# Design and Implementation of a Parallel Registration Algorithm for SAR Images

Fadiran Oladipo O.

A Project report submitted to the Department of Electrical Engineering, University of Cape Town, in partial fulfilment of the requirements for the degree of Master of Science in Engineering.

Cape Town, 10 October 2001

# Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the degree of Master of Science in Engineering in the University of Cape Town. It has not been submitted before for any degree or examination in any other university.

Signature of Author . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

Cape Town
10 October 2001

## Abstract

Registration of two or more images of the same scene is an important step in image processing that seeks to extract information not obtainable from one of the images in question. This process is required in many Engineering, Scientific and Medical applications. The accuracy of this step is crucial to the reliability of subsequent image processing and or decisions made on its basis. The huge size of the data to be processed, the speed at which the processing is required and the accuracy requirements necessitates a quick, efficient, robust and in some respects automatic program which efficiently harnesses available computing resources. This is the object of this project - the design of an image registration algorithm with a bias for SAR/InSAR applications but also applicable for other registration purposes, implemented on a parallel cluster of computing nodes.

# Dedication

Dedicated to the memory of my Late Father David Oladokun Fadiran

1939 - 1998

He would really have loved to see this.

# Acknowledgements

I wish to aknowledge my supervisor and group leader of the group within which I worked Prof M.R. Inggs, he has been of immese help, giving required and timely guidiance. He also facilitated the funding for my Masters program. Dr. Andrew Wilkinson was also of great help in clarifying theoretical concepts entailed in the project.

I appreciate De-Beers Consolidated Mines Limited for the partial funding of the Masters program. Anthon Voigt, the industry link person showed such interest in the project and my progress.

Mark Gebhardt and Thomas Bennet are members of the MultiComputer Technology Initiative MTI sub-group and they were of great assistance in my getting started on the project, Thomas is a particularly nice chap. Dr. Richard Lord was also a "handy reference" for Radar processing issues.

I appreciate the contributions of my family, financial and otherwise, I am really grateful to you all.

# Contents

# List of Figures

# List of Tables

# Nomenclature

**Azimuth**—Angle in a horizontal plane, relative to a fixed reference, usually north or the longitudinal reference axis of the aircraft or satellite.

**Range**—The radial distance from a radar to a target.

**Synthetic Aperture Radar (SAR)**—A signal-processing technique for improving the azimuth resolution beyond the beamwidth of the physical antenna actually used in the radar system. This is done by synthesizing the equivalent of a very long sidelooking array antenna.

**ERS**—European Remote Sensing Satellites.

**Tandem Images**—Images acquired 1 day apart.

# Problem statement and Scope

- **Survey/Research image registration algorithms, identify techniques specifically suited for SAR/InSAR purposes but also possibly useful for other purposes.**

- **Design parallel image registration algorithm based on researched techniques above. Implement algorithm on multiple computers in a clustered architecture.**

- **Test program with images and discuss the suitability of the algorithm as regards the accuracy of the obtained registration and the suitability of the parallelisation scheme for the registration problem. Both of these based on objective bench-marks.**

# Chapter 1

# Introduction

It is often necessary to use data from more than one image of the same scene to obtain required information about that scene. An example is the use of difference images in medicine to identify anatomical differences over time or from two or more viewing angles [5]. The formation of interferograms and height maps from stereo SAR information from images of the same scene taken at the same time with antennas mounted on the same platform or at different times during different "passes" also requires two images of the same scene [44][4]. A similar application is the detection of relatively minute changes in ground heights using Differential Interferometry requiring three(3) or more images. These are some examples of the processes which require more than one image to obtain needed information. For a valid comparison of any sort or further processing in all these cases mentioned and more, the images have to be properly "aligned"/"matched". These words aligned/matched as used here are vague, the process for forming interferograms and height maps from two(2) SAR images is briefly described in Section 1.1 to clarify this point.

Although various attempts have been made at reducing the computational complexities of the various algorithms employed in registering images, they are still generally very computationally intensive. The design of parallel algorithms which implement the multiple tasks involved in the registration on multiple computers is an approach to solving this problem. Parallelization however does not ensure a speed-up in the computational time and the efficiency of the algorithm as executed on multiple computing nodes must be justified in the light of the added computing resources. Such envisaged improvement has to be shown theoretically and or empirically by performing a series of tests using the algorithms. The work to be carried out in this project is as is stated in the problem statement and scope.

## 1.1 Background of Image Registration (Using SAR/InSAR)

SAR images are acquired in one of many possible modes, in all cases, the antennas used in acquiring the images are separated by a base-line $b$ as shown in Figure 1.1. More details on InSAR can be obtained from [44][4][38][37]. The brief description here only seeks to highlight the problem of image registration. The points $p$ and $p'$ as represented in Figure 1.1 are points represented on the ground(scene). For antennas transmitting at a carrier wavelength $\lambda$, the phase delays resulting from the two-way propagation delay from each of the antennas located at $(y_1, z_1)$ and $(y_2, z_2)$ is given by Equation 1.1 and the absolute interferometric phase $\psi$ is given by Equation 1.2 .

$$\psi_1 = -\frac{4\pi r_1}{\lambda}; \psi_2 = -\frac{4\pi r_2}{\lambda} \tag{1.1}$$

$$\psi = \psi_2 - \psi_1 = \frac{4\pi(r_2 - r_1)}{\lambda} \tag{1.2}$$

Also, assuming a flat-earth geometry for simplicity, the point $p$ (y,z) as in Figure 1.1a can be located within the plane by the relations :

$$y = y_1 + r_1 sin(B + C) \tag{1.3}$$

$$z = z_1 - r_1 cos(B + C) \tag{1.4}$$

where

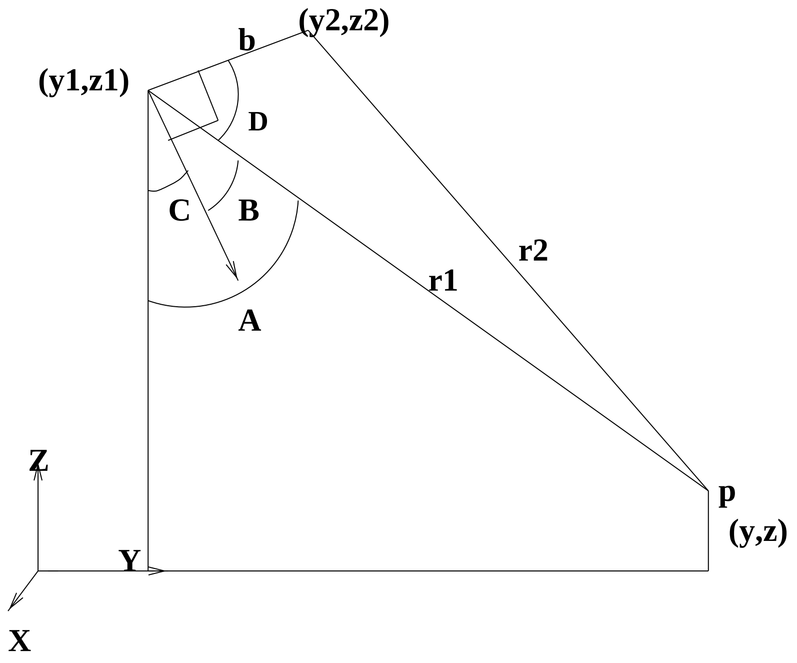
$(y_1, z_1)$ and $(y_2, z_2)$ are the coordinates of the locations of the antennas

$p(y, z)$ and $p'(y', z')$ are points on ground in the imaged scene

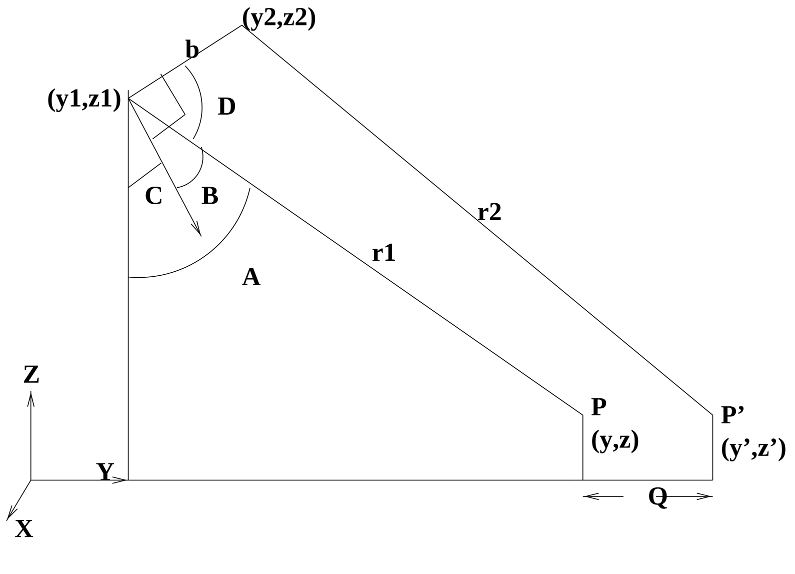
$r_1$ and $r_2$ are measured ranges from antennas 1 and 2 respectively to points on ground

$C$ is the base-line elevation angle

$B$ is the direction of arrival of the received echo from point $p$

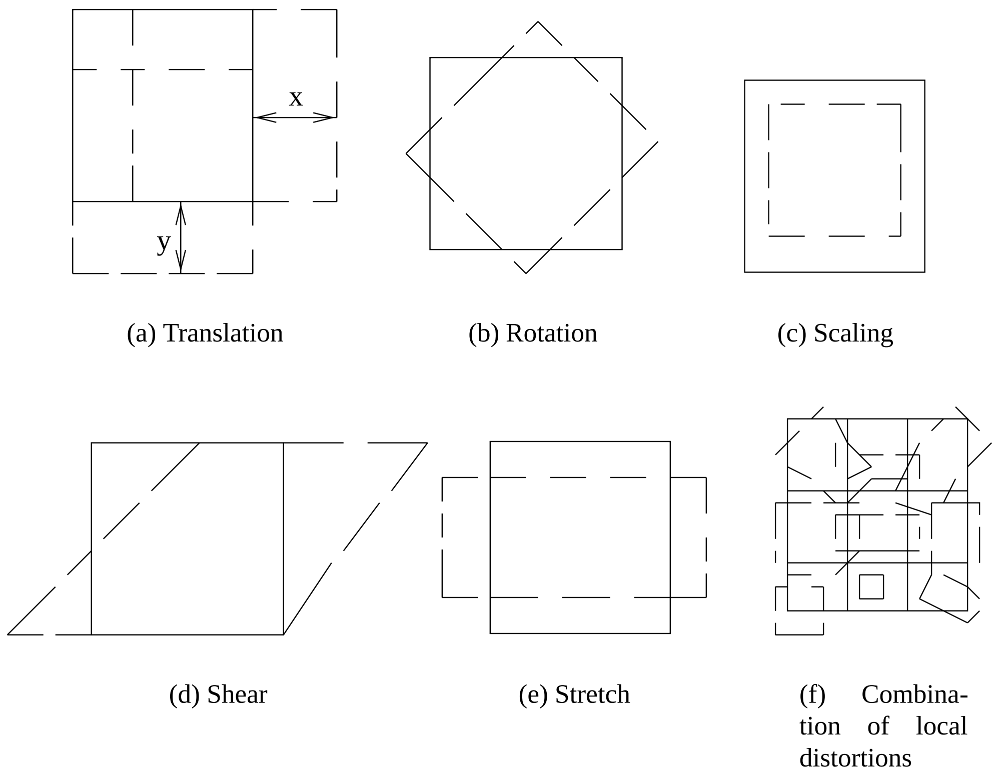


(a) Accurate registration of images



(b) Mis-registration of images

Figure 1.1: Registration and Mis-registration of Images acquired by Antennas with position $(y_1, z_1)$ and $(y_2, z_2)$



(a) Translation (b) Rotation (c) Scaling

(d) Shear (e) Stretch (f) Combination of local distortions

Figure 1.2: Possible distortions in images

These relations (Equations 1.1,1.2,1.3) will however not be valid for the geometry as shown in Figure 1.1b because of the spatial misregistration of the points being considered by the distance $Q$ (this time in the $y$-direction, note that the distortion could be in other directions or of a different type). Calculations made on the basis of misregistered images like Figure 1.1b will result in errors. Possible differences in the referenced points may be caused by changes in the mode and or device for the image acquisition, positioning of the device or some other deviation from the planned course of the platform on which the sensor is mounted. This can result generally in ***translation*** in the $x$- and $y$-axis, ***rotation***, ***scaling***, ***stretching*** and ***shearing*** in one image with respect to the other. For example, a deviation from the planned altitude of the platforms on which sensors (antennas) are mounted for SAR image acquisition may result in an error in scale in the images obtained. Image distortions can be generically classified [6] as **RADIOMETRIC** - due to the effect of the atmosphere on radiation and thus on remote sensing imagery and also instrumentation errors, and **GEOMETRIC** errors which are as a result of variation in platform altitude, velocity and attitude, etc. Some of these possible distortions are shown in Figure 1.2.

The distortions shown in Figures 1.1a and 1.1b (translations and rotations) are caused by orbit changes of the sensor (e.g. antenna) from the desired orientation while acquiring the image. The distortion shown in Figure 1.1c (scaling) is as a result of change of altitude of the sensor/platform on which it is borne, and those in Figures 1.1d and 1.1e (stretching and shearing) are as a result of distortions in the sensor itself.

These distortions may also be classified as global - continuous over the image as in Figures 1.1a - 1.1e or local as represented in Figure 1.1f.

The registration algorithm thus seeks to transform one of the images such that points on it (as represented by pixel positions/locations in digital images) align with or are represented by the same pixel locations on the other image. The image left unchanged is called the Reference/Master image and the image to be transformed is called the input/slave image, a third image is then produced from the transformation of the input image.
By "comparing" the input and the reference images, a relationship (a function) can be obtained which *warps*/*deforms*/*transforms* the input image into an image that is geometrically similar (in dimensions and pixel positioning) to the reference image. Thus, for a 2D reference image with pixel coordinates $(x, y)$ , there is a relationship which defines the location of the same pixel on the input image $(x', y')$ such that

$$x' = f(x, y) \tag{1.5}$$

$$y' = f(x, y) \tag{1.6}$$

Obtaining this function $f$ which may model distortions as simple as a global translation or as complex as a local combination of all the distortions earlier discussed is one of the main objects of image registration. All further operations in the process of obtaining further information from the images as described in Chapter 1 are then performed using the reference and the transformed input image (warped image).

The resolution of the images to be processed makes it necessary to do some oversampling or interpolation of the given data, effectively resulting in larger data sets. For SAR images for example, the resolution is in the order of meters(eg........ $1\ resolution\, cell\ = 10m)$ , accurate registration of this image at the pixel level will be inadequate as the points referenced for further processing may still be meters apart, resulting in appreciable post-processing errors. Sub-pixel image registration accuracy is thus a requirement for many purposes, this leads to a huge increase in data to be processed.

The processes described above are computationally intensive and thus require the most efficient use of computing resources. This can be achieved by the design and efficient implementation of registration algorithms. This process can also be sped up by harnessing the ***potential*** gain of using multiple computers for the computations.

## 1.2 Distributed and Parallel Computing

Rapid development in many areas of Medicine, Engineering and Science have resulted in the need to process more data and at a faster rate. Such computing was typically done on main-frame supercomputers as the available resources on PCs were inadequate. The relatively huge cost of supercomputers have put them out of the reach of many organizations which need to do the kind of processing described above. Another disadvantage of supercomputers is that they are usually very customized in hardware and sometimes in the software installed on them thus making maintenance difficult and expensive and upgrading almost impossible. Using multiple PCs with relatively low cost, off-the-shelf components organized as a loosely distributed system or more closely in parallel/clustered systems have been attempted as a way of obtaining the computing resources of the supercomputer and over-coming its cons. Solving the problem of maintenance and ease of upgrade have been achieved. The price/performance ratio for distributed/parallel computers have also surpassed that of supercomputers in many instances [41][27].

### 1.2.1 Classification of Parallel/Distributed computers

A useful criteria for classifying parallel/distributed computers is that of the ***Data*** set they process and the ***Instructions*** executed by the computing nodes that make up the system [9].

1. Single Instruction Single Data set/stream (SISD): this is basically the Von Neumann computer architecture, it consists of a single instruction being executed on a single set/stream of data on a processing node.

2. Single Instruction Multiple Data sets/streams (SIMD): All processing nodes execute the same instruction on different data sets/streams.

3. Multiple Instruction Multiple Data sets/streams (MIMD): All processing nodes execute different instructions on different data sets/streams. This group can be further sub-divided into MIMD-Shared Memory (MIMD-SD) and MIMD-Distributed Memory(MIMD-DM). In MIMD-SM, the processing nodes all have access to a central memory through a local bus system. In MIMD-DM, each processing node has its own local memory and communication with other processing nodes is through an external interconnection network.

4. Multiple Instruction Single Data set/stream (MISD): Different instructions on each processing node but operating separately on the same data set/stream.

Another classification criterion divides Parallel/Distributed computers into either:

1. SPACE SHARING : Tasks are distributed amongst computing nodes, each node processes a separate task/sub-task at a given time.

2. TIME SHARING : The different multiple tasks are given access to the computing nodes and they compete for the processor cycles/time by some designed time-sharing formula.

### 1.2.2 Parallelization/Distribution schemes

Parallelization/Distribution of tasks is done in one of or a combination of the ways stated [9][49]:

1. Pipe-Lining: Processes that have many sequential module tasks may be pipe-lined such that a processing node waits for the preceding processing node to perform its own task and then passes the data resulting from the processing at that stage to the subsequent stage. This arrangement is obviously only suited for processes with modular sub-tasks and in which the data to be worked on is “streaming”, that is, continuous input data and expected continuous output processed data. Real-time feature identification in input images may involve (i) filtering, (ii) rebining, (iii) thresholding and the (iv) identification of features. Each of these stages of the processes may be pipe-lined; the first processing node performing the filtering and the last processing node performing the feature identification. The overall throughput of the system is increased but the latency between input and output is not improved by this arrangement. The slowest link in the pipe-line determines the

performance of the system. The other draw-back of this arrangement is that it is non-scalable; a process designed to work on 5 processing nodes can not be easily scaled to run on 3 or 7 nodes.

2. Algorithm Parallelism: Sub-tasks of a process which can run concurrently can be distributed on a number of computing nodes. For example, operations on a 2D image to be performed in both a horizontal and vertical directions may be done on the same image concurrently. The result of each operation is then combined in some logical way to complete the process. This arrangement also suffers from the disadvantage of difficulty of scaling.

3. Data Parallelism: This fits the SIMD architecture . The data set to be processed is divided amongst the processing nodes which all execute the same instruction set. Such data distribution can be done *Statically* or *Dynamically.* Segments of digital images may be allocated to processing nodes for full processing then the whole image re-assembled to form the whole processed image. Considerations have to be given for boundary information between the adjacent image segments when re-assembling.

However, some other problems which are associated with distributed/parallel computing include:

1. **Load Balancing**: The efficient distribution of tasks and the allocation of computing resources . This ensures that the processing nodes have the least possible idle clock cycles.

2. **Scalability** of algorithms to be implemented: Design which ensures that increase in computing resources for the same task results in commensurate timing/throughput gains.

3. Obtaining adequate capacity on **Inter-connecting Network** between computing nodes. The communicating channel between processing nodes should not be a bottle-neck.

4. **Granularity**: Minimization of time taken for communication between nodes as compared to time for actual computation. Actual processing does not take place during communication between the nodes, this results in idle processing cycles also resulting in lower efficiency of the system. The object is to reduce the communication/computation ratio or vice-versa.[See Figure 1.3]

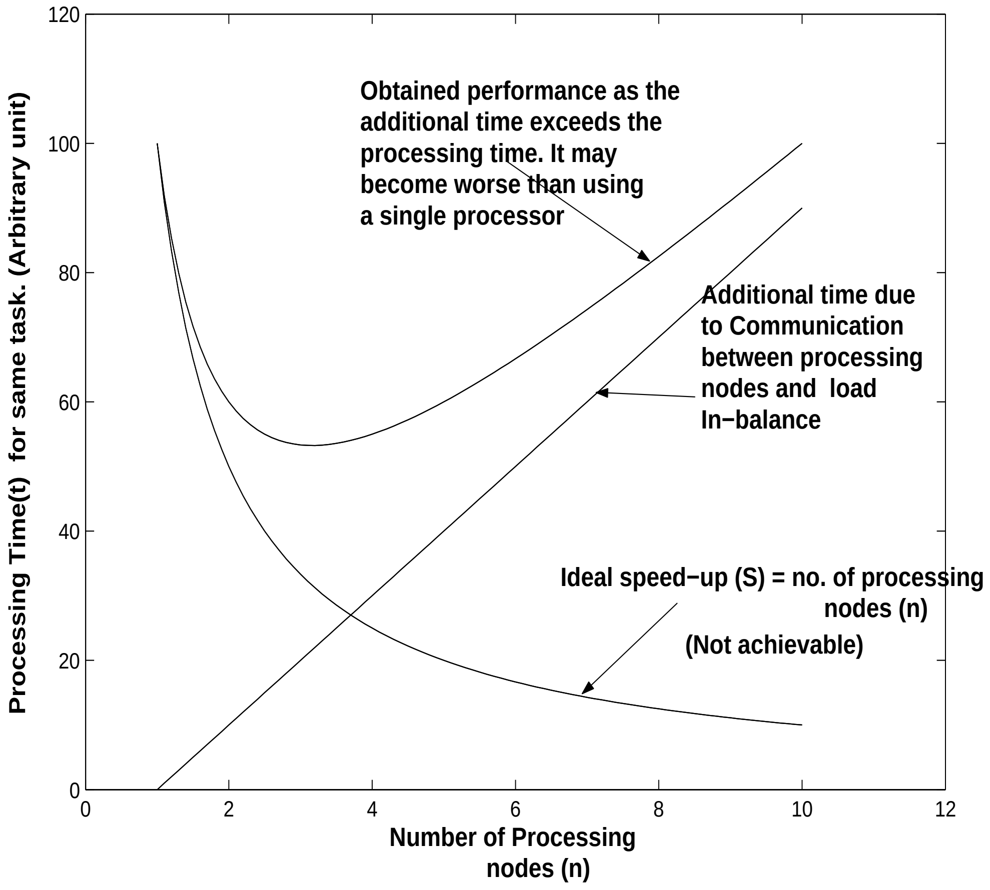


Figure 1.3: Plot showing performance of Parallel system with linear time increase (assumed) due to communication between nodes and load imbalance

The specific scheme or combination of schemes to distribute/parallelize the tasks to be done depend on the nature of the problem to be solved; data type - streaming data (continuous input and output) or a static source (e.g. image on disk), the type of operation - low level (e.g. processing at the pixel level), medium level (e.g. edge detection), high level (e.g. object recognition), and also depends on the existing hardware architecture. All these are taken into consideration with a view to harness the potential gains of parallelization, yet reduce to a minimum the basic problems of parallelization as stated in Section 1.2.2. The scheme for this thesis work is stated and the components are highlighted and justified in the program design process in Chapter 3 of this project report.

## 1.3 Thesis Organization

The rest of the thesis is arranged thus:

- Chapter 2 is a literature review and survey of image registration algorithms and related research. Algorithms that have been designed and implemented in the work/research done by others and the results obtained are discussed as they relate to this specific work.
- Chapter 3 describes the choice of the various components/steps to be implemented in this particular work, each component is justified on a theoretical basis and or practical experience as shown under discussions on obtained results from implemented algorithms in Chapter 2 and also in keeping within the scope of work to be done as stated in the Problem statement and scope of work. Theoretical and practical parallel computing considerations are stated and made especially as regards the specific algorithm/program design. Considerations made for implementation in C using PVM message passing libraries are also stated. A flow chart/pseudo-code for the proposed algorithm to be implemented is then shown. A brief discussion on the general and peculiar problems encountered in the actual coding in C and PVM are stated here if they exist.
- Actual tests of the program using real image data are carried out in Chapter 4. Records of necessary data are displayed and discussed under the headings:
  - Accuracy of the registration: the test of how accurately the test SAR images are registered will be determined by the quality of interferograms (amount of interferometric noise) formed by the registered images. The registered images

  will thus be used as input to an existing routine which forms interferograms. The registration accuracy will be assessed qualitatively and quantitatively. The bench mark will be images registered using a commercial software.

  – Speed-up, Scalability and Overall Efficiency of the Parallel System: Execution time of the implemented code will be compared to that of an existing serial implementation of a similar algorithm. Speed-ups will also be measured as the number of processing nodes performing the registration tasks are increased (Scaling). These will be linked to the overall efficiency of the system. Theoretical optimum performance will form the basis for measuring the parallel systems performance. Results are discussed and possible reasons for marked differences from the optimum are stated if they exist.

- Chapter 5 concludes the dissertation by summarizing the object of the work and how well the defined problem has been investigated/solved. The suitability of the algorithm for parallelization is discussed in the light of results from Chapter 4. Further work that can be done in this area, especially as regards improving the method for parallelizing the algorithm or amending the registration algorithm itself for the purpose of better overall efficiency are mentioned.

# Chapter 2

# Literature review and Survey of Image registration algorithms

A survey of image registration algorithms reveals that they are usually very problem-specific and are consequently numerous, each suited for specific purposes. This makes their classification difficult. Brown[5] identified the basis which cut across all image registration algorithms and attempted to classify them using this criteria. All image registration algorithm design involves the following.

## 2.1 Basic Image Registration considerations

- Decision on a **FEATURE SPACE** which extracts the information in the images to be used for the "matching". This could be magnitude/intensity as in [31][6], edges/segments/regions as in [15][21] or some other appropriate feature of the images to be registered.
- A **SEARCH SPACE** which defines the possible transformations to the input/slave image which will result in the best match to the reference image. This could be translations, scalings, rotations etc. or some complex combination of any of the distortion already mentioned.
- The **SEARCH STRATEGY** defines how the search space will be navigated. The search may be selective, exhaustive, hierarchal or some combination of these.

- **SIMILARITY METRIC** determines the merit or quality of match of the two images as registered after transformation of the input image. The two images are hardly ever perfectly registered, but a similarity metric such as an absolute difference of the two images after transformation may indicate how well the images have been registered. Note that taking the absolute difference between two registered images is not always a good measure of the accuracy of registration, it is mentioned here as one of the many options.

## 2.2 Reviewed materials

All image registration algorithms are combinations of instances of the four components listed above. An algorithm design involves identifying a suitable feature space, deciding on the necessary search space, navigating such a search space in the most efficient manner and determining which of the transformations produces the best match using the appropriate similarity metric. The choice of each component to be used is primarily dependent on the kind of images to be registered. A brief description of some registration algorithms highlighting these components follows.

Fornano et al. [14] use intensity information in SAR images as the feature space. With prior knowledge of SAR imaging, 2D translations and minimal scaling were chosen as the search space. Fourier methods were applied for the search strategy and the similarity metric was a measure of how well the spectrum of the two images aligned. The processing in this algorithm is on the raw SAR data which is usually very huge (order of gigabytes). The method is well suited for space-borne SAR images which suffer from the kinds of distortions accounted for in the scheme, the processing however takes appreciable time.

Registering SAR images by modeling the sources of distortions in SAR imaging and incorporating this in the SAR image processing scheme was attempted by Fernandes et al. [12]. Estimated translation, rotations and scaling based on the knowledge of SAR image acquisition scheme are included in the processing to compensate for the distortions. The estimations/modeling are analogous to the search space and search strategy, the raw image pixels serve as the feature space. A comparison of the spectrum of the registered images serves as the similarity metric. The accuracy of the model for the distortions is crucial to the accuracy of the registration scheme. The models may not always be accurate or empirical, tracing the sources of errors in registration could thus be difficult.

Baggs et al. [2] attempt to reduce the computation time and yet retain the accuracy in registering images by using what they called a “Length code algorithm”. The length code of objects in images is defined as the distance between the centroid of the object and the object boundary, this is used as the feature space. Translational and rotational features of the length code are used for the search space, the search strategy is an exhaustive search over all the possible translations and rotations for all the objects identified in the images. Geometrically measured translation and rotational errors between input and reference images are used as the similarity metric. Minimum matching errors with huge computational gains were recorded for simulated image data. The validity of this for any real image registration problem was not shown. Non-existence of objects in parts of the images may also result in errors.

Obtaining “Tie points” (accurately identified common points on both images to be registered) which is a step prior to obtaining a warping function which defines the relationship between input and reference images is very difficult and even thought impossible by some [15, Pg.1]. Goshtaby et al. [15] attempt to register images by segmenting the images in question and finding the centre of gravity of similar closed segments in both images. The established centre of gravity points in both images serve as the feature space, the number of coefficients defined in the warping function/polynomial serve as the search space as they represent translation, rotation, scaling and other distortions which need to be accounted for to register the input/slave image to the reference image. Least squares regression method is used to calculate the polynomial coefficient and this serves as the search strategy. The absolute difference of both registered images was used as a similarity metric and also a test of the accuracy of the algorithm. Growe et al [17] also worked in the same direction by attempting to solve the problem of accurately identifying “Tie-points” as a feature space by using prior knowledge from GIS data to identify features in images such as road intersections. Such points are identified through the use of defined semantics which are to be satisfied. Such accurately identified points then serve as feature space and registering the images follows a similar process as in the work done in [15] . In both algorithms, the distortions that lead to misregistration are assumed to be global and continuous over the whole image, this is usually not true for most images especially remotely sensed images.

Le Moigne [31][32] introduces the dimension of parallel processing to image registration, parallelization itself is basically not an image registration issue but it creates new possibilities in image registration algorithms. Raw pixel values are used as the feature space but a reduction in the data set is achieved by resolution reduction using wavelet decomposition. The search space was a combination of rigid translations and rotations. The search strategy is hierarchal - a large search space is navigated at lowest resolution and in larger "steps", as the resolution of the same image is increased, the search space is navigated more accurately ("finer" steps through) but is centered around the solution found in the prior resolution level, this is repeated until the full image is aligned as accurately as defined at full resolution. The algorithm was tested by applying known translations and rotations to Thematic Mapper (TM) and Advanced Very High Resolution Radiometry (AVHRR) images and the algorithm attempts to register these intentionally mis-aligned images with the original ones. Accuracy results and timing gains resulting from parallelization are recorded.

Chalermwat [7][6] extends on the algorithm implemented by Le Moigne [31] by reducing the search space. Genetic algorithms are applied such that the search even within the defined search space is not exhaustive yet the ability to find the best match between both images is not compromised. The algorithm is tested on more data in addition to those in [31], including SAR images and images from Geostationary Operational Environmental Satellite - (GEOS-8). Both algorithms have the great advantage of being able to register multi-resolution images using wavelet decomposition, huge timing gains are also recorded because of parallelization. Their tests are however on images in which known translations and rotations are introduced, no tests were carried out in registering non-simulated unregistered images. The algorithms are also limited to rigid, global and continuous distorted images, this is usually not the case with remotely sensed images. The algorithm is inappropriate for SAR images in particular.

The algorithms surveyed have some good features but none seems usable for SAR images with typical image sizes in the order of a gigabyte and a complex combination of local distortions. Some components of the algorithms are however extracted and modified for use in the algorithm design of this dissertation.

# Chapter 3

# Algorithm Theory,Design and Implementation

## 3.1 Introduction

This chapter designs a parallel algorithm for the registration of SAR images. The four basic considerations for any registration scheme earlier discussed are addressed, these are made with specific relation to the nature of typical SAR images, the required accuracy for SAR image post-processing and the most efficient parallelization schemes.

## 3.2 Centre Offset Module

### 3.2.1 Theory

The backscatter (amplitude/magnitude) from remotely sensed images may differ markedly for the same image scene, this is usually due to Radiometric errors [6]. This is particularly true for SAR images. The magnitude of the backscatter of the images to be registered is not a good choice as a "feature space" as information in the images which is invariant to illumination is required. Phase correlation which utilizes the phase information in the images backscatter is used. This is achieved by using one of the properties of the Fourier transform. The translational property of the Fourier transform (Shift Theorem) states that, given two $1 - dimensional$ signals $f_1$ and $f_2$ which are functions of an independent variable $x$, and differ only by a displacement $dx$, i.e.

$$f_2(x) = f_1(x - dx) \tag{3.1}$$

then the Fourier transforms $F_1$ and $F_2$ of the signals are related by

$$F_2(\omega_x) = e^{-j(\omega_x d_x)} F_1(\omega_x) \tag{3.2}$$

where $\mid F_2(\omega_x) \mid = \mid F_1(\omega_x) \mid$.

The two displaced signals thus have the same Fourier transform magnitude but differ in phase. This phase difference can be shown to be directly related to the displacement $dx.$ Computing the normalized cross-power spectrum of the two signals and then taking the inverse Fourier transform of this, we obtain a signal which is approximately zero everywhere else except at the point at which the signals are spatially displaced.

$$\frac{F_1(\omega_x) F_2^*(\omega_x)}{\mid F_1(\omega_x) F_2^*(\omega_x) \mid} = e^{(\omega_x dx)} \tag{3.3}$$

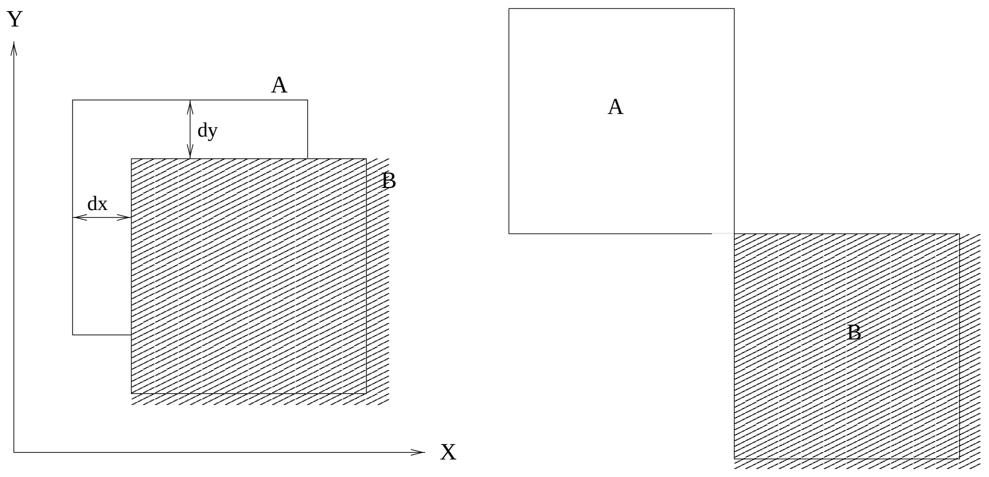
$$F^{-1}\{e^{(\omega_x dx)}\} = dx \tag{3.4}$$

The $1 - dimensional$ displacement $dx$ is thus obtained. This is easily extended to $2 - dimensions$ for images as shown in Equation 3.5.

$$F^{-1}\{e^{(\omega_x dx, \omega_y, dy)}\} = (dx, dy) \tag{3.5}$$

where $F_2^*(\omega_x)$ is the complex conjugate of $F_2(\omega_x).$

Rigid $2 - dimensional$ translation distortion in one image with respect to the other can be obtained using this method. This method for finding the translation in one image with respect to the other is immune to noise limited to narrow bandwidths, it is relatively scene independent and its accuracy is only affected by significant white noise [5, Pg. 346]. Thus, given two images A and B of the same scene, rigidly mis-aligned as shown in Figure 3.1a , the phase correlation method as described above will indicate the displacements $dx$ and $dy$



(a) Centre offset calculation possible

(b) No common points, centre offset calculation not possible.

Figure 3.1: Rigid Translation in one image with respect to the other

.

Note that this has to be done with the prior knowledge that the images at least have some common points, an application of the method to images as shown in Figure 3.1b will not be valid as the two signals are completely uncorrelated, this will be indicated by the normalized Cross-correlation $(CC)$ value $e^{(\omega_x dx, \omega_y dy)}$ which ranges $0 \leq CC \leq 1$. This is a figure of merit for the result of the phase correlation method for specific images(signals) and would be approximately zero for uncorrelated images.

Also, as rotation and other distortions earlier mentioned are not accounted for, the offset calculated for the images will be crude, this only serves as a first step to register the images, other distortions will be accounted for in the algorithm. For space-borne SAR images, (eg. ERS images), rotation in images of the scene acquired by antennas on the same platform is usually less than $5^o$ [45], this is minimal, and the result of the initial offset calculated using the phase correlation method will be a valid crude estimation.

### 3.2.2 Implementation

Typical SAR images are of size 1GB $(2048_{(range-bins)} \times 26,000_{(azimuth-lines)})$ samples, each represented by $8 bytes (complex\ float)$ , this makes performing the phase correlation as described for the whole $2-dimensional$ image impracticable for many comput-

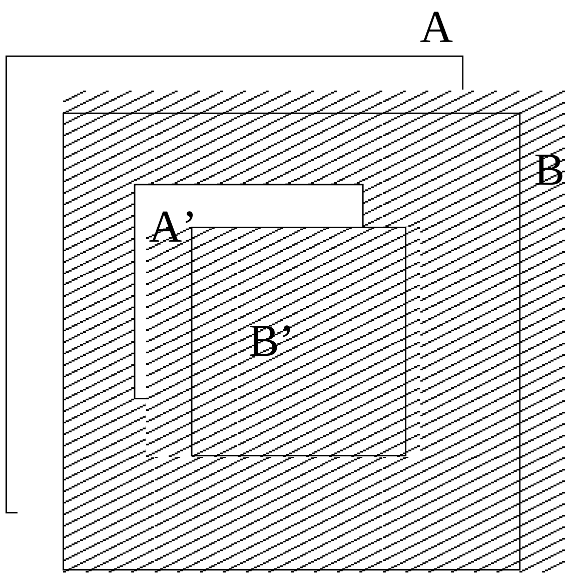


Figure 3.2: Sub-image A' and B' extracted from the centre of images to be registered.

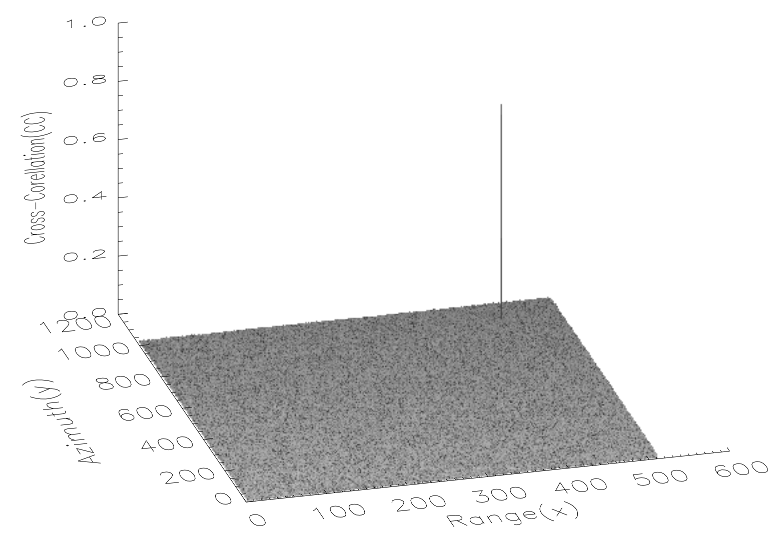
ing nodes and for most purposes, this is unnecessary. Again, a prior knowledge of the estimated displacement of the images in question in the $2$ dimensions is needed. This determines the size of the sub-images to be extracted from the centre of the images as shown in Figure 3.2 for the crude calculation of the displacement (offset) of one image with respect to the other in both dimensions.

The magnitude correlation method should then give a crude indication of how the whole images are translated with respect to each other. Typical worst case scenario translations for ERS SAR images are $1000$ pixels in the azimuth and $50$ pixels in the range [45], typical centre size extracted for the offset calculation is $512_{(range-samples)} * 1024_{(azimuth-lines)}$ samples, these sizes could be changed depending on prior knowledge of the translation between images to be registered.

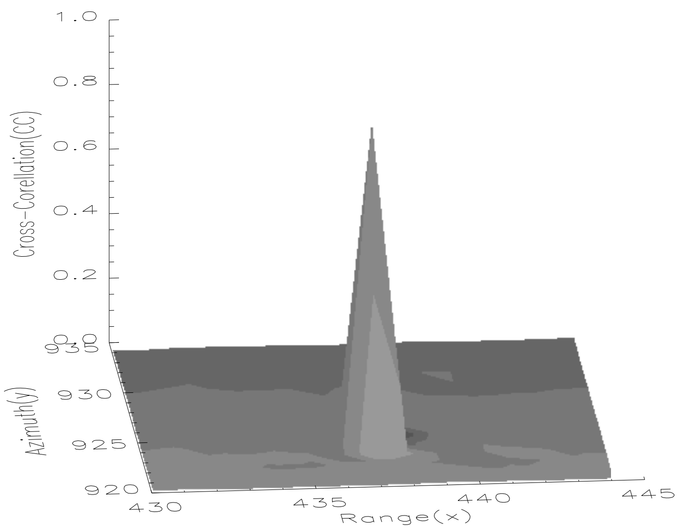
Details of the implementation are:

- Extract centre patches from both (Master and Slave) images.
- Obtain translation(offsets) in both dimensions as described. Store offsets in both axes for further use.

**Note** The plots as shown in Figures 3.3 and 3.4 are "folded over", that is, these are mirror images of the original. Thus, the translation is $(x - dx, y - dy)$, the obtained

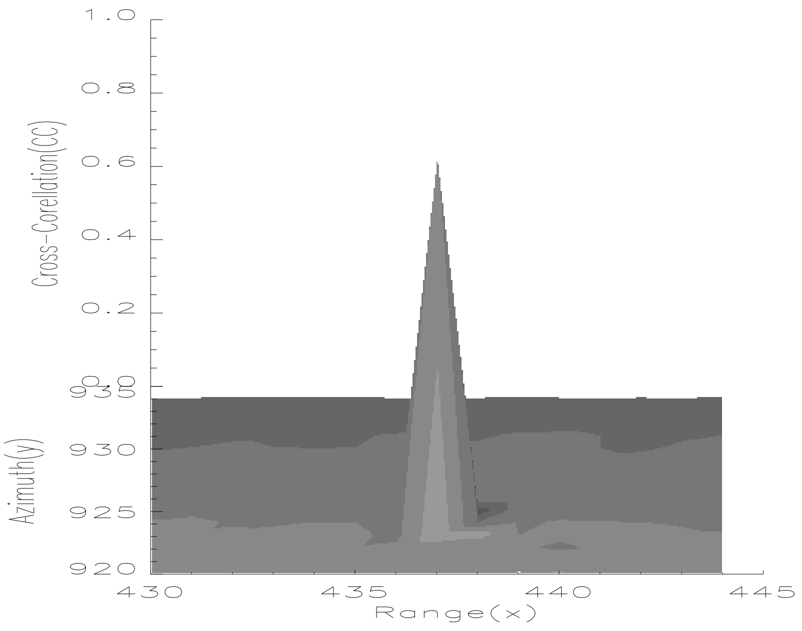


(a) Plot of all 1024*512 samples showing single "spike" indicating offsets in the 2-dimensions



(b) Zoomed-in version of offset plot

Figure 3.3: Plots of Offsets obtained from magnitude correlation of 2 images of size $1024 * 512$ samples



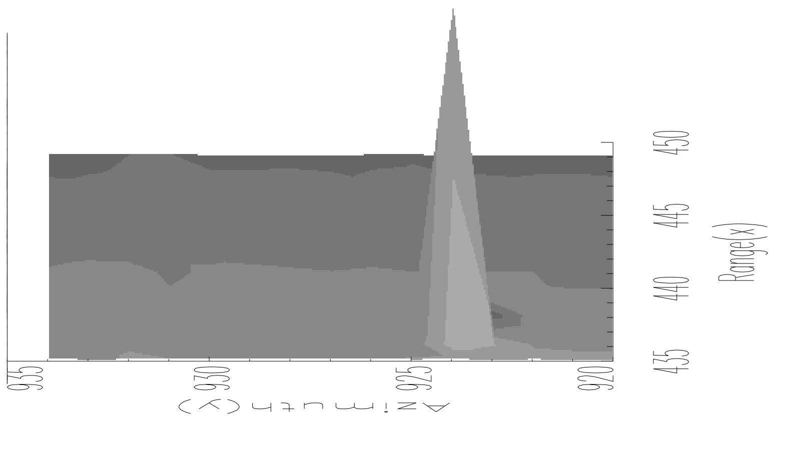


Figure 3.4: Zoomed-in plots showing offsets in both dimensions

translation from the plots shown are $(512 - 437 = 75)$ pixel samples in the range(x) and $(1024 - 924 = 924)$ in the azimuth(y).

The $3 - dimensional$ plots of the result of phase correlation between a SAR image and a second which is obtained by pre-shifting the original image in the two dimensions is shown in Figure 3.3, zoomed-in views (Figure 3.4) show the exact translations in the range(x) and azimuth(y). The translations obtained here are at best (assuming no rotations or any other distortion) accurate to $1 - resolution$ cell. The second image was obtained by shifting the original image by $75$ pixel samples in the range(x) and 100 pixel samples in the azimuth(y).

## 3.3 Tie-point Offset Module

### 3.3.1 Theory

Application of the calculated centre offset to the displaced image will register the images accurate to $1 - resolution$ cell at best. The rigid translation of one image with respect to the other may not be global, thus, the centre offset calculated may not be valid all through the images. Also, the accuracy of registration to $1 - resolution$ obtained is not adequate for SAR/INSAR applications [34][45].

After the crude offset is applied, a regular grid of “Tie-points” are defined all through both images in question, this is done to obtain the offsets between the images at these specific regions in the images. Generally, the higher the number of Tie-points defined, the higher the probability of the accuracy of the warping function determined from the offsets at these points, this is because there is more statistical information, and the data is generally more representative [5, Pg.353]. The lower limit on the number of Tie-points in the range axis for ensuring accurate results for SAR images is deduced in [45].

A patch of the image is then extracted around each defined Tie-point on the Slave and Master images. The size of the patch to be used for the local correlation is variable, a size of $64 \times 64$ samples in both axes has been determined to produce accurate results for SAR images with normalized cross-correlation greater than $0.2$ [20].

The extracted patch is interpolated by zero-padding of the signal in the frequency domain. An interpolation by a factor of $8$ ensures that the local offset calculated is accurate to $1/8$ of a resolution cell of the image, this is required for SAR/INSAR applications [46]. An example of interpolating $4 \times 4$ samples in the frequency domain by a factor of $2$ is shown. The same interpolation scheme for a $64 \times 64$ sample image patch will result in a $512 \times 512$ samples patch for the local correlation.

$$\begin{array}{cccccccc}
x & x & 0 & 0 & 0 & 0 & x & x \\
x & x & 0 & 0 & 0 & 0 & x & x \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
x & x & 0 & 0 & 0 & 0 & x & x \\
x & x & 0 & 0 & 0 & 0 & x & x
\end{array}$$

**Note** $x$ represents the image sample in the frequency domain

The local offset is then calculated using the magnitude correlation method as described in Section 3.2.

### 3.3.2 Parallelization

The numerous offset calculations can be done in parallel, the results are then collated for use in subsequent calculations.

The two images are divided equally amongst the computing nodes, this ensures that the number of local offset to be calculated on each computing node is the same. Since the processing to be done is at the pixel (low) level [9], the ***load balancing*** for this task is trivial.

Consideration is given for the size of the RAM on the computing nodes. Depending on the size of this, it may be necessary to send the portions of the images multiple times. A protocol is also established for file access by the computing nodes as they load their assigned portion of the image into memory for processing, this is necessary because only one node can access the image files at once. A node loads its image portion for processing into memory then sends a confirmation to the spawning (controlling) node when it is done, a signal is then sent to the next node to perform the same operation until all of them have done this.

### 3.3.3 Implementation

- Divide images (equally to the last row) using Column/Rows method [49, Pg. 102] among the computing nodes.
- Apply crude centre offset to slave image
- Load assigned portion of images into memory and send confirmation signal to spawning (controlling) computing node.
- Computing nodes perform the following tasks:
  - Define Tie-point positions.
  - Extract image patch around Tie-points.
  - Interpolate, do magnitude correlation and obtain $2 - dimensional$ offset for each image patch.
  - Record Tie-point Offsets, signal to spawning computing node when done, send Tie-point offsets to spawning computing node for further processing.

## 3.4 Warp Module

### 3.4.1 Theory

As stated in Section 1.1, the main objective of registration is to obtain the relationship between the two images in question; what function applied to a slave image will warp it to a reference master image. The warping process can be divided into two:

#### 3.4.1.1 Calculation of warping Co-efficients

There is no prior knowledge of the ***exact*** nature of the relationship between the slave and the reference image, this is usually represented by a general polynomial [5, Pg. 353][23].

For the reference image with sample co-ordinates $(x, y)$, the corresponding sample position on the slave image to be warped is $(x', y')$. The relationship between the co-ordinates may then be of the form

$$x' = a_o + a_1x + a_2y + a_3xy + a_4x^2 + a_5y^2 + a_6x^2y^2.............. \tag{3.6}$$

$$y' = b_o + b_1x + b_2y + b_3xy + b_4x^2 + b_5y^2 + b_6x^2y^2.............. \tag{3.7}$$

where $a_1, a_2.......b_1, b_2.........$ are the warping co-efficients which are the unknowns in the polynomial. When these are obtained, the positions $x', y'$ can then be calculated for all $x, y$.

The order of the general polynomial chosen is a trade-off between accuracy and speed. Generally, the higher the order of the polynomial, the better the accuracy of the warping function. Higher order polynomial functions may however be unpredictable and usually, the limit is the $3rd$ order [23][35]. Also, higher order polynomials give better warping accuracy around the Tie-points but large warping errors may occur at positions far away from the Tie-points [35]. A single order polynomial is chosen for this warping function, this accounts for translations, rotations, stretch, shear and scaling which are the typical distortions which may occur in SAR images. The polynomial used is:

$$x' = a_o + a_1x + a_2y + a_3xy \tag{3.8}$$

$$y' = b_o + b_1x + b_2y + b_3xy \tag{3.9}$$

Linear regression approximation method is used to determine the values of the co-efficients; the polynomial is fitted to the data provided by the offsets from the Tie-points by minimizing the error for each data point from the function defined by the polynomial. The minimum number of Tie-points data required to obtain the warping co-efficients is the number of co-efficients to be calculated, in this case, many more Tie-points are used in practice as earlier mentioned.

The least square regression method applied is ***weighted***, that is, a figure of merit (in this case the Cross-correlation $CC$ ) is used to determine what statistical contribution the data from each Tie-point offset makes in the linear regression approximation. Generally, the higher the value of the $CC$, the more reliable the offset results obtained from the magnitude correlation method. Temporal de-correlation in SAR images are due to bodies of water on the images, layed-over regions etc.[46][29], the offsets obtained from these regions are not reliable, the $CC$ values are very low $(\approx 0)$ and they make little or no contribution in the regression approximation. The matrices used for the calculation are of the form:

$$\begin{bmatrix} x'_0 \\ x'_1 \\ | \\ | \\ x'_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & x_0 & y_0 & x_0 y_0 \\ 1 & x_1 & y_1 & x_1 y_1 \\ | & | & | & | \\ | & | & | & | \\ 1 & x_{n-1} & y_{n-1} & x_{n-1} y_{n-1} \end{bmatrix} * \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{3.10}$$

$$W = \begin{bmatrix} w_0 & 0 & - & - & - \\ 0 & w_1 & - & - & - \\ | & | & | & | & | \\ | & | & | & | & | \\ 0 & 0 & 0 & 0 & w_{n-1} \end{bmatrix}$$

where $n$ is the number of Tie-point offset results used, and W is the weighting matrix in the curve fitting process (GNU Scientific Library `http://sources.redhat.com/gsl/ref/gsl-ref_toc.html`). A similar matrix is formed for $y'$ to solve for the co-efficients $b_0, b_1, b_2$ and $b_3$ .

#### 3.4.1.2 Warping

The calculated values of the co-efficients are used to create a third image (warped image) by calculating the positions for the samples on the slave image as represented on the reference image using Equations 3.8 and $1.9$. The corresponding sample pixel positions to be read from $(x', y')$ may be:

- $x'$ and $y'$ are integer values:-use exact image sample represented at this position in the slave image on the warped image.
- $x'$ and or $y'$ fall outside the range of the image:- set the value of the sample position in the warped image to zero(0).
- $x'$ and or $y'$ is/are non-integer value(s):- the pixel position to be read from is not defined on the slave image(see Figure 3.5). It has been determined that the $6-point$ cubic interpolation method is the best way for computing the value which will be

represented in pixel position for SAR images [24][4]. This method is employed for all the positions that fall in this group. The 2-dimensional $6 - point$ cubic interpolation is achieved by applying the following function

$$g(x,y) = \sum_{l=-1}^{2} \sum_{m=-1}^{2} c_{j+l,k+m} u\left(\frac{x - x_{j+l}}{h_x}\right) u\left(\frac{y - y_{k+m}}{h_y}\right) \tag{3.11}$$

$$u(s) = \begin{cases} \frac{3|S|^3}{2} - \frac{5|s|^2}{2} + 1 & 0 <| s |< 1 \\ \frac{-1|s|^3}{2} + \frac{5||^2}{2} & 1 <| s |< 2 \\ 0 & 2 <| s | \end{cases} \tag{3.12}$$

where $u$ is the interpolation kernel, $(x, y)$ is a point that falls in the rectangle whose vertices are defined by $[x_j, x_{j+1}]X[y_k, y_{k+1}]$,$h_y$ and $h_x$ are the $x$ and $y$ sampling increments (both $= 1$ in this case), and $j, k, l$ and $m$ are dummy variables used as counters [24].

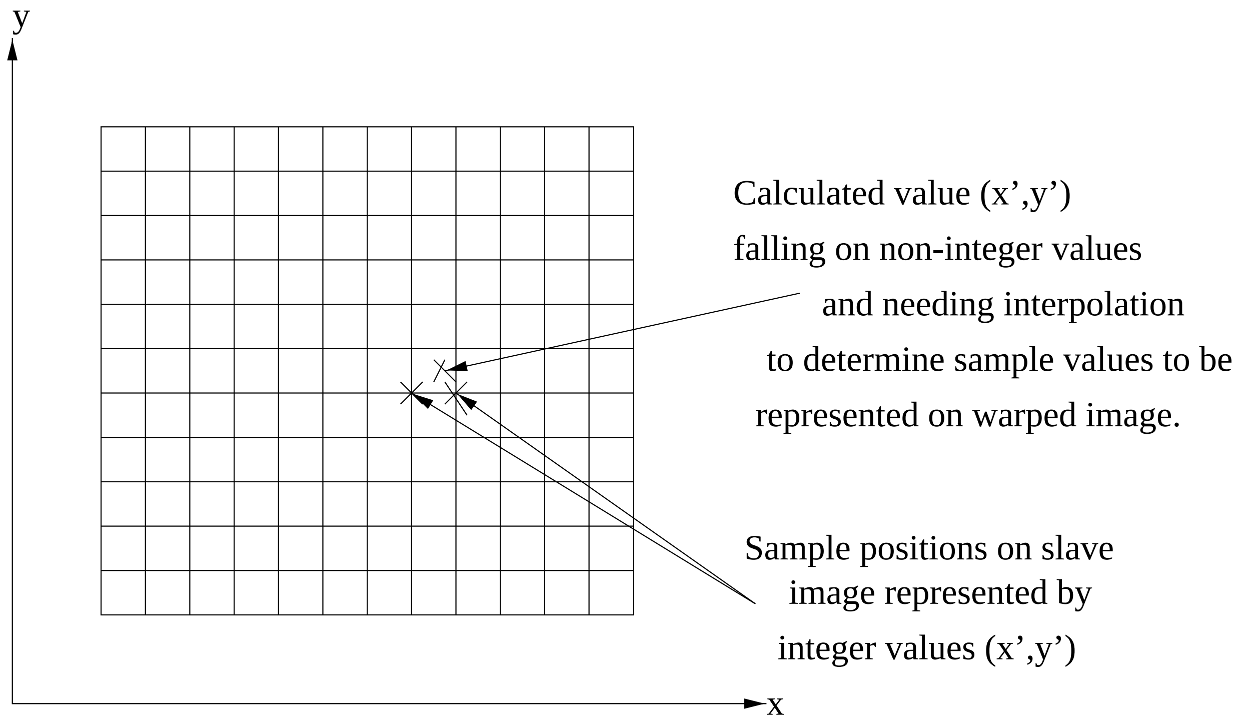
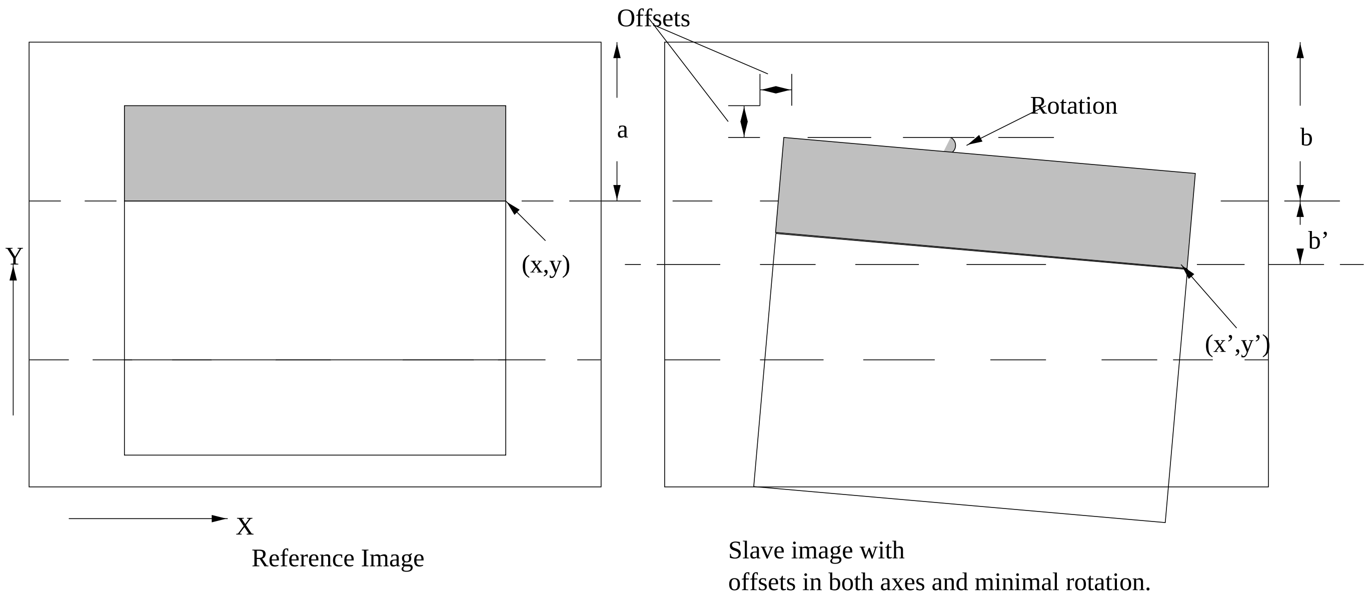

y
x
Calculated value (x',y')
falling on non-integer values
and needing interpolation
to determine sample values to be
represented on warped image.
Sample positions on slave
image represented by
integer values (x',y')

Figure 3.5: Diagram showing need for interpolation after applying warping function

### 3.4.2 Parallelization

The calculation of the warping co-efficients is intrinsically serial.

The actual warping (and interpolation when needed) is parallelized. The slave image to be warped is divided amongst the computing nodes, load balancing, file access and limitations of available RAM considerations are made as described in Sub-section 3.3.2.



a= portion of image in reference image whose samples x,y, are to be located in the slave image

b= portion of image in slave image equal to the same size in reference image

b'=portion to be added (in number of rows) to the portion of slave images sent to be warped in order to preserve edge information

The sample located at (x,y) in the reference image will be located at (x',y') in the slave image

Figure 3.6: Typical ERS SAR translation and rotation distortions

The new pixel positions of the samples on the slave image to be represented on the warped image depend on the obtained warping function and are not known prior to the actual warping. It is thus necessary that the portion of the slave image sent to each computing node is greater than the portion of the image to be warped. This takes account for the ***edge effect*** [8][10][49]. How much more has to be sent is determined on the basis of the initial calculated centre offset plus some more to account for rotation and other distortions reflected in the final warping polynomial used.

In this algorithm, the portion of the slave image to be warped plus twice the number of rows calculated as the azimuth centre offset is assigned to each processing node. This is based on the envisaged worst case rotation in SAR images (see Figure 3.6) and should ensure that no information is lost at the edges of the adjoining image portions sent to the different computing nodes for warping.

### 3.4.3 Implementation

- Calculate warping co-efficients using results from Tie-points offset calculation.
- Divide slave image as in Sub-section 3.3.3. Add $(2 * azimuth - centre - offset)$ rows to the size of the slave image to be loaded into memory by the processing nodes.
- Access file and load assigned image portion into memory. Observe file access protocol as described in Sub-section 3.3.3.
- Obtain sample value, (interpolate when needed) from slave image for new position in warped image. Do for each pixel position on the reference image to produce warped image portion on each computing node.
- Send signal to spawning (controlling) node when done, controlling node ensures file access protocol, and proper assembly of the different image segments.

## 3.5 Algorithm Summary

- The magnitude correlation technique utilizes the phase information in the backscatter of the SAR images to be registered, this is the ***Feature space***.
- The ***search space*** is ultimately determined by the distortions (translations,rotation, shear) accounted for in the determined warping polynomial.
- The ***search strategy*** involves an initial crude centre-offset calculation, followed by a more accurate Tie-point offset calculations, and the determination of a warping polynomial based on this.
- The ***similarity measure*** will be mentioned under the discussions on accuracy.

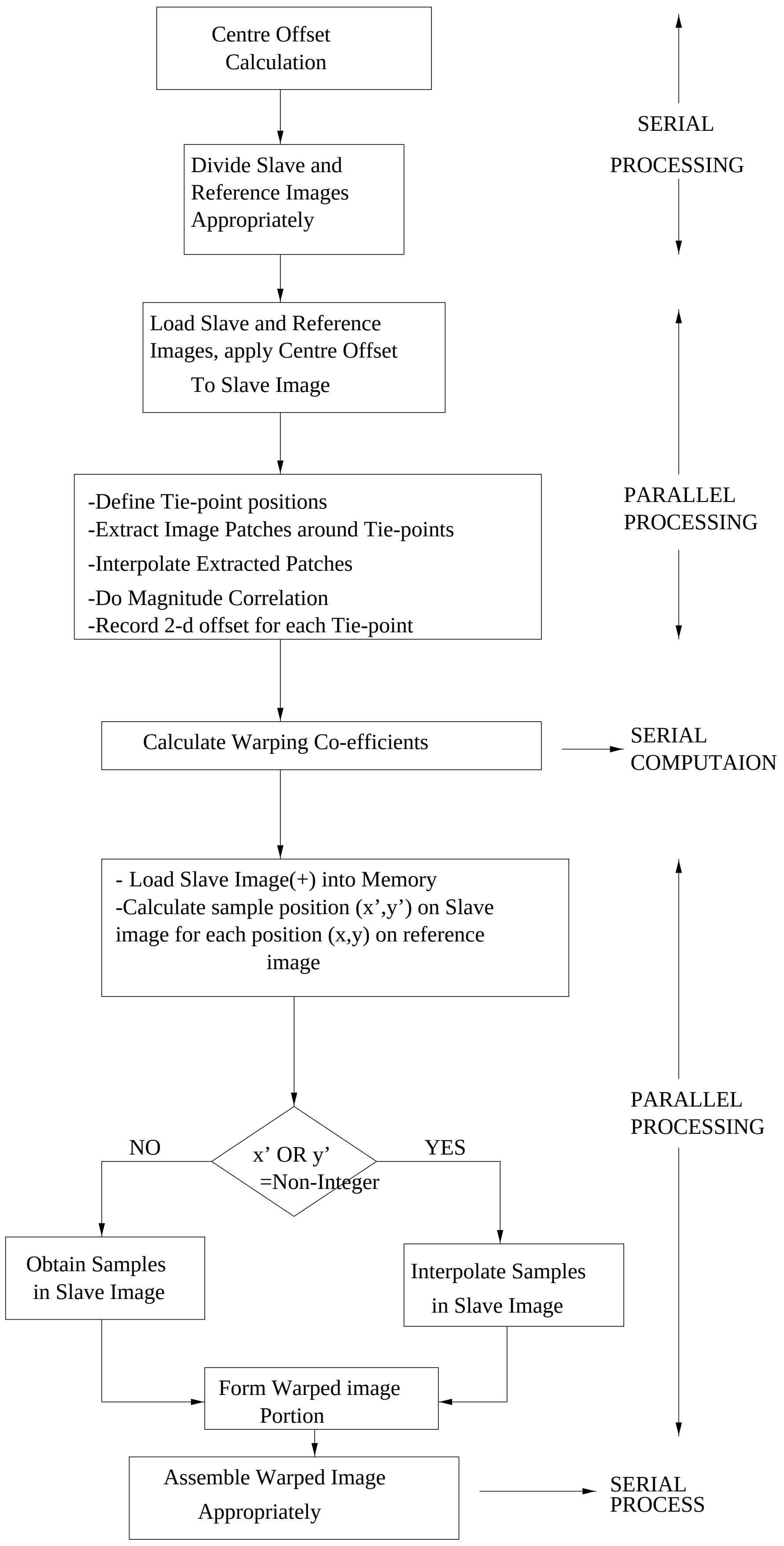

Centre Offset Calculation
Divide Slave and Reference Images Appropriately
Load Slave and Reference Images, apply Centre Offset To Slave Image
-Define Tie-point positions
-Extract Image Patches around Tie-points
-Interpolate Extracted Patches
-Do Magnitude Correlation
-Record 2-d offset for each Tie-point
Calculate Warping Co-efficients
- Load Slave Image(+) into Memory
-Calculate sample position (x',y') on Slave image for each position (x,y) on reference image
x' OR y' =Non-Integer
NO
YES
Obtain Samples in Slave Image
Interpolate Samples in Slave Image
Form Warped image Portion
Assemble Warped Image Appropriately
SERIAL PROCESSING
PARALLEL PROCESSING
SERIAL COMPUTAION
PARALLEL PROCESSING
SERIAL PROCESS

Figure 3.7: Algorithm Flow Diagram

## 3.6 Conclusion

The algorithm thus designed based on theoretical and practical considerations will be implemented and tested to determine its effectiveness for the required purpose

# Chapter 4

# Implemented Algorithm Tests

## 4.1 Introduction

The parallel algorithm designed in Chapter 3 is implemented in C language on a parallel cluster of 6 processing nodes (all processing nodes with the same configuration- pentium II $350Mhz$ processor with $512Mbytes$ RAM) with the Linux operating system installed on all processing nodes. The Parallel Virtual Machine (PVM) is the middle-ware for communication and other parallel processing constructs.

The images used for testing the implemented algorithm are tandem images of Cape Town, South Africa obtained by the ERS satellite, the images relevant parameters are listed in the Appendix. The raw images obtained by the satellites were processed into fully focused SAR images. Test images were then extracted from both images to be registered. Figure 4.1 shows the full image and an approximate area extracted from both images for registration.

This chapter discusses the results under two major headings of accuracy of the registration and timing results.

## 4.2 Image Registration Accuracy Tests

The GAMMA SAR processing software (GAMMA Remote Sensing Research and Consulting AG `http://www.gamma-rs.ch/`) is a well used commercial SAR process-

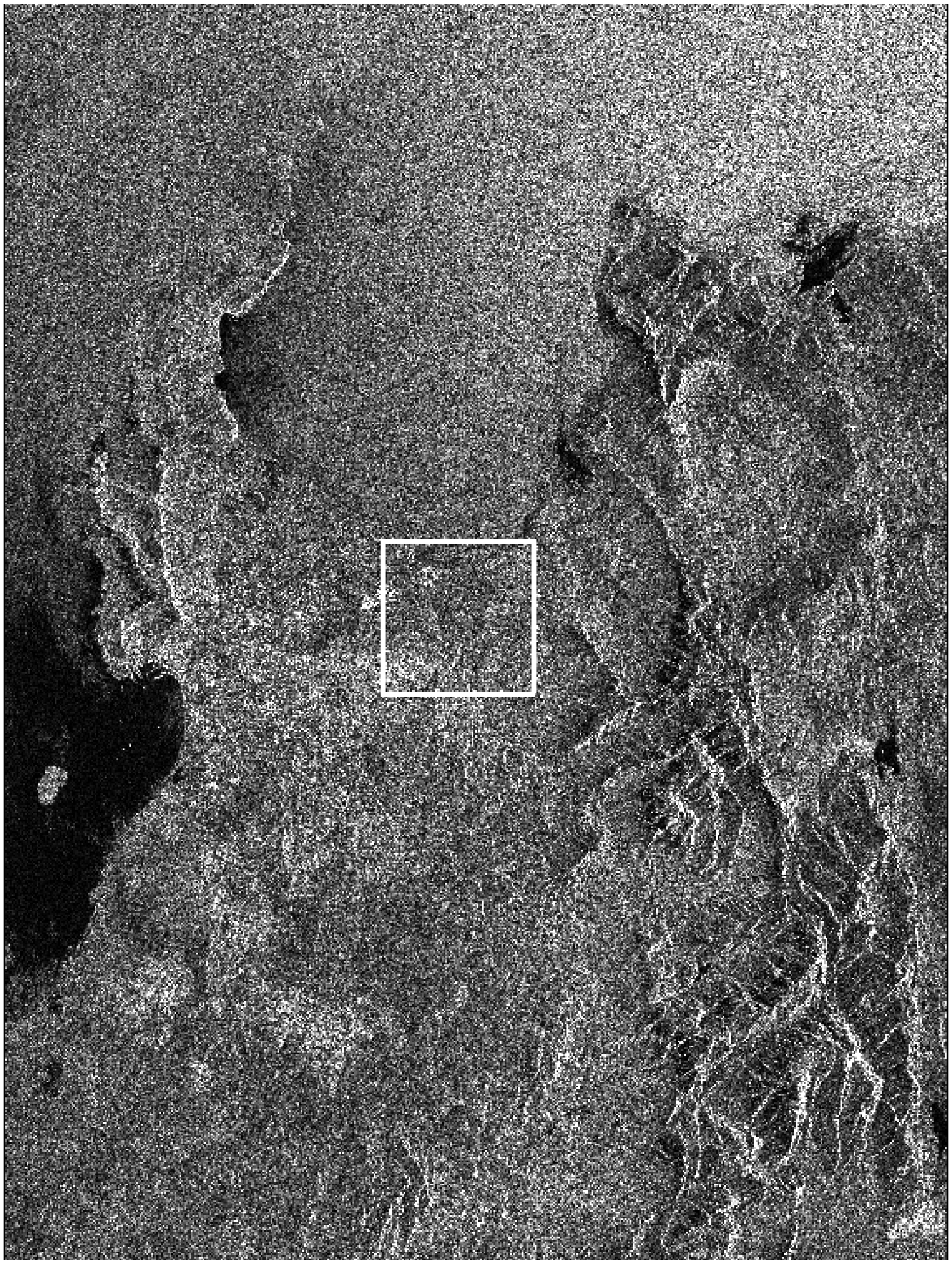


Figure 4.1: Cape Town ERS SAR Image showing extracted portion for registration

ing software. One of the functions it is capable of is the registration of fully focused SAR images which is what the algorithm implemented in this project does. The result of registering the test images using the GAMMA software is used as a benchmark for assessing the accuracy result of the algorithm as implemented in this project. The comparison is both qualitative(visual inspection) and quantitative.

### 4.2.1 Qualitative Measure

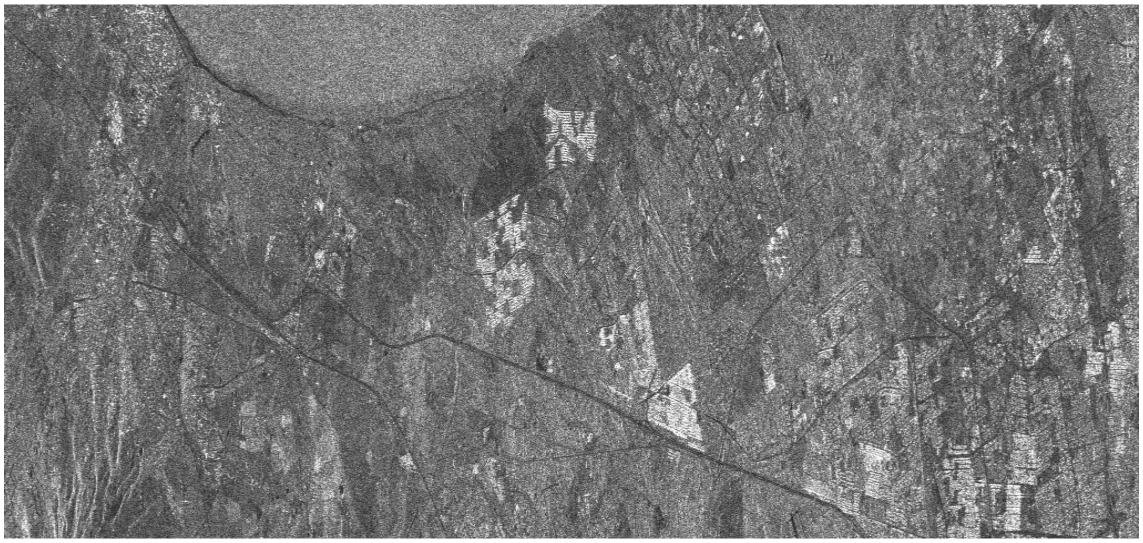
The quality of interferogram images [46] formed from registered images of the same scene is a measure of how accurately the images have been registered. The quality of this also depends on other parameters, especially the imaging baseline (see 1.1)[38]. However, for the same set of images registered by different algorithms, the quality of interferogram is a valid basis for comparison. Inaccurate registration results in interferometric noise - random nature of the phase plots over a region resulting from non-coherence and non-clarity of interferometric patterns resulting from such phase plots where they exist. Properties to look for in interferogram images formed from accurately registered images are:

- Appearance of interferometric patterns such as “contour-like” lines where there are height changes (for example mountains) on the images.
- Clarity of these patterns.

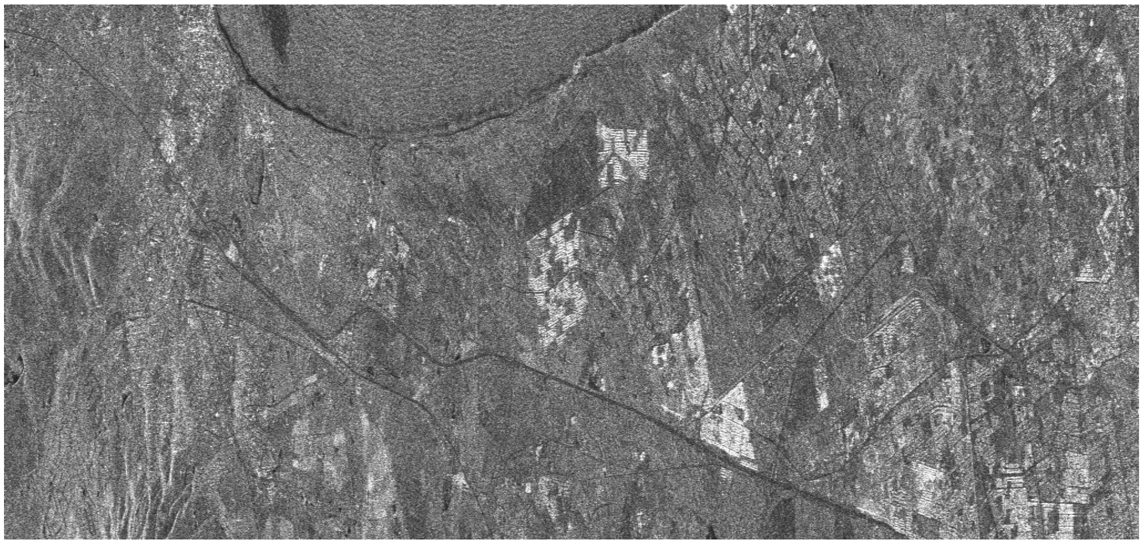
The quality of both of these are an indication of the values of the correlation co-efficient (CC) of the registered images

Figure 4.2 shows the original extracted images and the resulting warped (registered) slave image using the GAMMA software and the implemented algorithm. The reference master image does not change.

The regions with contour lines on the interferograms shown indicate regions of height changes, for example due to mountains (this may be apparent from the original images Figure 4.3). The interferograms in both cases show good visual similarity in all regions. Based on the formed interferograms from the registered images, the implemented registration algorithm compares well with the GAMMA software.

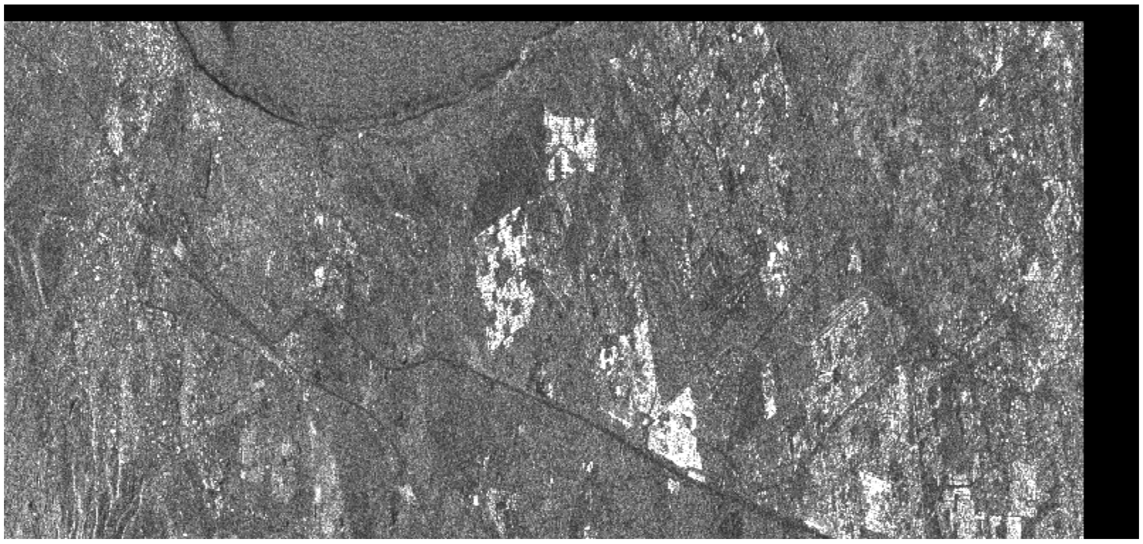


(a) Extracted portion of Master/Reference Image (05715.slc)

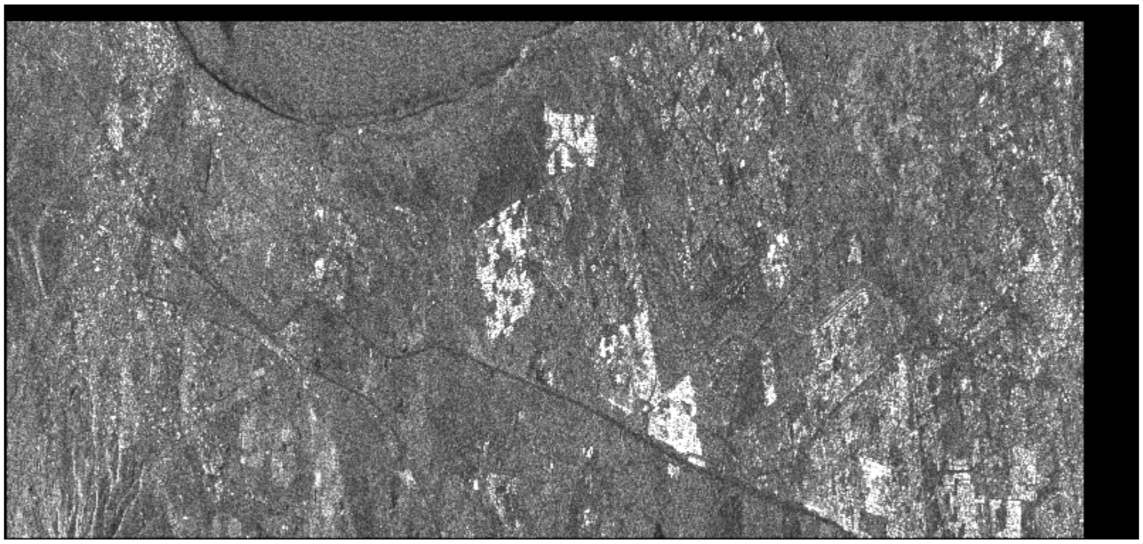


(b) Extracted portion of Slave Image (25388.slc)

Figure 4.2: Original extracted images for registration and the resulting registered images using GAMMA and the Implemented Algorithm

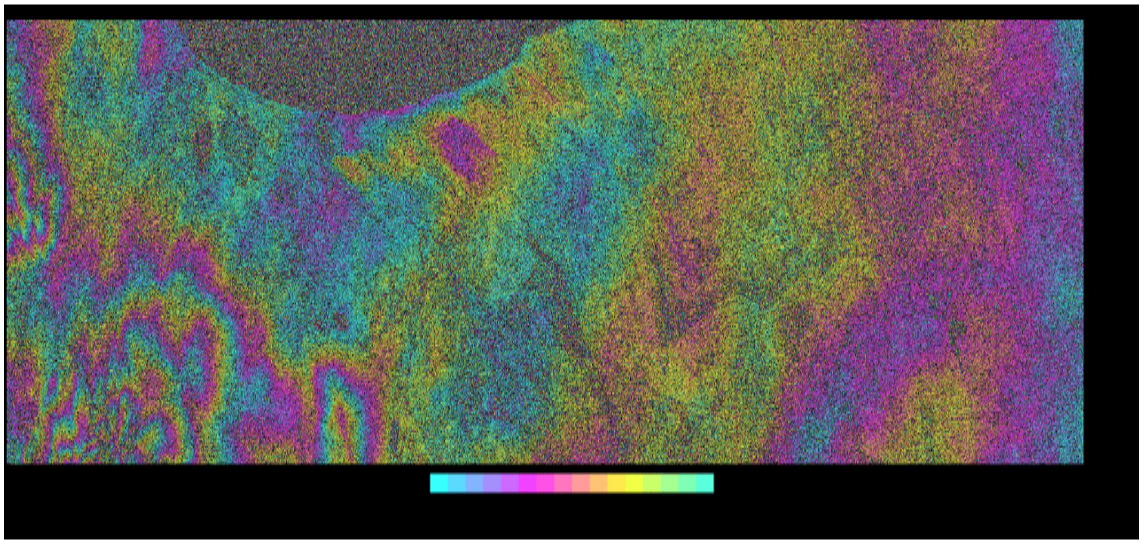


(a) Registered Slave Image Using GAMMA software

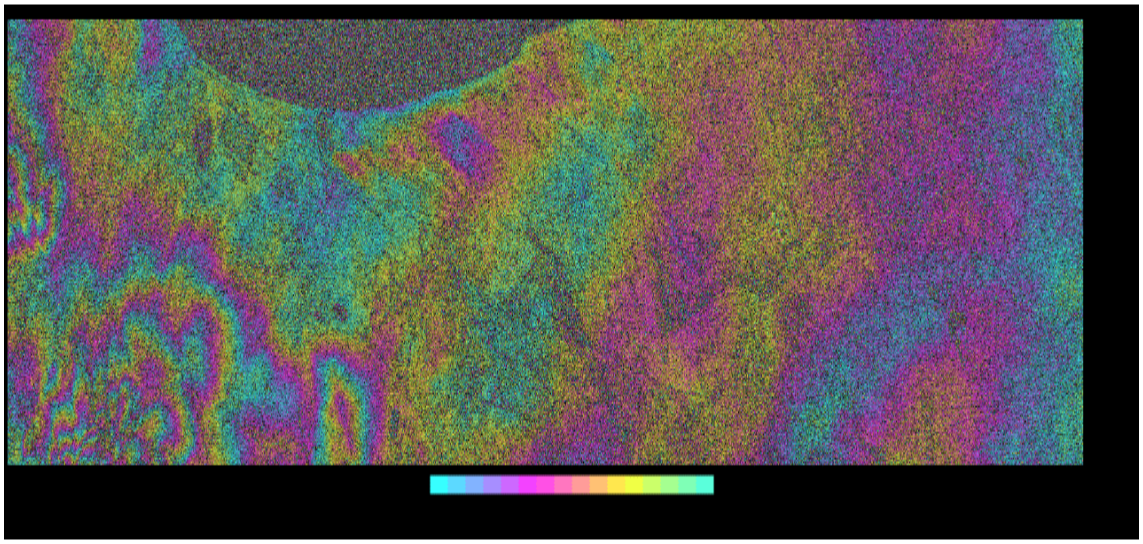


(b) Registered Slave Image Using Implemented algorithm

Figure 4.3: Registered Slave Images using the GAMMA and Implemented algorithm



(a) Interferogram from images registered by GAMMA



(b) Interferogram from images registered by implemented algorithm

Figure 4.4: Flattened Interferograms formed by images registered by GAMMA and the implemented algorithm

Colour code: Blue $(0)$ -> Light Green$(2\pi)$

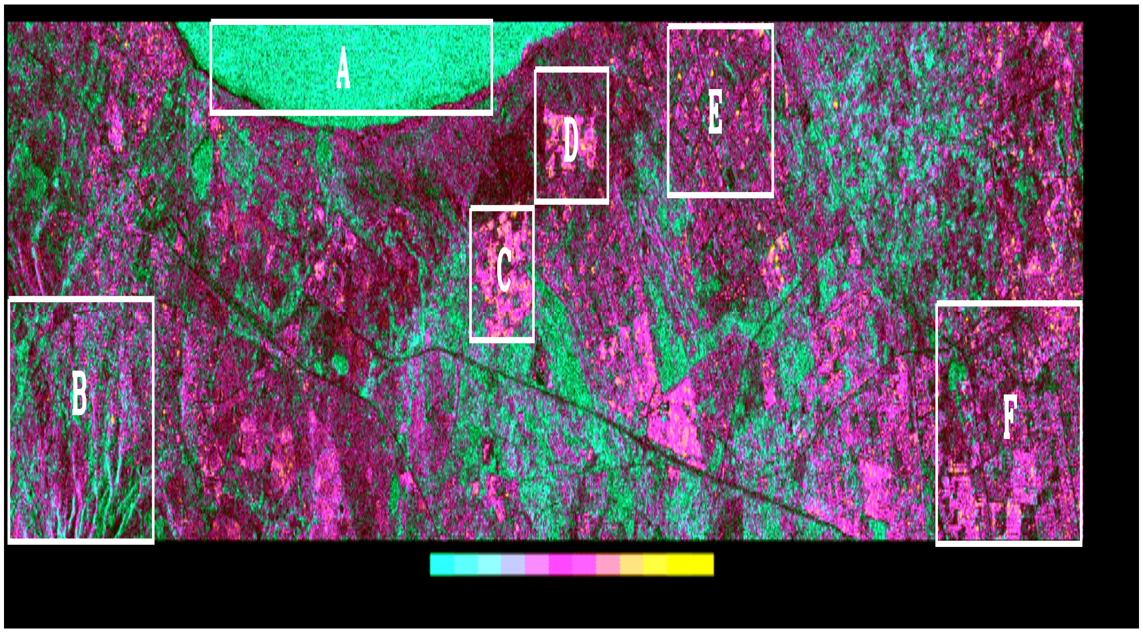


Figure 4.5: Coherence Map showing the regions whose average values are compared
Colour Code: Blue (Lowest CC) -> Yellow (Highest CC)

### 4.2.2 Quantitative Measure

The quantitative measure of accuracy of registration is the coherence co-efficient $(CC)$ 3.2. A single figure such as the average coherence co-efficient over a whole image may however not be a valid indication of how well the images in question have been registered. A good example of situations for which a single average value will not be valid is found in images for which a large region of the images have non-coherent backscatter due to the nature of the scene being imaged, eg. bodies of water. This will result in a low value of $CC$ (tending to $0$ on a per unit scale of $0 \leq CC \leq 1$ ) over the whole image. This may hide the fact that small but relevant regions of the images are accurately registered with relatively high values of $CC$ . For the purpose of quantitative comparison of the implemented algorithm and the GAMMA software, the average $CC$ values are computed for different regions of the generated coherence co - efficient maps. The regions considered are as marked in Figure 4.5.

As is shown in Table 4.1, the $CC$ average values of the regions marked in Figure 4.5 compare well. Note the $CC$ values for region A which is a body of water, the average $CC$ value is very low as expected, if this region formed a larger percentage of the images, it will be an example of the earlier described possible situation and will justify the argument for marking specific regions for comparison. The body of water is however a relatively

Table 4.1: Average Coherence Coefficient values ( on a scale of $0 \rightarrow 1$ ) for marked regions on the Coherence map in Figure 4.5

| Region | GAMMA | Implemented Algorithm |
|---|---|---|
| A | 0.155 | 0.156 |
| B | 0.366 | 0.381 |
| C | 0.507 | 0.514 |
| D | 0.464 | 0.470 |
| E | 0.414 | 0.416 |
| F | 0.463 | 0.448 |
| Whole Image | 0.347 | 0.350 |
| Whole Image (without water region A) | 0.367 | 0.370 |

small percentage of the whole image and the average CC value over the whole image is thus a valid measure in this case.

Based on the averaged $CC$ values over the marked regions, the implemented algorithm registers the images in question at least as accurately as the GAMMA software.

Note that numerous parameters can be changed in the process of registering images using the GAMMA processor in a bid to obtain the best accuracy possible. The parameters set for the registration used as a bench mark here were chosen with a bid to get the best registration accuracy possible, this information was obtained by consulting the GAMMA software reference manual and also from consultation with colleagues who have wide experience in the use of the software.

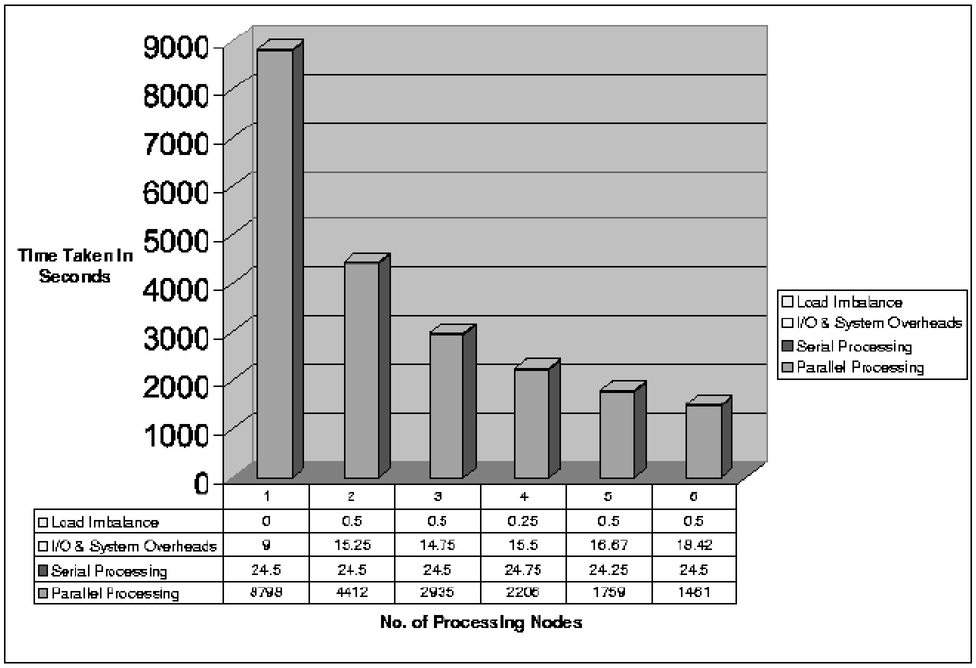
## 4.3 Timing Results

Even though the underlying principles for registering images using the GAMMA software are very similar to that employed in this project, the exact nature of the relationship between the numerous sub-tasks cannot be established, this makes exact timing comparison impossible. The GAMMA software's parameters were however set to make the tasks performed to do the registration to be very similar to the tasks performed by the algorithm. The GAMMA software was found to be slightly faster than the implemented algorithm on a single processing node. (Average time for GAMMA processor $= 8687 seconds$ , Average time for implemented algorithm $= 8700 seconds$ ). The time difference average $= 13 seconds$ represents a minute percentage ( $\approx 0.15\%$ ) of the total time taken for the processing. The total time taken for the image registration process is measured for different numbers of processing nodes $(n)$. The tasks carried out in each case is the same. The same test is carried out for each number of processing nodes four $(4)$ times to ensure statistical validity. Table 4.2 shows a total time budget for each number of processing nodes for the identifiable tasks in the registration process. A summary of the budget, logically categorizing the different tasks into four$(4)$ groups - *(i)Parallel processing*, *(ii)Serial processing*, (*iii*)I*/O and System overhead* and *(iv)Imbalance* are shown in the Tables adjoined to the plots in Figures 4.6 and 4.7 which show the plot of time results in seconds and as percentages of a whole.

Table 4.2: Average values of recorded time for the different tasks that make up the registration process. All recorded values are in Seconds.

| Tasks | 1 Node | 2 Nodes | 3 Nodes | 4 Nodes | 5 Nodes | 6 Nodes |
|---|---|---|---|---|---|---|
| Extract Centre (I/O) | 0.25 | 0.25 | 0.25 | 0.50 | 0.25 | 0.25 |
| Centre Offset Calculation (SC) | 23.50 | 23.50 | 23.50 | 23.25 | 23.25 | 23.50 |
| Register Processing Nodes (SO) | 6.75 | 7.25 | 7.25 | 7.00 | 7.00 | 7.25 |
| Part of Tie-point Offset Comp.(I/O & SO) | 3.75 | 3.50 | 3.50 | 3.50 | 3.25 | 3.75 |
| Part of Tie-point Offset Comp. (PC) | 8700.70 | 4361.50 | 2901.00 | 2180.67 | 1738.00 | 1444.70 |
| Part of Tie-point Offset Comp.(I/O & SO) | 1.00 | 0.50 | 0.25 | 0.75 | 1.67 | 2.00 |
| Part of Tie-point Offset Comp.(Im) | 0.00 | 0.50 | 0.00 | 1.00 | 0.50 | 1.00 |
| *Compute Warping Coefficients(SC) | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Part of Warp Slave Image(I/O & SO) | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.67 |
| Part of Warp Slave Image (PC) | 97.00 | 50.50 | 34.00 | 25.25 | 21.00 | 16.50 |
| Part of Warp Slave Image (I/O & SO) | 2.00 | 3.00 | 2.50 | 3.25 | 3.50 | 3.50 |
| Part of Warp Slave Image(Im) | 0.00 | 0.50 | 1.00 | 0.50 | 0.50 | 0.50 |
| Total | 8835.95 | 4452.00 | 2974.25 | 2246.17 | 1799.92 | 1504.12 |

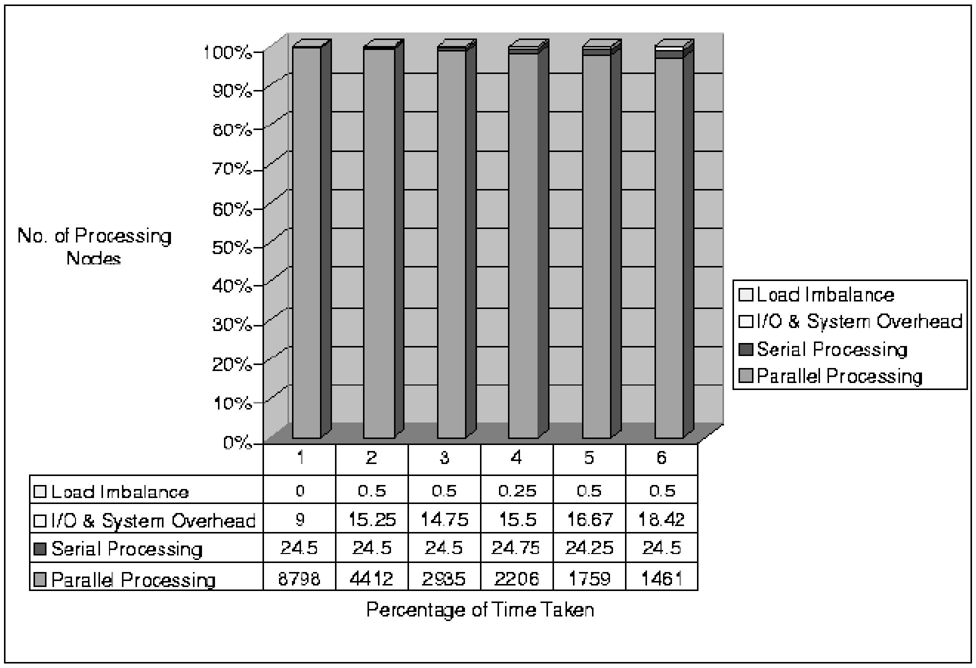
I/O:Input-Output SC:Serial Computation PC:Parallel Computation
SO:System Overheads Im:Imbalance
* The four averaged values were much less than 0.00 seconds



| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| ☐ Load Imbalance | 0 | 0.5 | 0.5 | 0.25 | 0.5 | 0.5 |
| ☐ I/O & System Overheads | 9 | 15.25 | 14.75 | 15.5 | 16.67 | 18.42 |
| ■ Serial Processing | 24.5 | 24.5 | 24.5 | 24.75 | 24.25 | 24.5 |
| ▨ Parallel Processing | 8798 | 4412 | 2935 | 2206 | 1759 | 1461 |

(a) Percentages of time spent on different categories for different number of processing nodes

Figure 4.6: Plots showing the time budget for the registration process

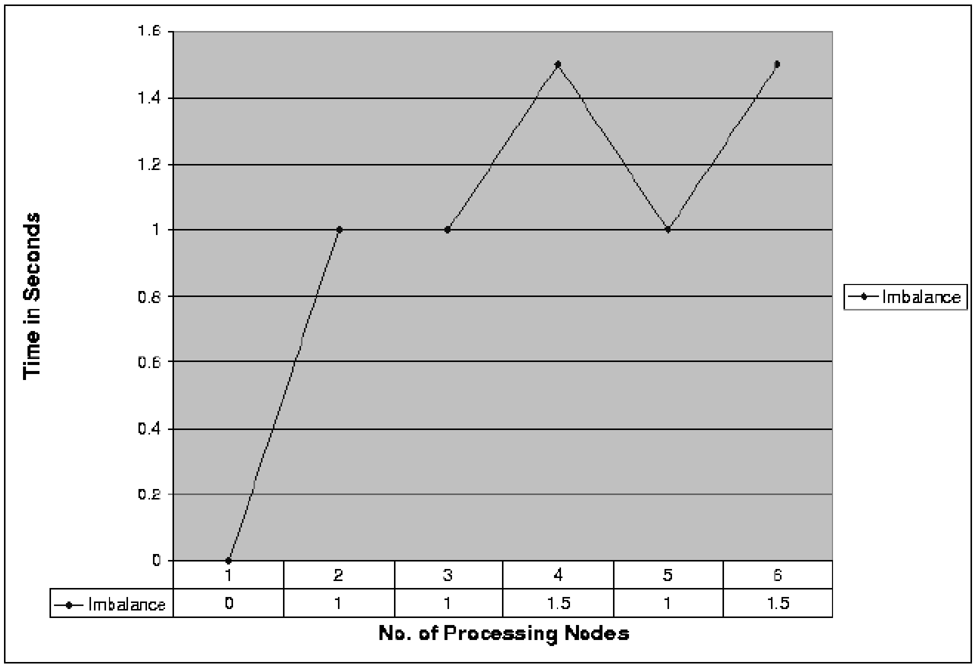


| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| ☐ Load Imbalance | 0 | 0.5 | 0.5 | 0.25 | 0.5 | 0.5 |
| ☐ I/O & System Overhead | 9 | 15.25 | 14.75 | 15.5 | 16.67 | 18.42 |
| ■ Serial Processing | 24.5 | 24.5 | 24.5 | 24.75 | 24.25 | 24.5 |
| ▨ Parallel Processing | 8798 | 4412 | 2935 | 2206 | 1759 | 1461 |

Figure 4.7: Percentage of Time taken by different Categories of the processes

The main focus of any parallel algorithm design is the maximization of the percentage of the time taken for parallel processing and as a result minimizing the time taken for the other listed 3 categories. This ensures that the processing nodes “enrolled” in the parallel cluster are actually processing for most of the time and not losing idle CPU cycles(processor utilization). This determines the overall efficiency of the parallel system and thus indicates justification for the parallelization scheme(or otherwise).

### 4.3.1 Load Imbalance



|  | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Imbalance | 0 | 1 | 1 | 1.5 | 1 | 1.5 |

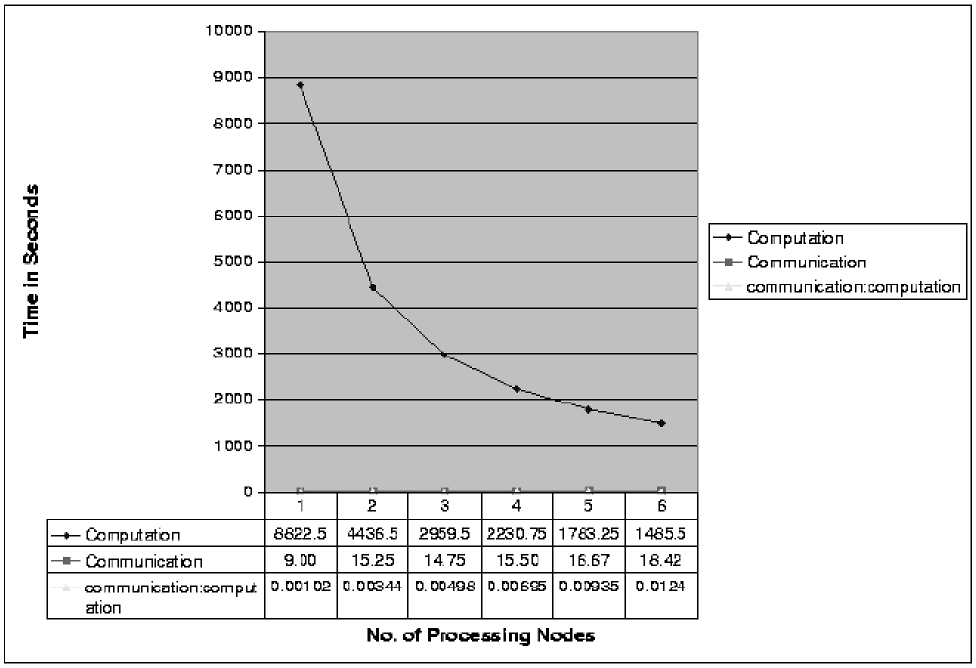
Figure 4.8: Plot showing extra time overhead due to load imbalance

As earlier explained, the tasks performed in parallel - tie-point offset calculation and image warping are at the low level (pixel level) and are evenly distributed 3.3.2. This ensures that the computing load distributed amongst the computing nodes are well balanced.

The recorded average values of imbalance for all values of $n$ are very low ($\approx 0$ seconds), they seem random, following no particular trend and cannot be attributed to any identifiable specific cause.

The time lost as a result of imbalance in load on processing nodes for this implemented algorithm as a percentage of the total time taken for the processing is negligible for all values of $n$ and can be ignored.

### 4.3.2 Communication versus Computation



| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Computation | 8822.5 | 4436.5 | 2959.5 | 2230.75 | 1783.25 | 1485.5 |
| Communication | 9.00 | 15.25 | 14.75 | 15.50 | 16.67 | 18.42 |
| communication:computation | 0.00102 | 0.00344 | 0.00498 | 0.00695 | 0.00935 | 0.0124 |

The plots of communication and communication:computation are hardly distinguishable because of their relatively low values, the adjoining table however shows the values

Figure 4.9: Plots showing the communication:computation ratio as the number of processing nodes is scaled

An objective of all parallel algorithm is to reduce the communication : Computation ratio.

The time taken for I/O and the accompanying system overhead are shown in the adjoining table to the plot shown in Figure 4.9, the Figure also shows the communication : computation ratio ( this is hardly distinguishable from the plots for the communication, the values are shown in the adjoining Table ). As is expected, the ratio is recorded as increasing as

$n$ is increased - the more the number of processing nodes, the more time is spent on I/O and system overheads.

A great percentage of the time in the I/O and system overhead category is accounted for by the time taken to “enlist” the processing nodes in the parallel cluster (see Table 4.2), this is a function of PVM and is unavoidable. The time taken for file I/O and information exchange between the processing nodes is almost negligible.

Overall, the communication : computation ratio is very low, the time taken for I/O and system overheads as compared to the time taken for actual computation for this implemented algorithm is almost negligible.

### 4.3.3 Serial versus Parallel Computation

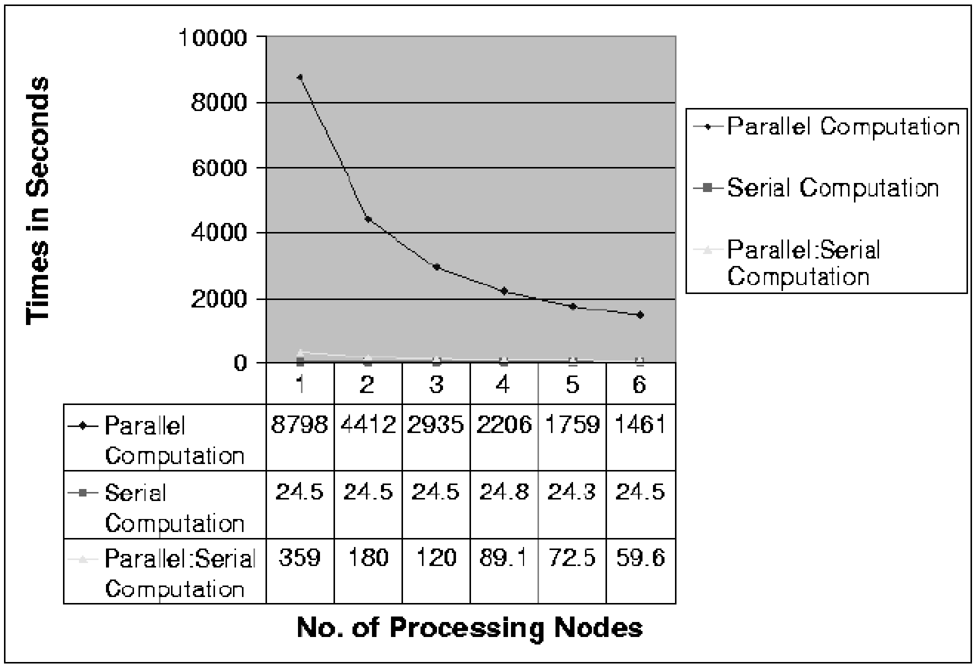
If the time taken for intrinsically serial computation accounts for a relatively high percentage of the whole processes, parallelization of the algorithm may not be justifiable. For example, by Amdahl’s law, the maximum obtainable speed-up by parallelizing an algorithm is given by

$$S = \frac{n}{1 + (n+1)f} \tag{4.1}$$

where $f$ is the serial fraction of the parallelized algorithm and $n$ is the number of processing nodes.

Applying this law, an algorithm with a serial fraction of $5\%$ has a maximum speed-up of $20$ if the remaining $95\%$ are completely parallelized. This is the upper limit for the speed-up irrespective of the number of the number of processing nodes employed in performing the tasks. Note that Amdahl’s law makes a number of assumptions which are not valid for all situations [1], the concept of a rapid decrease in achievable speed-up as intrinsically serial parts of an algorithm increases is however valid for all cases.

For many parallel algorithms (as in this case), the time taken for the serial computation is higher than that for I/O and system overheads and time lost due to imbalance in the computational load on the processing nodes. It is seen that the percentage of the serial computation as a part of the whole process increases fairly rapidly as $n$ is increased ( note



| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Parallel Computation | 8798 | 4412 | 2935 | 2206 | 1759 | 1461 |
| Serial Computation | 24.5 | 24.5 | 24.5 | 24.8 | 24.3 | 24.5 |
| Parallel:Serial Computation | 359 | 180 | 120 | 89.1 | 72.5 | 59.6 |

Figure 4.10: Plots showing Time for Parallel and the Serial Computation and the Parallel:Serial ratio

rapid drop in parallel:serial computation ratio in Figure 4.10 ), this is the main consideration for the upper limit of $n$ to employ for processing and still maintain reasonable efficiency, it determines how scalable a parallel algorithm implementation is. The greater percentage of the time taken for the serial computation is accounted for by the time taken for the centre offset calculation ( see Table 4.2 ), this will be given further considerations under discussions of Speed-up and Scalability in Subsections 4.3.4 and 4.3.5.

### 4.3.4 Speed-up (S)

As a bench mark for assessing the recorded speed-up as $n$ is scaled, a theoretically derived maximum speed-up is used. The underlying assumption in deriving the Gustafson-Barsis law is that the serial portion of the algorithm does not change for different sizes of the algorithm [18], this is valid for this particular algorithm as the time for the serial computation is accounted for by the centre offset calculation and it remains constant for all sizes of the problem. The derivation also assumes that the I/O and system overheads are negligible, this is also valid for this algorithm (see Subsection 4.3.2). If the time spent for the total processing on $n$ computing nodes is

$$T = t_s + t_p \tag{4.2}$$

where $t_s$ is the constant time taken for the serial portion and $t_p$ is the time taken by $n$ processors to complete the parallel portion, then the time taken by $1$ processor is

$$T = t_s + nt_p \tag{4.3}$$

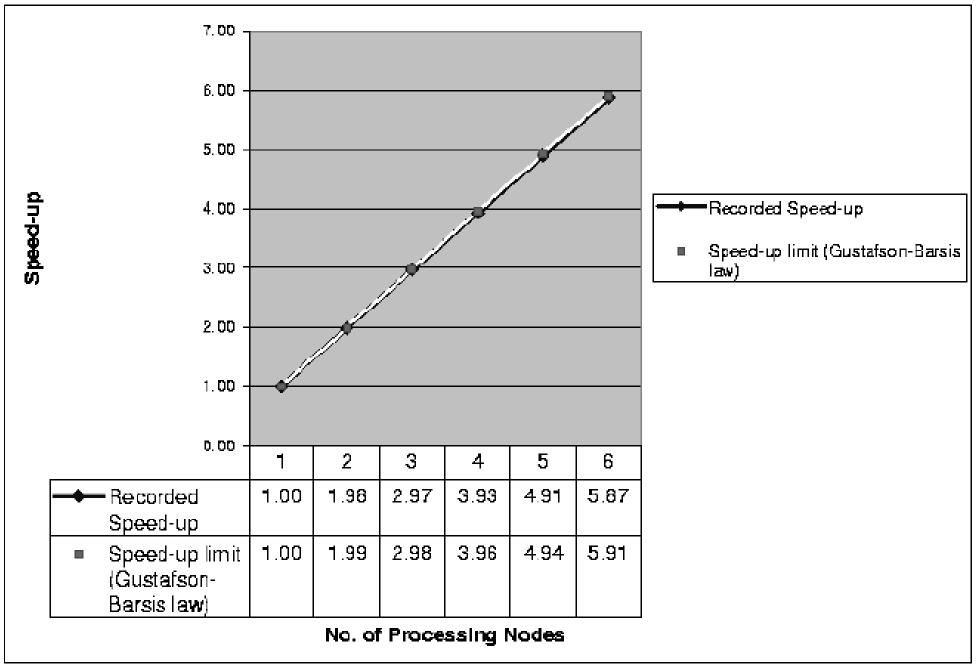
and the speed-up $S$ is given by

$$S = \frac{t_s + nt_p}{t_s + t_p} \tag{4.4}$$

if for simplicity we set $t_s + t_p = 1$, then,

$$S = t_s + nt_p = t_s + n(1 - t_s) \tag{4.5}$$

This is the Gufstafsons-Barsis law.



| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Recorded Speed-up | 1.00 | 1.98 | 2.97 | 3.93 | 4.91 | 5.87 |
| Speed-up limit (Gustafson-Barsis law) | 1.00 | 1.99 | 2.98 | 3.96 | 4.94 | 5.91 |

Figure 4.11: Plots showing Theoretical speed-up limit and recorded value for the implemented algorithm

The adjoining Table to Figure 4.11 shows recorded values obtained by running the implemented algorithm along with theoretical values calculated using Equation 4.5. The plots show that the recorded values are very close to the theoretical limit for all values of $n$ .

### 4.3.5 Scalability

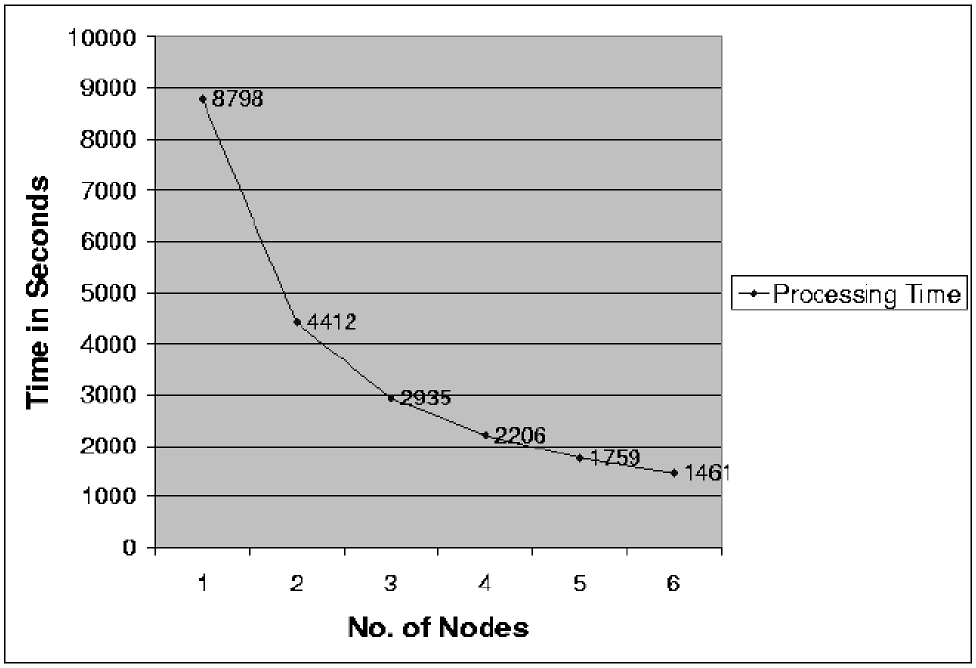


Figure 4.12: Plot showing Scalability of Implemented algorithm as the number of processing nodes is scaled

The scalability of the parallel program indicates how much computing resources can be increased to obtain commensurate increased performance. In this case, the computing resources increased is the number of processing nodes, and the required performance is increased speed-up (reduction in total processing time).

The parallel program as implemented scales vary well for the values of $n = 1\, to\, 6$, the obtained reduction in total processing time has an almost linear relationship with $n$ for this range(see Figure 4.11).

### 4.3.6 Efficiency ($\varepsilon$)

This gives the fraction (per unit) for which the computing node(s) is/are actually processing (performing computational tasks). For a single processing node, there are no overheads associated with communication and tasks distribution, all the time is spent processing tasks, its efficiency is thus theoretically $1$ $(100\%)$. As $n$ is increased, the mentioned overheads increase, thus reducing the efficiency of the system as a whole. Depending on the nature of the problem and the way the parallel algorithm is designed, the efficiency of the system may deteriorate slowly or rapidly as $n$ is increased. The efficiency of a parallel system is defined as

$$\varepsilon = \frac{T_1}{T_n \times n}(PerUnit) \tag{4.6}$$

where $T_1$ is defined as the time taken for the whole process using $1$ processor, $T_n$ the time taken by using multiple processors and $n$ is as earlier defined.
substituting for the speed-up factor defined as $s = \frac{T_1}{T_n}$,

$$\varepsilon = \frac{s}{n} \tag{4.7}$$

Figure 4.13 shows the plot of the calculated efficiency of the implemented program as $n$ is scaled between $1\, and\, 6$. The efficiency is seen to remain close to $1$ (Per-Unit) for $n = 6$ ($\varepsilon = 97.84\%\, for\, n = 6$). The recorded values of $\varepsilon$ are acceptable for most purposes.

### 4.3.7 Summary of Timing Results

The required speed-up, scalability and the recorded efficiency of a parallel system are all inter-linked. The upper limit of the speed-up is usually set by the number of processing nodes $n$, super-linear speed-up (where speed-up > $n$) exists in theory but is usually a result of a suboptimal sequential algorithm implementation [49]. An increase in $n$ results in an increase in speed-up, with the maximum speed-up for a particular algorithm as earlier described by Amdahl/Gustafson-Barsis laws. In theory, $n$ may be continually scaled upwards without any limit in the hope that a particular speed-up will be achieved. However the addition of every extra processing node without a commensurate increase in speed-up results in a decrease in the overall efficiency of the system.

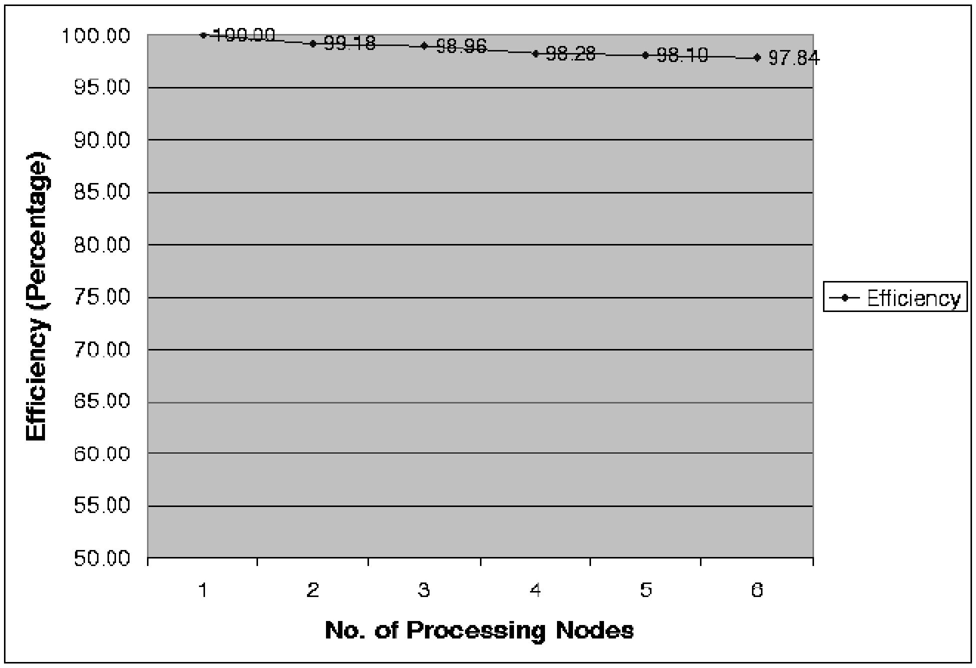


Figure 4.13: Plot of Overall Efficiency of the Parallel system as the number of processing nodes is scaled

To determine the optimum configuration for a specific parallel system, especially as regards the number of processing nodes to be employed, the minimum acceptable efficiency for the system may be set. $n$ is then scaled upwards and the speed-up factor $S$ is recorded for each value of $n$. $\varepsilon$ is then calculated for each case using Equation 4.7. $n$ is further scaled up until the minimum allowable efficiency for the parallel system is reached.

Efficiency is usually used as the limiting factor rather than the obtained speed-up, this is so taking into consideration the theoretical maximum speed-up for any parallel system as stated by the Amdahl's/Gustafson-Barsis laws. If for example, the minimum required efficiency for the implemented algorithm in this project is set at $98\%$ , then the maximum value of $n$ is $5$ ( $n = 6$ results in an efficiency of $97.84\%$ which is less than the minimum required ) ( see Figure 4.13 ). $n = 5$ results in a speed-up of $4.91$ . If the required speed-up is $7$ for example, the minimum overall efficiency requirement has to be reduced in order to allow for further upward scaling of $n$ in the hope of obtaining higher speed-up.

### 4.3.8 Extrapolating

A mathematical exercise is performed here to give an indication of the upper limit of scaling $n$ up and still maintain good overall efficiency of the parallel system in the implemented algorithm in this project. The two main parts of the process are the time taken for centre offset calculation (serial computation) and the tie-points offsets computation (parallelized). Assuming all other portions of the process are negligible, the time taken for centre offset calculation is approximately $t_s$ and the time for tie-points offsets calculation is $nt_p$. As $n$ is increased for the same task, $t_p$ reduces (see Equation 4.3). Since $t_s \, and \, t_p$ form a ratio adding up to $1$ (per-unit), a reduction in the parallelized fraction $t_p$ results in a relative increase in the serial fraction $t_s$. Referring to Equation 4.5 , increase in $t_s$ results in reduction in the speed-up $S$ and the overall efficiency of the system ( Equation 4.7 ).
If the time taken for computing $1$ tie-point offset is $t$ and there are $k$ tie-point offsets to be calculated in all:

Total time to calculate all tie-point offsets $= k \times t = t_p$ for single processor

Time taken for $k$ Tie-point offset computations by by $n$ processing nodes= $\frac{k \times t}{n} = t_p$ for multiple processors

As $n$ increases for multiple processing nodes, $t_p$ reduces i.e. $t_s = 1 - t_p$ increases thus reducing the obtainable speed-up and the overall efficiency of the system.

In other words, $n$ can be scaled up and the system will continue to perform relatively efficiently for as long as the time taken by each processing node to compute their allocated tie-points offsets in parallel is much greater than the time which was taken for the serial computation of the centre offset ($t_p >> t_s$) .The system becomes much less efficient when the number of processing nodes employed is so large that the time taken by each processing node to compute its allocated tie-points offsets is comparable to the time taken to compute the centre offset. This however only occurs for very large values of $n$ eg. $n = 100$ as the typical number of tie-point offset calculations for SAR images is about $1000$ . Dividing this amongst $50$ nodes will result in each processing node calculating 20 tie-point offsets which still results in a relatively low $t_s : t_p$ ratio, and a relatively good overall efficiency of the system.

## 4.4 Summary

The accuracy of the registration is tested qualitatively and quantitatively, the obtained results compare well to that obtained using the GAMMA software, a commercial software whose registration results are used as a bench mark.

Timing results of implementing the parallel algorithm on a variable number of multicomputers ( $n = 1\,to\,6$ ) are shown and discussed. Specific attention is paid to the time spent on I/O and system overheads, communication, time lost due to imbalance of tasks distributed amongst computing nodes, and the time taken for the serial and parallel parts of the computation. Tasks which contribute to these four (4) major portions of the process are identified, trends are highlighted and explained where they exist. Relationships between these and the evaluation parameters for the parallel system - speed-up,scalability and overall efficiency are established. The parallel program implementation scaled-up well as an almost linear relationship exists between the the number of processing nodes and the recorded speed-up for all the values of $n$ used. The overall efficiency of the system for the same range of $n$ was also high. ( lowest recorded efficiency $= 97.84\%$ for $n = 6$ ).

# Chapter 5

# Conclusion

## 5.1 Summary of Work done and Results obtained

An image registration algorithm with registration accuracy good enough for SAR images post-processing, especially interferometry was investigated. An algorithm with underlying principles that utilizes the phase information in the SAR images backscatter is designed, this is based on theoretical principles and subsequent practical test. The information resulting from this is used to further model and and correct typical distortions in SAR images.

The basic principles as identified are developed into a parallel algorithm capable of running on multiple computers. Distribution of the tasks in the registration process is done in the parallel program design to ensure efficient use of the available computing resources.

The designed algorithm is implemented in C programming language on systems with the Linux operating system installed. PVM is the middle-ware used for communication between the different computing nodes and other parallel computing constructs.

Regions of tandem SAR images of Cape Town are used as test images for the implemented algorithm. Registration accuracy obtained using the implemented algorithm in this project is compared to that registered using the GAMMA software, a commercial software with image registration functions. This is the bench mark for the accuracy of registration.

Qualitative and quantitative results show that the implemented algorithm is at least as accurate as the GAMMA software in registering the test images.

Timing results are recorded as the number of processing nodes are scaled between $1 \; and \; 6$ .The implemented parallel algorithm scaled well with an almost linear relationship between the number of processing nodes for the range of processing nodes tested, the efficiency for all these cases was also high with the lowest efficiency recorded when the number of processing nodes is $6$ (as expected) being $97.84\%$. Theoretical extrapolation showed that the upper bound of the number of processing nodes to be employed and still maintain relatively high efficiency is reached as the time taken for each processing node to compute its required number of tie-point offsets in parallel is in the order of the time taken for the serial operation of computing the centre offset.

The implemented algorithm results in registration accuracy adequate for the purpose of SAR post-processing, especially interferometry. Parallelization of the algorithm is very efficient for all the numbers of processing nodes tested. The problem is “embarrassingly parallel”.

## 5.2 Further Work

The implemented algorithm is tested on a tightly coupled cluster of parallel computers. The nature of the designed algorithm however gives the possibility of implementation on distributed computers. The numerous Tie-point offset computations can be done on partially available computing nodes - A central master computing node may send out the Tie-points data to the nodes as they become available, and gather the information from them for further processing. This may be investigated, not necessarily for better performance than is obtained on clustered computing nodes, but for working in the background to harness wasted CPU cycles.

It is noted in this work that most of the time spent for the registration process is for the Tie-point offset computation. This primarily involves $2 - dimensional$ Fourier transforms. The FFTW routine (`www.fftw.org`) is employed in this project. If a more efficient routine is investigated and employed, it will result in marked improvement in the total time spent for the registration process.

In the designed algorithm, the warping polynomial used is derived from global information all over the images to be registered and is also applied globally for warping the slave image to the reference. This may not be accurate, especially for air-borne SAR images with local distortions. It may be possible to divide the image into regions, compute warping polynomials for each region, and then warp separately. This will pose great challenges for parallelization considerations, especially edge effect considerations when arranging the different regions back into the complete warped image.

# Bibliography

[1] G.M. Amdahl. Validity of the single-processor approach to achieving large scale computing capabilites. In *AFIPS Conference proceedings*, volume 30, pages 483–485, 1967.

[2] R.A. Baggs, D.E. Tamir, and T. Lam. Image Registration Using Length Code Algorithm . In *Proceedings of IEEE 1996, Southeastcon '96*, pages 257–260, 1996.

[3] R. Bamler and Ramon Hanssen. Decorrelation Induced by Interpolation Errors InSAR Processing . In *IGARSS'97*, pages 1–3, 1997.

[4] R. Bamler and P. Hartl. Sythetic Aperture Radar Interferometry . *German Aerospace Research Establishment (DLR)*, 14(4):1–54, August 1998.

[5] Lisa G. Brown. A Survey of Image Registration Tehcniques . *ACM Computing Surveys*, 24(4):325–376, December 1992.

[6] P. Chalermwat. *High Performance Automatic Image Registration for Remote Sensing* . PhD thesis, George Mason University, Fairfax,Virginia, 1999.

[7] P. Chalermwat, A. El-Ghazawi Tarek, and Le Moigne Jacqueline. Wavelet-based Image Registration on Parallel computers . In *Super Computing '97*, pages 1–11, 1997.

[8] A. Downton. Generalised Approach to Parallelising Image Sequence Coding Algorithms. In *IEE Proc on Image Signal Processing*, volume 141, page 438, December 1994.

[9] A. Downton and D. Crookes. Parallel Architectures for Image Processing . *Electronics Communication Engineering Journal*, 10 (3):1–21, 1998.

[10] A. Downton, R. W. S. Tregidgo, and A. Cuhadar. Top-down Structured Parallelisation of Embedded Image Processing Applications. In *IEE Proc on Image Signal Processing*, volume 141, pages 431–, December 1994.

[11] N.R. Draper and H. Smith. *Applied Regression Analysis* . John Wiley and Sons, New York, Chichester, Brisbane,Toronto,Singapore, 1981.

[12] David Fernandes, Gunter Waller, and Joao Roberto Moreira. Registration of SAR Images Using the Chirp Scaling Algorithm . In *IGARSS '96*, volume 2, pages 799–801, Lincoln, Nebraska, 1996.

[13] L. M. G. Fonesca and C. S. Kenney. Control Point Assessment for Image Registration . In *Brazilian Symposium of Computer Graphics*, October 1998.

[14] G. Fornaro and G. Franceschetti. Image Registration in Interferometric SAR Processing . In *IEE Proceedings on Radar, Sonar and Navigation*, volume 142, pages 313–320, December 1996.

[15] A. Goshtasby, G. C. Stockman, and C V Page. A Region-Based Approach to Digital Image Registration with Subpixel Accuracy. *IEEE Trans on Geosc. and Remote Sensing*, GE-24(3):390–399, May 1986.

[16] A. Laurence Gray and Peter J. Farris-Manning. Repeat-Pass Interferometry with Airborne Synthetic Aperture Radar . *IEEE Trans on Geosc. and Remote Sensing*, 31(1):180–191, January 1993.

[17] S. Growe and R. Tonjes. A Knowledge Based Approach to Automatic Image Registration . In *ICIP'97*, Santa Barbara, U.S.A, October 1997.

[18] J. L. Gustafson. Reevaluating Amdahl's law. In *CACM*, volume 31, pages 532–533, 1988.

[19] P. Hartl, K.H. Thiel, X. Wu, and Y. Xia. Practical Application of SAR-Interferometry; Experiences made by The Institute of Navigation. In *Proceedings Second ERS-1 Symposium*, pages 717–722, Via Galileo Galilei, 1-00044 Frascati Italy, October 1993. ESA Publications Division.

[20] J. Homer and I.D. Longstaff. Minimising the Co-Registration Window Size for SAR Image Pairs . *IEE Proceedings on Radar, Sonar and Navigation*, 142(6), December 1995.

[21] Li Hui, B.S. Manjunath, and K. Mitra. A Contour-Based Approach to Multisensor Image Registration . *IEEE Trans on Image Processing*, 4(3):320–334, March 1995.

[22] Ton J. and A.K. Jain. Registering Landsat images by point matching . *IEEE Trans on Geosc. and Remote Sensing*, pages 642–651, September 1989.

[23] R. John Jensen. *Introductory Digital Image Processing - A Remote Sensing Perspective*. Prentice Hall, Englewood Cliffs, NJ 07632, 2 edition, 1996.

[24] R. C. Keys. Cubic Convolution Interpolation for Digital Image Processing . *IEEE Trans on Acoust., Speech, Signal Processing*, ASSP-29(6):1153–1160, December 1981.

[25] S. P. Kim and W.Y. Su. Subpixel Accuracy Image Registration by Spectrum Cancellation . In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93 IEEE International Conference*, volume 5, pages 153–156, 1993.

[26] Daumas Marc and Evripidou Paraskevas. Parallel Implementations of the Selection Problem: A Case Study . *Computer Science Dept., University of Cyprus*, 1999.

[27] C. Herbordt Martin, Cravy Jade, Sam Renoy, Kidwai Owais, and Lin Calvin. A System for Evaluating Performance and Cost of SIMD Array Designs. *Journal of Parallel and Distributed Computing*, 60:217–246, 2000.

[28] L. Barrett Martin and H. Wagner Clifford. *C and Unix* . John Wiley and Sons, New York, Chichester, Brisbane,Toronto,Singapore, 1996.

[29] Didier Massonnet, Thierry Rabaute, and Didier Massonnet. Radar Interferometry: Limits and Potential . *IEEE Trans on Geosc. and Remote Sensing*, 31(1):455–464, March 1993.

[30] Al-Mouhamed Mayez and Najjari Homan. Adaptive Scheduling of Computations and Communications on Distributed-Memory Systems. *Journal of Parallel and Distributed Computing*, 60:716–738, 2000.

[31] J. Le Moigne. Towards a Parallel Registration of Multiple Resolution Remote Sensing Data. *Goddard Space Flight Center, NASA*, pages 1011–1013, 1995.

[32] J. Le Moigne. Parallel Registration of Multi-Sensor Remotely Sensed Imagery Using Wavelet Coefficients. *Goddard Space Flight Center, NASA*, 2001.

[33] M.G. Montonya, C. Gil, and I. Garcia. Parallel thinning algorithms on multicomputers:experimental study on load balancing. *Concurrency: Practice and Experience*, 12:327–340, 2000.

[34] K. S. Rao and Y. S. Rao. Introduction to SAR Interferometry and Applications . *Indian Institute of Technology*, pages 2–11, January 1999.

[35] A. John Richards. *Remote Sensing Digital Image Analysis:An Introduction* . Springer, 1985.

[36] J.L. Roda, C. Rodriguez, D.G. Morales, and F. Almedia. Predicting the execution time of message passing models . *Concurrency: Practice and Experience*, 11(9):461–477, 1999.

[37] E. Rodriguez and J. M. Martin. Theory and Design of Interferometric Synthetic Aperture Radars . In *IEE Proceedings on Radar, Sonar and Navigation*, volume 139, pages 147–159, April 1992.

[38] Paul A. Rosen. Synthetic Aperture Radar Interferometry . In *Proceedings of the IEEE*, volume 88, pages 333–382, March 2000.

[39] M.Figueira Silvia and Berman Francine. A Slowdown Model for Applications Executing on Time-Shared Clusters of Workstations. *IEEE Trans on Parallel and Distributed Systems*, 12(6):653–, June 2001.

[40] Bischof Stefan, Ebner Ralf, and Erlebach Thomas. Parallel Load Balancing for Problems with Good Bisectors . *Journal of Parallel and Distributed Computing*, 60:1047–1073, 2000.

[41] Erich Stromaier, J. Jack Dongarra, W. Hans Meuer, and D. Horst Simon. The market place of high performance computing. *Parallel Computing*, 25:1517–1544, 1999.

[42] V.S. Sunderam and G.A. Geist. Heterogeneous parallel and distributed computing . *Parallel Computing*, pages 1699–1721, 1999.

[43] Thierry Hoja Toutin, E. Hoeppner, A. Remond, and Christine King. GCPs Selection from Multi-Source Data Over Mountainous Topography . In *IGARSS'98*, pages 2339–2341, Seattle, USA, July 1998.

[44] A. J. Wilkinson. Development of an Interferometric SAR Processing Facility Phase 1: Theoretical Principles and Algorithms. Technical report, UCT Radar Remote Sensing Group, March 1996.

[45] A. J. Wilkinson. Development of an Interferometric SAR Processing Facility Phase 2: Implementation of an Image Registration Algorithm. Technical report, UCT Radar Remote Sensing Group, March 1996.

[46] A. J. Wilkinson. Development of an Interferometric SAR Processing Facility Phase 3: Formation of Interferograms. Technical report, UCT Radar Remote Sensing Group, March 1996.

[47] A. J. Wilkinson. Development of an Interferometric SAR Processing Facility Phase 4: Implementation of a Phase Unwrapping Algorithm. Technical report, UCT Radar Remote Sensing Group, March 1996.

[48] A. J. Wilkinson. Development of an Interferometric SAR Processing Facility Phase 5: Implementation of an Algorithm to Produce Height Images. Technical report, UCT Radar Remote Sensing Group, March 1996.

[49] Barry Wilkinson and Michael Allen. *Techniques and Applications using Networked Workstations and Parallel Computers*. Prentice Hall, Englewood Cliffs, NJ 07632, 1 edition, 1999.

[50] C. Wimmer, R. Siegmund, M. Schwabisch, and Joao Roberto Moreira. Generation of High Precision DEMs of the Wadden Sea with Airborne Interferometric SAR. *IEEE Trans on Geosc. and Remote Sensing*, 38(5):2234–2245, September 2000.

[51] Yifeng Zhou, H. Leung, and Patrick C. Yip. An Exact Maximum Likelihood Registration Algorithm for Data Fusion. *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, 45(6):1560–1572, June 1997.

[52] Zhang Zuxun, Zhang Jianqing, Liao Mingsheng, and Zhang Li. Automatic Registration of Multi-Source Imagery based on Global Image Matching. *Photogrammetric Engineering and Remote Sensing*, 66(5):625–629, May 2000.

\appendix

# Appendix

## C Code Functional Layout

### Master Tasks

- **int CheckAndRegisterHosts(char **hosts);**
- *Checks and registers computing nodes in parallel Machine*
  - Function arguments-array of host names
  - Returns int(Number_of_hosts)

- **int ExtractCenter(float* master_center_ptr,float* slave_center_ptr);**
- *Extracts and stores Image centres in Arrays*
  - Function argument-float pointers to master and slave image centres
  - Returns int(confirmation)

- **int Compute_Centre_Offset(float* master_center_ptr, float* slave_center_ptr, int* array_dim);**
- *Centre Offset Calculation*
  - Function arguments-
    - Float pointers to master and slave image centres
    - 2-element integer array of image centre dimensions.
  - Result- sets value of 2-element array(Range and Azimuth Offset)

- **int Partition_Images(int no_of_hosts);**
- *Divides equally amongst Hosts*
  - Function arguments-int(Number of hosts)
  - Results-sets value of partition_dimensions_dimensions - 3-element array(Number of rows per session, Number of sessions, Number of residual rows to be divided in last session)

- **int SpawnSlaves_TiePointOffsetCompute();**
- *Starts Tie-point computation tasks on slaves, sends necessary information-file offset from beginning of image file to start reading from, number of rows to read,unique-id of slave process.*
  - Results-
    - Confirmation of started slave processes with necessary information
    - Resulting Tie-point offsets calculation results from slave processes.

## Slave Tasks

- **int ExtractSubImageAndShift(int nhosts,int starting_point, int sent_rows, int* center_offset);**
- *Extracts allocated portion of images in slave and master images and shifts slave image by earlier calculated centre offset.*
  - Function arguments-
    - nhosts-unique identity for spawned slave
    - starting-point-point to start reading in slave and master images for specific portion of image to calculate tie-points offsets for.
    - sent_rows-number of rows to be loaded into memory from master and slave images for processing, sent by spawning process.
    - centre_offset-pointer to 2-element integer array containing earlier calculated centre offset.

  - Result-
    - Portion of image to be processed loaded into slave machines memory
    - Shifting of slave image by centre_offset

- **int DefineTiePoint(int tiepoints,int sent_rows);**
- *Defines positions for Tie-point offsets calculation on slave and master images.*
  - Function arguments-
    - Number of tie-points offsets for the number of sent_rows
    - sent_rows
  - Results-sets value of integer array of tie-point offsets positions.

- **int CalculateTiePointOffset(int* TiePointPosition_ptr,int count);**
- *Does Tie-point offset calculation for all defined tie-point positions.*
  - Function arguments-
    - TiePointPosition-integer pointer to array of defined tie-point positions
    - Count-Number of tie-point offsets calculation to be done
  - Results-sets value of $3 \times count$-element array, Tie-point position, azimuth offset, range offset.

## Master Task

- **int ReadAndPrintResult(int count);**
- *Reads and prints tie-points offsets results from all slaves.*
  - Function argument-integer count, number of elements in tie-point offsets file
  - Results-Tie-point offsets results printed to screen.
- **int CalculateWarpingCoefficients(int count);**
  - Function argument-integer count, number of elements in tie-point offsets file
  - Results-warping polynomial coefficients
- **int SpawnSlaves_warp(int no_of_hosts);**
- *Spawns slaves to warp slave image based on warping coefficients, each slave is sent information similar to that sent in the function Spawnslaves, the slaves warp their allocated portion of the slave images, signify to the spawning process when done. The spawning machine assembles the warped images appropriately.*
  - Function arguments-Number of slave machines
  - Results-Assembled warped slave image.

## Slave Task

- **Warp_Main()**
- *Slave process warps allocated portion of slave image and sends it to spawning process. Contains function for Interpolating values when necessary.*
- Results- Warped version of allocated slave image.

## Program Input Parameters

Master_File ->

Slave_File ->

Number_of_rows ->

Number_of_Columns ->

Centre_Size_Azimuth ->

Centre_Size_Range ->

Number_of_Tie_Points_Range ->

Number_of_Tie_Points_Azimuth ->

List_of_Hosts ->

Output_Directory ->

## Program Outputs

Tie-Points Offsets (Tie_points_offset.txt)

Warping Polynomials (Warping_Polynomials.txt)

Registered Slave Image (warped_slave_image.slc)

## Relevant Test Image Parameters

### Master/Reference Image

| | | |
|---|---|---|
| title: | 05715 | |
| date: | 1996 05 24 | |
| raw_data_start_time: | 08 38 31.309 | |
| channel/mode: | VV | |
| platform_altitude: | 797392.5146 | m |
| terrain_height: | 0.0000 | m |
| range_pixel_spacing: | 7.905919 | m |
| range_resolution: | 13.495 | m |
| offset_to_first_echo_to_process: | 0 | echoes |
| echoes_to_process: | 27200 | echoes |
| range_offset: | 0 | samples |
| range_looks: | 1 | looks |
| azimuth_looks: | 1 | looks |
| azimuth_offset: | -0.28314 | s |
| azimuth_pixel_spacing: | 4.057588 | m |
| azimuth_resolution: | 5.072 | m |
| range_pixels: | 4912 | samples |
| azimuth_pixels: | 26139 | lines |

### Slave Image

| | |
|---|---|
| title: | 25388 |
| date: | 1996 05 23 |
| raw_data_start_time: | 8 38 31.918 |
| channel/mode: | VV |

```
platform_altitude:            797372.9890   m
terrain_height:                    0.0000   m
range_pixel_spacing:             7.905919   m
range_resolution:                  13.495   m
offset_to_first_echo_to_process:        0   echoes
echoes_to_process:                  27200   echoes
range_offset:                           0   samples
range_fft_size:                      8192
range_looks:                            1   looks
azimuth_looks:                          1   looks
azimuth_offset:                  -0.17541   s
azimuth_pixel_spacing:           4.057587   m
azimuth_resolution:                 5.072   m
range_pixels:                        4912   samples
azimuth_pixels:                     26139   lines
```