# An On-line Velocity Flow Profiling System using Electrical Resistance Tomography

Timothy Matthew Long

A dissertation submitted to the Department of Electrical Engineering,
University of Cape Town, in fulfillment of the requirements
for the degree of Master of Science in Engineering.

Cape Town, August 2006

# 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.

Signed by candidate

Signature of Author .....

Cape Town

31 August 2006

# Abstract

A system to display velocity profiles of flows through the CPUT FPRC pipe test loop on-line has been developed. An electrical resistance tomography instrument developed at the University of Cape Town is used to generate conductivity profiles through cross sections of the flow in a Ø57mm pipe at two points, separated by either 50 or 100mm. The two concentration profiles are cross correlated on a pixel-wise basis to produce a velocity profile.

The software was developed using C++ and employs a highly modular structure that allows different image reconstruction and cross correlation algorithms to be implemented without substantial changes to the rest of the application.

Results showing the speed performance of the system are presented as well as typical velocity profiles from a sliding bed flow regime.

To my family

# Acknowledgements

My sincere thanks to all the people who had input to this thesis is various ways.

Specifically, thanks to:

- Dr. Andrew Wilkinson, my supervisor, for all his help at many stages and pointing me in the right direction when I needed it.
- Mr. Bill Randall for being, essentially, a second supervisor and providing very many useful comments.
- Mr. Andrew Sutherland at the Flow Process Research Center at the Cape Peninsula University of Technology for the use of the test facility and the data that he spent many hours capturing.
- Mr. Granville De La Cruz for his diverting company after many hours in front of the computer.

# Contents

# List of Figures

# List of Tables

# List of Symbols

| | | |
|---|---|---|
| $\rho$ | — | Resistivity |
| $\rho^{-1}$ | — | Conductivity |
| $\sigma$ | — | Standard deviation |
| $f$ | — | Frame capture rate |
| $\Delta t$ | — | Frame capture period |
| $t^{dp}$ | — | Time to do data processing |
| $t^{r}$ | — | Time to do image reconstruction calculation |
| $t^{cc}$ | — | Time to do cross correlation calculation |
| $\mathbf{M}$ | — | Precomputed reconstruction matrix |
| $V_0$ | — | Boundary voltages |
| $D_0$ | — | Boundary current density |
| $v$ | — | Voltage measurement set |
| $\mathbf{Y}$ | — | Finite element interpolation matrix |
| $J$ | — | Jacobian matrix |
| $f(\rho)$ | — | Forward operator that maps resistivity to measured voltages |
| $\mathbf{Q}$ | — | Regularisation matrix |
| $L$ | — | Electrode plane separation |
| $cc$ | — | Correlation coefficient |

# Nomenclature

**2D**—Two dimensional

**3D**—Three dimensional

**CPUT**—Cape Peninsula University of Technology

**DAC**—Digital to Analog Converter

**DAQ**—Data acquisition device

**ERT**—Electrical Resistance Tomography

**FFT**—Fast Fourier Transform

**FPRC**—Flow Process Research Center

**GPU**—Graphics Processing Unit

**USB**—Universal Serial Bus

# Chapter 1

# Introduction

## 1.1 The Flow Measurement Problem

In many industrial processes, coarse materials are transported using a fluid carrier. The way in which this mixture of fluid and solids, called a slurry, flows through the transport pipes can vary substantially. For example, in some cases the solids move in a bed along the bottom of the pipe, while in others the slurry is completely mixed and the solids are suspended in the fluid. It is useful to be able to predict a slurry's flow pattern as it has ramifications on the mass flow rate as well as wear on the transport pipes.

The flows of high concentration slurries with wide particle distributions are particularly not well understood [56]. They often become non-Newtonian[1] and operate at turbulent transition velocities or even completely in laminar flow. In this case, the coarse solids are transported as a sliding bed, even though the carrier fluid yield stress is high enough to statically support them [51, 14]. Although pseudo-homogeneous axi-symmetric and stratified plane symmetrical models may predict gross transport characteristics, such as pressure drop versus flow rate for these flows, they cannot be used to differentiate between different flow patterns in the pipe (turbulent or laminar flow for example). Better models, based on physical parameters of the real flow mechanisms, are needed.

To develop these models, and to attain a physical description of the flows, a pipe test flow facility has been built by the Flow Process Research Center at the Cape Peninsula University of Technology (CPUT). The facility has a comprehensive set of measurement instrumentation installed on the pipe, including an electrical resistance tomography (ERT)

[1]The viscosity of a non-Newtonian fluid changes with the applied strain rate, in contrast to that of a Newtonian fluid which does not.

instrument, to determine the concentration profile of the flow.

Electrical resistance tomography has also been used by various groups [23, 16, 19] to measure the velocity profiles of flows. The instrument installed on the pipe test flow rig, developed by the University of Cape Town, is capable of much higher data capture rates, approaching 1000 frames per second, than the instruments used by these groups and should be able to measure a much wider range of flow velocities.

The high speed nature of the UCT instrument has led to the development of a real time visualisation system [65, 36] by the author, giving the user almost instant feedback. To build on this, an on-line system to measure velocity profiles in the CPUT facility, has been developed.

This thesis describes the development of this on-line velocity profile measurement system using electrical resistance tomography.

## 1.2 Electrical Resistance Tomography

Electrical resistance tomography [70] is an imaging technique that can be used to determine the internal structure of a object or vessel non-intrusively. Typically, electrodes are positioned around the periphery of the region to be imaged and an electrical field is generated in that region. This is most commonly done by driving a sinusoidal current between two of the electrodes. Figure 1.1 is a plot of the equi-potential voltage lines in the electrical field resulting from a typical current injection. The peripheral voltages resulting from this electrical field are then measured using the remaining electrodes. By repeating this process, selecting successive adjacent eletrode pairs to drive current through, a data frame is built up that can be used to reconstruct an image or resistivity map of the measurement region.

The non-intrusive nature of ERT makes it ideal for use in the investigation of the aforementioned flows because the high solid concentration of the fluids prevents the use of other measurement instruments such as laser doppler (the fluids are often opaque) or intrusive instruments.

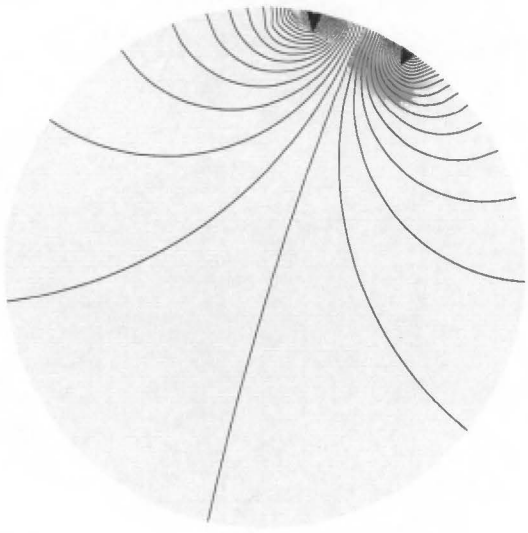


Figure 1.1: Equi-potential voltage lines in the electric field generated by a current injection

## 1.3 The UCT Instrument

The UCT instrument [66] differs from this typical setup in that it uses a bi-directional DC current pulse, instead of injecting a sinusoidal current. The switched DC pulse is effectively a single cycle of a square wave. This waveform allows the instrument to operate at high speeds, capturing data at up to 1000 measurement frames per second. Chapter 2 describes the instrument in more detail.

## 1.4 An On-line Velocity Profiling System

By placing an electrode ring around the test pipe, the UCT instrument can capture a resistivity profile in the the plane of the electrodes. The resistivity profile corresponds to the concentration profile of the flow. By placing a second electrode ring further down the pipe, the concentration profiles from both rings can be cross correlated to generate a velocity profile. This is possible if that the electrode rings are sufficiently close together, the image reconstruction is done accurately enough and that sufficient structure exists in the flow.

So an on-line velocity profiling system must capture ERT data from two electrode rings. This data must then be processed by an image reconstruction algorithm to generate cross

sectional images, or concentration profiles, of the flow in the plane of the electrode rings. The two images are then cross correlated on a pixel by pixel basis to produce an estimation of the velocity profile. The on-line system must be able to display this profile while the next set of ERT data is captured.

Thus the components of the system are:

1. data capture,
2. image reconstruction,
3. cross correlation and,
4. visualisation

This thesis describes these components and their integration into a single system. It should be noted that the data capture aspect is not examined in great detail as it has been well described elsewhere [54, 53, 67].

## 1.5 Structure of this Thesis

The thesis is structured as follows.

Chapter 2 describes the hardware used in the system. This includes the UCT ERT instrument and the pipe test facility.

Chapter 3 deals with reconstruction algorithms and the components that generally make up these algorithms. Both iterative and one step methods are described using the Newton-Raphson algorithm as an example.

Chapter 4 discusses the use of cross correlation in the estimation of velocity profiles. Two methods of performing cross correlation are presented, the more conventional time domain technique, and a so called 'fast correlation' method that involves multiplication of the Fourier transforms of the two signals being correlated.

The next two chapters describe the development of the software for the on-line system.

Chapter 5 investigates the implementation of two algorithms for image reconstruction and two methods of cross correlation described in Chapter 4. This chapter also examines the performance of these algorithms.

The second software chapter, Chapter 6, describes the integration of the reconstruction and cross correlation algorithms into the measurement application. This involves detailing the structure of program including the class layout. Various visualisations for viewing data at various stages of the sytem are also presented in this chapter.

Chapter 7 is the results chapter which examines the performance of the overall system. This is followed by the final chapter, Chapter 8 which presents conclusions and suggests some recommendations for future work.

# Chapter 2

# System Hardware

The velocity profiling system is based on a high speed data capture instrument connected to the pipe test rig. This chapter briefly describes the ERT instrument developed at the University of Cape Town and the pipe test rig at the Cape Peninsula University of Technology.

## 2.1 UCT Tomography Hardware

A high speed tomography instrument has been developed at the University of Cape Town that can capture data at up to 1000 frames per second. Although detailed description of this instrument is beyond the scope of the thesis, an overview of the fundamental aspects is given in this section. For more information about the instrument, the reader is referred to [65, 66, 54, 52, 13].

The UCT instrument uses the four electrode measurement model where current is injected between two electrodes and measured between another two electrodes as shown in Figure 2.1. The injected current is a bi-directional pulsed constant current which means that the waveform of the driving current is a single cycle of a square wave. The resultant potential difference between the two high impedance measurement electrodes is also a square wave (in a predominantly conductive medium). The use of pulsed direct current in this manner is somewhat unconventional as other instruments, such as those developed at the University of Leeds [61], the University of Manchester [71] and those developed by Industrial Tomography Systems, use a sinusoidal current, but this technique does afford some benefits. By sampling the resultant waveform during the stable parts of the square wave, there is no need for complex AC circuit analysis or demodulation [13]. This means

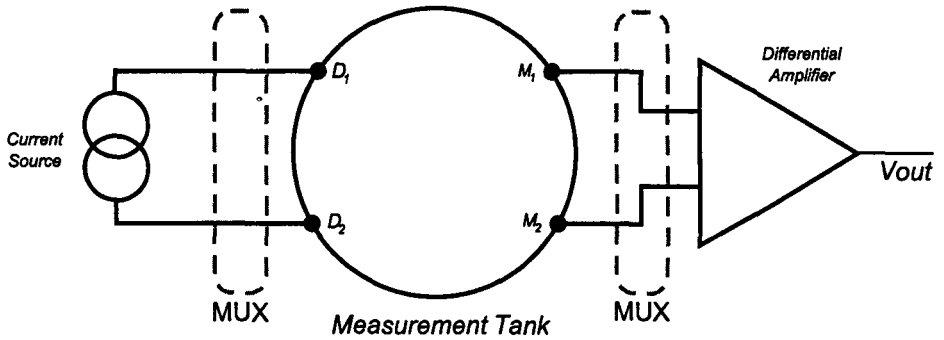



Figure 2.1: Four electrode measurement scheme

reduced circuit complexity and filters with relatively long time constants are unnecessary which makes high speed operation possible. A disadvantage of using direct current is that polarization effects at the electrodes start to affect measurements, however this is eliminated by using the bi-directional technique with pulses of equal amplitude and length, so the net injected current is zero.

To achieve further speed gains, the UCT instrument is fitted with sixteen parallel amplifier channels, one for each electrode pair in a 16 electrode ring.[1] Each amplifier channel has dual sample and hold circuitry for the positive and negative pulses meaning that the potential difference between each pair of electrodes can be captured simultaneously for each current injection. Analogue to digital conversion of the sample and hold values is done by an Eagle Technology USB-30B device which can sample at 400kHz on sixteen channels. This device also facilitates data transfer to the controlling PC via a USB interface and sets, via digital I/O lines, the operating mode of the embedded micro-controller. This micro-controller, a Motorola HC12 device, controls the current source and sample and hold circuitry, ensuring the consistent pulse widths required to make the electrolytic or polarisation effects of the DC current negligible. The micro-controller also sets the multiplexer channels as specified by a measurement sequence table supplied by the user.

The bank of sixteen amplifier channels is connected to the electrode system through sixteen 8 way multiplexers allowing the amplifiers to be connected to any one of eight electrode layers. A schematic of the UCT instrument configuration is shown in Figure 2.3.

The advantages that the pulsed current technique and the sixteen parallel amplifier channels provide mean that the UCT instrument is able to capture data on a single layer at up to 1000 frames per second. As the measurement circuitry is multiplexed to the electrode layers, this rate is halved when using two layers, and reduced by eight when all the electrode layers are in operation. Although the instrument can operate at 1000 frames

[1] It should be noted that the UCT instrument is not restricted to sensing rings of 16 electrodes.

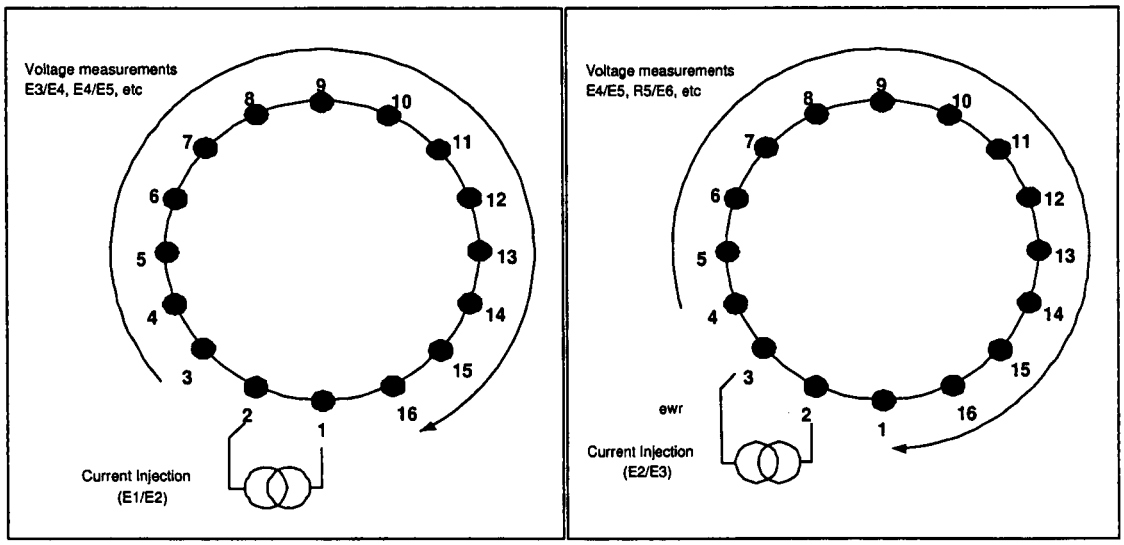



Figure 2.2: Measurement sequence

per second for a minimal independent set of 104 measurements, modifications made to the data capture sequence to improve the instrument's noise performance, including increasing the data set size to 208 measurements, have reduced this data rate. The highest capture rate used for the results presented in this thesis is 566 frames per second.

The measurement sequence used is an independent layer adjacent pair strategy, where current injections and voltage measurements are made on the same layer. For each current injection, on an adjacent pair of electrodes, 16 voltage measurements are made between neighbouring electrodes. Three of these measurements are then discarded, corresponding to the electrode pairs which share an electrode with the injection pair to result in 13 measurements per injection. Current injection is done on all sixteen electrode pairs resulting in a measurement set of 208 samples for each layer.

The UCT instrument is controlled by a data capture and visualisation application developed by the author. This software allows real time display of a number of representations of the conductivity profiles from multiple planes as well as information regarding the measured voltages. Figure 2.7 is a screen shot of this application.

## 2.2 CPUT Pipe Test Loop Facility

The pipe test loop facility at the Flow Process Research Center (FPRC) at CPUT consists of a $4.5m^3$ mixing tank with a 18.5kW variable speed dual disc mixer and baffles, centrifugal slurry pumps and 5 horizontal test loops of varying diameter including two

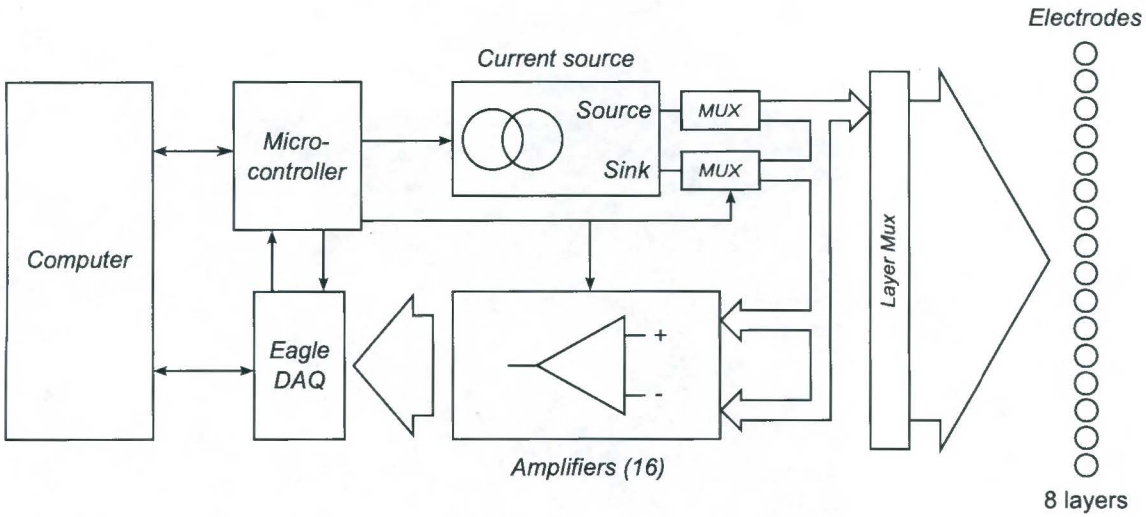



Figure 2.3: High speed data configuration

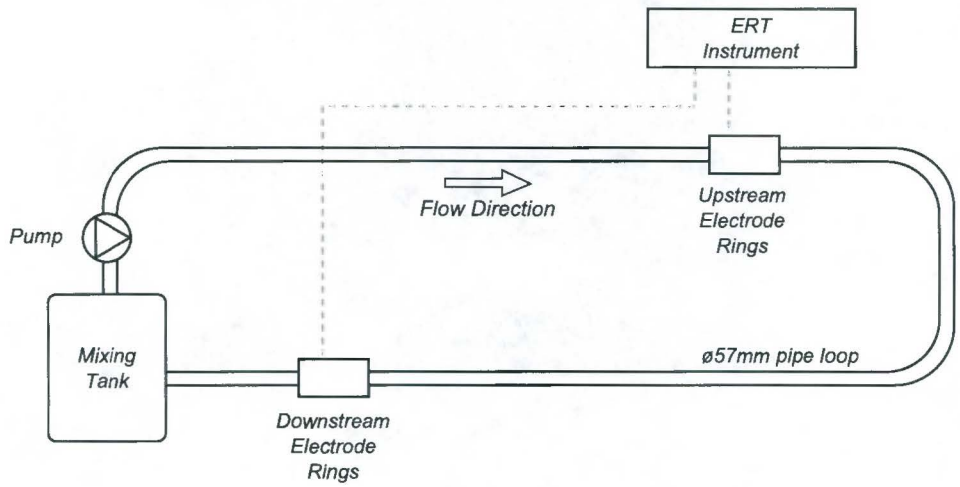


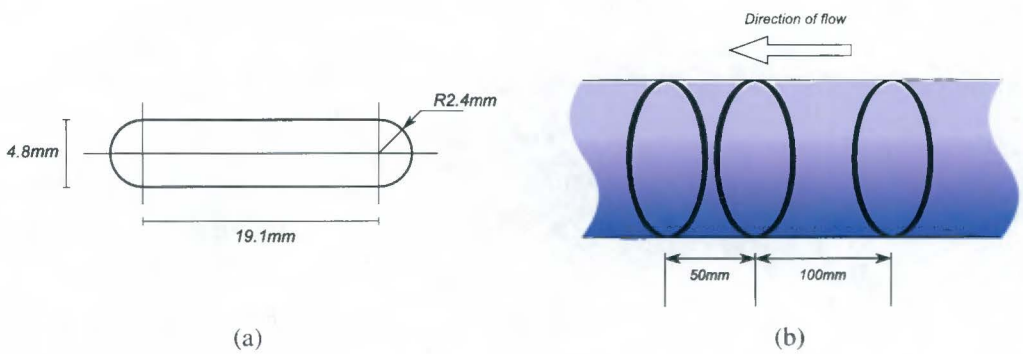

Figure 2.4: FPRC pipe test loop layout

Ø$57mm$ ID pipes. A schematic of the relevant subsection of the loop is shown in Figure 2.4. Sutherland *et al* describe the complete facility comprehensively in [56].

Mounted on the Ø$57mm$ horizontal pipe are six ERT sensing rings. These rings are positioned in groups of three, one group on the upstream leg, the other on the downstream leg. The first two rings in each group are axially separated by $100mm$, while the last two are $50mm$ apart. (See Figure 2.6(b)). A sensing ring is comprised of sixteen stainless steel electrodes equally spaced around the circumference of the pipe. The electrode dimensions are shown in figure 2.6(a). Sensing plane spacings of $50mm$, $100mm$ and $150mm$ can be selected by reading from the appropriate pairs of electrode rings.

The velocity of the flow is expected to range from 0.1 to 6m/s.

Figure 2.5: FPRC pipe test loop




(a) (b)

Figure 2.6: (a) Electrode dimensions (b) Electrode ring spacing

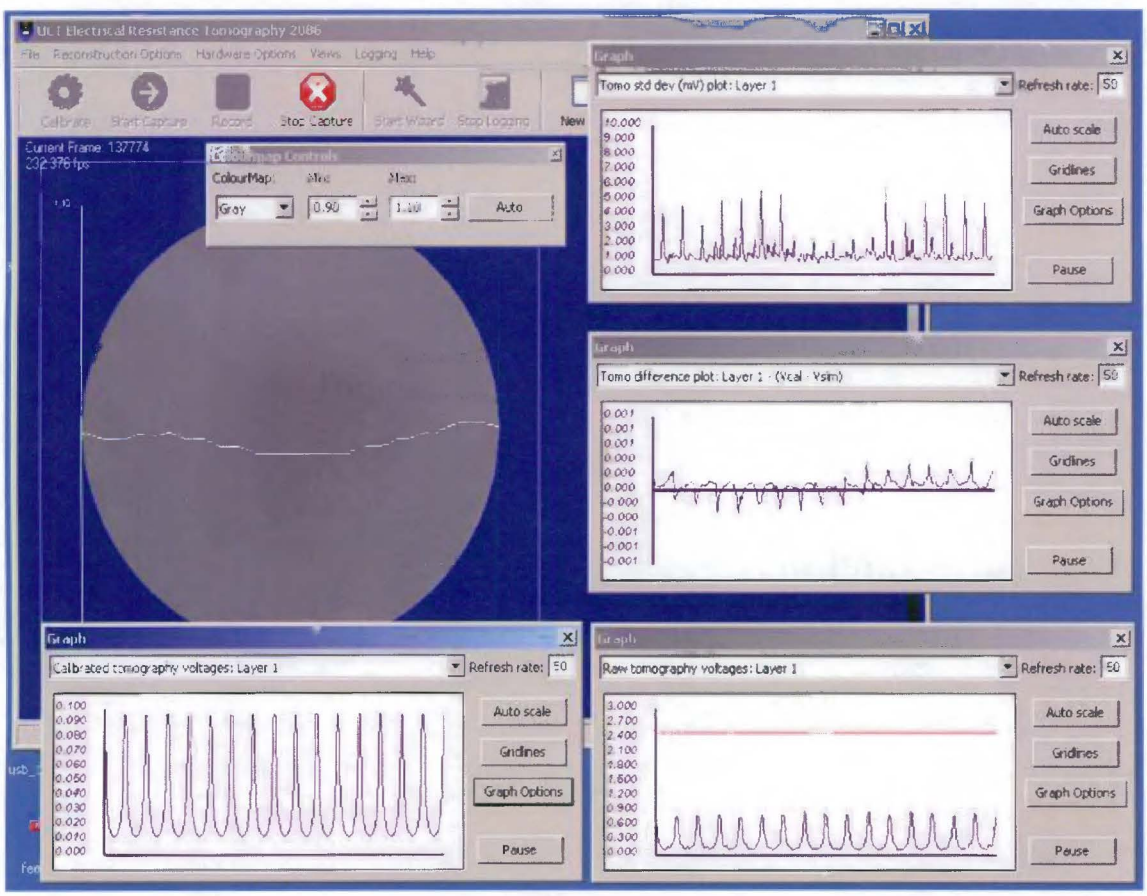



Figure 2.7: Data capture application

# Chapter 3

# Reconstruction Algorithms

Once data has been captured by the ERT instrument it must be manipulated to generate a resistivity map of the imaged region. This chapter introduces the concepts involved in the reconstruction and highlights work done by researchers in developing reconstruction algorithms. The Newton-Raphson algorithm is presented as an example of the one such algorithm.

## 3.1 Introduction

As mentioned previously, tomographic images are generated of the resistivity distribution by injecting current into the region to be imaged. The resulting electrical field is determined by Poisson's equation[1]

$$\nabla \cdot \rho^{-1} \Delta V = f \tag{3.1}$$

with boundary conditions

$$V = V_0 \; on \; \partial A \tag{3.2}$$

$$\rho^{-1} \partial V / \partial n = D_0 \; on \; \partial A \tag{3.3}$$

where $\rho$, $V$ and $f$ are the resistivity, voltages and distribution of current sources respectively and $V_0$ and $D_0$ are the voltage and current density at the boundary. However, because there are no current sources within the region, Equation 3.1 can be reduced to

$$\nabla \cdot \rho^{-1} \Delta V = 0 \tag{3.4}$$

[1] These equations are taken from [27].

Solving Equation 3.4, i.e. finding $\rho^{-1}$ given $V_0$ and $D_0$, the boundary voltages and currents is often called the 'inverse problem'. The 'forward problem', on the other hand, is the converse to the 'inverse problem', and involves finding the voltage distribution $V_0$ for a given conductivity distribution, $\rho^{-1}$.

Typically one is not able to solve the inverse problem directly, instead, the forward problem is solved iteratively, changing $\rho$ in some fashion until $V_0$ is sufficiently close to the measured voltages. As no analytical solution exists to the forward problem for a given resistivity distribution, a numerical method is used. Although some researchers have used finite difference and finite volume techniques, the most common technique is to use the finite element method [33] to model the physical situation as accurately as possible.

## 3.2 The Forward Problem

Briefly, the finite element method involves discretising the region to be imaged into a finite number of elements and assuming that the resistivity or conductivity of the region is constant across the element. This allows Equation 3.1 to be expressed as a linear system of equations, namely

$$\mathbf{Y}v = c \tag{3.5}$$

where $\mathbf{Y}$ is the interpolation matrix that maps the voltages, $v$, at the nodes of the mesh across the region for a given injected current, $c$. For a more detailed description of the finite element technique, see [35].

The way in which the region is discretised is very important. A finer mesh, with more elements, results in a more accurate description of the resistivity distribution, but it also increases the size of $\mathbf{Y}$, placing higher demands on the computer system solving for $v$. A compromise is for the mesh to be fine in areas where the resistivity changes rapidly and coarse in places where there is little change. Molinari *et al* have developed an adaptive mesh refinement algorithm which matches the local scale of the mesh to the scale of the image structures in two and three dimensions [40, 41]. Further optimisation can be achieved because matrix $\mathbf{Y}$ is sparse and the position of its non zero elements is determined by the numbering of the elements in the mesh. Pinheiro and Dickin [47] describe a number of techniques for renumbering the mesh to take advantage of sparse matrix methods. Figure 3.1 shows a typical three dimensional mesh.

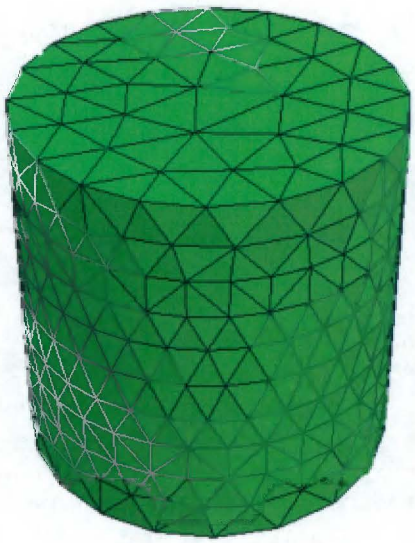


Figure 3.1: An example of a three dimensional mesh

## 3.3 The Inverse Problem

Finding the inverse solution is typically carried out by minimising an error function, often with a Newton-type method. Consider,

$$E(\rho) = \frac{1}{2}(f(\rho) - V_0)^2 \tag{3.6}$$

where $f$ is a forward operator that maps the resistivity to the measured voltages. To minimize this function, one could take the derivative and equate it to zero,

$$E'(\rho) = [f'(\rho)]^T(f(\rho) - V_0) = 0 \tag{3.7}$$

where $f'(\rho)$ is the Jacobian matrix and is made up of the partial derivatives of $f$, the calculated voltages, with respect to the resistivity distribution (i.e. the result of the forward problem). The Jacobian matrix, will be referred to as $J$ hereafter.

### 3.3.1 Finding the Jacobian

The importance of an efficient forward model becomes apparent at this stage. Although optimised methods for calculating the Jacobian have been developed [49], the basic algorithm involves determining how the voltages at the electrodes vary for a slight change in the resistivity distribution. More specifically, the conductivity of each element in the finite element mesh is varied slightly in turn, solving the forward problem each time to

build up the Jacobian matrix. This means that for a mesh of $N$ elements, the forward problem must be solved $N$ times.

### 3.3.2 The Newton-Raphson Algorithm

In the Newton-Raphson algorithm, the linear terms of the first Taylor polynomial around $\rho_0$ are taken, to get

$$E'(\rho) = E'(\rho_0) + E''(\rho - \rho_0)\Delta\rho \tag{3.8}$$

where $E''$ is the Hessian matrix which can be expressed as

$$E''(\rho) = J^T \cdot J \tag{3.9}$$

according to [27].

Substituting Equations 3.7 and 3.9 into Equation 3.8 leads to

$$\Delta\rho = -\frac{J^T(f(\rho_0) - V_0)}{J^T \cdot J} \tag{3.10}$$

The resistivity guess, $\rho$, must be adjusted by $\Delta\rho$, the resistivity update, to reduce the error function $E(\rho)$ at each iteration.

This process can be applied iteratively until some stopping criteria is met, this is typically once the output error falls below measurement noise [33]. Figure 3.2 shows the steps involved in the Newton-Raphson algorithm.

### 3.3.3 Regularization

The inverse problem is ill-posed [57], so to some extent the obvious approach is to regularize [33]. This is often done by modifying the error function, Equation 3.6, by some factor that penalises rapidly changing resistivities, such as

$$E(\rho) = \frac{1}{2}(f(\rho) - V_0)^2 + \lambda Q(\rho) \tag{3.11}$$

Where $Q$ is the regularisation operator and $\lambda$ is a number known as the regularisation parameter that determines the degree of regularisation. The resistivity update function then becomes [48]

$$\Delta\rho = -\frac{J^T(f(\rho_0) - V_0)}{J^T \cdot J + \lambda Q} \tag{3.12}$$

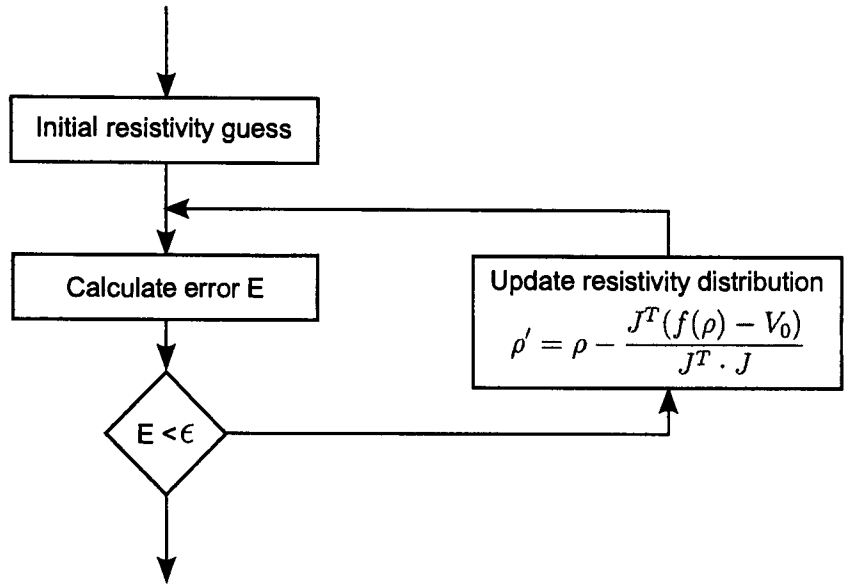



Figure 3.2: The Newton-Raphson algorithm successively updates an initial resistivity estimate until the error function falls below some value $\epsilon$.

This is an example of Tikhonov regularization [57]. Tikhonov regularization is widely used and many different regularisation operators have been suggested, Pinheiro et al [48], for example, describe a smoothness constrained inversion method to generate $Q$. Borsic [9] gives a detailed examination of regularisation techniques for electrical tomography.

### 3.3.4 Other Inverse Solutions

The Newton-Raphson method is just one of many multi-step algorithms that have been developed to solve the inverse problem. Wang [59], for example, has produced impressive results using conjugate gradient methods. More algorithms and techniques are described in reviews by Hua and Woo [27] and Lionheart [33]. Borcea [8], gives a detailed review of mathematical results for the inverse electrical tomography problem.

As an aid to researchers in this area, the EIDORS suite of Matlab functions has been developed [49, 1] which includes implementations of different algorithms to allow new techniques to be compared with existing methods.

### 3.3.5 One Step Algorithms

Although multi-step algorithms are widely used because of their accuracy, they are generally slow because of the amount of computation required to invert the large matrices

used in the forward and inverse problems. In applications where speed of calculation is important, one step or linear methods can be used.

One of the first techniques developed was that of linear back projection along equipotential lines by Barber and Brown, briefly described in [3]. This method has generally lost popularity, in favour of variations of the one step Newton technique, or NOSER algorithm [11]. Le Hyaric and Pidcock [29] have extended the 2D NOSER algorithm to three dimensions.

The idea behind the NOSER algorithm is that first step of the Newton-Raphson algorithm can be expressed as a linear problem. Consider the first step

$$\rho = \rho_0 - \frac{J^T[f(\rho_0) - V_0]}{J^T \cdot J} \tag{3.13}$$

this can be written as

$$\rho = \rho_0 - \frac{J^T f(\rho_0)}{J^T \cdot J} + \frac{J^T V_0}{J^T \cdot J} \tag{3.14}$$

or

$$\rho = \mathrm{s} + \mathrm{M}(V_0) \tag{3.15}$$

where both c and M can be determined before the first measurement is taken or 'precalculated'. This technique is very fast because a one reconstruction requires a simple matrix multiplication once the s and M have been calculated. The trade off for speed however, is accuracy, and the NOSER algorithm is only accurate for resistivity distributions that are close the the initial guess ($\rho_0$).

## 3.4 Difference Imaging and Calibration

The difference between quantitative and qualitative imaging should be mentioned here. In quantitative imaging, the aim is to determine the actual resistivity of the imaged region. To achieve this, a multi-step algorithm should be used that converges on the real resistivity. However, in many cases the aim is simply to find a relative distribution or identify a non-conducting object. In these instances, although the initial resistivity guess may differ significantly from the actual resistivity, one step algorithms can still be used to give a qualitative result. Often difference imaging is used for this.

In difference imaging the final image is the difference between two reconstruction results. The first reconstruction is done on data captured while the measurement tank contains an homogeneous resistivity distribution, and the second is done on data captured when it

contains the disturbance to be imaged.

Difference imaging can be viewed as a type of calibration as artifacts introduced by the algorithm and the measurement system (excluding noise) are effectively removed by the difference. Another type of calibration is that employed by Wilkinson *et al* [66] which scales the measured voltages by a set of factors that would match a data set taken from a homogeneous tank exactly with that predicted by the forward problem.

## 3.5 Noise Performance of One Step Algorithm

It is generally difficult to predict the relationship between measurement noise and image quality for any reconstruction algorithm, especially in those cases that employ iterative techniques. However, in the case of a one step algorithm, with a relatively simple regularisation technique, each pixel value is just a weighted sum of the measured voltages. The value of a particular pixel $\rho_i$, is

$$\rho_i = \sum_{n=1}^{N} m_{n,i} v_n + constant \tag{3.16}$$

Assuming independent noise on all measurements, the variance of the image pixel value is a weighted sum of the variances of the measurements

$$\sigma_{\rho_i}^2 = \sum_{n=1}^{N} m_{n,i}^2 \sigma_n^2 \tag{3.17}$$

where $\sigma_n$ is the standard deviation of the noise. Assuming further that it, $\sigma_n$, is the same for each measurement

$$\sigma_{\rho_i} = \sigma_n \sqrt{\sum_{n=1}^{N} m_{n,i}^2} \tag{3.18}$$

The standard deviation of the image pixels is proportional to the standard deviation of the measurement noise. Note however that the degree of proportionality varies for each element.

This variation of standard deviation across the mesh is shown in Figure 3.3. In the case of part (a), Gaussian noise with a peak of 1V and a standard deviation of 2mV was added to a simulated voltage set (which corresponds to a uniform resistivity or conductivity solution) and an image reconstructed. This was repeated 1000 times and the standard deviations of each reconstructed pixel are shown in part (a). Figure 3.3(b) shows a similar result using

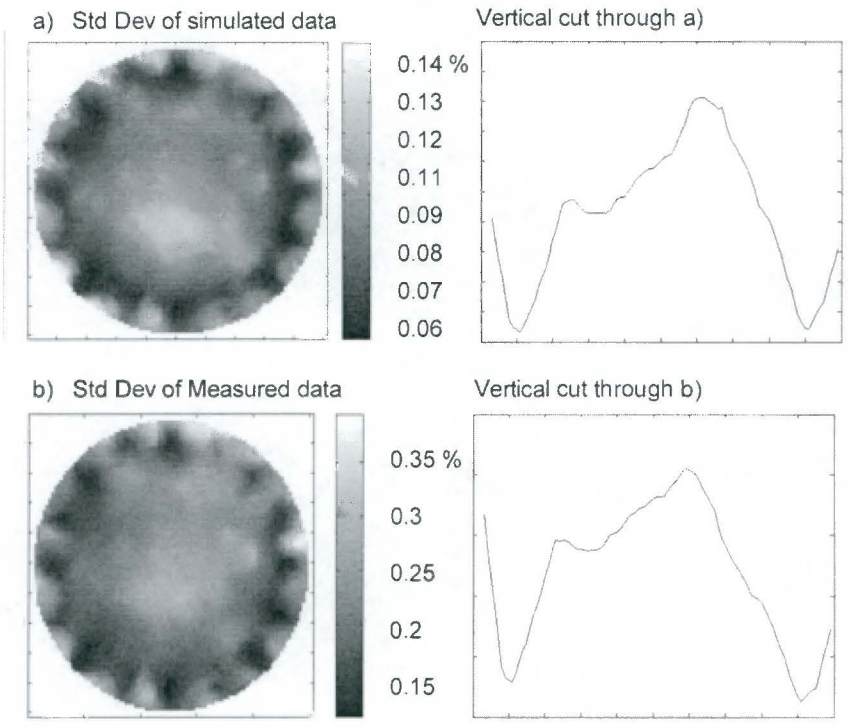



Figure 3.3: Images showing the resistivity standard deviation as a percentage of the conductivity of the solution, for cases of (a) with simulated voltage data and (b) real measured data. On the right hand side are shown vertical cross sections through the images.

real measurements taken under laboratory conditions. The asymmetry of the variation of the standard deviation across the mesh is due to the irregularity of the underlying finite element mesh. The specific reconstruction algorithm used to generate these images is described in Chapter 5.2.1.

# Chapter 4

# Velocity Estimation via Cross Correlation

This chapter describes cross correlation and its application in determining flow velocity using reconstructed tomographic images.

## 4.1 Introduction

Cross correlation has long been used as a technique for determining the velocity of a physical process [4]. Leitner [32] investigated the use of cross correlation in monitoring slurry flows and found that it was effective for this application.

Initial correlation applications often used a polar cross correlation technique implemented in hardware using analog components [69] due to its simplicity and limitations in computing power. However, advances in processing power and memory in computers have now enabled full cross correlation to be performed in software [16]. Yang & Beck [69] developed an early example of such a system for measuring flow in a pipeline. Armed with this extra computing power, researchers have been able to perform cross correlation on the results of process tomography reconstructions with considerable success. The reader is directed to [37, 16, 28, 7, 61, 23] for examples of some of these systems.

## 4.2 Principle of Cross Correlation in Velocity Measurement

Using cross correlation for velocity measurement is based on the principle that a particular disturbance in the flow produces a unique signature which can be identified in the flow further along the pipe. The time taken, say $\tau_d$, for this signature to move from one sensing plane to a second plane gives an indication of the velocity of the flow. Specifically,

$$v = L/\tau_d \tag{4.1}$$

where $L$ is the separation of the planes.

Consider the resistivity of a particular point, $i$, in the first sensing plane, $x_i(t')$; if the sensing planes are sufficiently close together, one can consider a corresponding point, $y_i(t')$, in the second plane, simply to be a delayed version of the first, $y_i(t') \approx x_i(t' - \tau_d)$, or alternatively, $x_i(t) \approx y_i(t + \tau_d)$ where $t' = t - \tau_d$. The difference between $x_i(t)$ and $y_i(t+\tau_d)$ arises from the tumbling motion of the particles. Thus, the relationship between $x_i(t)$ and $y_i(t)$ can be modeled as,

$$x_i(t) = y_i(t + \tau_d) + n(t) \tag{4.2}$$

where $n(t)$ is the component arising from the turbulence.

By reconstructing images through the sensing planes at $f$ frames per second, $x(t)$ and $y(t)$ are effectively sampled at $f$, and problem becomes a discrete one. If $x_i[m]$ and $y_i[m]$ are the values of pixel $i$ in the reconstructed images corresponding to $x_i(t)$ and $y_i(t)$ respectively sampled at $f$, Equation 4.2's discrete equivalent becomes

$$x_i[m] = y_i[m + d_i] + n[m] \tag{4.3}$$

where $d_i$ is the delay in samples at pixel $i$ and $n[m]$ now also accounts for errors introduced due to measurement noise and image reconstruction artifacts. The relationship between $\tau_d$ and $d$ is given by

$$\tau_d = d\Delta t + s \tag{4.4}$$

where $s$ is the sampling error and $\Delta t = 1/f$. Note that $s$ is negligible as long as $\Delta t \ll \tau_d$ i.e. as long as the sampling rate is sufficiently high.

As mentioned above, finding the value of $\tau_d$, or more specifically, $d$, allows the determination of the flow velocity. This is done by correlating $x_i[m]$ with $y_i[m + p]$ for various

values of $p$. Correlation is the degree to which two quantities are linearly associated [63], and the correlation of two discrete variables, $a$ and $b$, is defined as

$$R_{ab} = \frac{1}{N} \sum_{m=0}^{N} a[m]b[m] \tag{4.5}$$

For a meaningful correlation result, normalisation is necessary. This is done by ensuring the $a$ and $b$ have zero mean and scaling by their standard deviations as shown in Equation 4.6.

$$cc_{ab} = \frac{\frac{1}{N}\sum_{m=0}^{N}(a[m] - \bar{a})(b[m] - \bar{b})}{\sigma_a \sigma_b} \tag{4.6}$$

where

$$\sigma_a = \sqrt{\frac{1}{N} \sum_{m=0}^{N} (a[m] - \bar{a})^2} \tag{4.7}$$

$$\sigma_b = \sqrt{\frac{1}{N} \sum_{m=0}^{N} (b[m] - \bar{b})^2} \tag{4.8}$$

The normalised correlation coefficient can vary over a range of -1 to 1. For two signals that are well correlated, $cc$ will tend toward 1, while for two uncorrelated signals, $cc$ will be close to zero. The case where $cc_{ab}$ is negative could arise if $a \approx -b$.

So the aim is thus to find the value of the offset $p$ which maximises

$$cc_{xy} = \frac{1}{N} \sum_{m=0}^{N} x_i[m] y_i[m+p] \qquad p = -M... - 1, 0, 1, ...M \tag{4.9}$$

where $x_i$ and $y_i$ have been normalised to have zero mean and have standard deviations of 1.

This equation is simply a cross correlation [69]. By finding the offset of the peak correlation coefficient in the cross correlation for pixel $i$, namely $d_i$, the velocity of the flow through that pixel can be found by combining Equations 4.1 and 4.4,

$$v_i = \frac{L}{d_i \Delta t} \tag{4.10}$$

### 4.2.1 Maximum Measurable Velocity

According to Equation 4.10, the maximum velocity that can be measured is when peak offset $d_i$ is 1 which corresponds to

$$V_{max} = \frac{L}{\Delta t} \tag{4.11}$$
$$= 0.05/283^{-1} \tag{4.12}$$
$$= 14.15ms^{-1} \tag{4.13}$$

This is significantly more than the expected maximum flow rate of $6ms^{-1}$. Equation 4.10 also suggests that the discrimination of the velocity profile is $\Delta t/2$. However, better discrimination can be achieved if interpolation is done between points in the correlation profile, for example, $cc_{xy}(p + 0.5)$ can be estimated by interpolating between $cc_{xy}(p)$ and $cc_{xy}(p+1)$. In fact, the maximum velocity limit mentioned is only valid if no interpolation is done.

## 4.3 Interpretation of the Correlation Coefficient

In the ideal case, the cross correlation calculation produces an easily identifiable peak, and one has a high degree of confidence that the correct offset has been selected. However, in practice, the cross correlation result is degraded by measurement noise, image resolution, artifacts in the image reconstruction and non-axial movement of particles in the pipe. These factors may lead to errors in the location of the peak and if this degradation is severe enough, the upstream and downstream signals will not correlate at all and no peak will be produced. Simply finding the position of maximum value of the cross correlation is insufficient, one must have some measure of the accuracy of the offset and the corresponding velocity.

Figure 4.1 shows the relationship between the correlation coefficient and the variation of the position of the peak using different window sizes. The window size is the number of samples used in the correlation. It was generated by adding increasing amounts of noise to two identical signals, cross correlating them, and noting the correlation coefficient and the position of the peak. For each noise level, this was repeated 100 times. The standard deviation of the peak positions was then plotted against the mean of the correlation coefficients for each noise level.

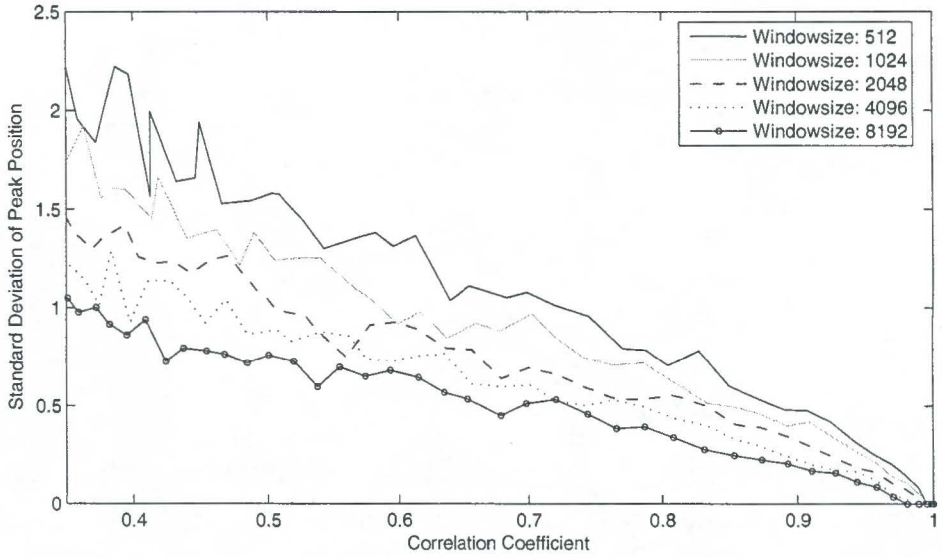



Figure 4.1: The relationship between the variation of the position of the peak and $cc_{xy}$

It is clear that a higher correlation coefficient yields a more accurate velocity estimation (based on the peak offset). Also, increasing the correlation window size reduces the variation of the peak position for a given value of $cc_{xy}$. This suggests that if the magnitude of the peak is low, the velocity estimation can be improved by increasing the window size.

In generating Figure 4.1 it is clear that increasing the measurement noise reduces the strength of correlation and thus $cc$. The relationship between the noise and $cc$ is examined below.

If $n_c(t)$ is taken as the variation of the flow, the two signals to be correlated can be represented as

$$x(t) = A + n_c(t) + n_x(t) \tag{4.14}$$

$$y(t) = A + n_c(t - \tau) + n_y(t) \tag{4.15}$$

in the presence of noise. Where $n_x(t)$ and $n_y(t)$ are additive independent noise with standard deviation of $\sigma_n$ and where $n_c(t)$ has zero mean and a standard deviation of $\sigma_c$. The correlation of these signals is

$$cc_{xy} = \frac{E\{(x - \bar{x})(y - \bar{y})\}}{\sigma_x \sigma_y} \tag{4.16}$$

$$= \frac{E\{(n_c + n_x)(n_c + n_y)\}}{\sigma_x \sigma_y} \tag{4.17}$$

$$= \frac{E\{(n_c^2 + n_c n_x + n_c n_y + n_x n_y)\}}{\sigma_x \sigma_y} \tag{4.18}$$

$$= \frac{E\{(n_c^2)\}}{\sigma_x \sigma_y} \tag{4.19}$$

$$\tag{4.20}$$

because the $n_c$ has zero mean and $\sigma_x^2 = \sigma_y^2 = \sigma_c^2 + \sigma_n^2$

$$cc_{xy} = \frac{E\{(n_c^2 - \bar{n_c})^2\}}{\sqrt{\sigma_c^2 + \sigma_n^2}\sqrt{\sigma_c^2 + \sigma_n^2}} \tag{4.21}$$

$$= \frac{\sigma_c^2}{\sigma_c^2 + \sigma_n^2} \tag{4.22}$$

Equation 4.22 indicates that should the standard deviation of the noise equal the standard deviation of the changes arising due to the flow, the best correlation coefficient one can expect is 0.5.

## 4.4 Fast Correlation

Cross correlation can also be performed by using the Fourier transform.

Consider the discrete Fourier transform of $f(k)$

$$F(n) = \sum_{k=0}^{N-1} f(k) e^{-j2\pi nk/N} \tag{4.23}$$

if $f(k) = R_{xy}(k)$,

$$F_n = \sum_{k=0}^{N-1} \left[ \sum_{m=0}^{N-1} x_i(m) y_i(m+k) \right] e^{-j2\pi nk/N} \tag{4.24}$$

Reversing the summation leads to,

$$F_n = \sum_{m=0}^{N-1} x_i(m) \left[ \sum_{k=0}^{N-1} y_i(m+k) e^{-j2\pi nk/N} \right] \tag{4.25}$$

setting $r = m + k$

$$F_n = \sum_{m=0}^{-1} x_i(m) \sum_{r=-m}^{N-1-m} y_i(r) e^{-j2\pi n(r-m)/N} \tag{4.26}$$

$$F_n = \sum_{m=0}^{N-1} x_i(m) e^{j2\pi nm/N} \sum_{r=-m}^{N-1-m} y_i(r) e^{-j2\pi nr/N} \tag{4.27}$$

For the case of circular convolution with window size $N$, $r = mod(r, N)$

$$F_n = \sum_{m=0}^{N-1} x_i(m) e^{j2\pi nm/N} \sum_{r=0}^{N-1} y_i(r) e^{-j2\pi nr/N} \tag{4.28}$$

Thus,

$$F_n = X_n^* Y_n \tag{4.29}$$

and

$$f = F^{-1}\{X_n^* Y_n\} \tag{4.30}$$

This result shows that it is possible to do cross correlation by multiplying the Fourier transforms of the two input signals. This is pertinent because it offers the potential for an alternate implementation, and considering the well known speed advantages of the FFT algorithm, this technique may be more suitable for an on-line system and is referred to as 'fast correlation' hereafter.

## 4.5 Best Correlated Pixel Method

The underlying assumption of the velocity estimation using cross correlation technique is that particles move in a straight line perpendicular to the sensing planes. To overcome this assumption, Mosorov *et al* [44] have suggested a 'best-correlated pixels' method. This method is based on the realisation that the signal from a pixel in the first sensing plane is sometimes more strongly correlated with a non-corresponding pixel in the second plane than the corresponding pixel, indicating that the particles are not always traveling parallel to each other.

This suggests that the cross correlation calculation should be repeated so that each pixel in the first plane is correlated not only with the corresponding pixel it the second plane, but also with neighbouring pixels. So the cross correlation calculation becomes

$$R_{x_i y_j}[p] = \frac{1}{N} \sum_{m=0}^{N} x_i[m] y_j[m+p] \tag{4.31}$$

where $j$ is the index a pixel in the neighbourhood of pixel $i$. Using this method also allows the flow *vectors* to be determined at each pixel providing greater information about the flow.

# Chapter 5

# Implementation

The elementary concepts involved in an on-line velocity profiling system have been described and a number of techniques for solving each have been introduced. This chapter identifies two algorithms for image reconstruction and two algorithms for cross correlation and describes their implementation. Factors influencing the speed of these implementations are also examined.

## 5.1 Overview

The steps for generating a velocity profile in the on-line system are:

1. Capture voltage data for both electrode planes,
2. Reconstruct two images from the voltage data,
3. Store pixel values of each image,
4. Cross correlate each pixel over a specified range,
5. Display correlation results as a velocity profile

This process is shown in diagrammatic form in Figure 5.1.

The time taken to generate a velocity profile is the sum of the time taken for each of these steps. From the software perspective, the data processing time, $t^{dp}$, is of most interest because it is determined by how the cross correlation and reconstruction algorithms are

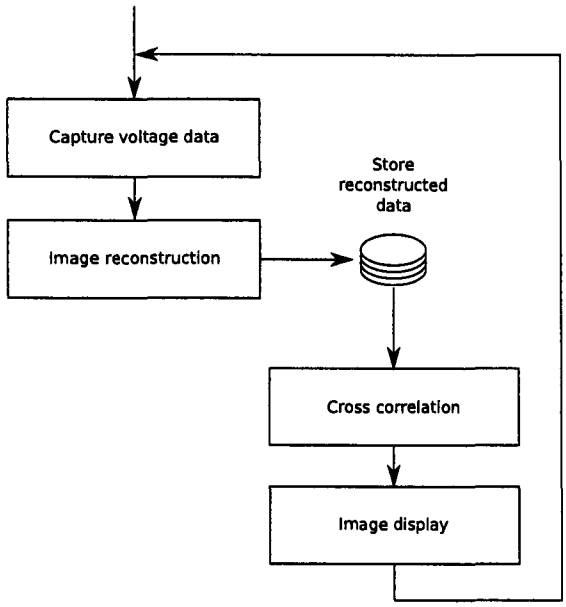



Figure 5.1: Generating a velocity profile

implemented. $t^{dp}$ can be expressed as the sum of the times taken for the reconstruction, $t^{r}$, and the cross correlation $t^{cc}$,

$$t^{dp} = t^{r} + t^{cc} \tag{5.1}$$

As the aim is to develop an on-line system, with performance as close to real time as possible, algorithms which reduce $t^{r}$ and $t^{cc}$ are preferred and the speed of implementation is paramount. However, due to the inevitable trade off, the speed is limited by accuracy requirements. Before describing the implemented algorithms, it is useful to examine what effect attempting to improve accuracy and resolution of the two parts has on the speed of the final result.

### 5.1.1 Implications of Resolution and Accuracy

#### Resolution of the Reconstructed Image

The spatial resolution of the final reconstructed conductivity image is affected by a number of factors; the reconstruction method and the number of independent measurements used in the reconstruction, the applied current and the system noise [64] for example. At the data processing stage, only the reconstruction method can be changed to influence the image resolution as the data has already been captured, so it will be the only factor examined here.

The resolution of the reconstruction method is ultimately limited by the number of independent measurements and the type of regularisation used because of the ill-posed nature of the inverse problem. However, because the finite element method assumes constant conductivity or resistivity across each element, in coarser meshes, the size of these elements play a much greater role in determining the resolution of the image. The number of resolution cells is effectively limited by the number of elements in the mesh. For this reason, only the mesh size is considered when determining the resolution of the image.

The relationship between mesh size and image reconstruction speed is determined by the reconstruction algorithm used and so it is difficult to examine here without specifying a particular algorithm. However, in the case of one step algorithms, where the resistivity is calculated by a simple matrix multiplication, one would expect the relationship to be linear.

The spatial resolution of the reconstructed image also affects the total time spent on cross correlation operations. The velocity profile is generated using each pixel, or element, in the image reconstruction. This means that the spatial resolution of the velocity profile is limited by the resolution of the reconstructed image, or in other words, the number of elements in the finite element mesh. So the time for the overall correlation calculation is essentially

$$t^{cc} = N t_i^{cc} \tag{5.2}$$

where $N$ is the number of elements in the mesh and $t_i^{cc}$ is time taken for the cross correlation of one pixel. Note that though it is possible to group pixels so the velocity profile is coarser than the reconstructed image, it is faster to simply use a coarser mesh in the reconstruction and avoid the extraneous processing at this step.

### Accuracy of Reconstructed Image

For the cross correlation algorithm to be successful, the pixel values from the downstream image must be a delayed version of that from the upstream image. If there is consistent correlation between a particular resistivity and the image it produces, a variation in the resistivity will produce the same variation in the upstream reconstructed image as it does in the downstream image, and the signals will correlate and produce a peak in the correlation profile regardless of their actual values.

What this implies is that the absolute accuracy of the reconstructed resistivities are not as important as they might be in other applications, as long as they are precise. This suggests a one step method or a difference imaging technique may be sufficient.

#### Resolution of the Correlation Result

The resolution of the correlation results, i.e. the temporal resolution of the time delay used to calculate the velocity profile, is limited by the data capture rate. However, velocity estimates at higher resolution can be made by interpolating between points in the correlation profile.

#### Accuracy of the Correlation Result

In Chapter 4 it was mentioned that the precision of the correlation result can be improved by increasing the correlation window size as shown in Figure 4.1. A larger window size increases confidence in the result but obviously slows any correlation calculations. It is worth mentioning again that regardless of the window size, using cross correlation for flow profiling assumes that the particles travel in straight lines between the measurement planes, i.e. that particle displacement in the plane of the cross section is significantly less that the spatial resolution of the resistivity reconstruction. The accuracy of the correlation result is only as good as this assumption.

As in the case of the resolution of the reconstruction algorithm, the effect of changing the window size has on $t_i^{cc}$, the time to calculate the correlation, is dependent on the particular algorithm used.

### 5.1.2 Choice of Algorithm

#### Reconstruction Algorithms

Due to the emphasis on speed, and the relaxed constraint on the accuracy of the reconstructed conductivities, a one step or linear algorithm is most appropriate for this problem.

Two algorithms were implemented, both based on the idea that s and M (Equation 3.15) can be pre-computed. In the first algorithm, a two dimensional finite element mesh was used, while the second uses a matrix calculated by the EIDORS package based on three dimensional geometry.

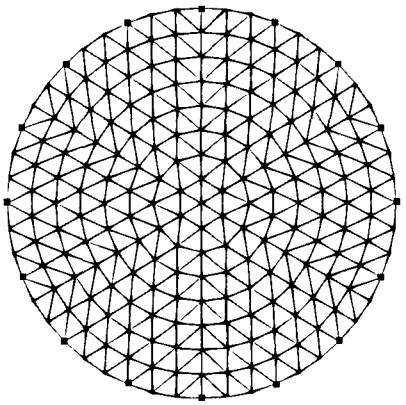


Figure 5.2: A 384 element, two dimensional mesh with 16 point electrodes

**Correlation Algorithms**

In the discussion about correlation, it was noted that it can be done in the time domain or by using an FFT-based method referred to as 'fast correlation'. Both these techniques were implemented and are described below.

In the interests of minimising time spent during the correlation stage, the 'best correlated pixels' method [44] was not implemented. The method requires that multiple cross correlations are done for each pixel which would result in the stage taking proportionally more time. The increase in calculation time would be by a factor equivalent to the number of pixels in the neighbourhood searched. Implementing the best correlated pixels method may be feasible if the data processing stages of the on-line program are fast enough. Further investigation into this is suggested.

## 5.2 Reconstruction Algorithms

### 5.2.1 2D Version

In the 2D version of the reconstruction algorithm, the finite element mesh is two dimensional and the electrodes are modeled as point sources. Figure 5.2 shows a typical mesh with 16 point electrodes and 384 elements.

**The forward problem**

The forward problem, namely solving $\mathbf{Y}v = c$, requires finding the inverse of the interpolation function matrix, $\mathbf{Y}$, which for meshes with many elements is a computationally expensive task. It is desirable therefore, to use techniques that reduce the amount of computation required.

The first thing that should be noted is that inverse is not actually required in itself; it is only used to find $\mathbf{v}$, thus performing a full Gauss-Jordan elimination is unnecessary. If $\mathbf{Y}$ can be converted to a triangular set, where finding the solution is quite trivial [50], calculating $\mathbf{v}$ would take less time. Consider an $\mathbf{LU}$ decomposition,

$$\mathbf{Y} = \mathbf{L} \cdot \mathbf{U} \tag{5.3}$$

where $\mathbf{L}$ is of the form

$$\begin{bmatrix} \alpha_{11} & 0 & 0 \\ \alpha_{21} & \alpha_{22} & 0 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{bmatrix}$$

and $\mathbf{U}$ is of the form.

$$\begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{13} \\ 0 & \beta_{22} & \beta_{23} \\ 0 & 0 & \beta_{33} \end{bmatrix}$$

Substituting into $\mathbf{Y}v = c$

$$\mathbf{c} = (\mathbf{L} \cdot \mathbf{U}) \cdot \mathbf{v} = \mathbf{L} \cdot (\mathbf{U} \cdot \mathbf{v}) \tag{5.4}$$

By splitting Equation 5.4 into two parts, the original problem is changed into two triangular set substitutions, namely

$$\mathbf{L} \cdot \mathbf{x} = \mathbf{c} \tag{5.5}$$

and

$$\mathbf{U} \cdot \mathbf{v} = \mathbf{x} \tag{5.6}$$

Equation 5.5 can be solved by *forward- substitution* and 5.6 by simple *back-substitution*. These substitutions are relatively simple to preform.

The algorithm used to implement the LU decomposition is taken from Numerical Recipes in C [50].

### The inverse problem

The Jacobian was generated in precisely the fashion described in 3.3.1, i.e. the forward problem solved for each small change in the resistivity.

A method similar to that described by Pinheiro *et al* [48], a smoothness constrained inversion was used to regularise the solution, in which M in Equation 3.15 becomes

$$\mathrm{M} = \frac{J^T}{J^T \cdot J + \lambda Q} \tag{5.7}$$

where $\lambda$ determines the degree of smoothing.

The regularisation operator, $Q$, was built based on the adjacency of each element. For each element in the mesh, all adjacent elements were identified and emphasised while others were penalised.

Finding M can also be done with an LU decomposition. Consider

$$\mathbf{F} \cdot \mathrm{M} = J^T \tag{5.8}$$

where F is the denominator of Equation 5.7. If matrix **F** is decomposed, M can be built column wise by doing the forward and back substitutions using columns of $J^T$.

### Calibration

As a relative resistivity distribution was deemed sufficient, a calibration technique was employed to improve the reconstructed image. This technique, already mentioned in Chapter 3.4 is described fully in [66].

It involves scaling the measured voltages by a set of factors that would force a measurement set from a homogeneous medium to equal that predicted by the forward model. Calibration in this manner reduces the erroneous influences of misaligned electrodes, varying amplifier gains and discrepancies between the forward model and the actual geometry.

Figure 5.3 shows a typical raw measurement set, the set of voltages predicted by the forward model (simulated measurements), the scaling factors and the measurement set after calibration. The difference between the calibrated measurements and the simulated measurements arises from a disturbance in the region being imaged. In this case it is caused by a non-conducting rod placed in a saline solution.

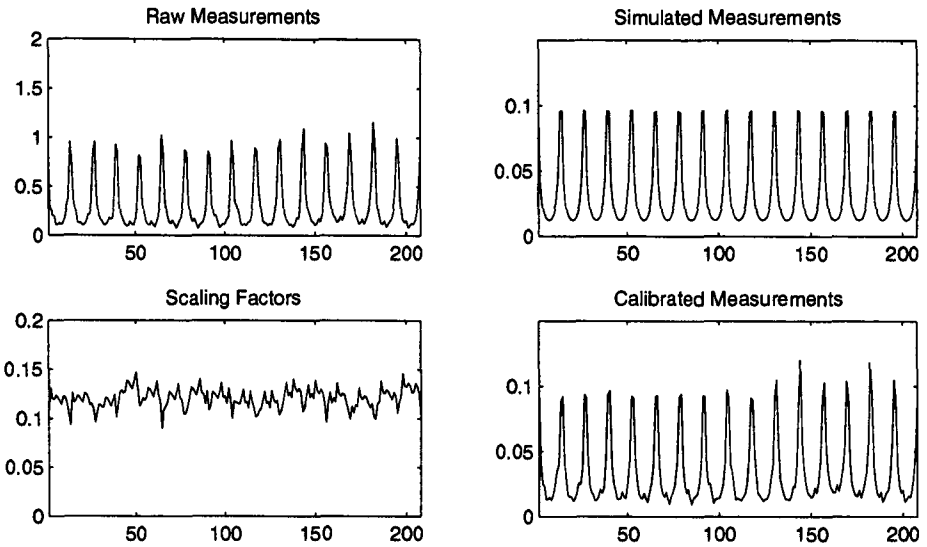



Figure 5.3: Calibration of measurement data

### Speed

After calibration, the resistivity of each pixel is calculated by

$$\rho_i = s_i - \sum_{j=1}^{M} M_{ij} V_j \tag{5.9}$$

where $V$ are the calibrated voltage measurements. $s$ and M are defined in Chapter 3.3.5.

What can be inferred from Equation 5.9 is that the complexity of the one step 2D algorithm is $O(n)$ in Big-O notation, where $n$ is the number of elements in the mesh. Increasing the size of the mesh linearly increases the calculation time as can be seen in Figure 5.4. For a 384 element mesh, the time to reconstruct a resistivity image is 0.923ms.

### Images

The results of the 2D image reconstruction with increasing mesh resolutions is shown in Figure 5.5. The reconstruction was done on data measured from a 350mm diameter tank containing a saline solution with a 50mm diameter non-conducting rod placed off center. The measurement sequence used is described in Chapter 2.1. The increase in resolution with mesh density is easily seen.

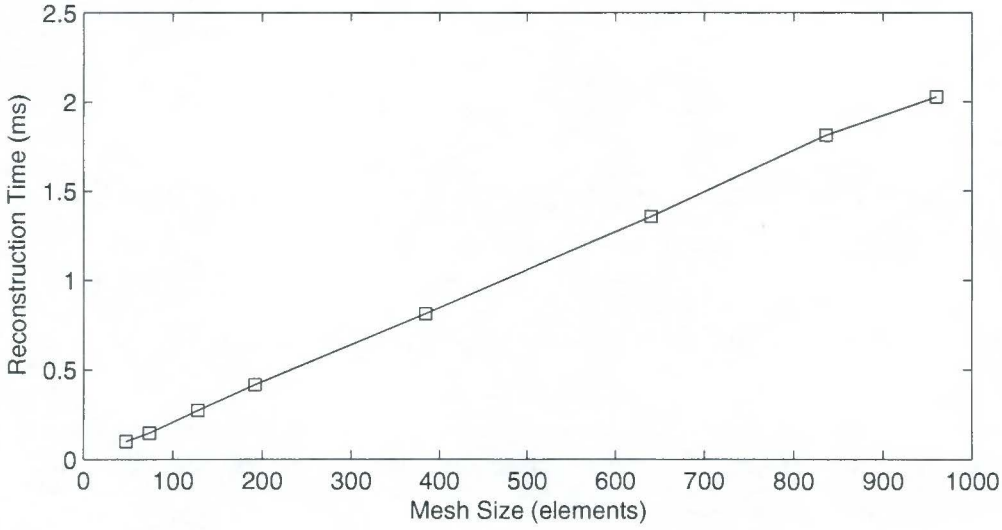
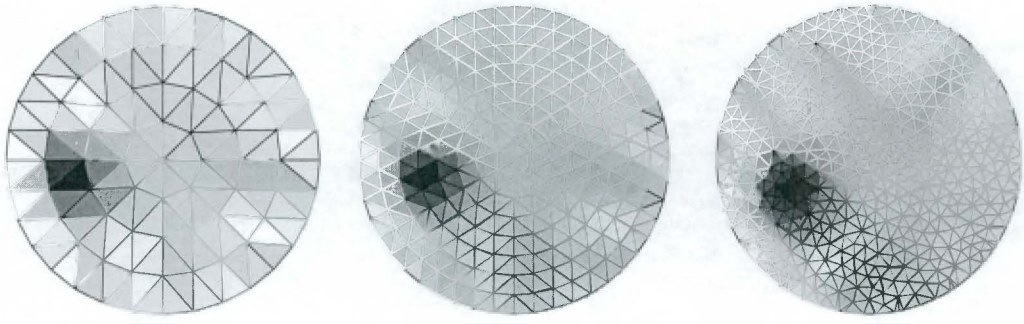



Figure 5.4: Image reconstruction speeds using a 2.67Ghz Intel Celeron PC with 500MB RAM



(a) (b) (c)

Figure 5.5: Results of a 2D image reconstruction with different meshes (a) 128 elements (b) 384 elements (c) 836 elements

### 5.2.2 3D Version

The 3D algorithm is based on the algorithm developed by Polydorides in the EIDORS package [49, 1] which is also a Newton-Raphson method. The resistivity update can be expressed as

$$\Delta\rho = \frac{J(\rho) \cdot (V - f(\rho))}{J(\rho)^T \cdot J(\rho) + \lambda L \cdot L} \tag{5.10}$$

The regularisation and Jacobian calculation techniques are described in [49] and will not be elucidated here. Note that this is similar in form to Equation 3.13 and can also be expressed as Equation 3.15, again allowing matrix M to be pre-computed. This is virtually identical to the 2D implementation, the only differences arising in the way that M is calculated.

To obtain M, a sample image reconstruction was done using the EIDORS package in Matlab and the appropriate matrix was saved to disk.

#### Calibration

Calibration of the data for this algorithm was done using difference imaging as described in Chapter 3. The relative resistivity at each pixel is taken as the difference between the result of the algorithm with data from a homogeneous medium and the result of the algorithm with data with the disturbance. This is better expressed as

$$\rho'_i = [c_i - \sum_{j=1}^{M} M_{ij} V_j^{hom}] - [c_i - \sum_{j=1}^{M} M_{ij} V_j^{inh}] \tag{5.11}$$

$$\rho'_i = \sum_{j=1}^{M} M_{ij} (V_j^{inh} - V_j^{hom}) \tag{5.12}$$

where $V^{inh}$ and $V^{hom}$ are the voltages from disturbance and homogeneous tanks respectively.

#### Meshing

The meshes typically used in 3D algorithms often seem to be very coarse compared to those used in 2D algorithms. This is because the number of elements in the mesh is usually limited by computer resources, so though 2D and 3D meshes might have similar numbers of elements, in the case of the latter, they are spread over three dimensions so

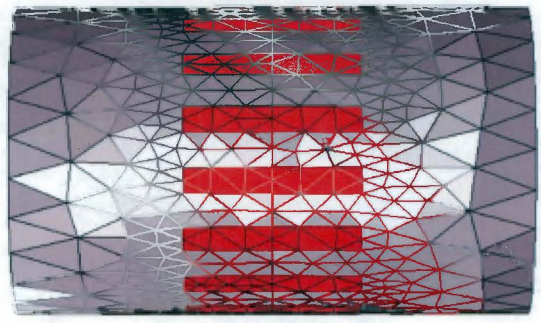


Figure 5.6: A 3D mesh with 8749 elements

the number of elements in a cross section is much less. However, because the velocity profiling algorithm only uses elements in the plane of the electrodes, once the M matrix has been calculated, the rows that calculate the resistivity of the elements outside of this plane can be dropped. This means that the 3D mesh can have the some cross sectional resolution as the mesh used in the previous algorithm without any loss of speed despite the fact that the forward model that generated it used a mesh with many more elements in it.

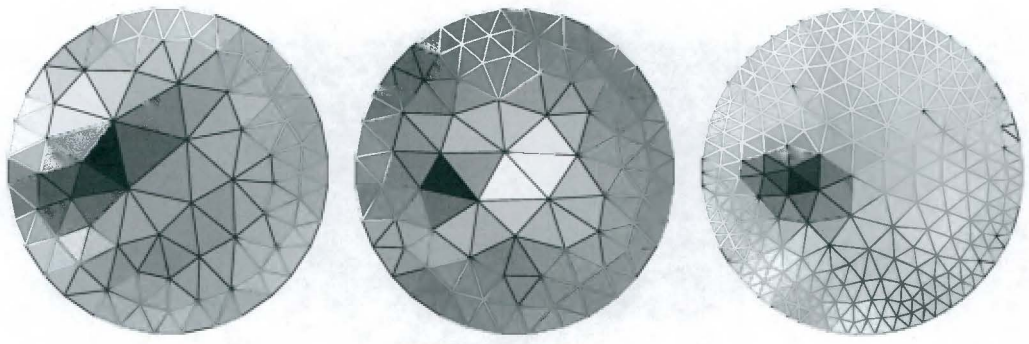
Figure 5.6 shows a typical 3D mesh generated using the NetGen [55] meshing program. A cross section through the plane of the electrodes reveals a mesh with 440 triangular elements, significantly less than the 8749 tetrahedral elements in the full mesh. The cross section can be seen in Figure 5.7(c).

### Speed

As the matrix multiplication operation of the 3D one step algorithm is virtually identical to the 2D version, the speed of operation is affected in the same way by increasing the number of elements in the mesh. Figure 5.4 shows the relationship.

### Images

The results of the 3D image reconstruction with increasing mesh sizes is shown in Figure 5.7. The reconstruction was done on data measured from a tank containing a saline solution with a non-conducting rod placed off center. The same data that generated Figure 5.5 was used.



(a) (b) (c)

Figure 5.7: Results of a 3D image reconstruction with different mesh sizes (numbers in brackets denote the number of elements in cross section) (a) 910 elements (126) (b) 1265 elements (136) (c) 8749 elements (440)

### 5.2.3 Comparison of Algorithms

Both algorithms have similar operating times and thus can only be differentiated on the basis of the reconstructed images. Though it is difficult to compare images generated using meshes of different resolutions, the 2D algorithm does seem to discriminate between the conducting and non-conducting regions better. However, this does not guarantee that the 2D algorithm will preform better on data captured from the test rig.

## 5.3 Cross-Correlation Algorithms

As mentioned above, two cross correlation algorithms were implemented, the first, the 'fast correlation' method, involves a multiplication of the Fourier transform of the signals, while the second operates in the time domain. These algorithms are described below.

### 5.3.1 Memory Management

Both these algorithms operate on pixel data from the last $N$ reconstructions, where $N$ is the window size of the cross correlation. This data must be saved in a memory store so that it is readily available. The size of this store is an issue as the UCT tomography instrument produces over 1.7MB/s[1] and the system will soon run out of memory if the store is continually increased to accommodate the latest data. To prevent this, a rolling data management system, shown in Figure 5.8 is used.

[1] 283fps $\times$ 2 planes per frame $\times$ 384 elements per plane $\times$ 8 bytes per element $\approx$ 1.7MB per second

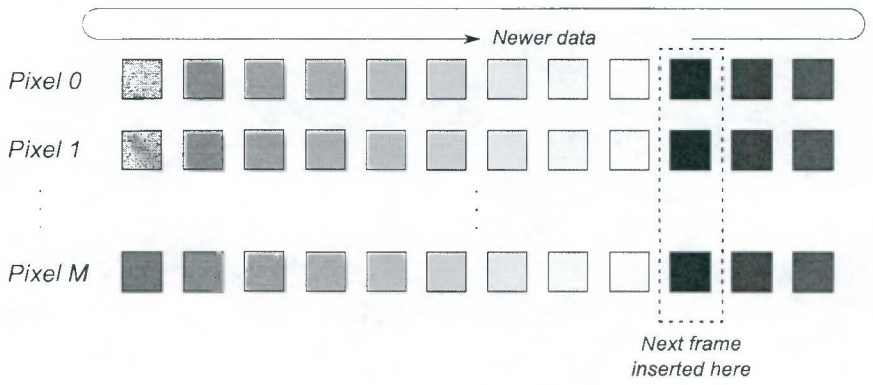



Figure 5.8: Rolling memory management

In this system, the size of the memory store is fixed. Once it is full, the new pixel values simply overwrite the oldest values and the index of the newest data is saved. When extracting the pixel data, copying begins after this index and loops around to the beginning of the pixel memory until every sample is copied.

This system saves memory and time as only the data to be used in the cross correlation is stored in memory and no superfluous copying or rearranging is done.

### 5.3.2 Normalisation

To calculate the normalised correlation coefficient, the mean and standard deviation of the two signals to be correlated need to be established. Instead of applying these values directly as described in Equation 4.6, i.e. subtracting the means during the correlation and then scaling by the product of the standard deviations, the input data is normalised before the correlation. In other words, the $x$ and $y$ are modified according to

$$x'(n) = \frac{x(n) - E(x)}{\sigma_x} \tag{5.13}$$

where $E(x)$ and $\sigma_x$ are the mean value and standard deviation of $x(n)$ respectively. This ensures that the signals have zero mean and standard deviations of 1.

This normalisation of the data is done as it is copied out of the memory store and into the cross correlation buffers.

### 5.3.3 Fast Correlation Algorithm

In Chapter 4 the cross correlation equation was expressed as

$$cc = F^{-1}\{X_n^* Y_n\} \tag{5.14}$$

where $X^*$ is the conjugate of the fast Fourier transform of the time series of a pixel in the first measurement plane and $Y_n$ the FFT of the corresponding pixel in the second plane.

This algorithm is implemented using the *Fastest Fourier Transform in the West* library, or FFTW [25]. This set of highly optimised routines was used to implement the Fourier and inverse Fourier transforms need to solve Equation 5.14.

#### FFTW

The FFTW library does not use a single algorithm for calculating all transforms. It aims to take advantage of any possible optimisations offered by the particular length the data, the type of data (real or complex), the type of transform and the machine the program is running on. To do this, it makes use of a *planner* which 'learns' the quickest way of doing the transformation and produces a transformation 'plan'. This plan may take a relatively long time to calculate, but only needs to be done once and it improves the efficiency of subsequent transformations of the same type. Once the plan has been created, the transformation is performed by executing it.

In implementing the cross correlation algorithm, plans for three transformations need to be created,

1. a plan for forward FFT of x
2. a plan for forward FFT of y
3. a plan for inverse FFT of $X_n^* Y_n$

The first two of these plans are similar, they both involve a forward transformation of real data. These plans where created with the `fftw_plan_dft_r2c_1d` routine. The last plan is an inverse transformation of complex data to real data and was created using the `fftw_plan_dft_c2r_1d` routine.

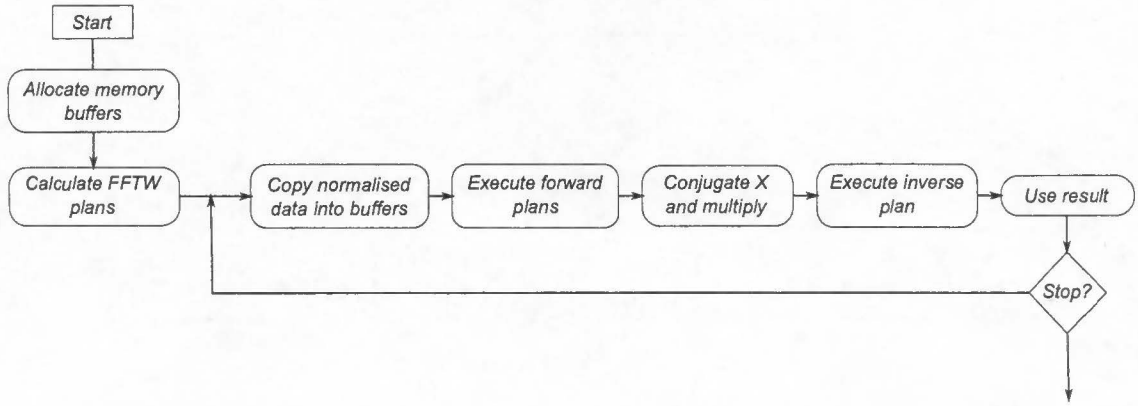



Figure 5.9: Fast cross correlation algorithm

**Algorithm Steps**

The steps that need to followed in the implementation are

1. Allocate memory buffers for two inputs and output
2. Calculate FFTW plan
3. Copy normalised data into input buffers
4. Execute forward plans
5. Conjugate X and multiply
6. Execute inverse plan
7. Use result
8. Get next data (step 3)

This is shown in Figure 5.9.

**Improving resolution**

Although the resolution of the correlation function is limited by the sample rate of the frame capture, a form of interpolation can be used to obtain an improved estimate of the location of the correlation peak.

Zero padding the result of the multiplication in the Fourier domain before calculating the inverse transform has the effect of oversampling in the time domain. This means that a higher sample rate can be simulated and a better velocity discrimination achieved.

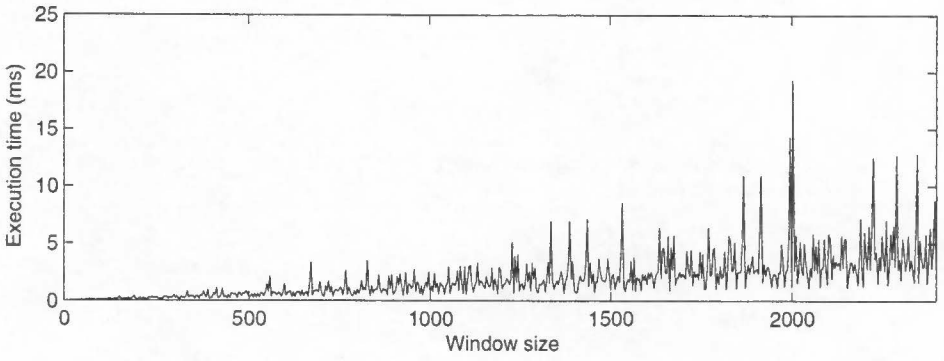



Figure 5.10: Fast correlation algorithm speeds for increasing window sizes

**Algorithm Analysis**

The FFTW uses a variation of the Cooley-Tukey algorithm, the complexity of which is $O(nLog(n))$. However, it must be noted that zero padding by some factor in the Fourier domain to improve resolution increases $n$ by that factor and that the range of the cross correlation is always equal to the window size, or length of the two series being cross correlated.

According to the development notes of the FFTW libraries [25], it is most efficient when the length of the vectors to be transformed have small prime factors. Figure 5.10 makes it apparent that even changing the window size by a small amount can have a dramatic effect of the time of calculation. This reduces the flexibility of the algorithm somewhat as the ratio of window size (or calculation speed) to correlation coefficient cannot be finely adjusted.

### 5.3.4 Time Domain Algorithm

Cross correlation in the time domain is based on Equation 4.9.

This algorithm is akin to 'sliding' a window of the most recent values from the second measuring plane pixel over the values from the first measuring plane pixel. At each step, the overlapping values are multiplied together and summed to produce the cross correlation profile.

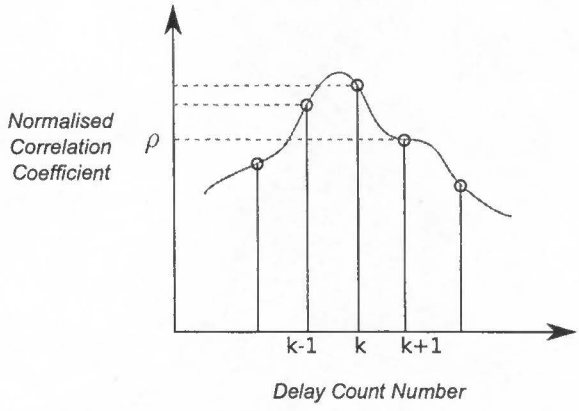



Figure 5.11: Interpolation of correlation coefficients

**Improving resolution**

The discrimination of this correlator is $\Delta t/2$, but it can be improved by interpolating between the values surrounding the correlation peak as shown in Figure 5.11.

In this case, the value at the interpolated peak is greater than the maximum value of $cc(k)$. This peak is found by fitting a quadratic function to the three values surrounding the peak in same manner as described by Yang and Beck. This is done as follows[2]

If the three values are $(k-1, cc_{k-1})$, $(k, cc_k)$ and $(k+1, cc_{k+1})$ they must be fitted to

$$cc = ax^2 + bx + c \tag{5.15}$$

the coefficients of which can be found by solving

$$cc_{k-1} = a(k-1)^2 + b(k-1) + c \tag{5.16}$$

$$cc_k = ak^2 + bk + c \tag{5.17}$$

$$cc_{k+1} = a(k+1)^2 = b(k+1)^2 + c \tag{5.18}$$

The maximum, or interpolated peak, occurs when

$$\frac{\delta cc}{\delta x} = 2ax + b = 0 \tag{5.19}$$

or

$$x|_{cc=max} = -\frac{b}{2a} \tag{5.20}$$

[2]These equations are taken from [69]

Solving equations 5.16 gives

$$a = \frac{1}{2}(cc_{k+1} - 2cc_k + cc_{k-1}) \tag{5.21}$$

$$b = \frac{1}{2}(cc_{k+1} - cc_{k-1}) - k(cc_{k+1} - 2cc_k + cc_{k-1}) \tag{5.22}$$

Therefore,

$$x|_{cc=max} = k - \frac{1}{2}\frac{cc_{k+1} - cc_{k-1}}{(cc_{k+1} - 2cc_k + cc_{k-1})} \tag{5.23}$$

**Improving Speed**

Unlike the previous algorithm, the range that the correlation window is 'slid' over can be different from the window size. This means that another technique used by Yang and Beck [69] can be employed, that of 'auto pre-delay'.

'Pre-delay' is based on the assumption that the location of the correlation peak will not vary significantly, this means that instead of sliding the window across the entire cross correlation range, it need only be slid across the expected variation of peak position. This is can result in a significant speed improvement, particularly if the variation is small.

The time domain algorithm also allows for incremental updates to the correlation profile. Wang *et al* [61] note that an extra data point (effectively increasing the window size) can be added to the cross correlation without repeating the entire calculation.

$$R_{xy}^{(k)}(n) = R_{xy}^{(k-1)}(n) + f_1(k-n)f_2(k), \quad n = 0, 1, ..., (N-1) \tag{5.24}$$

This a very useful observation with the potential to significantly reduce calculation time and increase the accuracy of the velocity estimation over time. However, as the cross correlation window effectively becomes the entire data set, variations in the velocity of the flow will not be identified. This means that this optimization should only be employed if the velocity is expected to be constant, or if $R_{xy}^{(k)}$ is periodically reset.

**Algorithm Analysis**

The complexity of the basic time domain algorithm, with the correlation range equal to the window size, is $O(n^2)$, which is somewhat worse than the frequency domain algorithm. This arises because the implementation involves a nested loop, with the inner loop doing the correlation and the outer doing the 'sliding'. However, intelligent use of the 'pre-

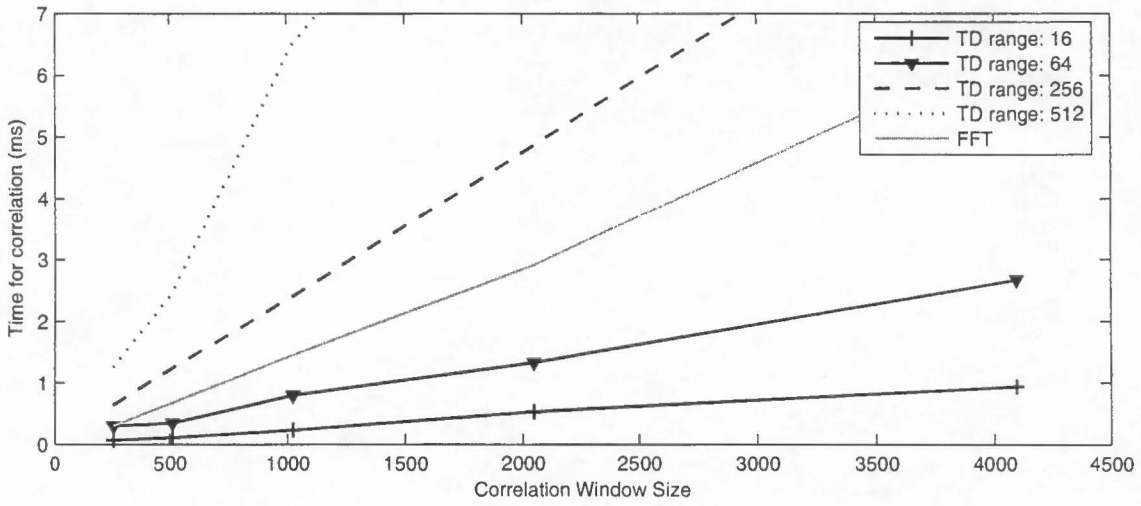



Figure 5.12: Time domain and fast correlation algorithm speed comparison

delay' can reduce the complexity to close to $O(n)$ as the number of iterations of the outer loop is minimised.

Using the update method suggested by Wang *et al*, also reduces the algorithm complexity to close to $O(n)$. In this case, the inner loop is replaced by an addition and a multiplication.

Furthermore, unlike the 'fast correlation' algorithm which is more efficient for window sizes with small prime factors, the time domain version does not favour particular window sizes. This means that it can be finely tuned for an optimal trade off between speed and confidence in the the location of the peak.

### 5.3.5 Algorithm Comparison

Figure 5.12 compares the algorithm speeds with varying window sizes. 'TD range' refers to the range for the cross correlation the time domain algorithm and FFT refers the 'fast correlation' method. Recall that improving the window size increases the confidence that the peak is in the correct position. To ensure the the fast correlation method operated efficiently, the window sizes used were powers of 2, namely 256, 512, 1024, 2048 and 4096.

As can be seen from the figure, if the expected variation of the peak position is low, and the range of the time domain algorithm used is small, for example 16 or 64 samples, the time domain algorithm is significantly faster. However, if the correlation range required is large, the fast correlation algorithm becomes more efficient. This suggests that the overall

system can benefit if both these algorithms can be used, depending on the window size required for confidence in position of the correlation peak.

# Chapter 6

# Software Integration

The previous chapter described the selection and implementation of the reconstruction and cross correlation algorithms. This chapter describes the integration of these algorithms into one system.

There are two aspects of the design of the system integration that need to be considered. The first is the structural integration. This deals how the routines that execute the algorithms previously described are grouped. Coupled with this is the second aspect, the dynamic design, or how the data flows through the system from when it is captured by the hardware, to when it is displayed on screen or saved to disk.

Before these aspects are examined, it is necessary to talk about the software generally.

## 6.1 Software Overview

The software was implemented in C++ using the wxWidgets library [26]. C++ is an appropriate choice because the executable files produced by the compilation process are among the most efficient produced by any third generation programming language. C++ has other benefits that are described below.

wxWidgets is an open source cross-platform set of libraries that allows developers to build applications that can be compiled to run on a variety of operating systems with little or no change to the source code.

Microsoft Visual Studio 6.0 was used to compile the software and all tests were done using Microsoft Windows XP SP2. This is relevant to the aim of achieving real time

operation because even if the time to process the data is less than that taken to capture it (a requirement for real time operation), the system, operating on Windows XP, will never be truly real time. This is because Windows XP is not a hard real time operating system, or in other words, the response time is not guaranteed for a request to perform work [46]. However, for the purposes of visual interpretation, the system can be considered to be real time if display update rate is faster than rate at which an observer can differentiate frames, this is typically between 20 and 30 fps.

## 6.2 Structural Overview

As mentioned previously, the steps involved in generating a velocity flow profile are:

1. data collection,
2. image cross section reconstruction,
3. pixel-wise cross-correlation and,
4. visualization

As such, it makes sense to make the software design modular to mirror these steps. In fact, implementing the software in separate modules or layers like this has an additional benefit. It allows each layer to be modified without affecting the other layers as long as the interfaces between each layer are clearly defined. This means that different algorithms can be swapped in and out of the application without having to re-program the entire sequence. C++, being an object-oriented language, provides concepts like inheritance and polymorphism that allow the developer to implement a layered or modular design such as this. Figure 6.1 shows the separation of the different software layers.

These software layers are now examined.

### 6.2.1 Data Provider Layer

The data provider layer is the point of entry of new data into the system. To ensure flexibility, this layer presents an interface that is independent of the data source. This means that the data source is not restricted to a tomography instrument, it allows a previously saved file to be used if desired. Figure 6.2 shows this abstraction.

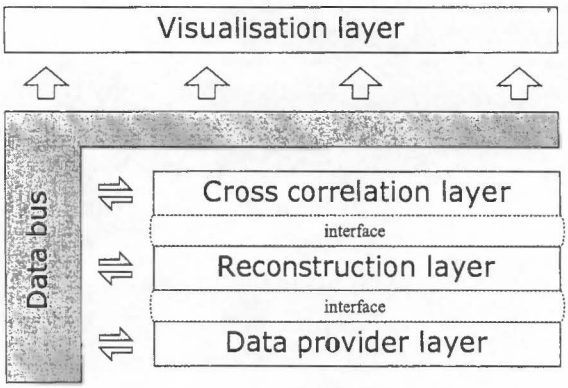



Figure 6.1: Software layout

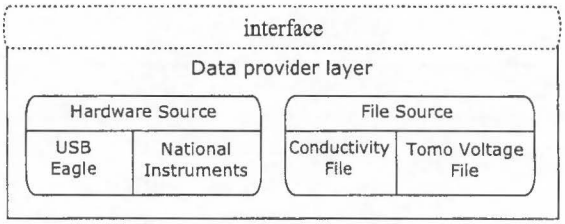



Figure 6.2: The data provider layer

By abstracting the hardware source, the software is not permanently tied to a particular piece of hardware. This is useful as different versions UCT tomography instrument have, in the past, been fitted with DAQs from different vendors such as National Instruments or Eagle Technology. Should the hardware change again, this software can be quickly be adjusted by writing a class appropriate for the new DAQ and slotting it in.

**Interface functions - Hardware Control**

The functions provided by the hardware control layer are shown in Table 6.1.

**Interface functions - Data Provider**

The functions provided by the data provider interface are shown in Table 6.2.

### 6.2.2 Reconstruction Layer

The reconstruction layer is the stage in which reconstruction of the voltage data from the data provider layer is done. The two reconstruction methods described in the previous

| Function Name | Description |
|---|---|
| `Reset` | Resets the tomography instrument |
| `Start` | Starts the tomography instrument |
| `StartCalibration` | Starts the tomography instrument running in calibration mode. That is, with no current applied |
| `Stop` | Stops the tomography instrument |
| `AdjustCurrent` | Sets the output current supplied by the tomography instrument |
| `GetData` | Copies a specified amount of data from the hardware buffer |

Table 6.1: The list of hardware control interface functions

| Function Name | Description |
|---|---|
| `SetSource` | Sets the data source. Either a hardware source or a file source |
| `GetFrame` | Gets a frame of data from the data source |

Table 6.2: The list of data provider interface functions

chapter have been implemented, but again, the modular nature of the design allows additional algorithm implementations to be included with minimal adjustment to the code.

**Interface functions**

The functions provided by the reconstruction interface are shown in Table 6.3.

### 6.2.3 Cross Correlation Layer

The cross correlation algorithm is implemented in this layer. Again, the abstraction enables both the techniques described in the previous chapter to included in the system.

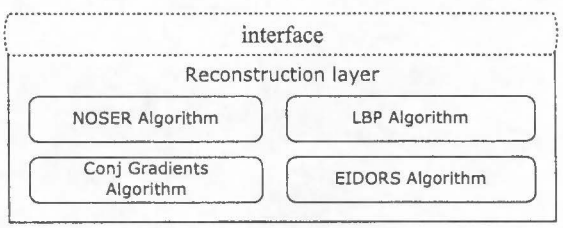



Figure 6.3: The reconstruction layer showing possible algorithm implementations

| Function Name | Description |
|---|---|
| `SetMesh` | Sets the mesh used in the image reconstruction. |
| `SetCalibrationMethod` | Sets the type of calibration to use |
| `SetCalibrationData` | Sets the calibration data. |
| `SetReconstructionMethod` | Sets the type of calibration method to use |
| `SetData` | Specifies the voltage data to reconstruct. |
| `SetAlgorithmParameter` | Different reconstruction algorithms use different parameters. This function allows these parameters to be set. For example, in the one step algorithms, the 'precomputed' matrix can be set using this function. |
| `DoReconstruction` | Executes the reconstruction algorithm and returns the image data. |

Table 6.3: The list of reconstruction layer interface functions

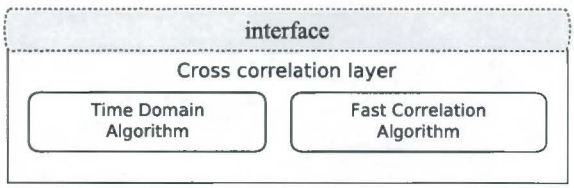



Figure 6.4: The cross correlation layer

**Interface functions**

The functions provided by the cross correlation layer interface are shown in Table 6.4.

### 6.2.4 Visualisation Layer

As its name suggests, the visualisation layer deals with visualising the data from each of the layers. All visualisation was done using the OpenGL graphics library. OpenGL is a set of cross platform libraries that allow the programmer to generate images based a set of primitives, namely points, lines and triangles in two dimensions or three.

This layer interacts with the rest of the program in a different fashion to the other layers. A visualisation is only updated when the screen is updated, so instead of the accessing the layers mentioned above directly, it only accesses a data bus as shown in Figure 6.1. This decouples the visualisation from the data processing which has benefits, as seen below, for the dynamic operation of the system.

The types of visualisation plots implemented are now described.

| Function Name | Description |
|---|---|
| `SetLayerData` | Adds the newest image data to the correlation data. |
| `SetAlgorithmParameter` | Different algorithms use different parameters which can be set using this function. For example, the 'pre-delay' offset can be set here. |
| `SetCorrelationMethod` | Sets the type of cross correlation method to use |
| `DoCrossCorrelation` | Executes the cross correlation algorithm and returns the result. |

Table 6.4: The list of correlation layer interface functions

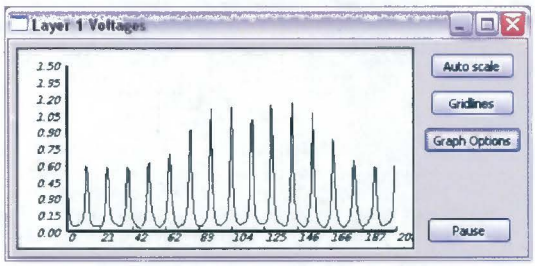



Figure 6.5: An example of the voltage plot

### Voltage Plots

Voltage plots allow the user to see the raw data being captured by the tomography instrument. This visualisation is very useful when debugging the hardware and when setting the level of applied current. Figure 6.5 shows a typical measurement set from one electrode plane.

### History plots

History plots show the change in value of a sample over time. This sample can be either a reconstructed pixel denoting the conductivity at a point, or a voltage data point. In the case of the reconstructed pixel, the plotted data is effectively the signal used in the cross correlation for that pixel. Figure 6.6 shows a history plot of the conductivity at a point near the top of the pipe.

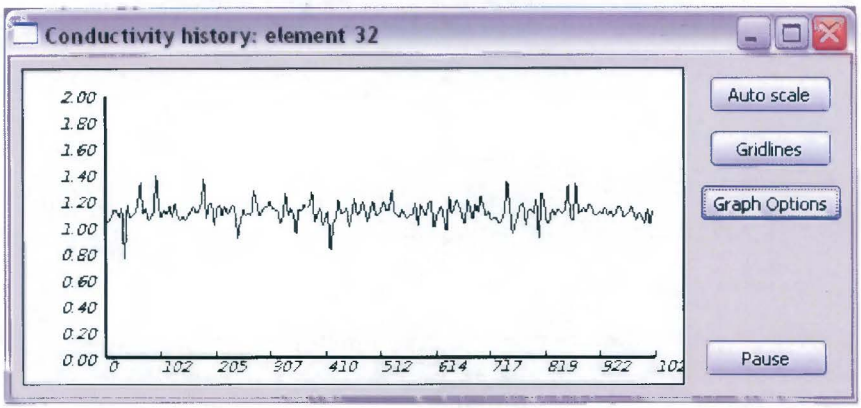



Figure 6.6: Conductivity history of an element near the top of the pipe

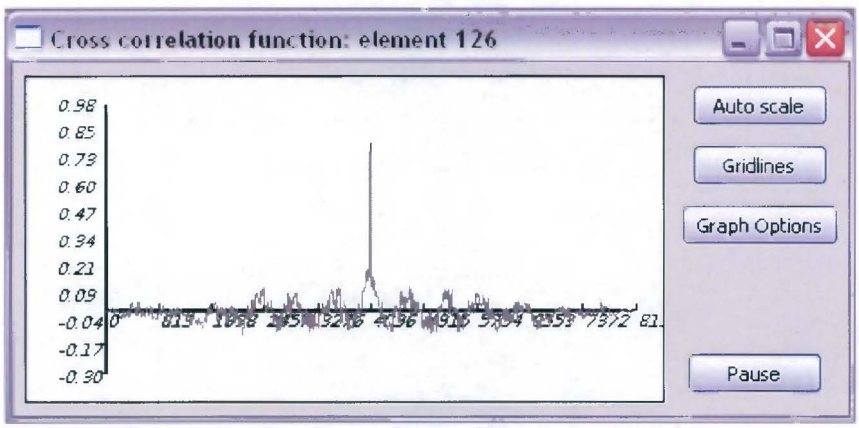



Figure 6.7: An example of a cross correlation plot showing a clearly visible peak

### Cross Correlation Plots

The cross correlation plot shows the cross correlation profile for a single correlation. Figure 6.7 is an example of this type of plot showing a clearly visible peak.

### Velocity Profile Graph

The final graph type visualisation is the velocity profile plot. It plots the estimated velocity values through a slice of the two dimensional velocity profile. An example is shown in Figure 6.8. The blocky appearance of the plot is due to the discretisation of the mesh.

### Mesh Plots

These are 2D plots of the finite element mesh. The colour of each triangle is a function of the value of the pixel at that point. Mesh plots can be used to represent the reconstructed

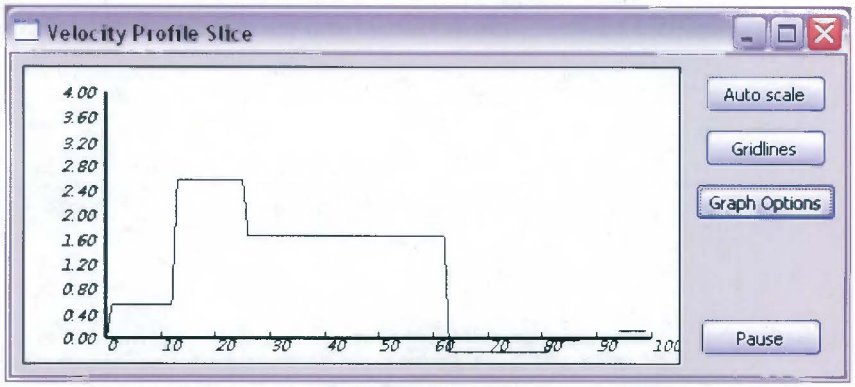



Figure 6.8: An example of a velocity profile slice running vertically from the top of the pipe to the bottom.

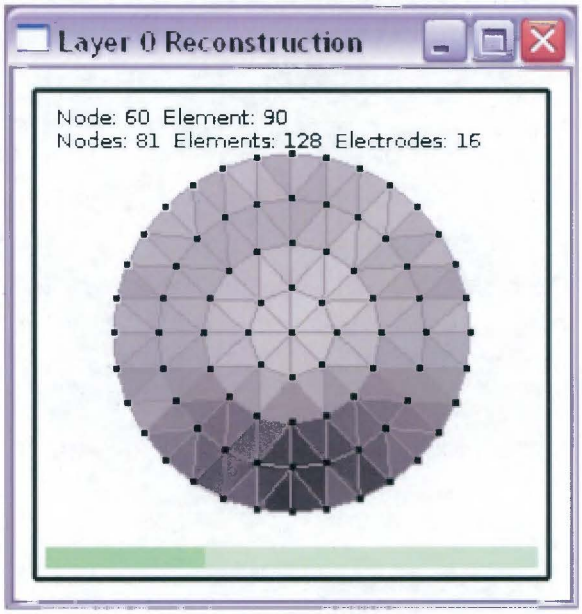



Figure 6.9: A typical mesh plot of the conductivity distribution showing the sliding bed.

conductivity or the velocity estimation. Figure 6.9 is a typical mesh plot of a conductivity distribution. The colourmap used is such that lower conductivities, or lower velocities, are represented by darker colours.

### Heightmap Plots

Heightmap plots are an extension of mesh plots. Instead of the colour of the pixel being a function of its value, height is now used. Again, either conductivity or velocity estimation can be represented. Figure 6.10 is an example of the heightmap being used to represent the velocity profile of the flow.

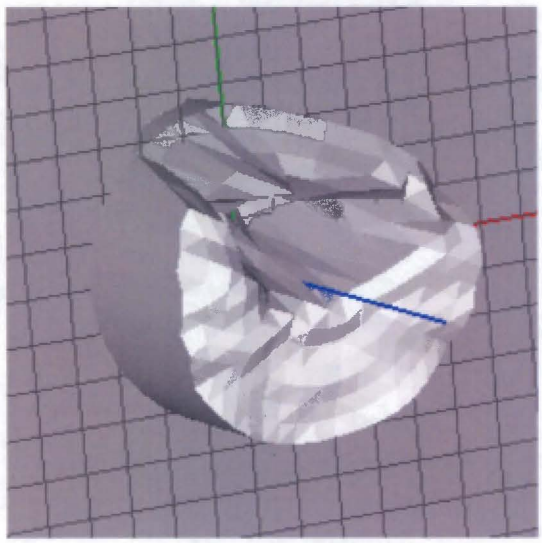


Figure 6.10: A heightmap plot showing the velocity profile of a flow

### 6.2.5 User Interface Layer

There is another layer, not included in Figure 6.1 as it is somewhat incidental to the data processing stages of the system. This layer includes the functionally that allows the user to interact with the program.

The actions that the user can perform through this layer shown in Table 6.5.

| Function Name | Description |
|---|---|
| Start | Starts the profiling process, this may be by starting the instrument or reading from a file |
| Stop | Stops the profiling process |
| SetCorrelationMethod | Choose the cross correlation algorithm to use |
| SetCorrelationParameters | Set parameters relating to the correlation algorithm |
| SetReconstructionMethod | Choose the reconstruction algorithm to use |
| SetReconstructionParameters | Set parameters relating to the reconstruction algorithm |
| SetDataSource | Choose the data source, from live data capture or a previously saved file |
| Add visualisation plot | Adds another visualisation plot to the display |

Table 6.5: Actions that the user can preform through the GUI.

Figure 6.11 is a screen shot of the application showing the cross correlation options for the fast correlation algorithm.

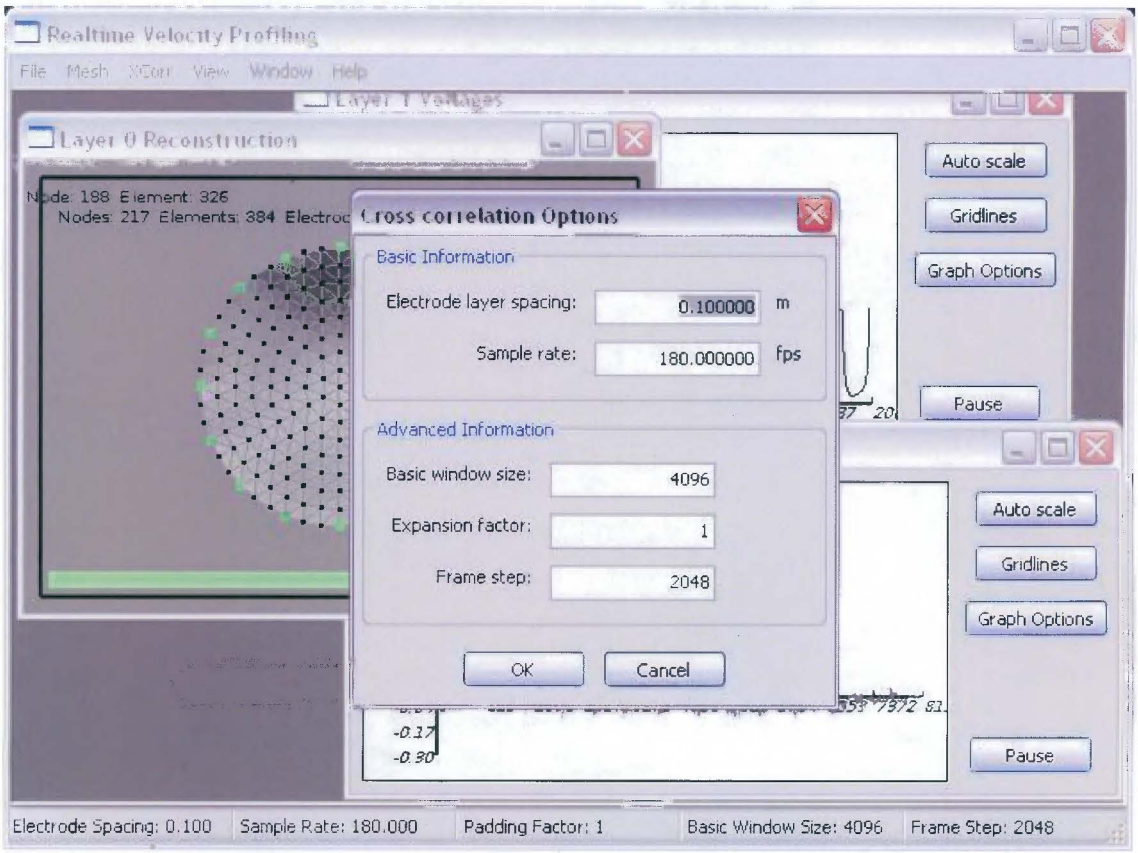



Figure 6.11: Part of the user interface

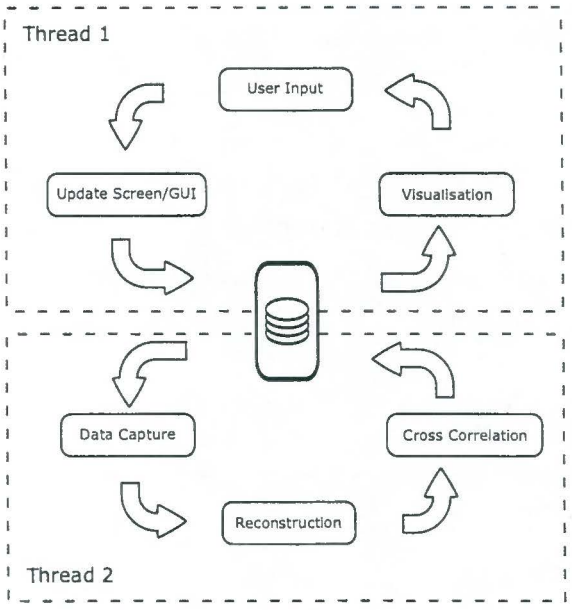



Figure 6.12: Program flow

## 6.3 Dynamic Overview

From a dynamic perspective, the software has been implemented using two control loops, as shown in Figure 6.12. There are a number of reasons for this, primarily due to the way the wxWidgets library deals with user interaction and when it updates the display.

The wxWidgets library follows an event driven model with a central message queue. Every time an event is generated (by the user or the operating system for example) it is added to this queue. When the application comes to process an specific event, it executes the code associated with that event (the event handler), afterwhich the application processes the next item on the queue. Types of events include:

- a button pressed event
- a refresh screen event
- a timer expired event

Should the application be implemented as a loop in which the data is continously captured, processed and the visualisation routine called, the screen would never be refreshed and the application would not respond to user input. This is because the flow of control would never go back to the message queue and the events that instruct the processor to do these things would never be processed.

To counter this problem, all the data processing has been moved to a separate thread, denoted by thread 2 in Figure 6.12. This will be referred to as the 'worker' thread hereafter, while the main thread, the thread which updates the interface and gets input from the use will be called the 'GUI' thread. With this arrangement, the worker thread can operate continuously, processing data as fast as it comes from the hardware, while the main thread only executes when handling user input or drawing a plot or visualisation on the screen. As the two threads operate asynchronously, the above problem is no longer an issue.

### 6.3.1 Race Conditions

Using more than one thread in a program introduces additional complexity because they operate asynchronously and the flow of control can switch between threads at any time. Consider a situation where the worker thread is copying the results of a calculation, say an image reconstruction, to the data bus but its time slice ends before the whole image is copied. If the GUI thread now tries to display this image, it will display a mixture of the new and old images. This is an example of a *race condition* and it gives an idea of the potential problems caused by multiple threads.

To prevent race conditions occurring *mutexes* are used. A mutex is an object which can be tested and set in one atomic operation and is effectively a guard for a memory block. When a thread wishes to write to a block of memory that is shared by more than one thread, it first checks to see if the mutex is set, or *signalled*. If it is not, the thread has access to the memory buffer and must signal the mutex before writing to the buffer. If the mutex is signaled, it means that the memory is being used by another thread and the current thread must wait until the mutex is un-signaled. This procedure only works if a thread un-signals the mutex once a write is complete. If the test and set of the mutex was not an atomic operation, the operating system could switch the flow of control between the test and signal steps resulting in possibly undesired behaviour.

## 6.4 Dealing with Bad Correlations

It was established that while the location of the peak correlation coefficient in the cross correlation determines the speed of the flow, the value of this peak, together with knowledge of the correlation window size, indicates the confidence in its accuracy. The question remains of how to deal with correlation results where this value is very low, i.e. when the

two signals do not correlate. Simply finding the maximum in a set of very low values is unlikely to produce the correct offset.

One option is to discard the pixels for which the peak correlation value is below some predetermined threshold and exclude them from any visualisation as the offsets are likely to be erroneous. Another option is to infer the likely value from surrounding pixels and what the velocity at that point was in the past. This is valid if the confidence of the accuracy surrounding pixels is high, the velocity gradients across the cross-section are small, and the variation of velocity over time at a point is low, i.e. if there is spatial and temporal coherence in the flow. If these conditions are met, the values of the pixels surrounding the problem element can be used to estimate a replacement value.

One method of estimating the replacement value is simply to define it as the mean of the surrounding pixels, giving each surrounding pixel equal influence on the estimated value. Alternatively one could relate the amount of influence a neighbouring element has to its correlation coefficient.

This can be expressed as

$$v' = \sum_{i=1}^{D} v_i \rho_i \tag{6.1}$$

where $v'$ is the estimate of the replacement pixel, $D$ is the number of surrounding pixels (including the previous value of this pixel) and $v_i$ and $\rho_i$ are the velocities and correlation coefficients of a surrounding pixel respectively.

Should the correlation coefficients of the surrounding pixels also be too low, this method cannot be used and the pixel should be excluded from the velocity profile.

The threshold for determining whether a pixel should be discarded is dependent on the flow and measurement conditions. Henningsson *et al* [23] consider cross correlations of lower than 0.45 as noise.

The results of this interpolation method can be seen in the heightmap visualisation in Figure 6.13. The pixels that have been interpolated are shown in green, while the discarded pixels are shown in red.

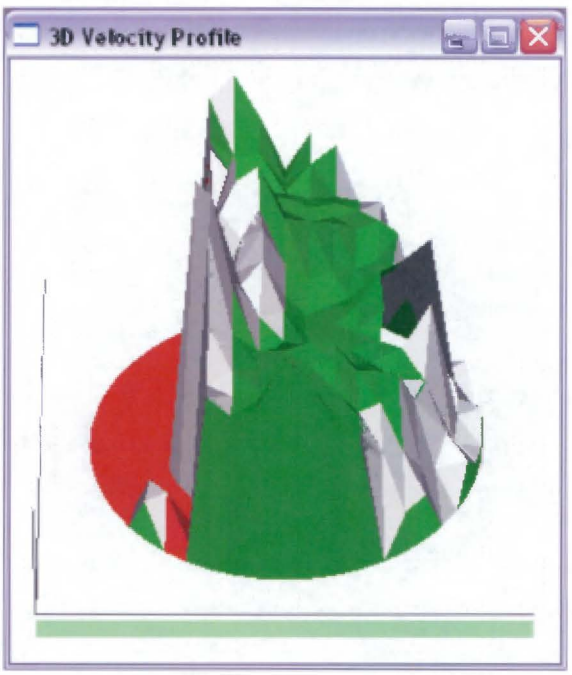



Figure 6.13: Interpolating across pixels with low correlation coefficients

# Chapter 7

# Results

This chapter examines the performance of the velocity profiling software and its suitability for on-line use on the CPUT pipe flow test rig.

The performance is examined from two perspectives: firstly, whether the program is fast enough to deliver on-line profiles and secondly, whether the these profiles reflect the nature of the flow.

In the tests below, four meshes were used, one fine and one coarse for each of the two reconstruction algorithms. Ideally, for the purposes of comparison, the 2D and 3D algorithms would use meshes with identical cross sections but unfortunately the nature of the meshing programs used makes this impossible. Instead meshes were chosen so that the number of elements in each were similar. Table 7.1 shows the number of elements in each mesh used.

| | 2D Algorithm | 3D Algorithm |
|---|---|---|
| coarse | 128 | 126 |
| fine | 384 | 440 |

Table 7.1: Mesh element count

### 7.0.1 2D Meshes

The two dimensional meshes were created using meshing software written by the author.

### 7.0.2 3D Meshes

The three dimensional meshes were created using NetGen, a commonly used meshing program. Note that only the number of elements contained in the extracted 2D slice are shown. The full 3D volumes contained many more elements, Figure 5.6 shows the complete 3D mesh that produced the 440 element cross section mesh. The meshes were based on the pipe geometry, that is, a 57mm diameter with 19.1mm by 4.8mm rectangular electrodes. The length of the pipe section modelled was 57mm, or equal to one diameter.

## 7.1 Speed

The overall time taken to generate a velocity profile is the sum of the times taken for the image reconstruction, correlation and visualisation stages. This section presents speed tests for these stages to ascertain how much time the processor of the PC spends on each and how they individually contribute to the operation times of the program.

The data capture rate is not mentioned in the above list because, although it does limit the operating speed, data capture happens independently of the controlling PC where all the data processing is done. The measurement data rests in the buffers of the Eagle Technology DAQ until it is transferred via the USB link into the PC memory, so the processor's involvement is limited to control of the USB and copying the voltage data into memory available to the velocity profiling system. In terms of the time taken to do further processing, the time spent on this stage is negligible.

The data capture rate is still important in determining whether the system can function on-line as it places a limit on how long the data processing stages can take. The tomography instrument is continually adding data to the Eagle Technologies buffer at 1.7MB per second, if the processor is spending too much time calculating the velocity profiles, it will not empty this buffer fast enough and the buffer will overflow.

To avoid this situation, and still use all the data, data processing (servicing of data in this buffer) must be at least as fast as the data capture. This is not to say that the time taken to generate one velocity profile must be less than the time to capture one frame as a profile need not be generated for each new frame. For the system to appear real time, the velocity profile need only be updated each time the screen is refreshed, typically at a rate of 15 frames per second, which, for the current setup, corresponds roughly to updating the velocity profile every 20 data frames. For an on-line system, the update rate can be substantially less. This means that the Eagle Technology buffer can be allowed to fill up

somewhat while a velocity profile is calculated as long as the buffer is emptied before the next profile is calculated. To ensure this, the data processing time, $t^{dp}$, must satisfy $t^{dp} < N * \Delta t$, where $N$ is the number of samples processed and used in the velocity profile and $\Delta t$ is the frame capture period.

A 2.67GHz Intel Celeron machine with 500MB of RAM was used to produce the results below.

### 7.1.1 Image Reconstruction Times

The image reconstruction times were estimated by recording the time taken to reconstruct data from both planes a 1000 times and averaged. These times are shown in Table 7.2.

| Mesh Size (elements) | Time for Image Reconstruction (ms) |
|---|---|
| 126 | 0.578 |
| 128 | 0.584 |
| 384 | 1.753 |
| 440 | 2.009 |

Table 7.2: Image reconstruction times

Although it was stated above that the velocity profile calculation need not be done every time a new data frame is captured, strictly speaking, it is only the cross correlation stage of this calculation that need not be done so frequently. The image reconstructions, however, must be done on every single frame as the correlation algorithm makes use of the resistivity data, not the voltage measurements. If all data is to be used, this means that the reconstruction rate must be greater than the data capture rate, i.e. $t^r < \Delta t$. Table 7.2 shows that even the mesh with 440 elements satisfies this condition as $\Delta t = 1/f = 1/283 \approx 3.534ms$.

The fact that image reconstruction can be done faster than the data capture rate means that this system could also be used in a real time process control application based on the resistivity profiles.

### 7.1.2 Cross Correlation Times

The speeds of the cross correlation algorithms were measured offline. By reading the tomography data from file, as opposed to live from the instrument, there was no danger

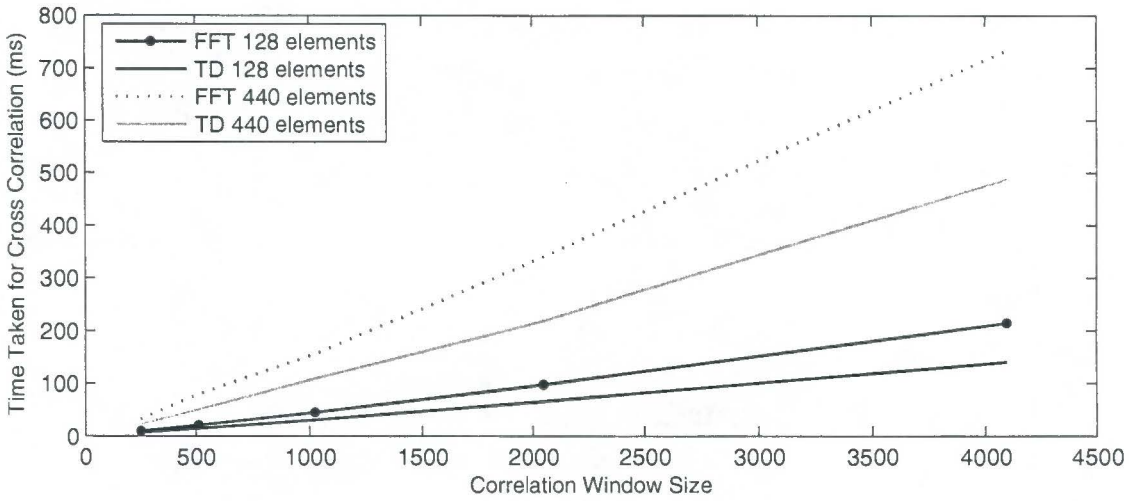



Figure 7.1: Cross correlation times. TD denotes the time domain algorithm using a maximum range of 64 samples, while FFT denotes the fast correlation method

of the Eagle Technology buffer overflowing if the calculations took too long.

Figure 7.1 shows the execution times for the correlation algorithms using window sizes of 256, 512, 1024, 2048 and 4096, i.e. powers of two so that the FFT algorithm is working at peak performance. Note that no interpolation was used by either algorithm and a correlation range of 64 samples[1] was used in the time domain version. Using a relatively small correlation range is justified as the flow velocity is not expected to vary rapidly. As long as the correlation peak is tracked (using the 'pre-delay' parameter) and kept within the correlation range, the system is able to adjust to changing flow velocities. A much larger range may be required to find the peak at the start of operation, but once found, the range can be reduced.

Once again the calculation times were estimated by repeating them a 1000 times and dividing the total time by 1000 to obtain an accurate estimate.

### 7.1.3 Velocity Profiling Times

The time to generate a complete velocity profile is made up of the image reconstruction times, the cross correlation time and any additional overhead times. Figure 7.2 shows how increasing correlation window sizes and using different meshes affects the overall computation time. Results for only the 128 element and 440 element meshes are shown.

The red line in Figure 7.2 represents the time taken to capture the number of samples

[1] representing a velocity range of $0.22ms^{-1}$ in the case where the electrode rings are separated by 50mm

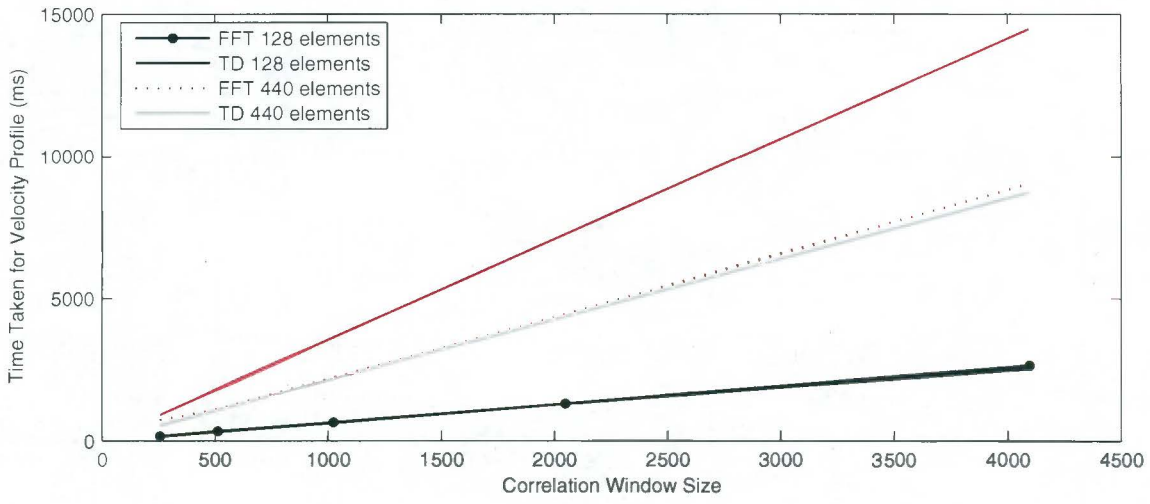



Figure 7.2: Times for velocity profiles to be generated using various window sizes for the cross correlation algorithm. TD denotes the time domain algorithm using a maximum range of 64 samples, while FFT denotes the fast correlation method

in the correlation window, or the 'on-line limit'. Combinations of mesh size, algorithm and window size that operate below this line can be successfully implemented on-line, with the correlation stage being performed at least once for every data set the size of the correlation window captured. Note that all the combinations tested operate below the line, indicating that the system is capable of on-line use.

The information in Figures 7.1 and 7.2 can be used to determine the fastest update rate of the correlation profile. Consider a combination of the fast correlation method, the 440 element mesh and a window size of 4096. In this setup, the 'on-line limit' for is $4096/283 = 14.46s$ and the time to calculate a complete velocity profile is $8.73s$. This leaves $5.73s$ for any additional calculations. As the time for the correlation calculation for this combination was recorded to be $0.73s$, the correlation can be done at most $5.73/0.73 \approx 8$ times during the data capture period ($14.46s$). This relates to an update rate of just over one frame every two seconds.

Figure 7.3 shows the maximum update rates for increasing correlation window sizes using both correlation algorithms on the 128 and 440 element meshes.

The above calculation does not take into account the time taken to visualise the data and thus the update rates that are possible in reality are likely to be much less than those indicated should visualisation be required. Accurately measuring the time taken to visualise the velocity profiles is a complex task as the operating system is responsible for when the display routines are called. As a result, the update rate of the velocity profile is left under the control of the user to determine heuristically.

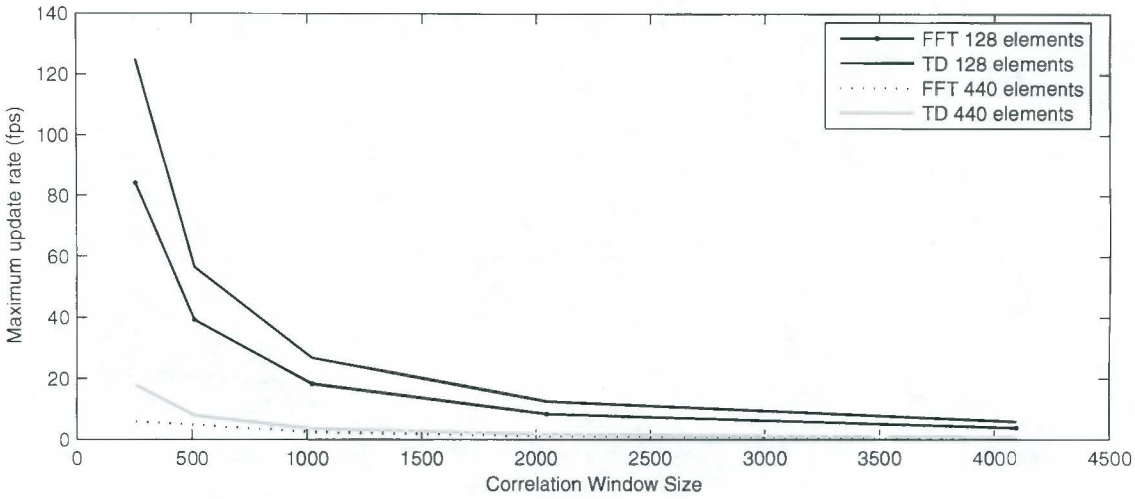



Figure 7.3: Maximum velocity profile update rates

## 7.2 Velocity Profiling

This section presents the results of the velocity profile calculations. At the time the data sets used below were captured[2], the on-line profiling software had not been developed so the data capture was done using the existing visualisation software mentioned in Chapter 2.

A mixture of mixture of 5% kaolin and 10% sand in water was pumped around the Ø57mm pipe at three different pump speeds, 800rpm, 1300rpm and 1700rpm. The UCT instrument captured data from two electrode planes separated by 100mm at a rate of 283 frames per second. At the slow pump settings, the coarse material (sand) settled on the bottom of the pipe as can be seen in Figure 7.4.

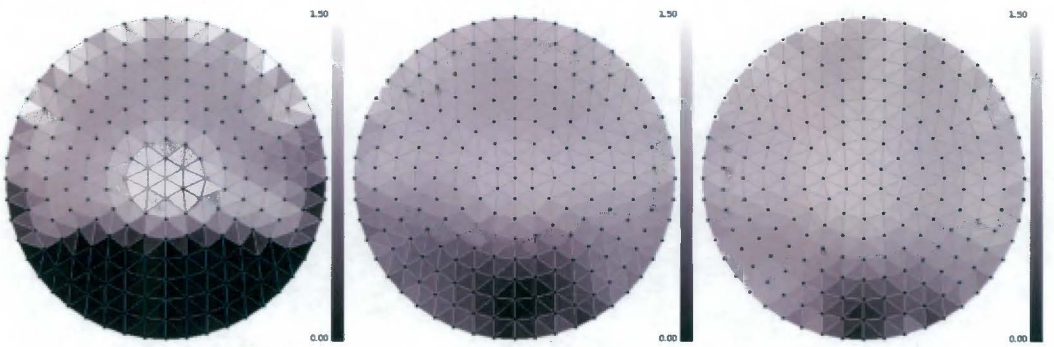
### 7.2.1 Conductivity Profiles

Figure 7.5 shows the reconstructed conductivity profiles of the slurry at the different pump speeds. The settling of the sand is clearly visible in the reconstructed images. As the pump speed was increased, more of the sand becomes suspended in the flow and the bed gets shallower.

[2]August 2005

(a) (b) (c)

Figure 7.4: The 5% kaolin, 10% sand slurry at a (a) low pump speed (800rpm) (b) medium pump speed (1300rpm) (c) high pump speed (1700rpm)



(a) (b) (c)

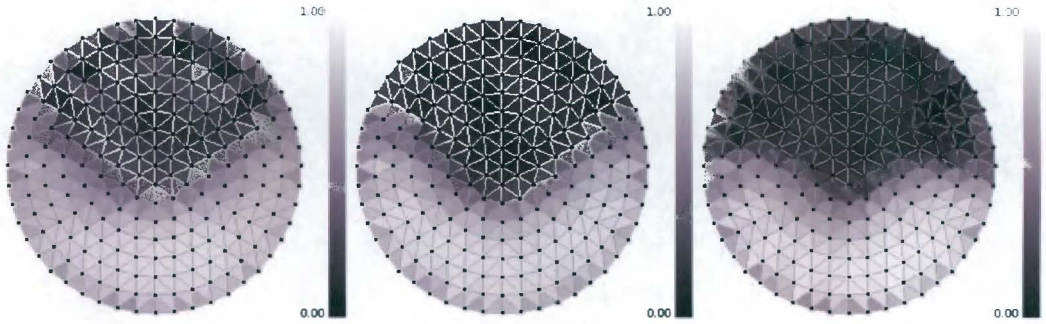
Figure 7.5: Conductivity profiles reconstructed with 384 element mesh (a) low pump speed (800rpm) (b) medium pump speed (1300rpm) (c) high pump speed (1700rpm)

### 7.2.2 Velocity Profiles

The results of the velocity profiling calculation on this slurry at different velocities are shown in Figures 7.7 and 7.8 in $ms^{-1}$ (note the scale is 0 to $3ms^{-1}$ in parts (a) and (b) and 0 to $5ms^{-1}$ in part (c) of Figure 7.7). The correlation window size used in these results is 4096 samples. These figures should be interpreted in conjunction with Figure 7.6 which represents the correlation coefficients at each reconstructed pixel. Lighter areas in Figure 7.6 correspond to higher correlation coefficients. Note that there is stronger correlation in the bottom of the pipe where there is a high concentration of solid particles. This is expected as the fluid flowing over the top of the bed contains fewer particles which cause less significant disturbances in the resistivity.

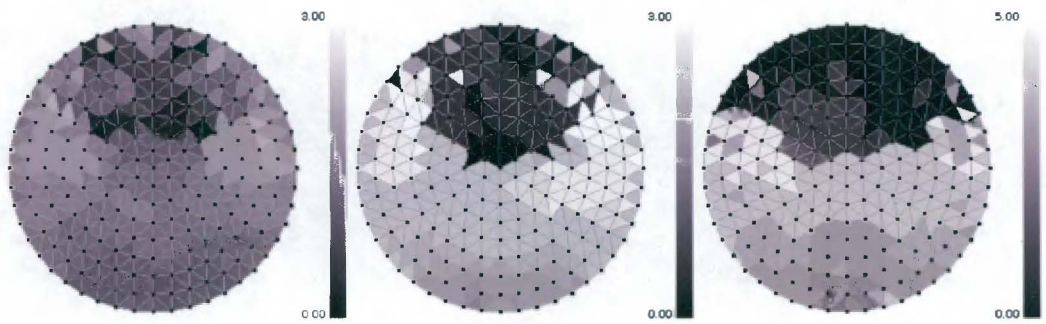
The correlation profile for a pixel in this region is shown in Figure 7.9, note the clear peak of 0.73 for this pixel, indicating good correlation. At the top of the pipe, the correlation is very low and the estimated velocities in this region should be discarded. Figure 7.10 shows a typical correlation profile for a pixel with poor correlation.

Figures 7.7 and 7.8 clearly show the increase in velocity as the pump speed is increased. Also note that in the sliding bed region of the flow, the velocity is shown to increase




(a) (b) (c)

Figure 7.6: Correlation coeffients in the flow (a) low pump speed (800rpm) (b) medium pump speed (1300rpm) (c) high pump speed (1700rpm)

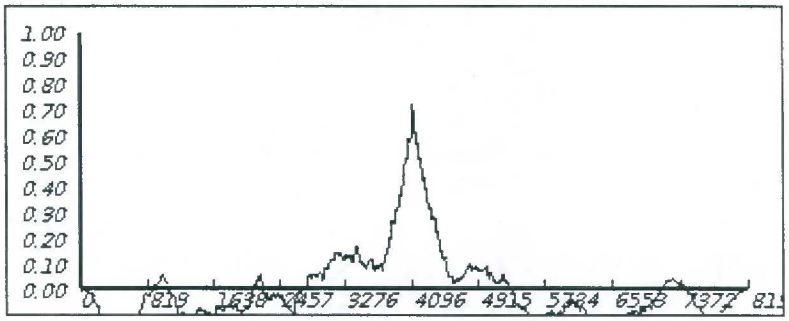



(a) (b) (c)

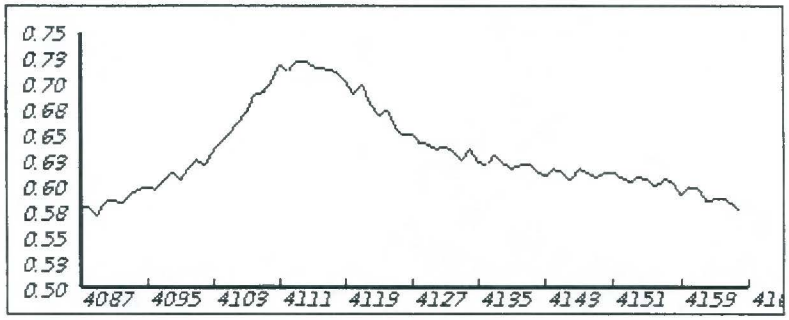
Figure 7.7: Velocity profiles of the flow (a) low pump speed (800rpm) (b) medium pump speed (1300rpm) (c) high pump speed (1700rpm) - Note the change in scale

(a) (b) (c)

Figure 7.8: Heightmap representation of the flow velocity profiles (a) low pump speed (800rpm) (b) medium pump speed (1300rpm) (c) high pump speed (1700rpm)




(a)




(b)

Figure 7.9: A correlation peak using the fast correlation algorithm of the pixel near the bottom of the pipe. (b) is a zoomed in version of the peak in (a)

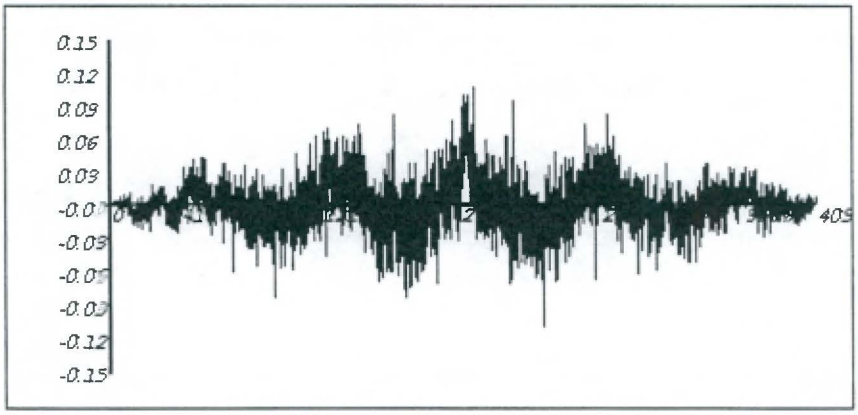
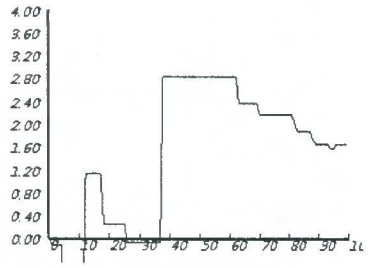



Figure 7.10: The correlation profile for a poorly correlated pixel




(a)

Figure 7.11: Velocity profile using the 384 element mesh.

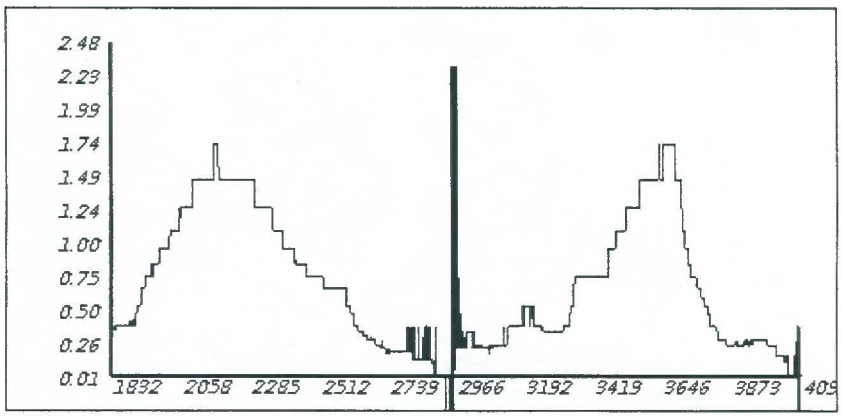
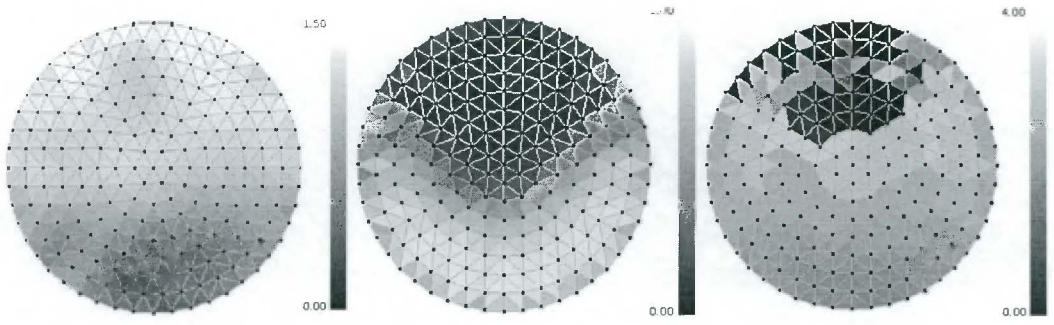



Figure 7.12: History plot of estimated velocities of a pixel near the bottom of the pipe

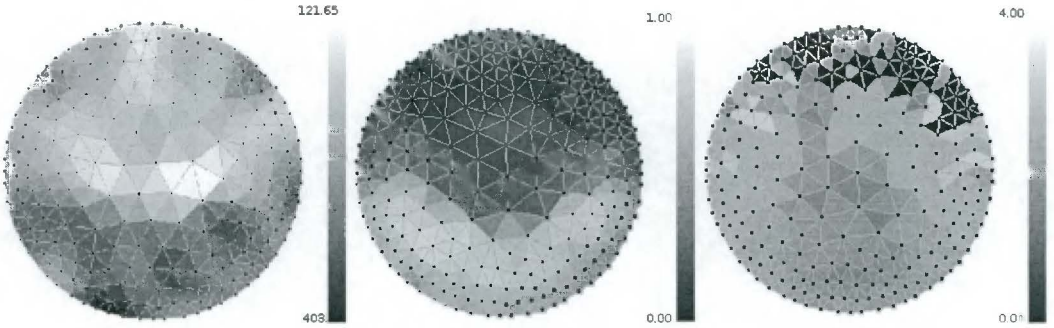
from bottom to top. This is more apparent in Figure 7.11 which represents the estimated velocities along a narrow slice vertically through the cross section of the typical sliding bed flow similar to the examples shown in Figure 7.7. The horizontal scale represents increasing depth, a value of 50 corresponding to the flow at the center of the pipe. Note once again that there was generally poor correlation in the upper regions of the flow and these velocity estimates should be again discarded. Observation of the flow supports these results as the particles at the top of the sliding bed were seen to move faster than those below.

The ability of the system to track changes in velocity is shown in Figure 7.12. This graph was generated by tracking the velocity of a pixel near the bottom of the pipe while the pump speed was varied. The spikes in the middle and at the end of the plot occur because the flow stopped and no correlation was possible. No interpolation of the correlation profile was done on the results for this image, hence quantized velocity steps.




(a) (b) (c)

Figure 7.13: Velocity profile using the 2D image reconstruction algorithm (a) Conductivity profile (b) Correlation coefficients (c) Velocity Profile




(a) (b) (c)

Figure 7.14: Velocity profile using the 3D image reconstruction algorithm (a) Conductivity profile (b) Correlation coefficients (c) Velocity Profile

### 7.2.3 Comparison of 2D and 3D Algorithms

To determine if there is any advantage in using the modified 3D algorithm, in which only the elements in the plane of the electrodes are used, over the 2D algorithm, both were used to generate a velocity profile using the same data. Figure 7.13 shows the results for the 2D algorithm using the 384 element mesh, and Figure 7.14 using the 3D algorithm with the 440 element mesh. A comparison of part (a) is both figures reveals that the 2D algorithm seems to produce a more accurate representation of the conductivity profile. This is most likely due to the fact that the 3D geometry and regularisation is more suited to a iterative algorithm. Part (b) of the figures show that the reconstructed conductivities using 2D algorithm correlate better over a wider area. The velocity profiles produced, shown in part (c), show that the 2D algorithm performs better.

### 7.2.4 Further Testing

Additional data sets were captured using different slurry concentrations, different electrode plane separations and different of pump speed settings. This was done on a later occasions to the first data capture series and in the presence of significant electrical noise. In most of these cases, there was very poor correlation between the measuring planes despite using increasing window sizes.

It is believed that the lack of correlation in these data sets arose from the fact that there was too much noise on the voltage measurements which corrupted the correlation signals. The level of measurement noise was significantly higher at the FPRC site on this occasion than under laboratory conditions. The standard deviation of the voltages was in the order of 20mV on a 1V peak, in contrast to the less that 2mV recorded under ideal conditions. The electrical noise is likely to be introduced into the system by the thyristor drives in the mixing tank and pump.

The FPRC pipe test facility is currently being fitted with appropriate noise supressors and an improved mains supply layout to reduce the noise in the system.

# Chapter 8

# Conclusions and Recommendations

This chapter draws conclusions about the system and makes some recommendations for future work.

## 8.1 Conclusions

A tool to visualise the flow of slurries at the Cape Peninsula University of Technology was developed.

The system is fast enough to allow real time display of the flow concentration, for example, when using the 440 element mesh, image reconstruction for two electrode planes takes approximately 2ms. This suggests that this aspect of the system could potentially be used in a real time control application to monitor the concentration in the pipe.

The results presented in Chapter 7 show that this system was also fast enough to produce velocity profiles of the flow in an on-line fashion without dropping any data. Although the fast correlation calculation with the 440 element mesh and a window size of 4096 samples can be completed in approximately 750ms, this combination in reality limits the system to a velocity profile update rate of 0.54 frames per second. The difference arising from the time taken to reconstruct the conductivity images used in the correlation.

The update rate can be increased using a coarser finite element mesh and smaller correlation windows.

The velocity profiles generated for a 5% kaolin, 10% sand mixture seemed to closely reflect the nature of the flow for pump speeds ranging from 800rpm to 1700rpm when

data was captured by electrode planes separated by 100mm. Attempts were made to generate velocity profiles for other mixtures and electrode plane separations, however the data used in these cases had significant amounts of noise and correlations were generally poor, making it impossible to produce coherent velocity profiles.

Using a modified one step 3D algorithm, in which only the elements in the plane of the electrodes are used, did not seem to add any benefit over the 2D algorithm with point electrodes. In fact, the results were considerably worse using this algorithm.

## 8.2 Recommendations

The accuracy of the velocity profiles that were generated was not verified. The FPRC pipe test flow loop is fitted with a number of other instruments [56] that might be used to verify the results of the on-line system.

Although the maximum update rates for the velocity profile are relatively low (0.54 frames per second), they are further reduced because of the time spent by the processor displaying visualisations of the profile. Using a dedicated graphics card would not only speed up the display of the images, it would also free up the CPU, allowing it to speed more time doing data processing. A dedicated graphics card has another potential benefit. Modern cards provide access to the graphics processor or GPU, a fact that has been utilised by a number of groups who provide FFT libraries that run on the GPU, for example [24]. Moving the correlation calculations to a graphics card has the potential for significant speed improvements and should be investigated.

Chapter 4 discussed the 'best correlated pixel' method which promises more accurate velocity profiles and allows flow vectors to be calculated. Having established that the current system is more than fast enough (for small window sizes) to operate on-line, implementation of this method is possible.

During this thesis, the UCT instrument was operating at 588 one layer frames per second although it has the potential to operate at up to 1000 frames per second. Using a higher frame rate improves the velocity discrimination and allows a wider range of velocities to be investigated.

Velocity profiles could not be established using the data sets from flows of many slurries. It is believed that this was due to the signals being swamped by electrical noise. The FPRC test rig is currently moving premises and it is hoped that good wiring practice will

supress some of the noise from the thyristor drives. Further studies in the new location should be done to evalute the system's preformance with reduced noise.

# Bibliography


[1] A. Adler and W. R. B. Lionheart. Uses and abuses of eidors: an extensible software base for eit. *Physiological Measurement*, 27(5):S25–S42, 2006.

[2] N. J. Avis and D. C. Barber. Image reconstruction using non-adjacent drive configurations (electric impedance tomography). *Physiological Measurement*, 15(2A):A153–A160, 1994.

[3] D. C. Barber and B. H. Brown. Applied potential tomography. *Journal of Physics E: Scientific Instruments*, 17(9):723–733, 1984.

[4] M. Beck and A. Plaskowski. *Cross Correlation Flowmeters*. Bristol: IOP Publishing Ltd, 1987.

[5] M. S. Beck and R. A. Williams. Process tomography: a european innovation and its applications. *Measurement Science and Technology*, 7(3):215–224, 1996.

[6] R. S. Blue, D. Isaacson, and J. C. Newell. Real-time three-dimensional electrical impedance imaging. *Physiological Measurement*, 21(1):15–26, 2000.

[7] G. T. Bolton, C. W. Hooper, R. Mann, and E. H. Stitt. Flow distribution and velocity measurement in a radial flow fixed bed reactor using electrical resistance tomography. *Chemical Engineering Science*, 59(10):1989–1997, May 2004.

[8] L. Borcea. Electrical impedance tomography. *Inverse Problems*, 18(6):R99–R136, 2002.

[9] A. Borsic. *Regularisation Methods for Imaging from Electrical Measurements*. PhD thesis, Oxford Brookes University, 2002.

[10] B. H. Brown. Electrical impedance tomography (eit): a review. *Journal of Medical Engineering & Technology*, 27(3):97–108, May/June 2003.

[11] M. Cheney, D. Isaacson, J. C. Newell, S. Simske, and J. Goble. Noser: An algorithm for solving the inverse conductivity problem. *International Journal of Imaging Systems and Technology*, 2(2):66–75, 1990.

[12] M. Cheney, D. Isaacson, and J. C. Newelly. Electrical impedance tomography. *SIAM Review*, 41(1):85–101, 1999.

[13] J. J. Cilliers, W. Xie, S. J. Neethling, E. W. Randall, and A. J. Wilkinson. Electrical resistance tomography using a bi-directional current pulse technique. *Measurement Science and Technology*, 12(8):997–1001, 2001.

[14] R. Cooke. Laminar flow settling: the potential for unexpected problems. In *15th International Conference on Hydrotransport*, 2002.

[15] J. L. Davidson, L. S. Ruffino, D. R. Stephenson, R. Mann, and B. D. G. T. A. York. Three-dimensional electrical impedance tomography applied to a metal-walledfiltration test platform. *Measurement Science and Technology*, 15(11):2263–2274, 2004.

[16] X. Deng, F. Dong, L. J. Xu, X. P. Liu, and L. A. Xu. The design of a dual-plane ert system for cross correlation measurement of bubbly gas/liquid pipe flow. *Measurement Science and Technology*, 12(8):1024–1031, 2001.

[17] F. Dickin and M. Wang. Electrical resistance tomography for process applications. *Measurement Science and Technology*, 7(3):247–260, 1996.

[18] F. Dong, Z. X. Jiang, X. T. Qiao, and L. A. Xu. Application of electrical resistance tomography to two-phase pipe flow parameters measurement. *Flow Measurement and Instrumentation*, 14(4-5):183–192, 2003.

[19] F. Dong, Y. Xu, L. Xu, L. Hua, and X. Qiao. Application of dual-plane ert system and cross-correlation technique to measure gas-liquid flows in vertical upward pipe. *Flow Measurement and Instrumentation*, 16(2-3):191–197, 2005.

[20] T. Dyakowski. Process tomography applied to multi-phase flow measurement. *Measurement Science and Technology*, 7(3):343–353, 1996.

[21] L. M. Heikkinen, J. Kourunen, T. Savolainen, P. J. Vauhkonen, J. P. Kaipio, and M. Vauhkonen. Real time three-dimensional electrical impedance tomography applied in multiphase flow imaging. *Measurement Science and Technology*, 17(8):2083–2087, 2006.

[22] L. M. Heikkinen, T. Vilhunen, R. M. West, and M. Vauhkonen. Simultaneous reconstruction of electrode contact impedances and internalelectrical properties: Ii. laboratory experiments. *Measurement Science and Technology*, 13(12):1855–1861, 2002.

[23] M. Henningsson, K. Ostergren, and P. Dejmek. Plug flow of yoghurt in piping as determined by cross-correlated dual-plane electrical resistance tomography. *Journal of Food Engineering*, 76(2):163–168, Sept. 2006.

[24] http://gamma.cs.unc.edu/GPUFFTW. *GPUFFTW: High Performance Power-of-Two FFT Library using Graphics Processors.*

[25] http://www.fftw.org. *The Fastest Fourier Transform in the West.*

[26] http://www.wxWidgets.org. *wxWidgets library.*

[27] P. Hua and E. Woo. *Electrical Impedance Tomography*, chapter Reconstuction Algorithms, pages 97–136. The Adam Hilger Series on Biomedical Engineering. Bristol: IOP Publishing Ltd, 1990.

[28] A. Hunt, J. D. Pendleton, and R. B. White. A novel tomographic flow analysis system. In *Preceedings of 3rd World Congress on Industrical Process Tomography*, 2003. See conference proceedings.

[29] A. L. Hyaric and M. K. Pidcock. A one step image reconstruction algorithm for electrical impedance tomography in three dimensions. *Physiological Measurement*, 21(1):95–98, 2000.

[30] O. Isaksen. A review of reconstruction techniques for capacitance tomography. *Measurement Science and Technology*, 7(3):325–337, 1996.

[31] G. A. Johansen, T. Frøystein, B. T. Hjertaker, and O. Olsen. A dual sensor flow imaging tomographic system. *Measurement Science and Technology*, 7(3):297–307, 1996.

[32] J. R. Leitner. *Slurry Flowmetering Using Correlation Techniques.* PhD thesis, University of Cape Town, 1979. In library.

[33] W. R. B. Lionheart. Eit reconstruction algorithms: pitfalls, challenges and recent developments. *Physiological Measurement*, 25(1):125–142, 2004.

[34] S. B. Lippman and J. Lajoie. *C++ Primer.* Addison-Wesley Professional, 3rd edition, April 1998.

[35] R. K. Livesley. *Finite Elements: An introduction for electrical engineers.* Cambridge University Press, 1982.

[36] T. M. Long. A software front end for a real-time electrical resistance tomography imaging system. Undergraduate Thesis, 2003.

[37] G. P. Lucas, J. Cory, R. C. Waterfall, W. W. Loh, and F. J. Dickin. Measurement of the solids volume fraction and velocity distributions in solids-liquid flows using dual-plane electrical resistance tomography. *Flow Measurement and Instrumentation*, 10(4):249–258, Dec. 1999.

[38] Y. Ma, Z. Zheng, L.-a. Xu, X. Liu, and Y. Wu. Application of electrical resistance tomography system to monitor gas/liquid two-phase flow in a horizontal pipe. *Flow Measurement and Instrumentation*, 12(4):259–265, Aug. 2001.

[39] The MathWorks, Inc. *Matlab ® Help.*

[40] M. Molinari, B. H. Blott, S. J. Cox, and G. J. Daniell. Optimal imaging with adaptive mesh refinement in electrical impedance tomography. *Physiological Measurement*, 23(1):121–128, 2002.

[41] M. Molinari, S. J. Cox, B. H. Blott, and G. J. Daniell. Comparison of algorithms for non-linear inverse 3d electrical tomography reconstruction. *Physiological Measurement*, 23(1):95–104, 2002.

[42] N. Morrison. *Introduction to Fourier Analysis.* John Wiley & Sons, 1994.

[43] V. Mosorov. A method of transit time measurement using twin-plane electrical tomography. *Measurement Science and Technology*, 17(4):753–760, 2006.

[44] V. Mosorov, D. Sankowski, L. Mazurkiewicz, and T. Dyakowski. The &lquot;best-correlated pixels&rquot; method for solid mass flow measurementsusing electrical capacitance tomography. *Measurement Science and Technology*, 13(12):1810–1814, 2002.

[45] T. Naidoo. Signal and image processing for electrical resistance tomography. Master's thesis, University of Cape Town, 2002.

[46] G. Nutt. *Operating Systems. A Modern Perspective.* Addison Wesley Longman, Inc, 2002.

[47] P. A. T. Pinheiro and F. J. Dickin. Sparse matrix methods for use in electrical impedance tomography. *International Journal For Numerical Methods In Engineering*, 40(3):439–450, 1997. Got the hard copy.

[48] P. A. T. Pinheiro, W. W. Loh, and F. J. Dickin. Smoothness-constrained inversion for two-dimensional electrical resistancetomography. *Measurement Science and Technology*, 8(3):293–302, 1997.

[49] N. Polydorides and W. R. B. Lionheart. A matlab toolkit for three-dimensional electrical impedance tomography:a contribution to the electrical impedance and diffuse optical reconstructionsoftware project. *Measurement Science and Technology*, 13(12):1871–1883, 2002.

[50] W. H. Press, S. A. Teukolsky, W. T. Vettering, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

[51] L. Pullum and L. Graham. A new high-concentration pipeline test loop facility. In *14th International Conference on Slurry Handling and Pipeline Transport*, 1999.

[52] E. Randall, A. Wilkinson, J. Cilliers, and S. Neethling. Current pulse technique for electrical resistance tomography measurements. In *Proceedings of 2nd World Congress on Industrial Process Tomography*, 2001. see proceedings.

[53] E. Randall, A. Wilkinson, T. Long, and A. Collins. The design of a flexible multilayer ert system and an evaluation of its performance. In *Proceedings of 4th World Congress on Industrial Process Tomography*, volume 1, pages 94–99, 2005.

[54] E. Randall, A. Wilkinson, T. Long, and A. Sutherland. A high speed current pulse electrical resistance tomography system for dynamic process monitoring. In *Proceedings of EDSA*, 2004.

[55] J. Schöberl. *NetGen -Automatic Mesh Generator*. http://www.hpfem.jku.at/netgen/.

[56] A. P. N. Sutherland, P. T. Slatter, A. J. Wilkinson, E. W. Randall, and T. M. Long. A pipe test facility to examine laminar shear settling using electrical resistance tomography. In *Proceedings of 4th World Congress on Industrial Process Tomography*, 2005.

[57] M. Vauhkonen. *Electrical impedance tomography and prior information*. PhD thesis, Kuopio University, 1997.

[58] T. Vilhunen, J. P. Kaipio, P. J. Vauhkonen, T. Savolainen, and M. Vauhkonen. Simultaneous reconstruction of electrode contact impedances and internalelectrical properties: I. theory. *Measurement Science and Technology*, 13(12):1848–1854, 2002.

[59] M. Wang. Inverse solutions for electrical impedance tomography based on conjugate gradients methods. *Measurement Science and Technology*, 13(1):101–117, 2002.

[60] M. Wang and Y. Ma. Over-zero switching scheme for fast data collection operation in electrical impedance tomography. *Measurement Science and Technology*, 17(8):2078–2082, 2006.

[61] M. Wang, Y. Ma, N. Holliday, Y. Dai, R. Williams, and G. Lucas. A high-performance eit system. *IEEE Sensors Journal*, 5(2):289–299, April 2005.

[62] M. Wang, W. Yin, and N. Holliday. A highly adaptive electrical impedance sensing system for flow measurement. *Measurement Science and Technology*, 13(12):1884–1889, 2002.

[63] E. W. Weisstein. Correlation. From MathWorld - A Wolfram Web Resource http://mathworld.wolfram.com/Correlation.html, 1999.

[64] J. L. Wheeler, W. Wang, and M. Tang. A comparison of methods for measurement of spatial resolution in two-dimensional circular eit images. *Physiological Measurement*, 23(1):169–176, 2002.

[65] A. Wilkinson, E. Randall, J. Cilliers, D. Durrett, T. Naidoo, and T. Long. A 1000-measurement frames/second ert data capture system with real-time visualization. *IEEE Sensors Journal*, 5(2):300–307, April 2005. See journal in Bill's collection.

[66] A. Wilkinson, E. Randall, D. Durrett, T. Naidoo, and J. Cilliers. The design of a 1000 frames/second ert data capture system and calibration techniques employed. In *Proceedings of 3rd World Congress on Industrial Process Tomography*, pages 504–509, 2003.

[67] A. J. Wilkinson, E. W. Randall, T. M. Long, and A. Collins. The design of an ert system for 3d data acquisition and a quantitative evaluation of its performance. *Measurement Science and Technology*, 17(8):2088–2096, 2006.

[68] M. Yang, H. I. Schlaberg, B. S. Hoyle, M. S. Beck, and C. Lenn. Parallel image reconstruction in real-time ultrasound process tomography for two-phased flow measurements. *Real-Time Imaging*, 3(4):295–303, Aug. 1997.

[69] W. Q. Yang and M. S. Beck. An intelligent cross correlator for pipeline flow velocity measurement. *Flow Measurement and Instrumentation*, 8(2):77–84, June 1998.

[70] T. York. Status of electical tomography in industrial applications. *Journal of Electronic Imaging*, 10:608–619, 2001.

[71] T. York, S. Murphy, A. Burnett-Thompson, and B. Grieve. An accessible electrical impedance imaging system. In *Proceedings of 4th World Congress on Industrial Process Tomography*, 2005.