer Vision from the University of East Anglia. He worked at the Colour & Imaging Institute, University of Derby as a Research Associate and as Senior Research Fellow at the School of Information Systems, University of East Anglia before joining the School of Computing and Informaticcs at Nottingham Trent University as a Senior Lecturer in 2001. His research interests include colour image analysis, physics-based vision, image retrieval, and image coding.

Adrian Hopgood is professor of computing and head of the School of Computing and Informatics at Nottingham Trent University, UK. He is also a visiting professor at the Open University. His main research interests are in intelligent systems and their practical applications. He graduated with a BSc (Hons) in physics from the University of Bristol in 1981 and obtained a PhD from the University of Oxford in 1984. He is a Fellow of the British Computer Society and a committee member for its specialist group on artificial intelligence.

Lars Nolle graduated from the University of Applied Science and Arts in Hanover in 1995 with a degree in Computer Science and Electronics. After receiving his PhD in Applied Computational Intelligence from the Open University, he worked as a System Engineer for EDS. He returned to The Open University as a Research Fellow in 2000. He joined Nottingham Trent University as a Senior Lecturer in Computing in February 2002.