An Automated Camera Tracking System for Single Bubble Velocity Profiling

Stewart Charlton Reid

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

Cape Town, February 2007

Declaration

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

signature removed

Signature of Author .....

Cape Town

February 16, 2007

Abstract

An apparatus has been developed for the automated velocity profiling of single bubbles in aqueous solutions. Using a mobile IEEE 1394 camera and a control algorithm, the apparatus can automatically track a bubble as it rises in a Perspex column. The resulting video footage is then post-processed to obtain the velocity profile of the bubble, amongst other data. This apparatus is to be used in research relevant to the minerals processing industry.

The software components were developed in Matlab and Microsoft Visual C++ and implemented in Visual C++.

The methodology used to develop the apparatus is presented, and typical examples of velocity profiles are given. An analysis to determine the effect of error of parrallax is also performed.

To my grandparents

Walter and Grace Switzer

In gratitude for all the support you have given me through the years.

Acknowledgements

I owe a great debt of gratitude to the many people who have contributed to this project in some way. In particular:

My supervisor, Dr Andrew Wilkinson, who was always capable of providing encouragement and inspiration when I most needed it, and helping oversee the project from beginning to end.

My co-supervisor, Mr Martin Harris, for his help with the chemical engineering aspects of this dissertation.

Prof. Martin Braae and Dr Fred Nicolls, who provided insight with regards to the control theory and imaging aspects for this project respectively.

Mr Bill Randall, whose continuing interest and contributions have greatly affected this project.

Sameer Morar, for his help in the operation of the apparatus during testing and operation.

Granville de la Cruz, Kenneth Maseko and the Chemical Engineering Workshop for their help with the construction and hardware components of the apparatus.

Heather Sundstrom, who provided financial administration during the course of this project.

My family, Grace, Peter and Liz Reid, for their continuing support and encouragement.

Contents

Acknowledgements

I owe a great debt of gratitude to the many people who have contributed to this project in some way. In particular:

My supervisor, Dr Andrew Wilkinson, who was always capable of providing encouragement and inspiration when I most needed it, and helping oversee the project from beginning to end.

My co-supervisor, Mr Martin Harris, for his help with the chemical engineering aspects of this dissertation.

Prof. Martin Braae and Dr Fred Nicolls, who provided insight with regards to the control theory and imaging aspects for this project respectively.

Mr Bill Randall, whose continuing interest and contributions have greatly affected this project.

Sameer Morar, for his help in the operation of the apparatus during testing and operation.

Granville de la Cruz, Kenneth Maseko and the Chemical Engineering Workshop for their help with the construction and hardware components of the apparatus.

Heather Sundstrom, who provided financial administration during the course of this project.

My family, Grace, Peter and Liz Reid, for their continuing support and encouragement.

List of Figures

List of Tables

List of Symbols

C	—	Resultant correlation image
D	—	Differential term (PID controller)
d_e	—	Effective bubble diameter
d_l	—	Lower distance from bubble to number (pixels)
d_u	—	Upper distance from bubble to number (pixels)
e	—	Error value (for controller)
$e(s)$	—	Error (laplacian)
$e(z)$	—	Error (z-transform)
e_i	—	Error (difference equation)
F	—	FFT operator
G	—	Gain (plant/mechanical system model)
g	—	Gradient
$g(s)$	—	Plant/mechanical system model (laplacian)
h_b	—	Calculated bubble height
I	—	Image array (cross-correlation)
$\overline{I}$	—	Mean of the elements in I
i	—	Horizontal co-ordinate in image (cross-correlation)
j	—	Vertical co-ordinate in image (cross-correlation)
K	—	Kernel array (cross-correlation)
$\overline{K}$	—	Mean of the elements in K
$k(s)$	—	Controller model (lapacian)
$k(z)$	—	Controller model (z-transform)
M	—	Kernel width
N	—	Kernel height
n	—	Tape measure number (height marking in cm)
P	—	Proportional term (PID controller)
P_C	—	Hydrostatic pressure due to column
P_R	—	Hydrostatic pressure due to reservoir
$r(s)$	—	Ramp setpoint (laplacian)
s	—	Lapacian variable
T	—	Time constant (PID controller)
T_m	—	Mechanical time constant

t	—	Time
U_{const}	—	Terminal bubble velocity
U_{max}	—	Maximum bubble velocity
$u(s)$	—	Control voltage (laplacian)
$u(z)$	—	Control voltage (z-transform)
u_i	—	Control voltage (difference equation)
u_x	—	Horizontal uncertainty
u_y	—	Vertical uncertainty
V	—	Control voltage (or simply, Volts)
$\hat{v_b}$	—	Estimated bubble velocity
x	—	Horizontal co-ordinate
y	—	Vertical co-ordinate
Γ	—	Integral term (PID controller)
σ_I	—	Standard deviation for image region
σ_K	—	Standard deviation for kernel array

Nomenclature

1D– One-dimentional

2D– Two-dimentional

AC– Alternating current

C/L– Closed loop

Column– In image processing, refers to arrays formed from an image by grouping elements vertically

DC– Direct current

Digit– Single numeral within a number, usually in the context of correlation kernels for the tape measure image processing

FFT– Fast fourier transform, an implementation of the discrete fourier transform

FPS– Frames per second

Number– Multi-digit length marker on the tape measure

O/L– Open loop

PGM– Portable grayscale map image format

PID– Proportional-integral-derivative controller

PPM– Parts per million

RAM– Random access memory (volatile, high speed memory in a computer)

Ray– Beam of light associated with a single point (pixel) in an image

Row– In image processing, refers to arrays formed from an image by grouping elements horizontally

RPM– Revolutions per minute

Chapter 1

Introduction

1.1 Froth Flotation Overview

Froth flotation is a selective process used widely in industry to perform separation in aqueous mixtures. It is primarily associated with the field of minerals processing, though it is also used in the processing of a variety of non-metals [11, 23].

The process begins with comminution, where the ore is broken down into a powder. This increases its surface area, exposing more of the ore for processing, unlocks and liberates the minerals from the gangue and reduces the size of the component mineral particles to be suitable for flotation (less than 500 μm according to [11], but often practically less than 100 μm).

The powder is placed in a flotation cell, where surfactants are used to render certain particles - usually the valuable minerals - hydrophobic. These particles attach to air bubbles dispersed in the cell and eventually gather in a froth layer that forms at the top of the cell. The froth layer is removed as concentrate for further processing.

Despite its extensive use, flotation is still one of the least understood engineering operations [8, 10], making control and efficient recovery in industrial flotation difficult. Work to mathematically model froth flotation effectively is underway in numerous institutions (see [10, 16]), including the Centre for Minerals Research (CMR) at the University of Cape Town (UCT). Research by Sam et al [17] has shown that bubble behaviour in the pulp phase (the mixture below the froth) is an important component of such a model, as it is in this phase in which, amongst other processes, bubble-particle attachment and detachment occur, as well as surfactant adsorption onto the surface of the bubble. These processes can affect a bubble's velocity as it rises, which has indicated the importance of studying of the axial velocity profiles of single bubbles in surfactant solutions.

1.2 Bubble Velocity Profiling

A bubble tracking apparatus was developed in the mid-1990s at the Department of Mining and Metallurgical Engineering, McGill University in Montreal, Canada. Using a mobile video camera, a single bubble rising in a column containing surfactant solution could be continuously observed over a distance of four meters. The velocity plots that were obtained with this apparatus indicated that bubble terminal rise velocity was not dependent on surfactant concentration, as had been proposed by Fdhila and Duineveld [4]. Instead, they indicated towards an entirely new model for bubble behaviour involving the adsorption of surfactant molecules onto the bubble's surface [24].

The work done in developing this new model (see [16, 17, 25] amongst others) has shown the importance to research of the ability to continuously monitor the rise velocity of a bubble.

As a result, the CMR at UCT commissioned the development of a bubble tracking camera system based on the one at McGill university. The method used would include a mobile video camera, mounted on a track next to a column in which the bubble would rise. While a method involving several stationary cameras (positioned along the height of the column) was considered, the single mobile camera method was preferred, as it was closer in design to the McGill apparatus. The multiple camera option is an interesting avenue for research, and is mentioned in the recommendations (Section 6.2), but this dissertation is concerned with the single mobile camera method implemented.

1.3 The UCT Bubble Tracking Camera System

The apparatus at McGill university was well designed in many respects. It was constructed out of flat perspex sheets to reduce optical distortion when observing the inside of the column. It was double layered, allowing for a temperature controlled environment, and it eventually made use of a differential pressure system to facilitate bubble release. These features were replicated in the design of the UCT apparatus.

There were, however, some inconveniences encountered with the McGill apparatus, mostly involving the processing of the video footage from tracking runs. Cesar Gomez (McGill University) has described that the tracking itself is a skill that can be learned, but that most people are discouraged by the idea of going through hundreds of frames (captured 30 times a second) and plotting the bubble's positions and velocity readings. Consequently, results published based on data from the McGill apparatus are often obtained only from every third image. While there may be advantages and disadvantages to sub-sampling in this way, it would be optimal to have all data processed so that it can be filtered if it is necessary or desired.

It was thus decided to automate the UCT apparatus so that all data from a tracking run could be obtained from every frame without the cost of human labour, and so that the

apparatus could be operated easily and efficiently.

Unfortunately, very little has been published with regard to the bubble tracking apparatus developed at McGill University - the focus has been on the results from research it facilitates. The description in [16] is somewhat outdated - much of the above discussion is from personal communication and other unpublished sources. Additionally, a literature survey conducted at the beginning of the bubble tracking project at UCT yielded only one result for such specialized functionality. This was a description of an apparatus used in the investigation of the rising speed and dissolution rate by carbon dioxide bubbles by Takemura and Yabe [18]. Their apparatus was capable of automated tracking, but once again the authors did not go into great detail about its functionality, and the video analysis still had to be performed by hand.

Thus, there was no possibility of using a prefabricated solution for the project of automating the apparatus. A solution would have to be developed from the ground up for the tracking of bubbles as they rise, and for the image analysis of the resultant video footage. This would constitute a novel application of automation to the bubble velocity profiling problem.

1.4 Problem Specification

The bubbles that are of primary interest for this project are small enough to be non-oscillating ($0.4 > d_e > 1.4$ mm [17]) and rise in isolation. Figure 1.1, from [8], gives a representative plot of the bubble velocity profile for such bubbles in various solutions.

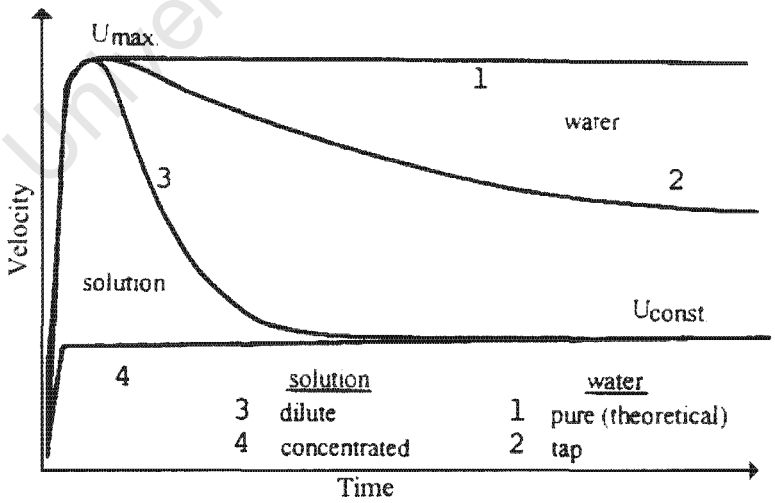

Figure 1.1: Typical single bubble velocity profiles [8].

After the bubble is released there is a sharp increase in velocity up to a maximum (U_{max}), followed by a decay to terminal rise velocity (U_{const}). In cases of high concentration, U_{max} may not even be reached, whereas for distilled water, U_{max} is theoretically the same as U_{const}.

The decay in velocity from U_{max} is caused by the adsorption of molecules in solution (surfactant molecules, or contaminants in tap and impure distilled water), which results in drag as the bubble rises. The rate of this decay, and the terminal velocity, are dependent upon a number of factors including bubble size, type of surfactant and concentration [17].

An analysis of some of the results obtained from the McGill apparatus [16, 17, 25, 24] indicated that, for the bubbles of interest, U_{max} never exceeded 40 cm/s. This limit was used as the design criterion for the maximum tracking speed - such a system would have to be able to track a 40 cm/s bubble and output a plot such as the one in Figure 1.1 with the bubble's actual velocity profile.

The main aim of this dissertation is to describe the system developed to allow this effect to be further studied, providing a record of its functionality as well as the problems encountered and the solutions provided.

1.5 Dissertation Structure

The format of this dissertation is as follows:

- Chapter 2 describes the hardware components and construction of the UCT bubble tracking apparatus.
- Chapter 3 describes the development and implementation of the image processing and control theory relating to the tracking aspects of the apparatus.
- Chapter 4 describes the algorithm used to post-process images from tracking runs and extract the bubble positional and velocity data.
- Chapter 5 shows the general performance of the apparatus and its application to experimental data to obtain the velocity profiles of bubbles rising in various conditions. This chapter also contains an analysis of the uncertainty in the post-processing algorithm due to error of parallax.
- Chapter 6 draws conclusions about the development and performance of the apparatus and makes recommendations for the further development of the apparatus.

Chapter 2

Component Description

This chapter describes the hardware components and design aspects of the bubble tracking apparatus at UCT. Supplementary descriptions of the exact design of certain components can be found in [14].

2.1 Overview

Figure 2.1 is a graphic giving a description of the system in its entirety.

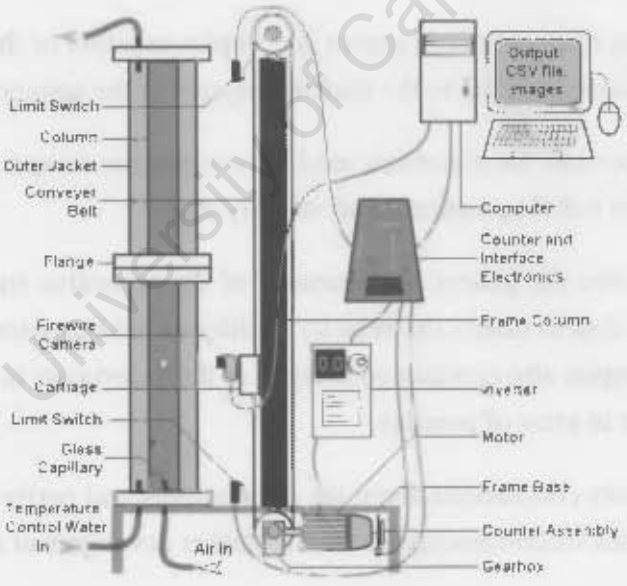

Figure 2.1: Graphic showing UCT's bubble tracking camera system.

The system contains a column which provides a controlled environment in which the bubbles can be generated and tracked. A more detailed description of the column and bubble generating equipment is provided in Section 2.2.

Alongside the column is a track with a conveyor belt. Attached to this is an IEEE 1394 (Firewire) camera. The Firewire camera is covered in more detail in Section 2.4.

The apparatus operates as a large feedback system. In order to track a bubble rising in the column, images of the bubble from the Firewire camera are processed by a PC (Section 2.3), which then uses a control algorithm to output an appropriate control voltage via an Eagle data acquisition (DAQ) PCI card (see Section 2.5). This control signal passes through an electronics interface box (Section 2.6) and operates a transistorized inverter, which drives a motor and gearbox assembly (Section 2.7), which in turn moves the conveyor belt to which the camera is attached. The resulting motion keeps the bubble in the frame of the camera.

The development of the tracking algorithm, described in the paragraph above, is detailed in Chapter 3. The images from these tracking runs are saved and later used in post-tracking processing, to obtain accurate bubble position and velocity data. The post processing algorithm is described in Chapter 4.

2.2 Column

The column consists of two nearly identical sections. Both sections are 2 m tall, and consist of two chambers. The inner chamber has a square horizontal cross section measuring 100 mm on each side, large enough not to have a significant effect on the behaviour of bubbles of the size interest and larger ($d_e > 3$ mm [17]) . The aqueous solution in which the experiment takes place is pumped into this chamber through an opening at the bottom. Provision is made for the release of bubbles through a glass capillary, also at the bottom of this chamber.

The second, outer chamber, completely surrounds the inner chamber. It also has a square cross-section, with each side measuring roughly 180 mm in length. Temperature regulated water can be pumped into the bottom of this chamber and out of the top. This process gradually regulates the temperature of the solution in the inner chamber. This design is based on the apparatus at McGill University [17].

It should be noted that the temperature control system was not yet set up for this apparatus. Furthermore, no temperature sensors have yet been installed in the column. The results obtained in Chapter 5 thus have no corresponding temperature data. This is not a problem for the development of the apparatus, which this dissertation is concerned with, but a temperature regulation system would need to be set up for experimental use. The outer chamber was filled with tap water for the duration of development.

The two column sections are attached with one on top of the other, giving a total height of four meters. The inner chambers of both columns are connected such that a bubble can rise through the connection uninhibited. The outer chambers are also connected, allowing the temperature controlled water to flow through both sections. The column, as it stands in the Milling and Floatation Laboratory at UCT, can be seen in Figure 2.2.

A tape measure is suspended in the inner chamber down the entire length of the column. It is read in software (using an image processing algorithm) and used to obtain accurate

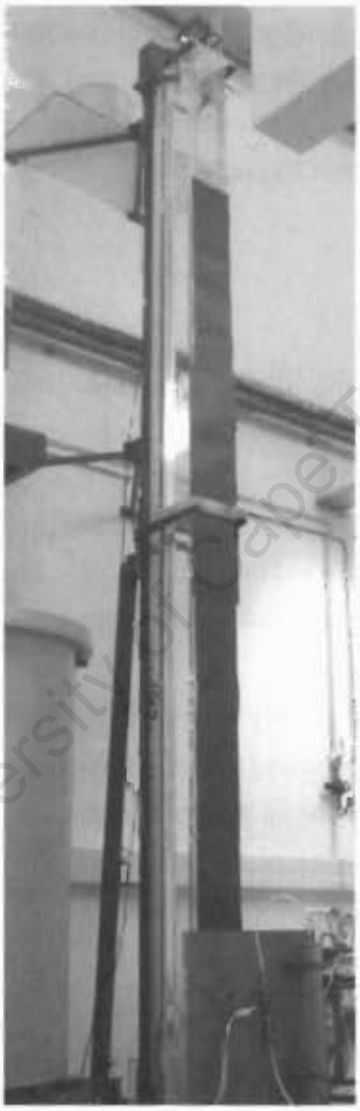

Figure 2.2: Photograph showing column.

bubble height readings during the post-processing of the images. This is described in detail in Chapter 4.

During operation, a 50 W, highly directional halogen lamp is placed underneath the column to provide lighting from below, and a 150 W halogen spotlight provides lighting from above. This method for lighting placement was chosen as it lights up the inside of the column effectively, and does not cause excessive visual noise that could interfere with the camera's view of the bubble and the subsequent image processing.

The connection between the column sections is a flange which is opaque. Consequently, the bubble cannot be seen for a short length of time during tracking. This proved to be a significant complication which had to be dealt with in both the tracking of the bubble and the post-processing of the images. The solutions in both of these cases are discussed in Sections 3.3.5 and 4.6 respectively[1].

2.2.1 Bubble Generation

Numerous methods for generating single[2] bubbles of consistent size can be found in the literature.

Sam [16] made use of a canister of compressed air and a micro valve to inject air into the column through a glass capillary. Bubbles would continuously form and release on the tip of the capillary, resulting in a bubble release frequency that could be adjusted by tuning the micro valve. Using this apparatus, Sam could lower the bubble frequency such that the bubbles could be considered to be rising independently. An early prototype of the UCT apparatus made use of a similar system, though nitrogen was used instead of compressed air.

Takemura and Yabe [18] described an electronic bubble generator for their studies of carbon dioxide bubbles. A thin steel plate with a tiny hole separated the bottom of their column from carbon dioxide gas at a slightly higher pressure. The hole was blocked by a rubber stopper on a rod held in place by a spring. They were able to generate single bubbles by injecting a brief current into a solenoid, which pulled the rod and stopper down long enough for a small amount of gas to escape. Since the camera in their automated tracking system was required to be in motion before the bubble was released, it was important for their apparatus to be able to release bubbles on command. This method could have been and could still be suitable for use with UCT's rig, and is discussed later in this section.

Wu and Gharib [22] used their "gentle-push" method[3] to inject a known volume of air

[1] Two 2 m sections, rather than a single 4 m column, were used for space and logistical reasons. Prototyping was performed with one of the sections only [14]. The column was extended to its full height once space in a laboratory with a high ceiling became available. Additionally, on the occasions when it was required, it was easier to move the column in two sections.

[2] "Single" refers to the fact that the bubble is rising in isolation, or is far enough from other bubbles that it can be considered independent.

[3] Strictly speaking, this method was pioneered by the researchers Saffman and Duineveld, who are cred-

through a glass capillary using a pair of syringes. This method also allowed bubbles to be generated singly, though the slow rate at which the syringes had to be operated would have reduced the predictability of the bubble release time, rendering it unsuitable for applications that require a determinable release time. Nonetheless, this method could also be considered for use with UCT's apparatus.

Finally, the rig at McGill University now makes use of a differential pressure method to generate bubbles, which is described in [21]. This method works by placing a pocket of air between two hydrostatic pressures - the pressure on the tip of the capillary in the column (P_C), and the pressure from a reservoir (P_R). By keeping P_R slightly greater than P_C, a very low flow rate of air through the capillary, and hence a low bubble rate, can be achieved. Figure 2.3 gives a functional diagram of this system.

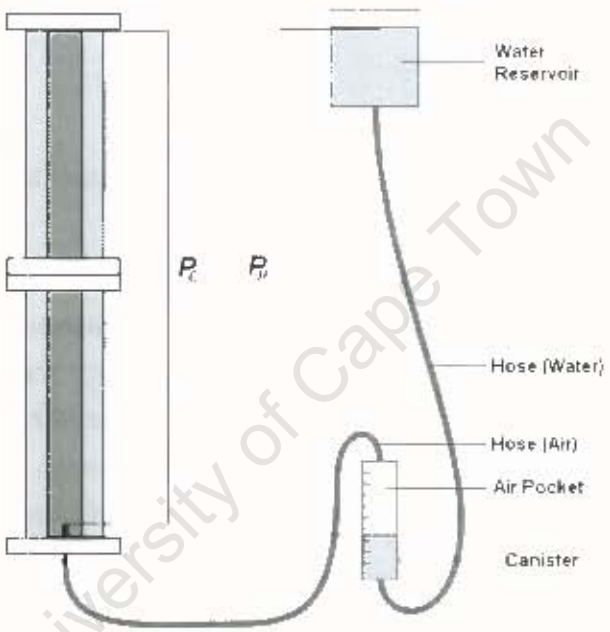

Figure 2.3: Functional diagram of bubble generation system.

It was the differential pressure method that was implemented at UCT, since the UCT design was based on the one from McGill University. This method had already been used at McGill University to generate bubbles of various sizes by using capillaries of different sizes - this functionality was required for the UCT apparatus, and the indications of success with this method in the literature surrounding the McGill apparatus was significant. Additionally, the apparatus was built out of components from the prototype gas canister system mentioned above, including the lower section of the column. This section already had facility for inserting a glass capillary, as required by the differential pressure method. Finally, this method was also easily implemented with equipment already in the UCT lab.

Figure 2.4 shows the equipment related to the maintenance of the air pocket mounted on a board near the base of the column. Besides the parts of the apparatus shown in Figure 2.3, there are also valves attached to the system to allow the air pocket to be replenished,

ited in [22].

since air continuously leaks out through the capillary during operation. The air inlet valve opens the pocket to compressed air at a pressure of approximately 0.5 kPa.

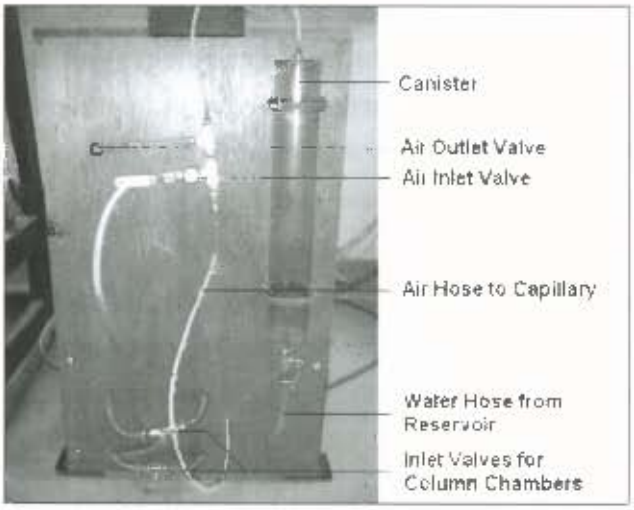

Figure 2.4: Photograph showing board mounted canister and valves.

In most cases where bubbles are released in surfactant solution, the differential pressure method works as expected. However, this is not always the case. In tap water, the bubble frequency does not decay uniformly as air leaks out of the air pocket, but reaches a critical level where bubbles cease to be released at all, despite a forward pressure remaining across the capillary. Single bubbles can then be released with a light tap on the column or frame. Cesar Gomez has also described some practical difficulties experienced with the bubble generating apparatus at McGill, though they have not been enumerated in the literature.

Nonetheless, in tap water, single bubbles suitable for use in the development of the apparatus can be generated using the "tap" mentioned above with one of the glass capillaries set aside for the project. Their rise velocities do not exceed 40 cm/s (initially this had to be estimated, though it was later confirmed by the apparatus itself), and upon release they are small enough that they do not exhibit turbulent motion as they rise, though as they climb higher and the pressure decreases, their volume increases and they begin to oscillate. Thus the bubbles generated with the "tap" method are on the upper limit of the size range of interest for this research. Other available capillaries generate bubbles larger than desired.

A different problem is experienced when using surfactant solutions in the column, particularly when the concentration is relatively high. When the column is filled, a relatively high flow of air through the capillary is necessary to ensure that water does not flow back into the capillary[4]. This implies a strong stream of bubbles that gradually dies away as the column level rises. The bubbles gather on the rising surface of the water/solution, and sometimes attach to the tape measure or the inner walls of the column. This effect is not so pronounced in tap water and lower concentrations of surfactants, but it is preva-

[4]Wetting the inside of the capillary can interfere with the the bubble release, and once water is in the capillary, it is difficult to force it out through such a small opening.

lent in high surfactant concentrations, to the point were it can interfere with tracking and post-processing.

Though development has taken place with the above conditions, the actual operation of the apparatus will require a modification of the bubble generating equipment. Due to the problems with bubble attachment mentioned above, a method that does not require a steady stream of bubbles during set up would be preferred. Both of the following methods meet that requirement, and thus could be considered.

An implementation of Takemura and Yabe's method would require a redesign of the bottom part of the column to put it in place. The ability of their bubble generator to generate bubbles of different sizes remains in doubt as this was not required for their experiments. Their method remains of interest due to the controllable bubble release time, but since the UCT tracking algorithm operates from a standing start, it is not an essential function.

Wu and Gharib's "gentle-push" method seems to hold more promise, since it allows the volume of the bubble generated to be varied. The bubble would be injected into the column through a glass capillary, so it is likely that no column modifications would be needed.

2.3 Personal Computer

A personal computer (PC) provided the platform for the apparatus software, and was used to control the electronic hardware. The PC had an Intel Celeron 1.7 GHz processor and two random access memory (RAM) chips: One 512 Mb, and the other 256 Mb.

The computer ran Microsoft Windows XP Service Pack 2 as an operating system (which resulted in only 632 Mb of the RAM being available).

Software included Matlab 7.1, which was used primarily for software development, Microsoft Visual C++ 6.0, used for development and software implementation, and a number of imaging applications.

All development and eventual results processing were performed on this PC. Timing results given in later sections are thus standardized to this PC.

2.4 Firewire Camera

An IEEE 1394 (Firewire) camera is used to obtain visual information about the position of the bubble in the column. The camera is mounted on a carriage on a track next to the column, allowing vertical mobility. The camera's motion is described in detail in later sections; this section deals with the camera's software specifications.

Figure 2.5 shows the camera as it is mounted on the track.

The driver used was developed at the Robotics Institute of Carnegie Mellon University (available at CMU's IEEE 1394 driver site [`http://www.cs.cmu.edu/~iwan/`

Figure 2.5: Photograph showing camera, carriage and track.

1394/], see also [20]). The camera is operated as follows:

- 640×480 pixels output image size
- 8-bit (grayscale) depth
- 30 frames per second (FPS)

The 640×480 pixels output size is the highest image resolution possible with the camera in use. It should be noted that the camera is turned on its side, so that the 640 pixel axis of the camera is aligned with the rise direction of the bubble. The camera is roughly 8 cm away from the wall of the column; approximately eleven centimeters of the rise axis of the bubble is visible in the camera's frame (see Figure 3.2).

2.4.1 Image Acquisition

The Carnegie Mellon Firewire driver presents the image data as a bottom-up 640×480 (width×height) 24-bit RGB image (despite the 8-bit operation; the grayscale value is simply repeated three times in the output) that is mirrored about the horizontal axis of the image. This is not an ideal format for the purposes of the bubble tracking apparatus, so a function was created to convert the image to a better format.

The output of the conversion function is a top-down 480×640 8-bit grayscale image with no mirroring. A top-down[5] image format was used as it allowed the images to be

[5]Top-down specifies the origin pixel (0,0) to be the top-left of the image, while bottom-up specifies the origin to be the bottom-left.

saved in the simple PGM image format (portable grayscale map [http://netpbm.sourceforge.net/doc/pgm.html]), which is top-down. The 480×640 image size allows the vertical dimension in space to be aligned with the vertical in the image. The converted image has two redundant RGB bytes stripped away, leaving a true 8-bit grayscale image which is smaller in size and allows processing to take place faster. Finally, the mirroring present is removed.

The conversion is performed quickly enough to take place without compromising the 30 FPS frame rate. Thus the converted image format is used in all processing operations including the tracking processing. All images from the Firewire camera in this dissertation are shown in the converted format.

2.4.2 Saving Images

Images are saved on the PC's hard drive in a date and time stamped directory. They are enumerated so that they can be post-processed and viewed in the correct order.

Unfortunately, writing an image to disk takes enough time to compromise the frame rate of the camera, which would cause problems during tracking. As a result, images are stored in the PC's RAM during the tracking run, and written to the hard drive after the run is complete.

2.5 Eagle DAQ Card

Apart from the Firewire camera, input and output to and from the PC is facilitated by a PCI730 DAQ card from Eagle Technology (http://www.eagle.co.za). The card has four analog outputs (only one was used), three byte-wide digital I/O ports, and 16 analog inputs (these were not used). The card also has a built in crystal oscillator counter, which can be used as a timing device.

The analog output is used to provide the control voltage for the inverter, governing the motor's speed, and thus the camera's speed on the track. Two of the digital I/O ports are used to read a counter which gives the position of the camera on the track (see Section 2.6), and the third is used for a series of output tasks, such as setting the inverter direction (allowing the camera to move up or down) and resetting the positional counter.

2.6 Electronics Interface Box

An electronics box, developed in conjunction with Bill Randall of UCT's Department of Chemical Engineering, provides the interface between the PC's DAQ card and the hardware components of the apparatus. It can be seen in Figure 2.6.

The electronics box has two main functions. They are described in the sections below:

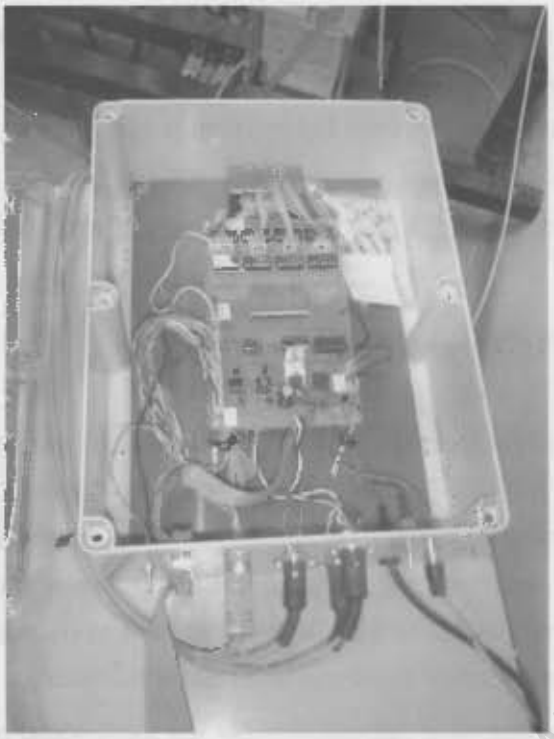

Figure 2.6: Photograph showing electronics box.

2.6.1 Motor Control

The tracking control voltage from the DAQ card is sent to the inverter via an operational amplifier (op amp) circuit for isolation. Direction control, performed digitally by the DAQ card, is passed through a circuit which optically isolates the fragile digital I/O port from the inverter. This circuit also incorporates hardware limit switches (visible in Figure 2.1) to stop the motor's operation should the carriage approach the end of the track.

2.6.2 Optical Counter

The electronics box also contains the circuitry for a counter mechanism that measures the height of the carriage on the track. A toothed wheel is attached to the motor's axle and turns through a pair of optical limit switches which are configured to increment or decrement a counter depending on the turn direction (and hence the vertical direction of the camera).

The counter value can be seen on the display in the electronics box (see Figure 2.6) or read into the computer via two of the digital I/O ports on the DAQ card.

The positional resolution obtained for the camera on the track with this mechanism is approximately 1 mm per counter unit.

The counter is used for the calibration of camera position before runs (so that the same starting point can be used each time), to determine when a tracking run needs to be stopped due to the camera approaching the top of the track.

Initially, the intention was to calculate the absolute bubble height from the counter value

combined with the bubble's frame position. This gave erratic results due to a time lag, and instead the current post-processing method (involving using a tape measure to provide an absolute reference) was used. The time lag problem is discussed in detail in Section 4.1.1.

The original design of the counter circuit was by Alistair Stewart, and the printed circuit board layout for the entire electronics box was performed by Bill Randall.

2.7 Inverter, Motor and Gearbox

The motor operation is performed by a 400 W Mitsubishi transistorized inverter. The inverter is set so that it can be controlled by the external electronics described above. The control voltage and directional information from the electronics box corresponds to a three-phase variable frequency signal. In addition, the inverter automatically filters all input - a step change will result in a smooth adjustment in output frequency to the new value, reducing wear and tear on the motor and related hardware.

The 90 W, three-phase, four-pole induction motor has a top speed of 1675 revolutions per minute (RPM). Combined with a 10:1 ratio worm drive gearbox and a conveyor belt cog radius of 40 mm, the top speed for the carriage was approximately 70 cm/s, which was more than adequate for the maximum anticipated bubble speed of 40 cm/s.

Chapter 3

Tracking Control

3.1 Overview

In order to obtain a continuous plot of bubble velocity, samples of the bubble's position as it rises are required. The tracking algorithm is designed to give such samples in the form of images of the bubble, taken at 30 FPS. This chapter describes the tracking algorithm and its development.

The tracking control loop can be visualized as a block diagram of the form given in Figure 3.1.

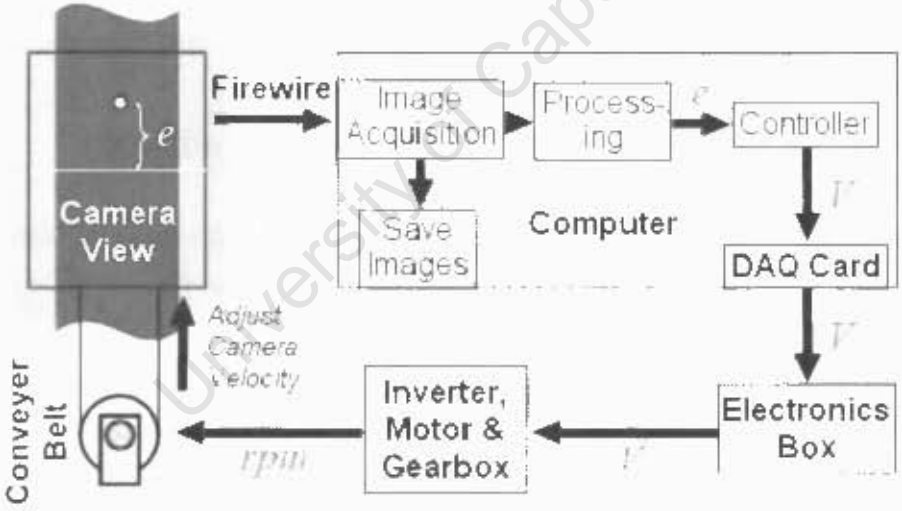

Figure 3.1: Block diagram of tracking control loop.

Section 3.2 describes the initialization of the tracking process. The following two sections describe the two main aspects of the tracking loop - image processing and motor control.

During tracking, images of the bubble in the column are acquired with the Firewire camera. These images are processed to obtain error values (e) for the controller (being the vertical displacement of the bubble from the center of each frame; shown in Figure 3.1). The image processing techniques used to obtain the error are described in Section 3.3.

The controller uses the error values to calculate a control voltage (V) which sets the speed of the motor. The modeling of the motor and other hardware and the development of the controller are described in Section 3.4.

3.2 Initialization

The camera is first moved to a position where the tip of the capillary (see Figure 2.1) is just out of sight below the bottom of the frame. This position is consistently found using the counter-based position control mentioned in Section 2.6.2. Alternatively, it can be set by the operator.

Computer memory (RAM) is set aside for the capture of images during the tracking run. After the tracking run, the images saved in RAM are written to the PC's hard drive. Writing directly to the hard drive during tracking is not possible (with the current PC, at least) as the operation takes too long and slows down the frame rate.

Once all necessary memory and variables have been initialized, the camera begins to acquire images at 30 FPS, and the algorithm searches the bottom of each image for a bubble using the bubble search method described in the following section. The system waits in this state until a bubble is detected near the bottom of the screen.

As soon as a bubble is detected, the algorithm begins tracking it. Images from this point on are saved, the bubble's frame positions are analyzed (Section 3.3) and the tracking control algorithm is started (Section 3.4).

3.3 Image Processing for Tracking Algorithm

A sample image from a tracking run is shown in Figure 3.2. It should be noted that the tape measure, while visible in all images from the tracking run (as is required for post-processing), is not actually used during tracking.

A 1D search algorithm is used to find the bubble in each frame. A 2D correlation was not considered, as it was too expensive in terms of processing time. It was also unnecessary, as only vertical information is important for the camera's motion. The 1D method proved to be both fast and robust.

The image processing to obtain the error value is described in the following sections.

3.3.1 Row Maxima

First, limits are set between which the bubble is expected to remain during the tracking run (shown in Figure 3.2). Since the bubble's lateral displacement seldom exceeds 2 or 3 cm, the region enclosed by the limits can be kept small. The maximum brightness value from the section of each row between the limits is placed into an array. The resulting 640

Figure 3.2: Sample image from a tracking run. Limits between which the bubble is expected are shown. Contrast in the image segment containing the tape measure has been increased for clarity.

element long array has peaks in it according to the heights of the bubbles in the image. A strong contrast between bubble and background can be achieved because the limits cut out most of the noise in the image, as well as the segments that contain bright objects, such as the tape measure.

3.3.2 Filtering and Threshold

The array of row maxima is low-pass filtered with a moving average function of a size based on the estimated size of the bubble. A kernel size of eight pixels was found to be most suitable for this apparatus configuration. The mean and standard deviation of the elements of the filtered array are then calculated, and a threshold level is set two standard deviations above the mean value. Most peaks smaller than the bubble are reduced to be below the threshold level by the moving average, while the bubble's peak is consistently detected. The filtered array and threshold level for the image in Figure 3.2 are shown in Figure 3.3.

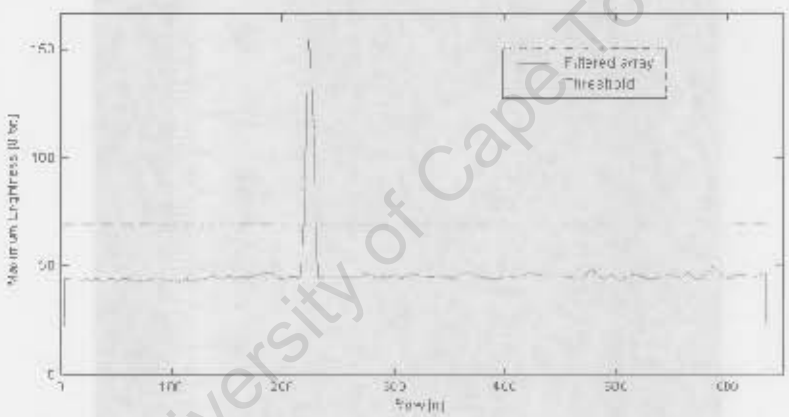

Figure 3.3: Tracking image processing, showing the array after filtering by moving average and the threshold level, two standard deviations above the mean.

3.3.3 Dilations and Erosions

The lighting situated above and below the column causes dual bright spots to appear on the top and bottom of the bubble, where the light is reflected, while the middle of the bubble usually remains dark, as can be seen in Figure 3.4. Occasionally, the binary array resulting from the threshold operation has gaps in it in the middle of the bubble, even with the use of the filter described above.

To solve this problem, the binary array undergoes further processing in the form of a two element dilation, followed by a two element erosion [13]. These steps have the effect of linking detected regions that are within four elements of each other while retaining the overall size of the regions, as shown in Figure 3.4.

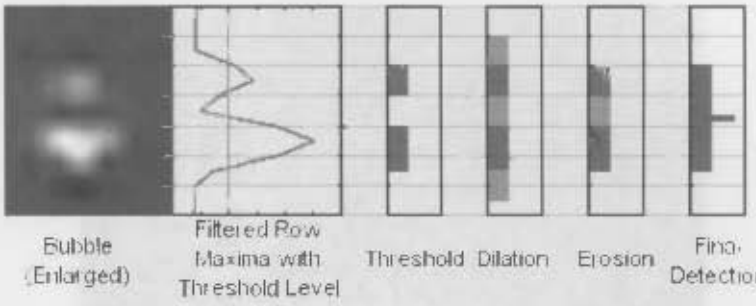

Figure 3.4: Processing for image section containing bubble, shown vertically.

3.3.4 Final Detection and Selection

Once the array of row maxima has undergone the processing described above, the final bubble search and selection takes place. The binary array is searched for continuous "set" regions. These regions are enumerated and their positions in the array, their sizes (lengths) and their centers are noted.

For example, in the frame in Figure 3.5, two bubbles were detected as "set" regions in the array of row maxima. The detected regions are drawn on the left side of the image. In this case, the first (top) region is 14 pixels in size (length), and the vertical position of the center of the region (shown as the mark protruding from the region indicator) is at a position 119 pixels from the top of the frame. The second region is nine pixels in length, and its center is 262 pixels from the top of the frame[1].

The list of detected regions is then used to determine the error for the controller. First, a prediction for the expected bubble position is generated based on the bubble frame positions from the two previous iterations of the tracking loop:

$$Y_n^p = Y_{n-1} + (Y_{n-1} - Y_{n-2}) = 2 \cdot Y_{n-1} - Y_{n-2} \tag{3.1}$$

where Y_n^p is the predicted value for the current frame and Y_{n-i} are the actual bubble positions used from the previous two frames. This equation effectively gives a linear estimate for the bubble's motion - it predicts that the bubble will move by the same amount each frame. This estimate was found to be a good prediction due to the relatively smooth motion of the bubble between frames.

The list of the detected regions is then searched, and the region (if there is more than one) with its center closest to the prediction is used for the controller error. The following equation is used to calculate the error of the bubble's position from the center of the screen:

$$e = \left(\frac{\text{FrameHeight}_p}{2} - \text{BubblePosition}_p \right) \cdot \frac{\text{VisibleLength}_{mm}}{\text{FrameHeight}_p} \tag{3.2}$$

[1] As an aside, it is the lower of the two bubbles that was being tracked when this image was taken. The top one was actually attached to the wall of the inner chamber of the column, and was thus stationary. It passed out of the view of the camera a few frames later.

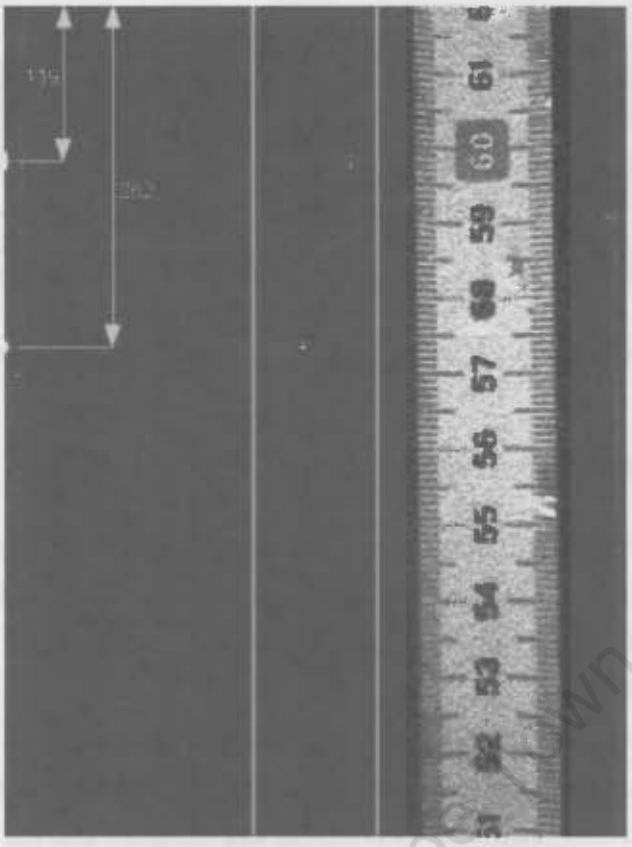

Figure 3.5: Tracking image with two bubble detections. Contrast has been increased in segment containing tape measure.

where all of the references with a subscript p are given in pixels. The $VisibleLength_{mm}$ refers to what length of the column is visible in the 640 pixel $FrameHeight_p$, and remains approximately 110 mm consistently throughout the run.

The controller was developed assuming that the error would be given in millimeters, so Equation 3.2 converts a top-down pixel value for the center of the detected region into a millimeter error measured from the center of the frame. The error is positive if the bubble is above the frame's center, and negative if below. It is this error value that is provided to the controller described in Section 3.4.

There are a number of contingencies built into the error calculation. For instance, occasionally the processing can fail to detect the bubble (the threshold value is not met), leaving the list of detected regions empty. In this case, the prediction itself (Equation 3.1) is used to calculate the error. Another problem that can arise is a missed detection for the tracked bubble at the same time as another, spurious detection in the same frame. This leads to the wrong target being tracked, which can completely compromise the tracking run. To counter this, the range of possible region positions is reduced. Only regions with centers within a certain range of the prediction are considered for use to calculate the error. If there are no regions within this range, the predicted value is used. 15% of the screen size (96 pixels) was found to be an effective range allowing a very high tracking success rate.

3.3.5 Tracking Flange Compensation

The opaque flange joining the two column sections together, mentioned in Section 2.2, is dealt with by attaching a piece of black cardboard over the side of the flange facing the camera. The cardboard is positioned in such a way so as to allow the bubble to be tracked for as long as possible, but then to obscure the bubble and flange structures before false detections could interfere with the tracking. The black cardboard keeps detections from occurring within the search range while the view of the bubble is obscured, allowing the predicted value for the bubble's position to be used. This effectively keeps the camera's velocity constant, since by this time (2 m into the run) the camera will have caught up to and centered the bubble in the frame, which means the predicted value for the bubble will also remain centered. An examination of the velocity profiles of bubble in the literature [8, 16, 17, 25, 24] shows that while the bubble's velocity at this stage will not necessarily be constant, all significant transients will have decayed, and the camera should be able to re-acquire the target bubble even if visuals are temporarily lost. This is generally the case, as shown in Section 3.4.5 and the results (Section 5.2).

A series of images from a tracking run past the flange is shown in Figure 3.6. The vertical area of the frame which is searched is shown between the lines. Note that the bubble remains the only detected region within the lines. If the bubble is obscured, no region is detected between the lines, allowing the predicted values for bubble position to be used and the camera velocity to remain constant. The bubble is reacquired on the other side. The images shown are not consecutive, nor were they taken at regular intervals - they were selected as the best to demonstrate the effect of tracking a bubble through the flange.

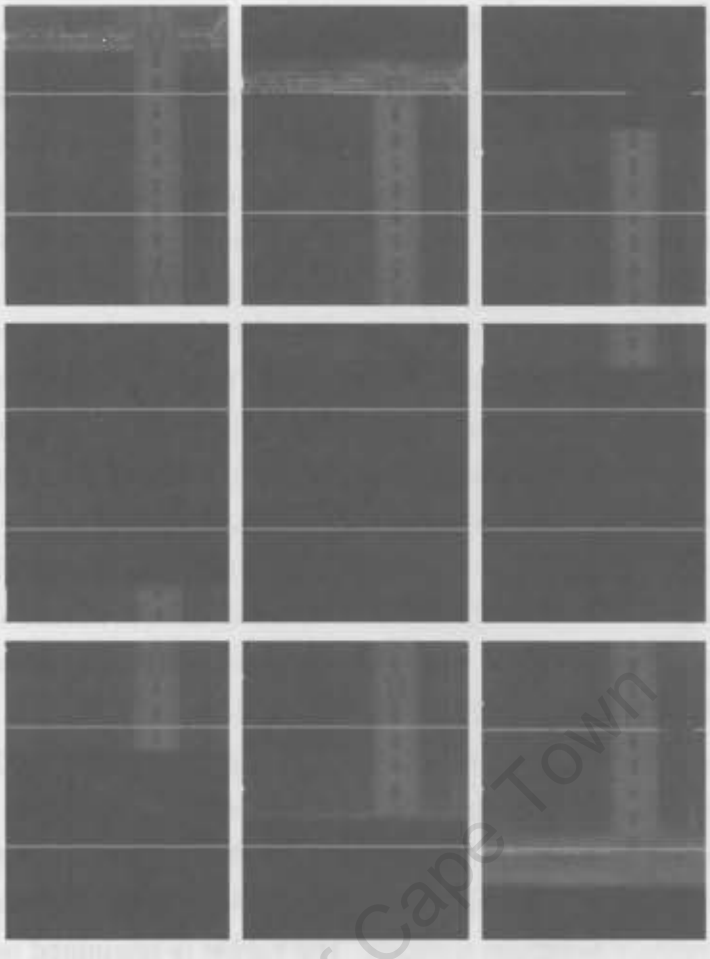

Figure 3.6: Multiple images of a tracking run through the flange.

3.4 Motor Controller

Once the error value for a frame is obtained as described above, it is used in a control algorithm to keep the error minimized and thus keep the bubble centered in the frame. The design, testing and performance of the control loop is described in this section.

3.4.1 Overview

The aspects of the apparatus relevant to the development of a controller can be modeled as a laplacian block diagram in standard feedback configuration, as shown in Figure 3.7.

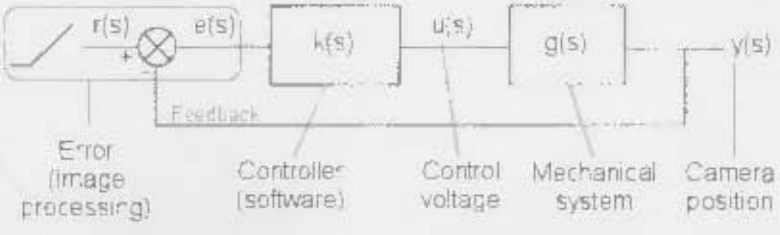

Figure 3.7: Laplacian block diagram modeling tracking control.

The setpoint (bubble) can be modeled as a ramp ($r(s)$), with a constant velocity of no more than 40 cm/s. The error ($e(s)$) calculation is performed in image processing, as described in the section above. The controller ($k(s)$) uses the error values to set the control voltage ($u(s)$). Since the controller is implemented in software, the lapacian controller model must be converted into a digital control law. This is described in Section 3.4.3. The mechanical system ($g(s)$) consists of all of the equipment related to the motion of the camera. It is the system that accepts a voltage from the DAQ card and outputs a velocity for the camera on the track. To develop a controller, this system must first be adequately modeled, and this is described in Section 3.4.2.

3.4.2 Motor Modeling

The main component of the mechanical system is the motor, and thus a model for this system was tested which was based on a standard form for a motor, from [1]:

$$g(s) = \frac{G}{s \cdot (1 + s \cdot T_m)} \tag{3.3}$$

where G is the steady-state gain and T_m is the mechanical time constant. The plant model ($g(s)$) is a ratio of input voltage to output speed (mm/V). It responds to a step input by exponentially approaching a steady state velocity. A plot of the step response for such a model can be used to estimate values for G and T_m. As a result, the mechanical system of the apparatus was tested with step inputs.

A ruler was positioned in front of the camera. From a stationary position, the input voltage was stepped using the DAQ card. At the moment it was stepped, the camera began to capture frames at 30 FPS. After the run, the images were examined and the exact position of the camera was read using the ruler[2]. These data were used to plot the camera velocity characteristic, shown in Figure 3.8.

Note that in most cases, the transfer function given in Equation 3.3 models a rotational velocity output. In this case, however, the motor's rotation (rad/s) is converted to a linear velocity by the conveyor belt pulley system (mm/s).

The motor model gives a good representation of the actual operation of the mechanical system. The steady state velocity is gradually approached, and doubling the step magnitude doubles the magnitude of the steady state velocity. However, there is a discrepancy: the model predicts that the steady state velocity will be reached in a constant amount of time, regardless of step magnitude. In Figure 3.8, the time constant appears to be smaller for the smaller step.

This deviation from expected performance is most likely caused by the inverter, which is used to convert the DC control voltage into a 3-phase AC signal. The inverter has

[2]In this case, the process was not automated, and the position data had to extracted by hand, unlike in post-processing.

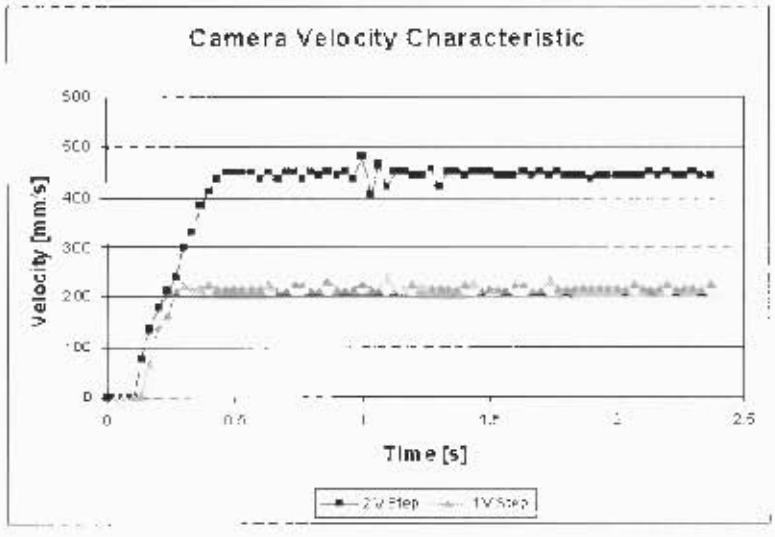

Figure 3.8: Camera velocity characteristic responses to step inputs.

control software designed to smooth out the motor's operation, and protect it from potentially damaging input signals. Unfortunately, the inverter's software is inaccessible and unavailable for modeling.

Despite the minor discrepancies, the model given in Equation 3.3 characterizes the behaviour of the mechanical components of the apparatus well enough to allow for an effective controller design.

As such, the following equation was used to describe the mechanical system:

$$g(s) = \frac{220 \cdot e^{-0.1s}}{s \cdot (1 + 0.35s)} \tag{3.4}$$

where G and T_m (from Equation 3.3) were estimated to be 220 mm/s/V and 0.35 s respectively. Note that a dead time of 0.1 s, clearly visible in the plots in Figure 3.8, has been included as well.

3.4.3 Controller Development

The controller was developed over time with an array of software packages, including n^{3}control (http://www.n3creations.com), Matlab Simulink (http://www.mathworks.com) and the *CAD software packages written by Professor Martin Braae of UCT (http://www.ct.ee.uct.ac.za/software/).

The design of the controller was an incremental process that gradually improved the controller until its performance provided the best tracking possible. This was done using a number of techniques including root locus design, simulation and trial and error, but there were a number of design criteria that were maintained.

The system needed to be capable of tracking a constant velocity setpoint - the rising bubble. Thus a Type 2 system [1] was preferred for zero steady state error when tracking

a velocity setpoint. This design called for two integrating s factors in the denominator of the open loop (O/L) system. One integrator already existed in the mechanical model (see Equation 3.4), so the controller design would need to include the second.

A standard design for a controller that includes an integrating term is the Proportional-Integral-Derivative (PID) controller. A PID controller consists of two zeros and a pole in theory, and two zeros and two poles in practice [1]. The form is given in the following equation:

$$k(s) = P \cdot \left(\frac{1 + \Gamma \cdot s + \Gamma \cdot D \cdot s^2}{\Gamma \cdot s \cdot (1 + T s)}\right) \tag{3.5}$$

where P is the proportional term, Γ is the integral term, D is the derivative term and T is a time constant that ensures the extra pole is non-dominant.

Using Equation 3.5 as a controller template in n^{3}control, the values of P, Γ, D and T were adjusted with the mechanical system model in feedback and ramp inputs of various gradients. n^{3}control allows the real-time editing of variables and pole/zero positions on the s-plane, and this proved invaluable in the controller design. The following transfer function was found to be the most promising controller:

$$k(s) = \frac{0.081s^2 + 0.405s + 0.371}{s^2 + 20.25s} \tag{3.6}$$

While the above function does not exactly fit the mold of the PID formula given in Equation 3.5, it has the integrating s factor allowing zero C/L velocity error, and it provided good tracking given the mechanical system model. The ramp response of the controller and mechanical model for the steepest ramp expected are shown in Figure 3.9.

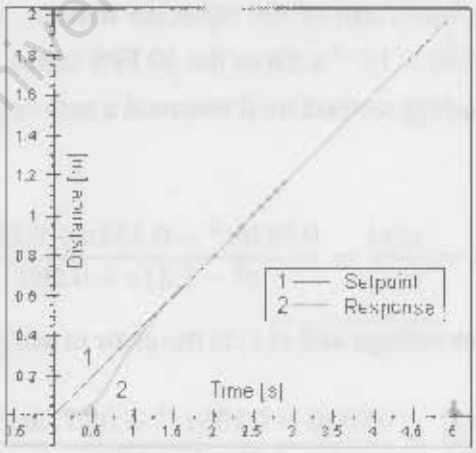

Figure 3.9: Ramp response for controller and system model in unity feedback configuration. Image is from n^{3}control.

The root locus diagram for this system in C/L configuration is shown in Figure 3.10.

The C/L poles (boxes) originate at the O/L poles and, dependent on loop gain, move along the loci mapped out in the diagram (crosses). The controller in Equation 3.6 results in a loop gain that places the C/L poles of the system at the position shown in Figure 3.10.

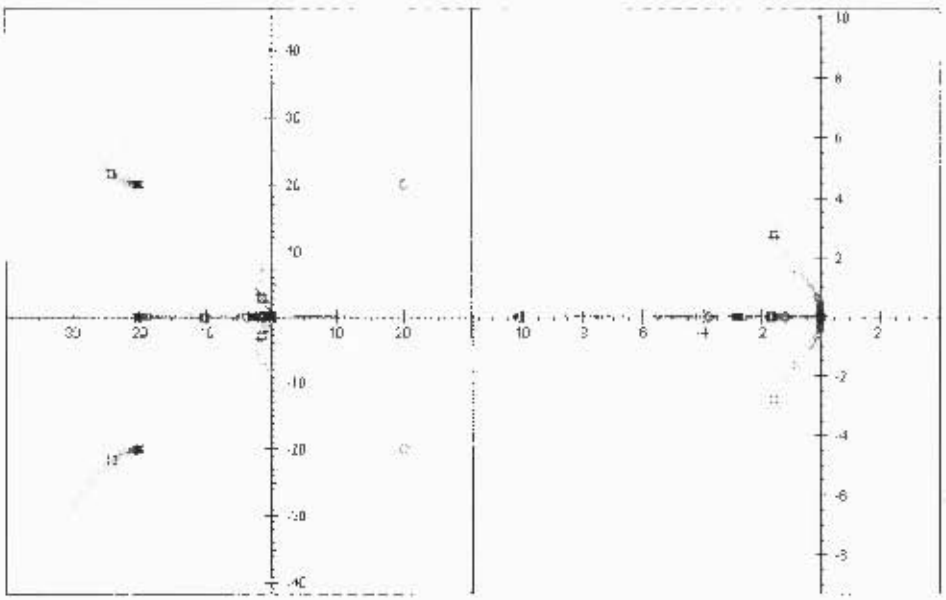

Figure 3.10: Root locus diagram for controller and mechanical system in C/L. The right diagram is a zoom of the left diagram. Images are from n^{3}control.

An examination of the C/L pole positions indicates that the performance of the controller is very close to optimum. The dominant poles in the configuration (the boxes at approximately $-1.6 \pm 2.8j$) have been pulled as far from the imaginary axis as possible without allowing them to move too far from the real axis or bend back into the positive half of the s-plane. This ensures that the system remains as fast as possible without becoming too oscillatory.

In order for the controller to be implemented by a computer, it needed to be converted into a discrete-time recursive difference equation. This was done in Matlab, using the *c2d* function to find the *z*-transform of the laplacian transfer function of the controller. Sample time was set at 33.33×10^{-3} s (from the 30 FPS frame rate of the camera during tracking), and the discretization method used assumed a zero-order hold on the input. The result was as follows:

$$k(z) = \frac{u(z)}{e(z)} = \frac{0.0810z^2 - 0.152z + 0.0713}{z^2 - 1.51z + 0.509} \tag{3.7}$$

where $u(z)$ is the controller voltage and $e(z)$ is the error in millimeters. Rewriting:

$$\frac{u(z)}{e(z)} = \frac{0.0810 - 0.152z^{-1} + 0.0713z^{-2}}{1 - 1.51z^{-1} + 0.509z^{-2}} \tag{3.8}$$

The above transfer function gives rise to the following recursive difference equation, suitable for implementation as a digital control law:

$$u_0 = 0.081e_0 - 0.152e_{-1} + 0.0713e_{-2} + 1.51u_{-1} - 0.509u_{-2} \tag{3.9}$$

Thus the control voltage for the current iteration (u_0) is calculated from a combination of

the current and previous errors (e_i) and the previous control voltages (u_i).

There is a final point about the controller design which requires attention. The bubble is released just below the bottom of the frame, and as a result, the bubble initially climbs upwards through the bottom half of the frame. This results in negative error values being sent to the controller in the early stages of tracking, which normally would result in negative control voltages - causing the camera to move downwards at first.

A number of techniques were considered to remedy this. Having the setpoint set to the bottom of the frame, and then adjusting it once the tracking was underway was a method employed in the earlier prototype of the apparatus, but it was eventually discarded as it was clumsy in implementation and performance. Setting the camera so that bubbles would be released from the middle of the frame resulted in the bubbles outrunning the camera - they would disappear beyond the top of the screen before the camera could build up any speed (partially a result of the dead time in the mechanical system).

Ultimately, the controller was biased with an initial steady state voltage. This forced the camera to begin moving upwards as soon as the controller was brought into action during tracking (once a bubble was detected), regardless of error. This method proved easy to implement and effective in providing the initial impetus for the camera while still allowing the controller to fine tune the steady state response and center the bubble accurately. (Many thanks to Professor Braae, who suggested the idea.)

The actual voltage value used to bias the controller is 1 V. Examining the transfer function for the mechanical system (Equation 3.4) shows why. In the steady state, a A 22 cm/s ramp will eventually be tracked by a 1 V signal. This falls in the middle of the range of bubble velocities of interest (which can be as low as 12 cm/s, and up to 40 cm/s). The 1 V signal was large enough to provide a starting bias for the fastest bubbles, while not overwhelming the controller in the case of slower bubbles.

3.4.4 Simulation

The controller was tested in simulation in a number of manners during its development. n^{3}control has simulation facilities, as can be seen in Figure 3.9. Matlab Simulink was used extensively, as it allows a detailed model to be tested, including effects of the bias voltage and the initially negative error. In addition to this, the bubble was simulated in the software of the apparatus, and the mechanical system was allowed to physically track a simulated bubble. The graphs in this section are all from the Matlab and apparatus simulations. While large numbers of simulations were performed during development, testing different versions of the controller and different aspects, only a selection will be presented here, chosen to best represent the development as a whole.

Matlab Simulink

The format for the simulations run in Matlab is shown in Figure 3.11.

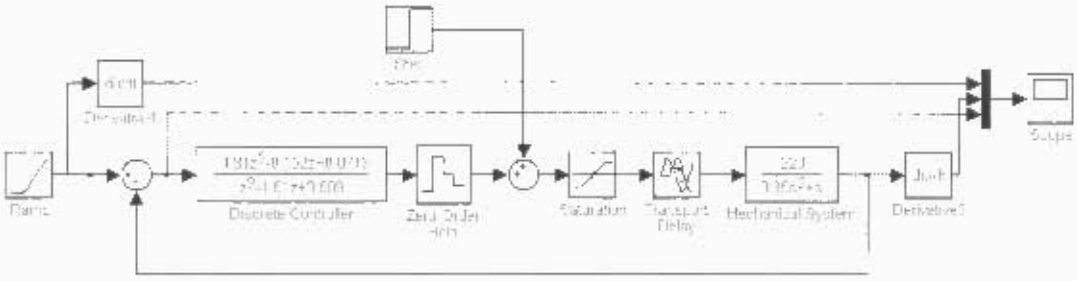

Figure 3.11: Matlab Simulink model for apparatus.

The discrete version of the controller is shown in this figure. Simulations were also performed with the continuous version of the controller, and attained similar results due to the high sample rate (30 per second). The results presented here are all from the discrete version, as this one is a better approximation to the true nature of the controller. The *Zero-Order Hold* block simulates the DAQ card, which holds the most recently set voltage on its output. The *Step* input simulates voltage bias mentioned in the section above. The *Saturation* block simulates the fact that, during tracking, the DAQ card cannot output a higher voltage than 3.5 V or less than 0 V[3]. The dead time in the system is modeled by the *Transport Delay*, and the remainder of the mechanical system by its laplacian transfer function. The *Scope* block provided the output channel from which the data from the simulations could be collected.

In the above configuration, the outputs are the velocities of the setpoint and response (acquired by sending the signals through derivative blocks) and the error signal. This was not always the configuration used, but it shows how the various signals desired from the simulations could be acquired. In the plots in Figures 3.12 and 3.13, the signals shown will always be mentioned.

Hardware Simulation

The simulations using the apparatus hardware were performed as follows.

The camera was placed at a known starting point using the position control (Section 2.6.2), and then the timer on the DAQ card was started. A loop was then started. Iterating every 33 ms, it contained the following algorithm:

- Read timer
- Calculate new setpoint position (in track units) from timer value
- Read camera position
- Calculate error, in mm, based on difference between setpoint and camera position

[3]A control voltage of less than 0 V would imply the camera is attempting to move downwards, which is not feasible for a rising bubble.

- Calculate new control voltage according to Equation 3.9

This method allowed the controller to be tested with the apparatus, without needing a bubble to act as a setpoint. Facility was provided in software to have an initial negative error and a bias voltage included, allowing comparison between the apparatus and Matlab simulations.

Note that this simulation method includes the reading of the position from the counter in real-time. This device was found to have problems with accuracy (see Section 4.1.1). However, in this case, the simulations were only attempting to give an indication of performance, and the inaccuracies were not large enough to compromise that.

Results

Figure 3.12 shows the simulation responses to 40 and 20 cm/s setpoints respectively. The error responses are shown in the graphs on top and the velocity responses are shown below.

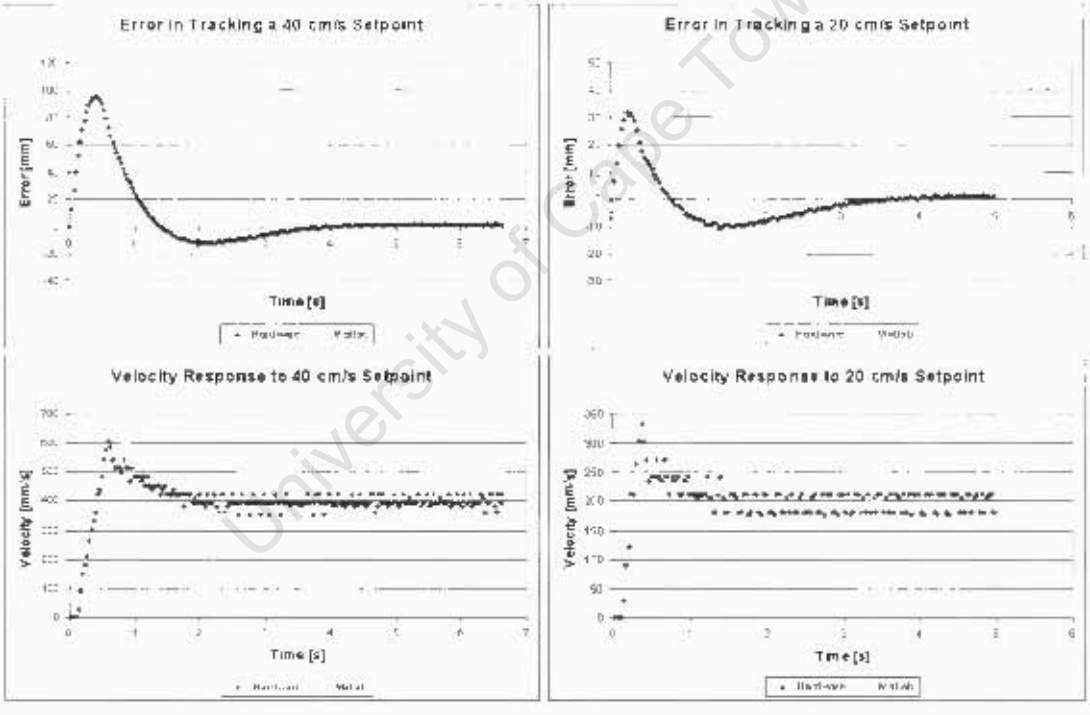

Figure 3.12: Simulation responses to 40 and 20 cm/s setpoint.

The two different simulators bear a marked resemblance to each other, particularly in the faster case. Timing and magnitude of the response are of a similar order. As predicted, the response in both cases is quick and close to having a critical damping factor.

To ensure that the camera would still be capable of tracking a setpoint with the initial error negative, as was expected in the practical situation, simulations were performed under that condition. Some plots of these can be seen in Figure 3.13.

Note that the response in 20 cm/s case is quite different between the hardware and software simulators. This is not entirely unexpected, as there are aspects of the mechanical

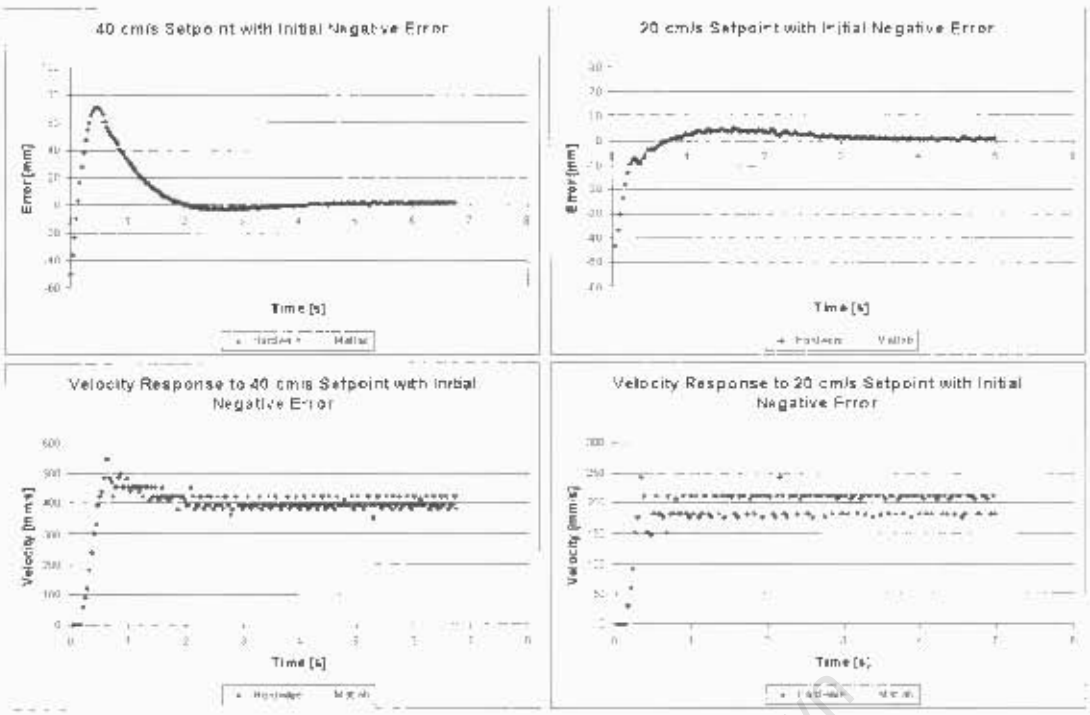

Figure 3.13: Simulation responses to 40 and 20 cm/s setpoints with initially negative errors.

system that have been neglected. The difference in response is most likely a result of friction from the conveyor belt moving against the track, or the control software of the inverter mentioned above in Section 3.4.2. Nonetheless, the setpoint was robustly tracked in both cases, and indicated that the controller would be suitable for use.

3.4.5 Performance

A typical profile from actual bubble tracking runs is shown in Figure 3.14.

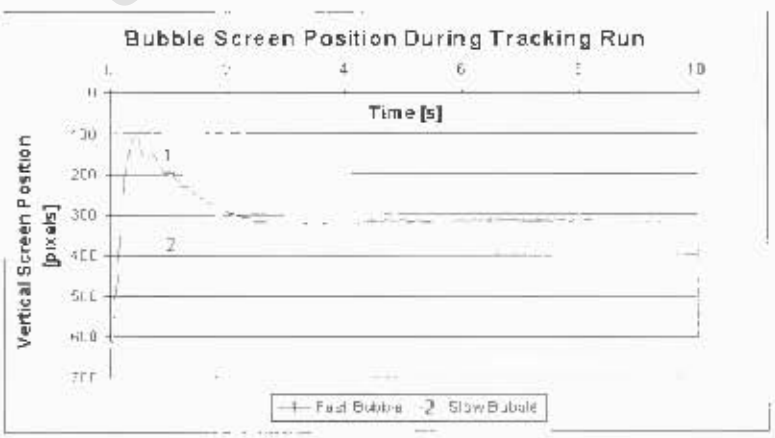

Figure 3.14: Plot of vertical screen position as a function of time for a bubble during a tracking run.

The two different plots are from different runs, one with a fast bubble (37 cm/s on average, tap water), one from a slower bubble (15 cm/s on average, MIBC solution, 30 Parts Per

Million [PPM]). More on the velocities of the bubbles can be found in the results (Chapter 5). This section is primarily concerned with the performance of the controller.

The plots measure the actual bubble position during the progress of the run. This is the reading that is used in Equation 3.2 to determine the controller error. Since the images are in top-down format, the bubbles enter the frame at the bottom with a high pixel value and are eventually centered in the image at about pixel 320 during the course of the run.

The controller performance was found to be fast and robust, as was indicated by simulation. Note that there is a gap in the readings in the plot of the response for the fast bubble around 5.5 s. This is due to the bubble passing behind the flange, and it was correctly reacquired on the other side. There is no gap in the response to the slower bubble as it had not reached the flange by the end of the tracking run (10 s). The one response that was not predicted, however, were the oscillations clearly visible in both graphs in the figure. Their cause is not immediately apparent.

They are not caused by the dominant poles in the root locus diagram in Figure 3.10. The pole positions correspond to a frequency of 0.45 Hz, which is an order of magnitude away from these oscillations.

It was also proposed that the oscillations could be a result of integral wind-up. If the controller is forced to saturate the control voltage at the top rail for too long (a result of a large error), it can cause the controller to continue to saturate at the rail until the integral term has been decreased by a negative error value. This can cause oscillatory behaviour, and is described, along with a possible solution, in [3]. The solution, involving desaturation of the integral term, was tested in the controller but had no effect.

It should be noted that the oscillations are largely absent in the simulations shown above, although smaller amplitude oscillations of a similar frequency can be seen in the 20 cm/s simulations. Thus the possibility exists that the oscillations are caused by some part of the model that has been neglected - the inverter, with its independent control, is a likely candidate.

But as it stands, the most likely cause for this behaviour is no single component, but simply a mode of operation which has not been included in the model. There is another set of C/L poles that has not been accounted for that is pulled towards the imaginary axis as the loop gain increases. This behaviour could not be replicated easily in simulation - that would require a much more detailed model of the system components, which in turn would defeat the purpose of the simulation - a general indication of controller performance.

An attempt was made to eliminate the oscillations with the use of a lag circuit [1] integrated into the controller, but unfortunately any circuit that affected the oscillations also compromised the controller performance. It was decided to ignore the oscillations, as the controller's performance was proven to be excellent despite them, and they ultimately had no effect on the post-processing of the images from the tracking runs.

Chapter 4

Post-Processing Algorithm

The development of the algorithm presented here is available in summarized form in [15].

4.1 Background

Originally, it was intended that the bubble velocity calculations be performed during the tracking run itself. Using the camera's track position according the the optical counter, and the bubble's vertical position in the frame, an absolute bubble position could theoretically be attained, allowing the velocity to be estimated by a discrete derivative operation. This method was implemented in the prototype version of the system [14], but unfortunately, the performance achieved ruled out this technique due to inaccuracies inherent in the method.

This section describes the problems that led to the implementation of a post-processing routine. It also discusses the advantages of performing post-processing as well as the disadvantages and difficulties encountered.

4.1.1 Source of Error

Early runs with the apparatus used the method described above to calculate bubble velocity, but accurate data resembling expected velocity profiles (based on [17]) could not be obtained. Errors are likely to arise from two main sources: Image processing and the optical counter.

Image Processing

Figure 4.1 (top plot) shows the bubble velocity, calculated from the camera's track position and the bubble's frame position (the method mentioned above), on the same time scale as the bubble's position in the frame. Below that is the camera velocity alone with the bubble frame position on the same time scale.

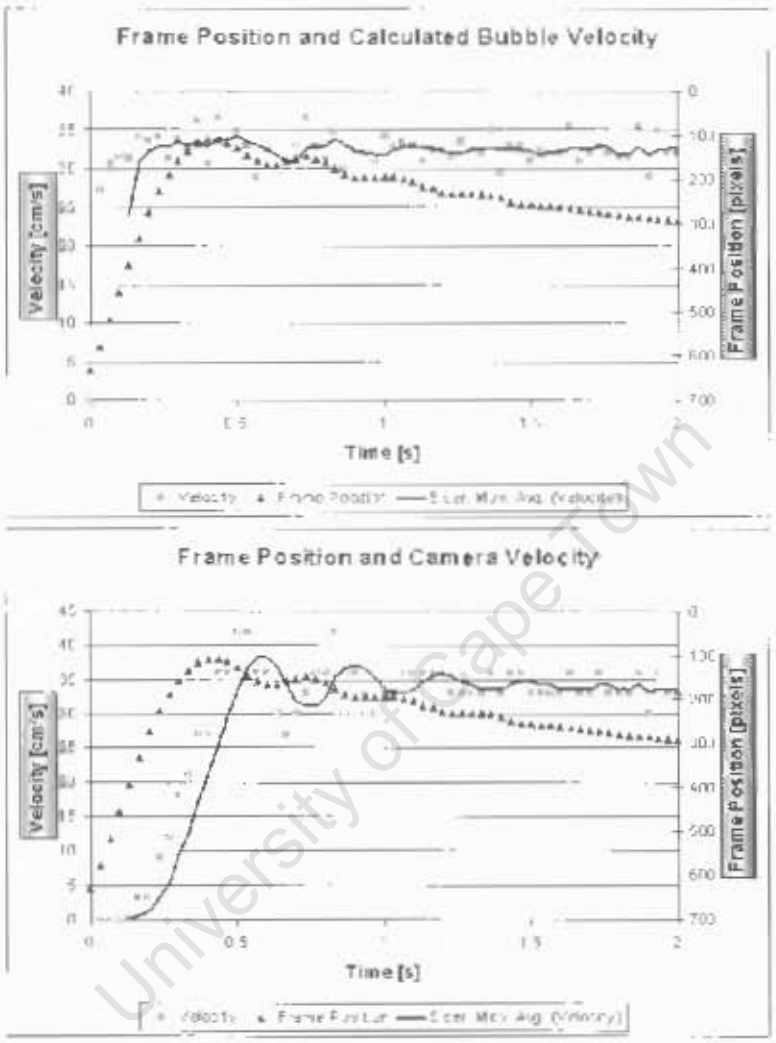

Figure 4.1: Frame position shown with calculated bubble velocity (top) and camera velocity (bottom). A trend line has been added to both velocity plots to better show oscillations in their behaviour.

Note that there is an oscillation present in the graph of calculated bubble velocity. It is subtle, but is clearly visible in the averaged version of the graph. It is at the same frequency of oscillation as the bubble position in the frame. Oscillations in the bubble velocity are not expected according to published findings so far, or from any current model for bubble rise velocity [16, 17, 25, 24].

The lower graph shows the camera velocity for the same tracking run. The oscillations are more pronounced, and are of the same frequency as the bubble frame position oscillations[1]. This graph is included to show the interrelation among all three sets of oscillations.

The correlation between the various sets of oscillations indicates that the oscillations visible in the bubble velocity arise from inaccuracies in the processing, not the actual bubble behaviour. An in-depth discussion of this can be found in [14], but the important point is that patterns in the motion of the camera during tracking, and patterns in the bubble frame position, are filtering through into the results. This does not pose a major problem for the tracking of the bubble, but it is unacceptable for the results, which need a high degree of accuracy.

Optical Counter

After the contamination of the results, shown in the section above, was noted, it was decided to test the accuracy of the optical counter mechanism, which had previously simply been assumed. A program was written to implement the test.

A ruler was placed in front of the camera. From a stationary position, the input voltage was stepped using the DAQ card, and camera frame capture was started. For every frame, the track position was read in from the optical counter through the DAQ card. Following the run, the images saved were examined and the actual camera position was noted by reading the ruler[2].

The two sets of position readings could then be compared by converting both readings to measurements in millimeters, and subtracting the initial position from every element in the set, for both sets (allowing both to start from 0). The set of positions according to the counter was then subtracted from the set according to the ruler. This was done for two velocities. The error plots are shown in Figure 4.2.

The images of the ruler are considered to capture the actual positions of the camera during the run. A positive error when subtracting the position read in from the counter indicates a time lag in that reading. This is confirmed by the fact that initially there is no apparent error, but as the camera velocity increases, the error develops. Additionally, the error is proportional to velocity - doubling the stepped velocity approximately doubles the error. This indicates an approximately constant time lag.

[1] The phase lag can be accounted for by the system dead time, as well as the lag introduced by the moving average function.

[2] Once again, the image analysis was done by hand.

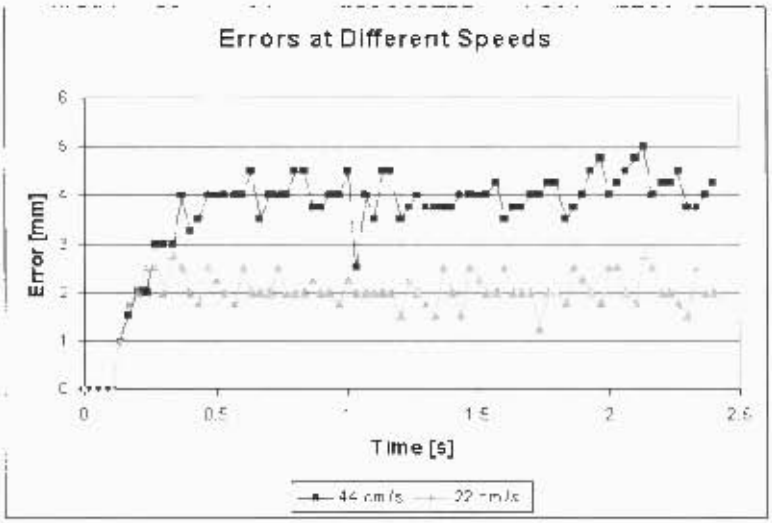

Figure 4.2: Error plots for optical counter for velocity steps.

The graph in the figure shows the inaccuracy of the optical counter. Compensating for this apparent time lag was considered, but this idea was ruled out: while the lag does appear to arise in the electronics box or the driver for the DAQ card, it cannot easily be characterized or guaranteed to remain constant. Even in the plots in Figure 4.2, a substantial jitter is apparent, especially in the response to the 44 cm/s step.

Height Measurement Using Tape Measure

Due to the apparent inaccuracies in the image processing and the camera track position, it was decided to test a method which eliminates both sources of error. This method involves placing a tape measure into the column in the same plane as the bubble's rise axis, and reading the tape measure in software. It gives an absolute reference for bubble height in the column, and theoretically allows the same accuracy to be obtained as a human operator performing the analysis (who would estimate the bubble height by reading the tape measure, the same method employed by the software).

It was decided to use a tape measure as an absolute reference for a number of reasons:

It provides a human-readable format. The bubble's position can thus easily be read in any frame by a human should the need arise. This is especially useful for debugging and error-checking.

It is not susceptible to aliasing. Each "reading position" (centimeter marks, in this case) on the tape measure is uniquely specified by a number. Other machine-vision solutions often use repetitive patterns that are non-unique, such as checkerboard patterns or other alternating-colour patterns. Using such a pattern could cause processing errors due to aliasing. This would occur if the positional change between frames (resulting from the motion of the camera) is greater than the pattern period.

Preliminary results were promising. The initial investigation into the feasibility of using an absolute reference was performed with a tape measure as it was easy to install in the column with a minimum of difficulty. Results from this investigation indicated that a high degree of accuracy could be obtained with the use of a tape measure.

As a result, development was carried out with a standard tape measure inserted in the column, and the algorithm is described as it operates on a tape measure. However, there are disadvantages to the use of a tape measure.

For instance, it is difficult to get it to stay in position. Since it is suspended down a distance of 4 m, small changes in the position at the top can result in large changes at the bottom. It is also bumped and jostled while the inner chamber of the column is being filled with water, which can leave it out of position. Due to the flexibility of the tape measure, it does not hold a straight line throughout the length of the column, but moves slightly to the left and right, and forward and backward, relative to the camera. This is not a major problem, but can affect the accuracy slightly (see Section 5.3) and causes problems when attempting to set the position. The flexibility can also allow it close enough to the bubble's rise path to affect the velocity, though this is rare.

In the light of these disadvantages, it is recommended that a new tape measure, tailored for the specific use of this apparatus be built and installed in the apparatus. A good measuring device would need to have the following characteristics:

- It would need to be attached to the column, in the plane of the bubble's rise axis relative to the camera, for the entire length of the column (or close to that).
- A rigid, non-corroding material would need to be used, such as Perspex.
- A machine vision solution for the position markers should be considered - this could be with unique markings (such as a bar-coded pattern), or possibly a non-unique repeating pattern (at the risk of aliasing - see above, and later in Section 4.4.5). If a non-human-readable format is used for the processing, the measuring device should incorporate the height in addition to the machine-vision pattern for easy debugging and error checking.

A better tape measure has not yet been implemented as this would likely require a long down time with the apparatus, which was not feasible. Additionally, the purpose of this dissertation is development, for which the present tape measure is adequate. The program code developed for the tape measure will probably require only minor modifications for the new measuring device, though this does depend on the design to a degree.

Another drawback, wholly unrelated to the tape measure, is that the time taken to process the 250-300 images generated in a tracking run is substantial (in the region of half an hour). For this reason, provision is made for the post-processing routine to process images

from several runs, one after the other, so that the PC can be left to perform the processing without requiring human input.

Once again, the time taken might be sharply reduced (see Section 4.4.5), but the aim of this dissertation is to demonstrate that the apparatus can perform the processing autonomously.

4.2 Overview

After a tracking run, the program saves the images from the run in a date-and-time-stamped directory. Currently the system is configured to limit the total number of images per run to 300. This allows 10 seconds of footage to be captured, which was found to be sufficient for development.

The images are numbered and processed in order from first to last. They are all of the form as the one shown in Figure 3.2[3]. The images include the bubble and the tape measure. The bubble's position remains roughly centered horizontally, but can vary substantially vertically before the bubble is caught and centered by the camera. Figure 3.14 gives examples of the pattern that can be expected. The tape measure remains in roughly the same position in the screen throughout the tracking run, though the numbers visible on it increase as the camera rises. The camera remains the same distance from the column at all times, thus the the camera's view always encompasses the same span of the column (and hence, the same span of the tape measure) - approximately 11 cm.

The image data set described above is the input to the algorithm which extracts the bubble velocity profile. The output is a comma separated values (CSV) file which contains the calculated velocity profile for the bubble, amongst other data used in the calculation, which can be used for debugging and error checking.

4.2.1 Algorithm Structure

The absolute height of the bubble in each frame is determined from the pixel position of the bubble's center in the frame, and the pixel positions of two of the numbers on the tape measure in the frame. The algorithm that does this has the following structure; for each frame:

1. Determine bubble position:
   (a) Predict likely position based on positions from previous frames.
   (b) Obtain region around prediction to search for bubble.
   (c) Search for the bubble using correlation.

[3]Note that the vertical lines are not included in the images saved by the system.

2. Read tape measure:

    (a) Find the tape measure in the frame.

    (b) Estimate the positions of the numbers on it.

    (c) Generate a set of numbers that is known to be visible on the screen.

    (d) For each possible set layout (based on the estimated number positions):

        i. For each estimated number position in the set layout:

            A. Perform a 2D correlation in the region around the position with a template shaped like the number expected in that position.

            B. Obtain a normalized "fitness" for the best fitting correlation in the region.

        ii. Use the fitness values to calculate an overall fitness for this set layout.

    (e) Compare the fitness values to determine the best fitting set layout.

3. Calculate the absolute bubble height from the bubble position and the tape measure number positions.

4. Estimate the bubble velocity from the position data, using a discrete derivative.

Number 1 in the list above is described in detail in Section 4.3. Number 2 is covered in Section 4.4. Numbers 3 and 4 are covered in Section 4.5. Finally, Section 4.6 describes how the flange problem was approached in terms of the post-processing algorithm.

4.3 Bubble Position Determination

Since searching each image in its entirety for the bubble would be time consuming and susceptible to spurious detections from smudges or other bubbles on the column walls, only a small portion of each frame is searched for the bubble. This requires a prediction for where the bubble is expected, and a range around the prediction to be searched.

4.3.1 Initial Prediction

The bubble's frame position is specified by two co-ordinates for the bubble's center, the horizontal x component and the vertical y component. The initial prediction for the expected bubble position contains predictions for both of these components.

The x component is initially set to the middle of the frame, as this is where the bubble is expected. In all subsequent frames, the predicted x value is simply the actual x position of the bubble in the previous frame, as the bubble is not expected to move much horizontally.

The y component for the prediction is found using the same method for prediction as the tracking algorithm. Initially set to near the bottom of the screen, like the tracking

algorithm, the predicted position of the bubble is given by Equation 3.1 after the second frame has been processed.

4.3.2 Search Region

The region around the prediction that is searched for the bubble is a rectangle centered on the prediction.

The horizontal range that is searched for the bubble is kept to 12 pixels on either side of the predicted value (5% of the frame size), although a slightly larger range is searched in the first frame (20% - between the vertical lines in Figure 3.2). This was found to be a compact, yet robust, range allowing the minimal horizontal movement of the bubble to be tracked, even on occasions when helical motion set in near the end of the runs.

The vertical range, once again, uses code from the tracking algorithm. Section 3.3.4 describes the array of "set" regions, one of which is the projection of the tracked bubble onto the vertical. The "set" elements in the array from this region correspond to the rows in the image inhabited by the bubble. The vertical search range is set to correspond to these rows.

The rectangle formed by the above ranges and centered at the predicted values is relatively small, thus reducing the time taken for the cross-correlation operations used to find the bubble, which are described in the following section.

Note that in the current configuration, the accuracy of the post-processing algorithm in finding the bubble does depend on the system's ability to track the bubble accurately. This is generally not a problem as the tracking algorithm is accurate. Results can deteriorate in the presence of noise (see Section 5.2.1 and Figure 5.3), but this can also allow the system to recover from an error (see Section 5.2.3 and Figure 5.7).

4.3.3 Cross-Correlations for Bubble

The region is searched with a series of cross-correlations [9, 13], using correlation kernels shaped like circles of various radii. Figure 4.3 is a zoom of the bubble in Figure 3.2, and shows the region described above (box) and some circles representing the cross-correlation kernels.

The circles on the left of the image represent kernels positioned at the first point in the 2D cross-correlation. Note that the kernels can overlap the region - the region specifies possible positions for the *center* of the bubble. The two circles shown also represent the smallest and largest kernel sizes that are considered in searching for the bubble. First, the smallest kernel is correlated against every center position in the region, then the kernel size is increased and the process is repeated for increasing kernel sizes up to the largest. The bubble's size and position are given by the best fitting kernel size and center position (shown as the circle around the bubble in Figure 4.3).

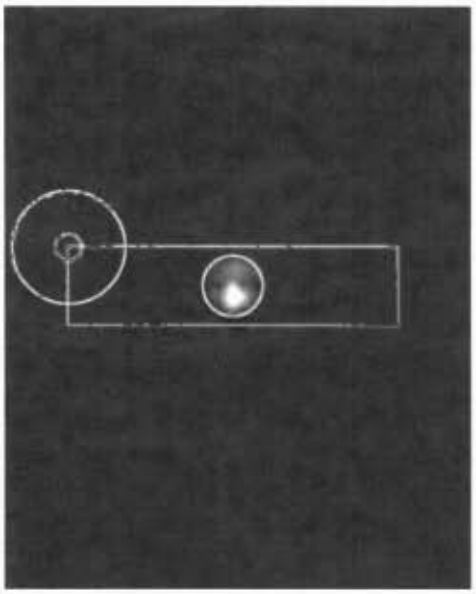

Figure 4.3: Zoom of bubble in image, showing search region and representations of correlation kernels.

Note that the full range of kernel sizes is only used in the first image. In the subsequent images, only kernel sizes close to the best fitting size from the previous frame are considered.

Practically, the kernel is generated as a square image with a side length equal to the diameter of the circle in the kernel. The image is binary, with the circle light and the background dark.

The cross-correlation operation [9, 13] used to find the bubble is based on normalized cross-correlation, a technique which ensures both data sets have means of 0 and standard deviations of 1 before the multiplication and summation take place. Normalized cross-correlation is used in the tape measure processing (Section 4.4.4 and Equation 4.2), so it will not be discussed in detail here. The main point relevant to image processing is that it is independent of changing intensity values across an image due to the normalization.

In this case, however, the bubble usually appears as an extra bright section in the region being searched. The correlation equation for a size $N \times M$ kernel shown below allows that fact to be exploited:

$$C_{x,y} = \frac{\sum_i \sum_j (I_{i,j}) \cdot (K_{x-i,y+j} - \overline{K})}{\sigma_K \cdot N \cdot M} \tag{4.1}$$

where the I refers to the image region and the K refers to the kernel. Note that the kernel has been normalized (by subtracting its mean and dividing by its standard deviation), while the image section of the image it is compared to is not. The final value is also divided by the size of the kernel, which allows a comparison between the fitness of kernels of different sizes.

Note that the radius of the bubble is also determined as a byproduct of this method. This

is relevant information, and is included in the output file.

4.4 Tape Measure Processing

Once the bubble's position has been determined, the tape measure must be read in software to provide a reference for height. This procedure involves several steps, which are described below.

4.4.1 Tape Measure Position

First, the position of the tape measure in the image must be determined. As mentioned previously, there is a slight lateral drift in the position of the tape measure throughout the height of the column. In addition, the tape measure is not always set in the exact same position for every tracking run. Thus, the position of the tape measure can not be assumed to remain fixed.

The tape measure appears as a lighter band against the dark background of the image. Since the tape measure is always approximately vertical, its position can be specified using only two values: The horizontal co-ordinate of each edge.

Figure 4.4 shows the plot obtained by averaging every column of the image in Figure 3.2. The tape measure's presence is clearly visible in the higher values on the right.

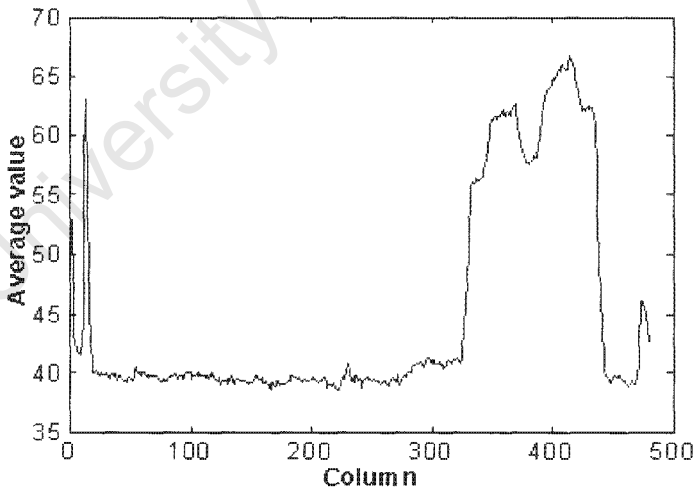

Figure 4.4: Graph showing column averages for Figure 3.2.

A number of 1D normalized cross-correlations are performed for *Rect* functions of varying widths. The function with the best fitting position and width gives the appropriate horizontal coordinates for the tape measure edges.

The apparent width of the tape measure in each frame does not change much through the set of images, so the range of width values is kept small to avoid spurious detections - such as from the spike on the left of Figure 4.4, or one of the single "lobes" of the tape measure section.

4.4.2 Digit Kernels

In the remainder of this section, references are made to "numbers" and "digits". "Number" refers to a multi-digit height marker on the tape measure. For instance, the tape measure section visible in Figure 3.2 contains the numbers 12 to 22. "Digit" refers to a single numeral within the number, often in the context of the kernels that are constructed as described here. This distinction is noted in the nomenclature.

In order to find the numbers on the tape measure in the frame, correlation kernels resembling the numbers had to be constructed.

An examination of Figure 3.2 shows some important practical facts about the tape measure:

- Numbers are of a consistent font (size and shape).
- Each number, while it varies in width, is nonetheless exactly centered on the line marker for the height it refers to.
- Multiples of ten appear as light numbers against a dark background, opposite to the usual.

The first two points above allow the kernels for the numbers to be constructed out of digit templates, made before the run and stored on the PC's hard drive. Each of the ten digits is kept as a separate file, stored as a binary image with the same format as the digit in the frame it is to represent. This means black feature against a white background, and rotated at 90 degrees. The templates can then be stacked to form the number required. Since the templates have different widths, the width of the number kernel varies to match the width of the number it is designed to search for. This can be seen in Figure 4.5.

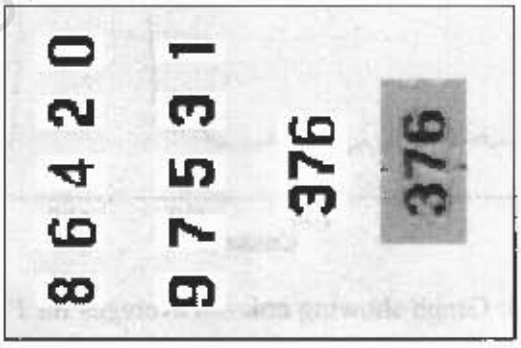

Figure 4.5: Digit templates for number kernels. A number kernel is constructed from stacking the templates, and is compared with a view of the number on the tape measure.

While the number kernel will never exactly match the image of the number it is modeled after, the match is generally better than any other number in the image.

In the case of the multiples of ten (with the opposite colour format), provision is made to negate the template in the function that performs the cross-correlation.

Because the templates are stored in files on the hard drive, they can be replaced should the tape measure format change, or should the performance need to be adjusted.

4.4.3 Initial Number Position Estimation

At this point in the algorithm, the image is negated (inverted in grayscale) to make the numbers appear as light features against a dark background.

Before correlation for actual numbers can take place, an estimate of the possible number positions, and a region around those estimates, is required, similar to the prediction and search region for the bubble.

The horizontal positions of the numbers are known to be exactly halfway between the edges of the tape measure, but the vertical positions of the numbers still need to be found. A frequency analysis of the image is used to determine the vertical period and phase of the number sequence.

First, a narrow vertical band is extracted from the middle of the tape measure, using the horizontal co-ordinates for its edges calculated in Section 4.4.1. The width of the band is usually made the same as the width of the digit kernels (26 pixels) - thus, the size of the band depends on the size of the features (numbers).

A blurring operation is performed on the extracted band, by using a fast fourier transform [12] (FFT - the library used was FFTW [5]) to cross-correlate a blurring kernel with the extracted band (for a description of the cross-correlation by FFT operation, see Section 4.4.4). This result is shown in Figure 4.6.

For each row of the resulting band, the maximum value is extracted and placed in an array. The plot of these values can be seen in Figure 4.7.

The periodicity of the numbers can clearly be seen. This 1D data array now undergoes frequency analysis to obtain a frequency spectrum revealing the fundamental frequency component. It is padded with zeros until its length has quadrupled, which increases the frequency resolution of the output, and an "FFT-shift" operation is performed, swapping the first and second halves of the padded array [12]. An FFT is then performed, the results of which can be seen in Figure 4.8.

The FFT array is then searched for the maximum absolute value for frequency. Since the length of the tape measure visible in the frame is known to be a certain value (11 cm), the search can be constrained to a small band of frequencies around the value corresponding to the expected value. Constraining the search ensures that low frequency spikes from ambient light variations do not cause a false detection.

The actual period for number repetition, in pixels, is given from the graph by dividing the length of the array by the index of the maximum value.

The phase information for the numbers is then extracted by taking the arctangent of the complex argument of the maximum value.

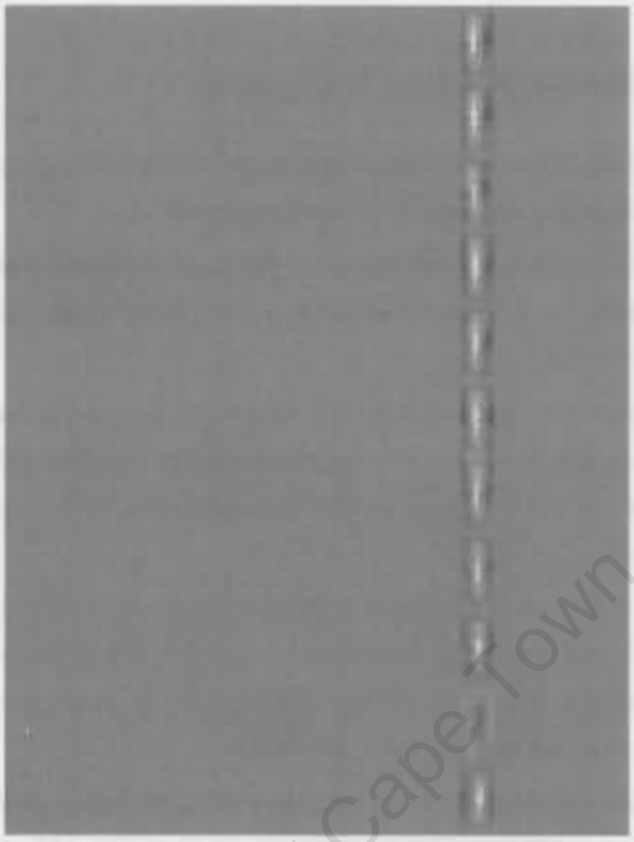

Figure 4.6: Segmented and blurred image of tape measure numbers.

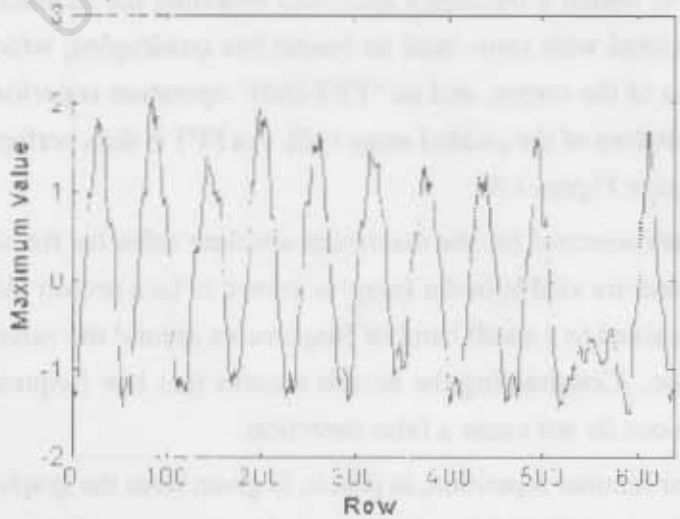

Figure 4.7: Plot of row maxima from blurred image. Data set has been normalized.

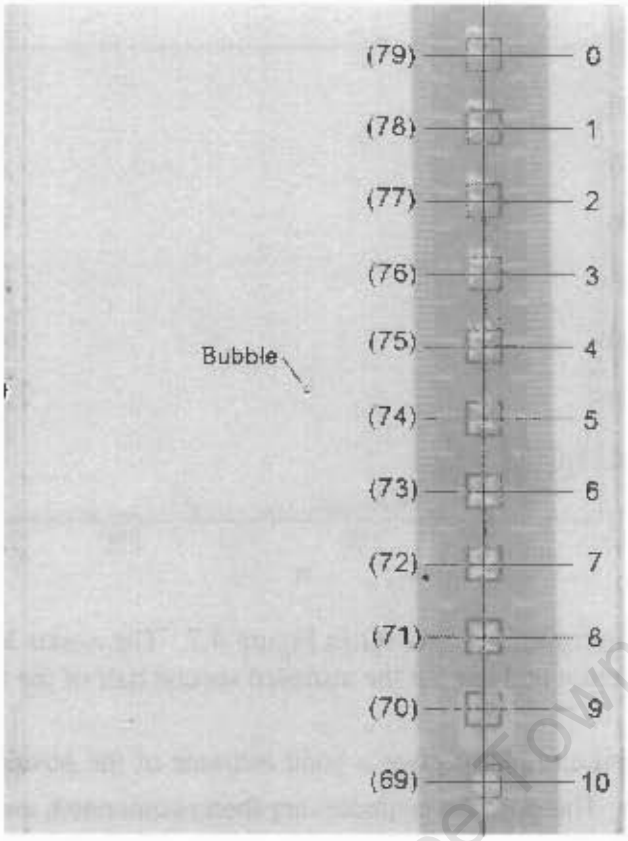

Figure 4.9: Negative image (blurred one from Figure 4.6) showing estimated number positions (crosshairs), search regions (boxes), and estimated position enumeration (numbers on right). Actual numbers on the tape measure are shown on the left.

4.4.4 Number Sequences

In post-processing, the absolute bubble height is calculated from knowledge of the positions of the numbers visible on the tape measure. The process of finding the numbers is described here.

The algorithm requires a set of "known" numbers - numbers the algorithm assumes to be visible on the tape measure in the current frame. The algorithm then finds the positions of the known numbers by searching in the regions around the estimated positions shown in Figure 4.9. The algorithm is given a constant set (that can be changed, if necessary) for the first frame - since the camera always starts from the same position on the track, the same set of numbers can always be used initially[4]. Following that, a new set of numbers is generated for every frame based on the top numbers visible in the previous frame.

These "known" sets consists of six consecutive numbers, though their length can be varied (six was found to be a good number in terms of performance). Averaging the "fitness" for multiple numbers increases robustness by ensuring that a single error does not compromise performance.

[4]The frame in Figure 3.2 actually shows the camera in its initial position. It has not yet begun to move in tracking the bubble, due to system dead time. The initial set given for the first image is thus the numbers 16 to 21.

The set for a given frame is tested in several layouts or positions according to the estimated number positions. These different layouts for a set of numbers are referred to as "sequences". Table 4.1 shows an example of a set of numbers for the frame in Figure 4.9 (the numbers 73 to 78 are likely to have been the top six numbers in the previous frame), as well as the considered sequences.

Est. Position	Actual #	Seq 0	Seq 1	Seq 2	Etc
0	79	78			
1	78	77	78		
2	77	76	77	78	
3	76	75	76	77	...
4	75	74	75	76	...
5	74	73	74	75	...
6	73		73	74	...
7	72			73	...
8	71				...
9	70				
10	69				

Table 4.1: A set of numbers and considered sequences for Figure 4.9.

A series of normalized cross-correlations is then performed for the numbers in the sequence according to their respective regions. For instance, in *Seq 0* in Table 4.1, a kernel shaped like "78" is used in the cross-correlation around estimated position 0, "77" is used around position 1, and so on. Once all six numbers in *Seq 0* have been correlated for, the process repeats for *Seq 1*, and so on for all of the sequence positions that allow all six of the numbers to be visible.

Normalized Cross-Correlation

The normalized cross-correlation operation [13] that is used to search for the numbers is described by the following equation:

$$C_{x,y} = \frac{\sum_i \sum_j (I_{i,j} - \overline{I}) \cdot (K_{x+i,y+j} - \overline{K})}{\sigma_I \cdot \sigma_K \cdot N \cdot M} \tag{4.2}$$

This is the fully normalized version, which has the means subtracted from both the kernel and the image region ($\overline{K}$ and $\overline{I}$), divides by both of their standard deviations (σ_I and σ_K), and divides by the size of the region ($M \times N$). This gives rise to the following advantages:

- It is insensitive to changing light intensity across an image. This is important in this case, as correlation results from various heights in an image are being compared.
- It outputs a value between -1 and 1, 1 being a perfect match. This uniformity allows comparisons to be made.

Note that the region of the image referred to by I in Equation 4.2 is not the same region as the one used to confine the search around the estimated number position. Correlation can be visualized as a lining up of the kernel with a part of the image, followed by a multiplication and summation. The region referred to by I is the region beneath the kernel for that particular correlation result ($C_{x,y}$). The confining region around the estimated number position is akin to the region in Figure 4.3. Similar to that region, it confines the *center* of the kernel to a certain area. Note that information in the image outside this region is still used in the correlation, if parts of the kernel overlap the outside of the region. Because of this, the edges of the resultant correlation image do not "fade out", a common effect in correlation software.

Each kernel will have a point of "best fit" in its (confining) region - this is given by the maximum correlation result for that region and kernel. These values are summed up for the sequences, and the sequence with the highest overall result for all six numbers is considered the correct choice. *Seq 1* from Table 4.1 is, of course, the correct result for Figure 4.9. The distribution of the calculation of the best fitting sequence across six numbers aids in robustness.

Once the correct sequence has been identified, the remaining numbers are inferred, and searched for using normalized cross-correlation to determine their exact positions.

The tape measure processing has finally given rise to a description of the tape measure in the frame. Each visible number has its position recorded. Tables 4.2 and 4.3 show the processing results for Figure 4.9 following the bubble search and the tape measure processing.

X Co-ord [pixels]	Y Co-ord [pixels]	Radius [mm]
233	296	0.690

Table 4.2: Table showing results of bubble processing for Figure 4.9.

Position [Enumerated]	Number [cm]	X Co-ord [pixels]	Y Co-ord [pixels]	Correlation Result [normalized]
0	79	365	30	0.808
1	78	365	87	0.735
2	77	366	142	0.751
3	76	367	200	0.765
4	75	367	257	0.730
5	74	368	315	0.717
6	73	369	373	0.810
7	72	369	431	0.800
8	71	370	490	0.790
9	70	372	549	0.710
10	69	371	607	0.684

Table 4.3: Table showing results of tape measure processing for Figure 4.9.

The method used to determine the millimeter radius of the bubble from the pixel radius (Table 4.2) will be covered in Section 4.5.

In reading Table 4.3, it should be remembered that the image is top-down and is 480×640 pixels in size. The numbers on the tape measure are, of course, centimeter markers. The x and y co-ordinates represent the center of the kernel when it best fit the image beneath. Since the numbers are centered on their marker lines on the tape measure, these co-ordinates are assumed to give the actual centimeter height of that pixel in the frame.

4.4.5 Methods Tested for Decreasing Process Time

The process of reading the tape measure takes between 5 and 30 s per frame, depending on the following criteria:

Kernel size: The size of the number affects the size of the correlation kernel. Larger numbers take longer to search for using correlation. Early frames in the run thus take less time to process than later frames because the numbers are double digits, rather than triple digits.

Region size: The region around the number can be made larger or smaller depending on how accurate the estimated number positions are (see below). Smaller search regions decrease processing time. Note, however, that region size is linked to kernel size (see Section 4.4.3).

Number of correlations: The more normalized cross-correlations performed per frame, the longer the time taken in processing. This is related to the number of sequences tested (see Section 4.4.4).

Processing method: A spatial domain normalized cross-correlation is computationally expensive, due to normalization taking place for every iteration (correlation offset [13]) of the operation. A possible trade-off between accuracy and processing complexity could save frame processing time.

Computer speed: A faster PC will also result in faster processing times. It should be noted that one PC was used for all results in this dissertation. It is described in Section 2.3.

The time taken per frame can add up to a significant length of time over the course of 300 frames. As a result, a number of methods were tested to decrease the time taken to process a frame. Some were incorporated into the software for the apparatus, some did not stand up to testing. They are described below.

Using Estimated Number Positions to Calculate Bubble Velocity

The normalized cross-correlations, described in Section 4.4.4, could be dispensed with entirely, in theory. The estimated number positions, shown as the crosshairs in Figure 4.9, could be used to calculate the bubble velocity.

In this case, the velocity would be inferred from determining how the bubble moves relative to the positions of the tape measure's markings (given by the frequency analysis of the tape measure). This method neglects the information relating to the actual numbers visible in the frame, and some accuracy would be lost in the use of the number position estimates. However, the trade-off for this loss of accuracy would be a significant decrease in processing time.

In the current configuration, however, this method would be subject to aliasing. At the maximum speed of 40 cm/s and a frame rate of 30 FPS, the bubble moves 1.67 cm between frames. Therefore, the camera, once it has caught the bubble, moves the same distance, and thus the tape measure will appear to move the same amount downwards between frames. The spatial resolution of the estimates is only 1 cm, so the change in height between frames cannot be calculated with absolute certainty. (A movement of 1.5 cm appears the same as 0.5 cm.)

This problem could be solved in software with the knowledge that the tape measure is initially stationary, and assuming that the change in bubble height remains similar between frames. Even so, this method is still error prone.

Due to these reasons, and since the development of this apparatus was primarily concerned with determining how accurate a velocity profile generated in software could be if it could even be done, this method is not investigated in this dissertation.

Should further development on the apparatus be commissioned, such a functionality could be developed. This would be especially useful if a new tape measure were to be fabricated, with a spatial resolution greater than 1.67 cm.

Decreasing Region Size

This refers specifically to the confining region around the estimated number positions (the boxes in Figure 4.9). The estimated positions were generally found to be very good, and consequently the side lengths of the regions searched could be reduced to half the size of the kernel.

Due to the large amount of processing required for a spatial-domain normalized cross-correlation, reducing the search area significantly decreased the time taken by a factor of four, provided accurate estimates could be guaranteed.

In most of the results processed, the tape measure was initially a relatively severe angle (an example can be seen in Figure 4.10). Due to this, the half kernel method could not guarantee that the center of the number would fall in the reduced region. Consequently,

only the vertical range search was reduced - the horizontal was left the same, trading processing time for robustness.

Reducing Number of Sequences

The bubble cannot move downwards between frames, and is not expected to move faster than 40 cm/s (corresponding to 1.67 cm between frames - this value is rounded up to 2 cm in the test algorithm). This knowledge is used to reduce the number of sequences that are considered, by eliminating sequences that would result in incongruous bubble behaviour.

The following algorithm is used to determine whether a sequence should be considered. It is placed in the code after the bubble's position and the estimated number positions have been found. The bubble's previous height, calculated from the last frame, is also known (see Section 4.5).

1. Determine which estimated number position is immediately below bubble. Label: *FirstBelow*.
2. Round down the previous height to the nearest integer. Label: *FloorPrevHeight*.
3. For each possible sequence (see Table 4.1):
   (a) If the sequence position results in a number less than *FloorPrevHeight* in the position *FirstBelow*, it is not considered.
   (b) If the sequence position results in a number greater than *FloorPrevHeight* + 2 in *FirstBelow*, it is not considered.
   (c) Otherwise, it is considered.

This algorithm was found to significantly reduce time taken for processing, as it allows a maximum of three sequences to be considered, instead of six or seven. It is, of course, not used in the first frame as in that case there is no previous record of the bubble's height.

Note that the rounding down of the previous height in Step 2 does technically allow a sequence to be considered where the bubble moves downward. This turned out to benefit the system, as it allowed it to recover from an error, shown in Section 5.2.3 in the results.

FFT Versions of Cross-Correlation

Certain signal processing operations, such as correlation, can often be performed faster by converting to the frequency domain. To test if a speed improvement was possible for the normalized cross-correlation, a number of FFT versions were written and tested using the FFTW library [5].

To eliminate edge effects in the search region (the box regions in Figure 4.9), the side lengths were increased by the size of the kernel before it was extracted from the image.

This larger region was then normalized by subtracting its mean from every point, and dividing every point by its standard deviation.

Following this, depending on the version being tested, the region was either left alone, or padded with zeros. In the padded case, the region was padded until its dimensions were powers of two.

Once the region had been normalized, and if necessary, padded, the kernel was then normalized and then padded with zeros up to the size of the (padded) region. The FFT based correlation was then performed in the usual way [12]:

$$C = F^{-1}[F(I) \cdot F(K)^*] \tag{4.3}$$

where C is the resultant image, I is the image array, K is the kernel array and F denotes the FFT operation. Note that the complex multiplication involves the complex conjugate of the kernel array elements.

In this case, normalization cannot happen in every iteration of the correlation operation. The fact that the entire region is normalized first saves processing time, but it also means that every data point in the output image (C) is normalized by an estimated factor. This results in a loss in accuracy for the correlation result used to compare sequences, and results in incorrect sequences being chosen. This was corrected by using the FFT to find the correct co-ordinates of the maximum value in the output, and then performing a single spatial domain normalized cross-correlation on that pixel, giving an accurately normalized correlation result that can be used in the comparison.

The first 300 correlation operations performed for a tracking run were timed using the Windows *clock* function in Visual C++ for the different methods (the same 300 operations were used each time for consistency). The region size used was a half of the kernel size, as described above in this section. The actual region and kernel dimensions vary, but the kernel was roughly 26×26 pixels, which means the search region (Figure 4.9) was around 13×13 pixels, and the actual amount of the image used (the "larger region", above, and see also Figure 4.3) was thus 39×39 pixels. These results can be seen in Table 4.4.

Type	**Padding**	**EST/MEAS**	**Mean Time**	**Standard Deviation**
			[ms]	**[ms]**
Spatial Domain	N/A	N/A	93.4	21.8
FFT	no	EST	87.6	8.32
FFT	no	MEAS	88.4	11.1
FFT	yes	EST	130	21.1
FFT	yes	MEAS	132	33.8

Table 4.4: Table showing times taken to perform correlation operations for different methods. Sample size is 300.

The *EST/MEAS* option in Table 4.4 involves the efficiency of the FFTW architecture. If the *FFTW_ESTIMATE* (*EST*) option is set for the operation, the library estimates the best

architecture for the algorithm (usually sub-optimally). If *FFTW_MEASURE* (*MEAS*) is set, FFTW executes several FFTs, measuring the time taken for each to determine the best algorithmic architecture. A further explanation can be found in [5], or at (http://www.fftw.org).

As can be seen, the *EST* option provides marginally better results in each case. This is likely because any advantages due to the *MEAS* option are offset by the overhead processing it requires, as well as the relatively small FFT sizes.

The padding (up to the next highest power of two) of the region was tested as it was suspected that an FFT that is an exact power of two is executed faster than a smaller one with factors of higher numbers in its size. As can be seen in Table 4.4, this is not the case.

In the test, the unpadded FFTs perform fastest. The spatial domain correlation is slightly slower, followed by the padded FFTs. Relative to its size, the spatial domain correlations are also more unpredictable with regards to process duration.

The unpadded FFT version thus seems preferable in terms of performance, and did work well in processing results from a tracking run with a uniformly lit tape measure. However, this did not hold up in all cases, and the FFT version did not exhibit the same robustness as the spatial domain method. The frame shown in Figure 4.10, the first from a tracking run, illustrates.

Figure 4.10: Image from a tracking run that compromised FFT performance. Section of image containing tape measure has been equalized for clarity.

Note the change in brightness across the width of the tape measure. The FFT region (expanded by a kernel size) searched around the number 20 is highlighted. It is slightly decentered due to the tilt in the tape measure, but note that it still contains the entire number, so edge effects play no part.

The region is normalized once, across the entire region, before the FFT correlation is performed. Due to this single normalization, changes in light intensity across the region do have an effect, and in this case, it compromises the correlation. The bright half of the tape measure results in a stronger result for the FFT correlation, resulting in the correlation giving incorrect co-ordinates for the number's center.

This effect was noted on several occasions per frame when processing this tracking run, and resulted in incorrect sequences being chosen, compromising the post-processing algorithm.

4.5 Bubble Height, Velocity and Radius Calculations

Two methods were tested for the bubble height determination. They are described here, followed by the methods used to estimate the bubble velocity profile and radius.

Method 1: Linear inference

The pixel co-ordinates of the bubble and the two numbers on the tape measure closest to the bubble (to reduce errors from refraction) are used to calculate the height of the bubble in each frame, according to the following equation:

$$h_b = \frac{(n_j - n_i) \cdot (y_b - y_i)}{(y_j - y_i)} + n_i \tag{4.4}$$

where h_b is the absolute height of the bubble (cm), n_i and n_j are the numbers i and j on the tape measure (cm), y_i and y_j are the y co-ordinates of numbers i and j in the frame (pixels) and y_b is the y co-ordinate of the bubble's center in the frame (pixels).

This method is demonstrated graphically in Figure 4.11. The height is given by the ratio d_a:d_i in pixels, which is the fraction of a centimeter that the bubble is above 76 cm.

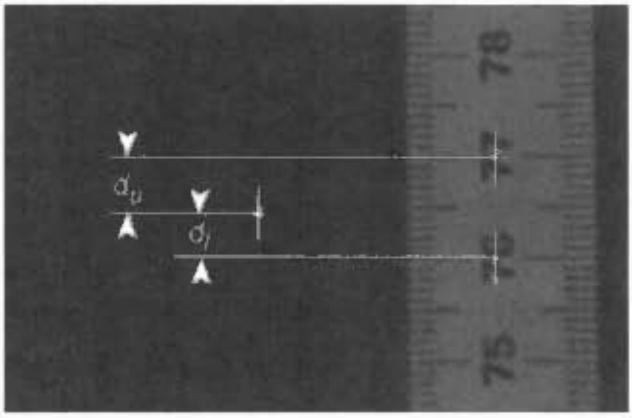

Figure 4.11: Graphical demonstration of Equation 4.4.

Note that Equation 4.4 will work for any two numbers on the tape measure, regardless of their order or positions, provided that the numbers are kept with their correct y co-

ordinates. This is useful, since a restriction is added to the calculation: the numbers used to determine the bubble height must have a correlation result (see Table 4.3) greater than 0.4. Numbers are occasionally cut off the top or bottom of the frame, or are obstructed by a bubble stuck to the tape measure. This can result in the wrong pixel position for the center of the number, and a lowered correlation result. The restriction ensures that only clear numbers on the tape measure are used.

Method 2: Ray Undistortion

The second method is actually an elaboration of the above method. A checkerboard pattern was fabricated, and placed in the column (in water) in front of the camera. (Checkerboard patterns are often used for calibration in image processing.) A set of screen shots were taken (examples can be seen in Figure 4.12), and these were used to calibrate the camera with the Matlab Camera Calibration Toolbox (see [7], also [http://www.vision.caltech.edu/bouguetj/calib_doc/]).

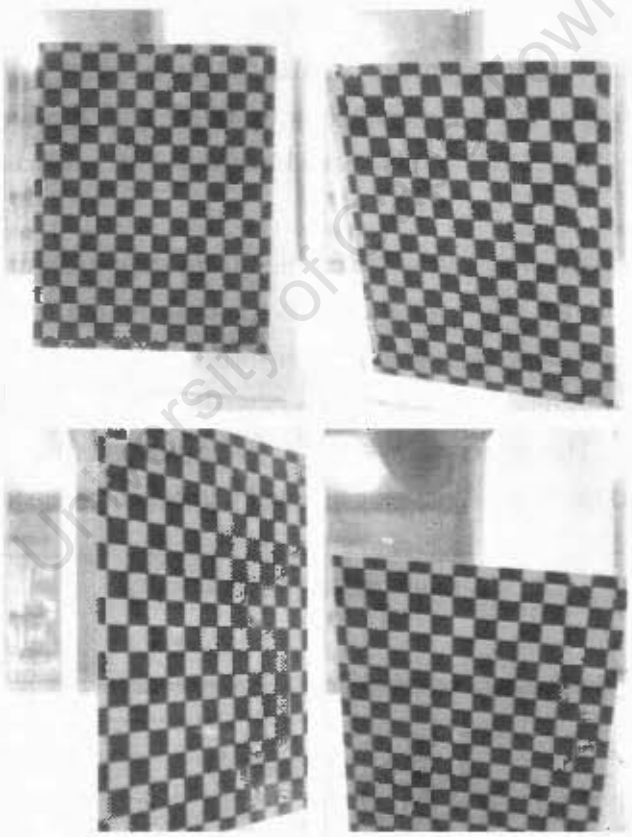

Figure 4.12: Examples of calibration images.

The toolbox is designed to compensate for optical distortion arising from the lens of a camera, but the calibration method involves a general polynomial approximation to the distortion, which could be used to compensate for errors due to the refraction of light as it exits the column.

The set of calibration parameters extracted from the calibration images using the toolbox was integrated into a function in the apparatus software that performed a ray undistortion

for given pixel co-ordinates. The co-ordinates for the bubble's center and those for the numbers on the tape measure were undistorted using this function before the application of Equation 4.4.

The effect of the undistortion algorithm is minimal, as can be seen in Figure 4.13. This is due to the fact that the bubble height is being calculated from a reference that is also inside the column, very close to the bubble, significantly reducing inaccuracy. The linear height calculation gives nearly the same accuracy and the height calculation with undistortion.

Velocity

Once bubble height information is obtained from one of the two methods mentioned above, the frame to frame velocity profile for the bubble can be estimated by performing a discrete derivative operation:

$$v_b := \frac{\Delta h_b}{\Delta t} \quad \frac{h_i - h_{i-1}}{t_i - t_{i-1}} \tag{4.5}$$

where v_b is the velocity estimate for the i^{th} frame, Δh_b is the change in the bubble's height from the previous frame, Δt is the change in time from the previous frame (1/30 of a second), and h_i, h_{i-1}, t_i and t_{i-1} are the actual bubble heights and times for the frames i and $i-1$.

Figure 4.13 shows position and velocity profiles from the processing of a portion of a tracking run (bubble in tap water, similar to the run in Section 5.2.1 in the Results). The line shows the position value (h_b) without undistortion. The undistorted position profile is not shown, as the difference can not be seen in position plot. It is slightly more evident in the velocity plots. The circles show the velocity estimated from the ray undistortion described above, and the dashes show it estimated without the undistortion.

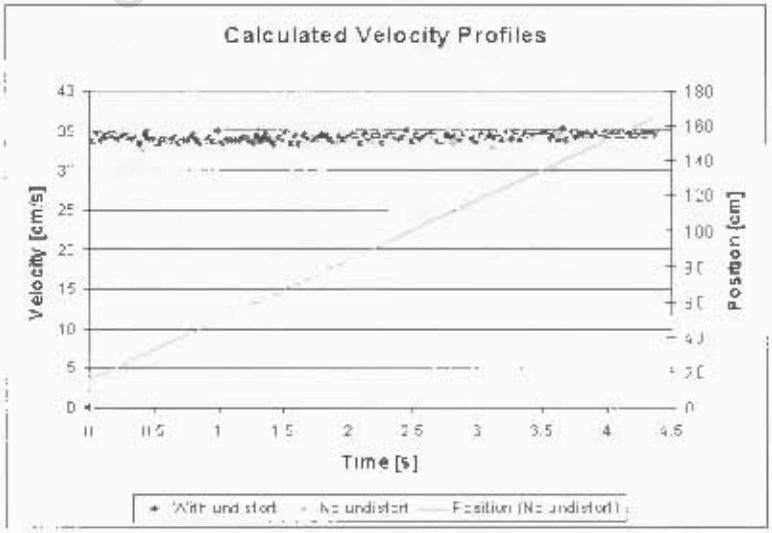

Figure 4.13: Position profile from a tracking run, and velocity profiles calculated with and without ray undistortion operation.

In most places, the two curves line up closely - the undistorted points (circles) are lined up with the unaltered points (dashes). However, in places, the effects of the undistortion can be seen, as there are dashes separated from their corresponding circles, which have been pulled into the curve more tightly by the undistortion. This indicates that, while the effect is minimal, there is some benefit to the use of the undistortion algorithm.

The undistortion takes negligible time to calculate, so there is no processing disadvantage to its use. The unaltered curve is still desired for comparison and error checking, thus both versions are included in the post-processing output file.

Bubble Radius

The pixel radius of the circular kernel used to find the bubble (Section 4.3.3) is used to estimate the bubble's radius. After the bubble's height is calculated, the height of a point above the bubble, corresponding to the bubble's edge, is also calculated using Equation 4.4. The difference in heights between the edge of the bubble and its center gives an estimated value for the radius.

This method was found to underestimate the actual radius (determined by human analysis of the images). While the underestimation was only slight for a well lit bubble, this method did contribute to a larger inaccuracy which affected results slightly. This is described in Section 5.2.2.

Similar to the velocity estimation, the radius estimates from both regular pixel values and the undistorted values are included in the output.

4.6 Post-Processing Flange Compensation

The flange that obstructs the bubble for a small portion of the run also obstructs the tape measure. This had to be dealt with in the post-processing, since certain frames before and after the flange allow the bubble's position to be calculated from the visible part of the tape measure, but undermine the processing by obstructing numbers from the "known" array (see Section 4.4.4). Furthermore, in certain saved frames the bubble is not visible at all; these frames would need to be skipped in post-processing as they contain no data for the velocity profile. Figure 4.14 shows the last frame before the flange and the first after the flange in which the bubble is visible.

The decision as to which frames should be ignored for post processing is again made based on the same code used during tracking to determine the bubble's position and the error (Section 3.3.4, see also Section 4.3.2). If no bubble is detected within a certain range of a predicted value during tracking, the predicted value is used. If no bubble is detected within the same region during post-processing, the image is ignored. This ensures that only frames where the bubble is visible are processed.

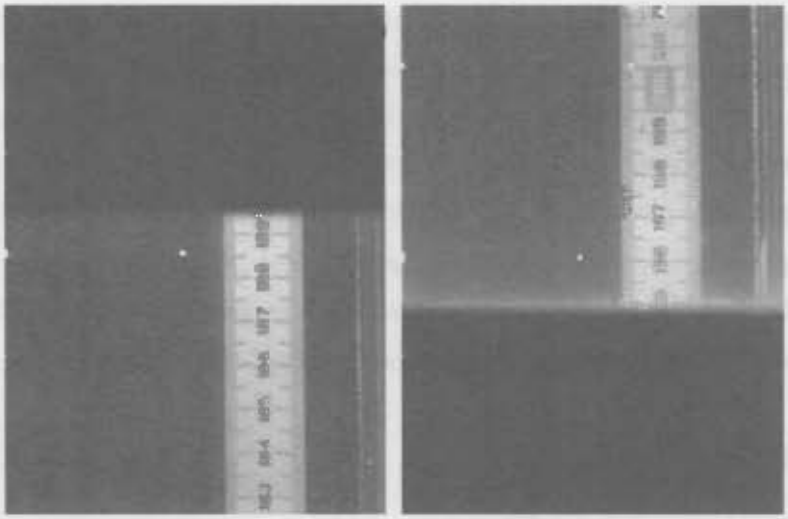

Figure 4.14: First image before and last image after flange in which bubble is visible. Contrast and brightness have been adjusted for clarity.

In the left frame in Figure 4.14, the top number visible is 189. Earlier frames before the flange allow a higher number to be visible due to the angle of view for the tape measure. Figure 4.15, from a few frames earlier, illustrates.

The numbers 190 to 192 can be seen. According to Section 4.4.4, the top numbers visible in a frame are used as the "known" numbers in the next frame. Just below the flange, this method does not work, as certain numbers visible in one frame will be obscured in the next frame, resulting in incorrect sequences being chosen.

To solve this problem, a simple fix was implemented:

If the top "known" number is higher than 188, or lower than 201 (see below), the "known" array is changed such that 188 is the top number. (The number 188 was always visible until the bubble disappeared - 189 was occasionally obscured.) The number position estimation method still works - there are sufficient numbers visible to ensure a correct frequency selection. The sequence reduction method described in Section 4.4.5 also still functions correctly, fitting seamlessly in with the slight code modification for around the flange. Since the tape measure is not moved vertically very much, the "flange numbers" (188 and 201) don't need to be changed, though the facility exists in the program should it become necessary.

After the camera has passed beyond the flange, the known number can't be generated from the previous frame easily. Again a simple fix was implemented:

As soon as the bubble is visible again, the "known" array is set to the numbers 196 to 201 (201 being the top number). These numbers are always visible on the tape measure as the bubble reappears after the flange. Post-processing behaviour returns to the normal algorithm once the bubble has reappeared.

The method used to determine if the bubble has reappeared from behind the flange (and whether the "known" array should be set as above), is to keep track of how many frames have been ignored in post-processing due to no bubble being present (see above). Occa-

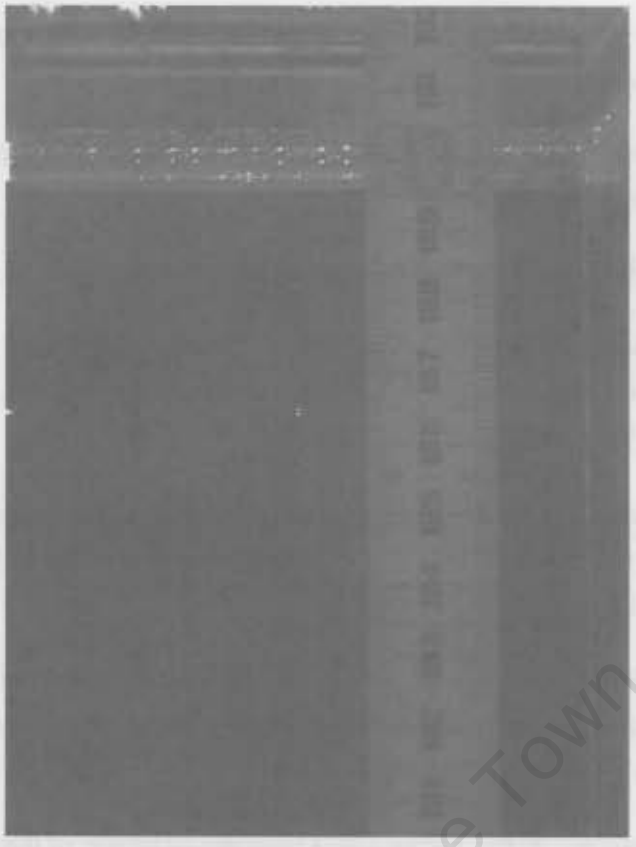

Figure 4.15: Image of tape measure from further below flange.

sionally the bubble detection can fail during regular tracking because the threshold isn't crossed (see Section 3.3.2), but this rarely happens for more than one or two frames in a row. Consequently, if a bubble is not detected for more than two frames, it is assumed to be behind the flange. Again, this number can be adjusted if necessary.

A more robust "flange detection" method, perhaps taking into account the bubble height or the numbers visible on the tape measure before the bubble disappeared, has so far not been considered because the method described above works adequately.

4.7 Output

Following the completion of the post-processing algorithm on a set of images from a tracking run, a CSV file is generated containing relevant data from the run. The following data are included:

Frame number: The data is listed according to the number of the image it was extracted from.

Time: The time, in seconds, that the frame was taken (calculated from assuming a 30 FPS frame rate).

Error: Numerical value associated with certain errors in processing the frame (such as from missing the bubble due to flange).

Height: Calculated height of the bubble (from Equation 4.4), in m, cm or mm, depending on output options.

Velocity: Estimated velocity of the bubble (from Equation 4.5), in m/s, cm/s or mm/s, depending on output options..

Radius: Estimated radius of the bubble (described in Section 4.5), in mm.

Bubble *x* co-ordinate: Horizontal component of the bubble's frame position, in pixels. If there is an error causing the frame to be skipped, this is left as the predicted value described in Section 4.3.1.

Bubble *y* co-ordinate: Vertical component of the bubble's frame position, in pixels. If there is an error causing the frame to be skipped, this is left as the predicted value described in Section 4.3.1.

Ray undistorted data: Ray undistorted (Section 4.5) versions of the bubble height, velocity and radius readings are included, as well as the camera-normalized versions of the bubble's x and y co-ordinates.

Estimated tape measure data: Certain data from the estimates of the tape measure number positions are included (Section 4.4.3). These include the number period, the position of the first number (giving phase), and the y components of the edges of the search regions.

Tape measure model: The full model of the tape measure is included. For each number on the tape measure, the following are included:

- The number
- Its maximum correlation result in that position
- The co-ordinates, in pixels, of where the maximum result was obtained
- The ray undistorted version of the co-ordinates

Chapter 5

Performance and Results

5.1 Overview

This chapter discusses and provides a consolidated description of the performance of the system as a whole. It will be primarily concerned with the performance of the post-processing algorithm and its application to experimental data. The performance of the tracking algorithm has been discussed in Section 3.4.5, so only aspects of it that are relevant to the post-processing and experimental data will be covered here.

The experimental data consist of the video footage from a number of tracking runs for bubbles rising in various conditions, including the following:

1. Tap water
2. Distilled water
3. Dowfroth 250 surfactant solutions, at concentrations of 0.06 and 30 PPM
4. MIBC (Methyl IsoButyl Carbinol) surfactant solutions, at concentrations of 0.06 and 30 PPM

In Section 5.2, the estimated velocity profiles resulting from the application of the post-processing algorithm to the experimental data are presented in the order above. Different aspects of the system's performance are highlighted by the different plots, and to enhance this, the velocity plots will sometimes include additional data. The general system performance will be discussed as the relevant points are demonstrated in the plots.

The date and time of the tracking run is included with the results, as well as the processing time and the total number of images processed. Note that ignored images (such as flange images) are still included in the number of images, and that not all data sets have exactly 300 images. This is due to the suppression of images or because the tracking run was terminated at a certain track position. These data have been included to give an indication of the size of the data sets and the time taken to process, and should not be considered as a benchmark.

Appendix A contains certain results for bubble velocity profiles from the apparatus at McGill university. The appendix will occasionally be referred to for comparison.

5.2 Experimental Data

5.2.1 Tap Water

Figure 5.1 shows a plot of the estimated velocity profile of a bubble rising in tap water. Also shown in the plot is a graph of the calculated absolute height of the bubble, from which the velocity is estimated. The juxtaposition of these graphs allows an overall impression of the operation of the algorithm to be gained.

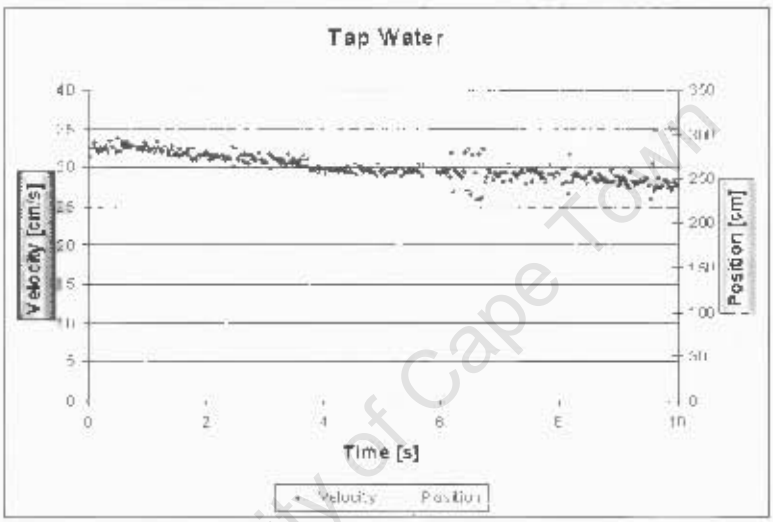

Figure 5.1: Velocity profile of bubble rising in tap water. Plot of position is also shown. Tracked December 12, 2006, 12:51 pm. Total processing time was 37:31 for 300 images.

Note the break in both of the plots at around the 6 s mark. This is where the bubble was temporarily obscured by the flange. These data points would be registered as errors by the post-processing, and ignored in the velocity estimation, giving a clearer profile for velocity.

Note that the position profile appears relatively smooth, whereas there is a more significant scatter to the velocity readings. This is due to the discrete derivative operation in the velocity calculation, which highlights even subtle discrepancies, such as errors due to pixellation, in the calculated bubble position between frames. This is useful for highlighting incorrect data points, as will be seen in later graphs. If it is desired, the general scatter can be smoothed with filtering operations.

In this graph, discrepancies can be seen in the form of an increased scatter following the flange break. This is due to an error resulting from the operation of the bubble detection method in the post-processing algorithm. It is shown and explained in Section 5.2.2.

As for the actual reading itself, the subtle velocity decay in this profile is very similar to the one for the warmer run in Figure A.1 and particularly for the profile for bubbles with

$d_r = 1.5$ mm in Figure A.3. The plot in Figure 5.1, above, arose from a bubble of similar size, as confirmed by visual analysis of the images.

This plot should be compared with that of a tracking run from a few months earlier, taken during earlier system development. It is shown in Figure 5.2. (Processing time is better, as this run was processed with smaller regions than the other results; see Section 4.4.5.)

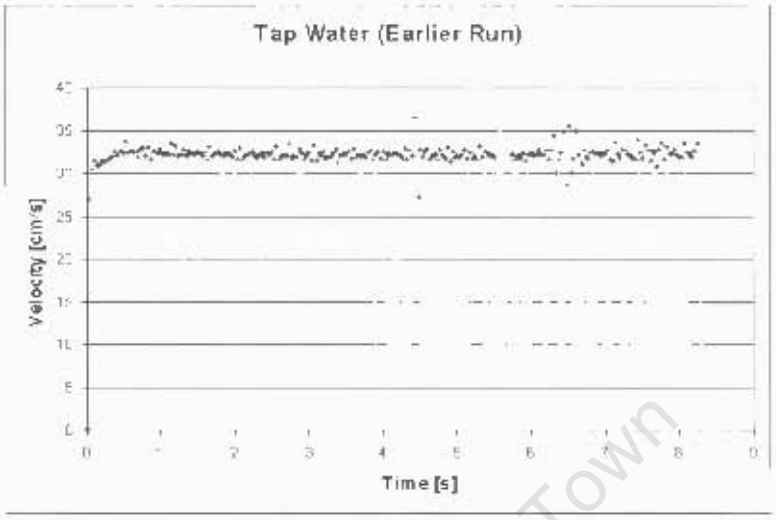

Figure 5.2: Earlier tracking run, also in tap water. Tracked August 24, 2006, 11:00 am. Total processing time was 20:38 for 247 images.

Note that there is no decay evident in the velocity profile. Interestingly, this corresponds with the colder profile from Figure A.1. This cannot be directly confirmed, since the temperature sensors and control equipment were not in place, as mentioned. Note however, that the decaying and non-decaying runs were taken in the (Southern Hemisphere's) summer and winter months respectively, which would affect the temperature of the water, and could explain the different behaviour.

Note the pair of spurious data points at around 4.5 s into the run. A sharp increase in velocity followed by a sharp decrease indicates that a single perturbation in the height reading is responsible. Figure 5.3 shows the relevant frame from this tracking run.

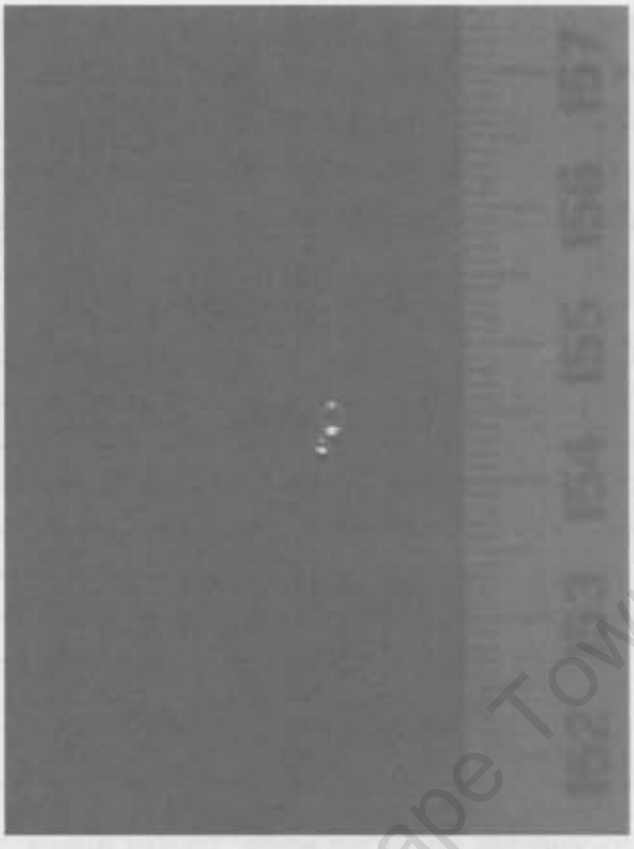

Figure 5.3: Enlargement of tracking image responsible for spurious data point in graph in Figure 5.2.

In this frame, the bubble being tracked (smaller, lower left) passes very close to a bubble attached to the wall of the column (larger, upper right). In this frame the two are close enough that part of the larger bubble is within the search region for the tracked bubble. The bright region of the large bubble provides a better correlation result according to the method described in Section 4.3.3. The center of this bright region is thus taken to be the center of the bubble for this frame, providing the spurious data point (and points in the velocity profile). In the frames before and after this one, the camera and tracked bubble are, of course, in different positions, and the correct bubble is detected.

5.2.2 Distilled Water

The estimated velocity profile of a bubble rising in tap water is shown in Figure 5.4.

In this case, the velocity profile is very close to the theoretical profile for pure water in Figure 1.1, in that the velocity does not decay at all over the course of the run. While it does not coincide with the profile for distilled water in Figure A.3, this can be explained by the presence of contaminants in the water, and this is specifically mentioned in [17], the source of the graph.

As can be seen, this plot of velocity also contains an enlarged region of scatter following the flange break (around 6 s into the run). This is a relatively common occurrence in the results presented here. Figure 5.5 helps provide an explanation.

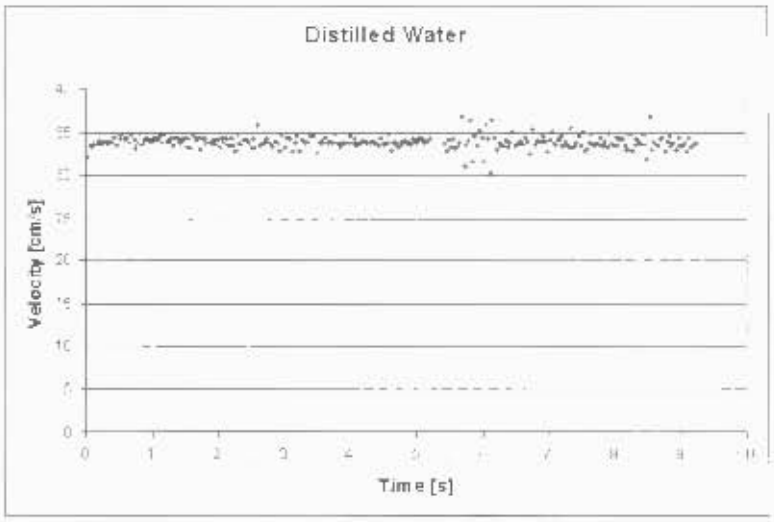

Figure 5.4: Velocity profile of bubble rising in distilled water. Tracked December 14, 2006, 1:28 pm. Total processing time was 34:10 for 277 images.

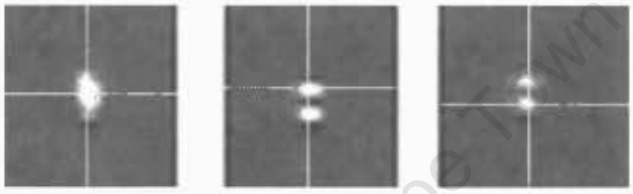

Figure 5.5: Zoomed images of the same bubble, showing calculated bubble center (crossbairs).

The leftmost image is a zoomed image of the bubble from the first frame of the distilled water tracking run, from which the graph in Figure 5.4 was created. As can be seen, the bubble is well illuminated, and the bubble search algorithm has found the approximate center of the bubble.

The middle and right images are two consecutive images corresponding to the first two data points that are noticeably more scattered (marked with the dashes in Figure 5.4). As can be seen, later in run the bubble reflects light from above and below in two distinct regions. This is due to the column lighting from above and below. Both regions are contained in the search region for the bubble, as they are linked by the dilations mentioned in Section 3.3.3. In the middle image, the algorithm determines the center of the bubble to be in the upper bright region, but in the following image, it is determined to be in the lower bright region.

The determined center's shift across the height of the bubble is the cause of the increased scatter visible in the results graphs. For most of the graph, the determined center remains on the same point on the bubble for most of the run. It is set on the actual center, if the bubble is well illuminated. Once the bright region bifurcates, the algorithm generally uses the brighter of the two regions as the determined center. Below the scatter, this tends to be the lower region, due to the lighting of the column from below. Above the scatter, the tendency is to use the upper region, due to lighting from above. The scattered region is

where the brightness of the two regions roughly balances, causing a stochastic selection to take place, and the noisier movement of the detected center is highlighted in the derivative operation used to give velocity.

The variable radius method for detecting the bubble's center is the root cause of this error. This method has a tendency to underestimate the bubble's radius, even when the bubble is well highlighted. The light reflecting off of the bubble appears as an ellipse, or a pair of ellipses, and the radius tends towards the smaller end of its range to maximize the correlation result, rather than the larger end to "fill out" the shape of the bubble from the curvature of the ellipse.

Reducing the range of radius values used could help force both regions to be used in the bifurcated bubble image, putting the detected center between them. However, this could affect the robustness of the algorithm, and it also can not be ascertained yet how the current method, or any modifications, would affect the systems response to larger or smaller bubbles, as the hardware is not yet in place to generate bubbles of different sizes.

Another possibility is to search for the bubble in a frame using a correlation kernel constructed from the previous image of the bubble. While the bubble's image does change across the course of the run, the frame to frame change is minimal. This method could help reduce the scatter in the results as the determined center of the bubble would remain in the center of the bubble, regardless of the bubble's appearance. This method, however, would require a substantial modification of the program's code. It is recommended that a more in-depth study of the bubble processing methods available be conducted, including the effect of different bubble sizes on the apparatus, before any significant changes are made.

Included in any new method for bubble detection should be the equation given for the bubble's effective diameter in [17]:

$$d_e = (a \cdot b^2)^{\frac{1}{3}} \tag{5.1}$$

where a and b are the minor and major axes, respectively. This would allow the apparatus to give more effective results for larger (non-spherical) bubbles. However, it is uncertain whether this equation could be put to effective use, as the irregular shape of the illumination on the bubble could cause problems with any algorithm's attempt to calculate a and b.

As it is, the current algorithm has allowed development to take place and provides results effective for comparison with the previous results shown in Appendix A.

All of the above is noted in the recommendations (Section 6.2). However, ultimately, the best judge of the bubble's d_e is a human analyzing the images, and it is recommended that this take place as a check on the system, at least until a degree of confidence in the system's ability to determine bubble size is reached.

5.2.3 Dowfroth 250

Figure 5.6 shows a plot of the results for a bubble rising in a solution of water containing the surfactant Dowfroth 250 at a concentration of 0.06 PPM.

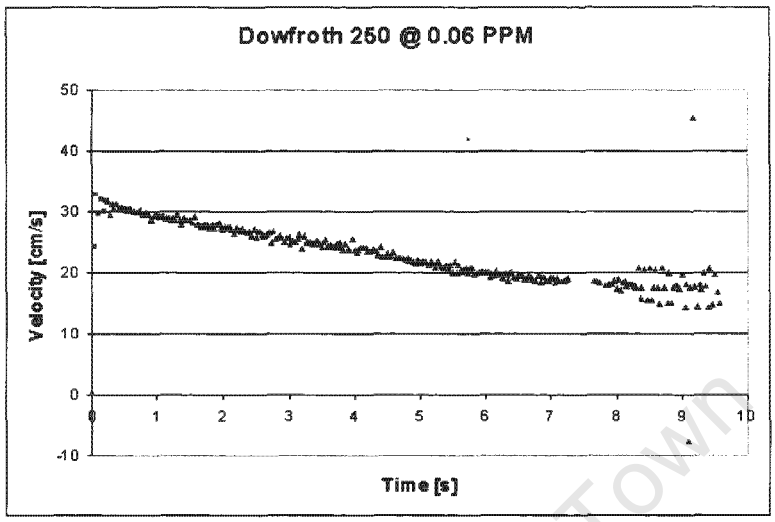

Figure 5.6: Velocity profile for Dowfroth 250 @ 0.06 PPM. Tracked December 19, 2006, 11:28 pm. Total processing time was 30:39 for 287 images.

This profile can be seen to correspond accurately with the plot in Figure A.7 for a concentration of 0.06 PPM. Both approach the supposed terminal velocity of 16 cm/s by approximately 10 s into the run. Human analysis of the images from the run confirmed that the bubbles from both runs are of roughly the same size ($d_e = 1.5$ mm).

Note that the first seven data points in the graph (squares) are a bit more scattered than would be expected from the earlier velocity profiles given. This is due to the fact that the bubble was not correctly detected in the first seven images of the run. Figure 5.7 shows the first image from the run.

The bubble is extremely low in frame, obscured by the edge such that that the correlation for the bubble could not find it effectively, and instead erroneously determined the bubble's center to be to the left of its actual position. The height calculation for the bubble is still approximately correct, because the search region was still constrained by the bubble's profile in the tracking algorithm, but the minor deviations due to noise within those constraints explains the increased scatter. The bubble was successfully acquired (by chance, once the horizontal range of the region overlapped the bubble) by the eighth image, which meant that the processing run was not entirely wasted.

The first image of the MIBC 0.06 PPM run also had this effect. In that case, the first image was suppressed from processing, allowing the bubble to be found accurately in second image, and as can be seen in Figure 5.11, there is much less scatter in the first few data points of the resulting graph. It is sometimes beneficial for brief human examination of the images from a run to take place, so that potential problem points can be isolated and images can be suppressed if necessary. This is usually only necessary at the beginning of

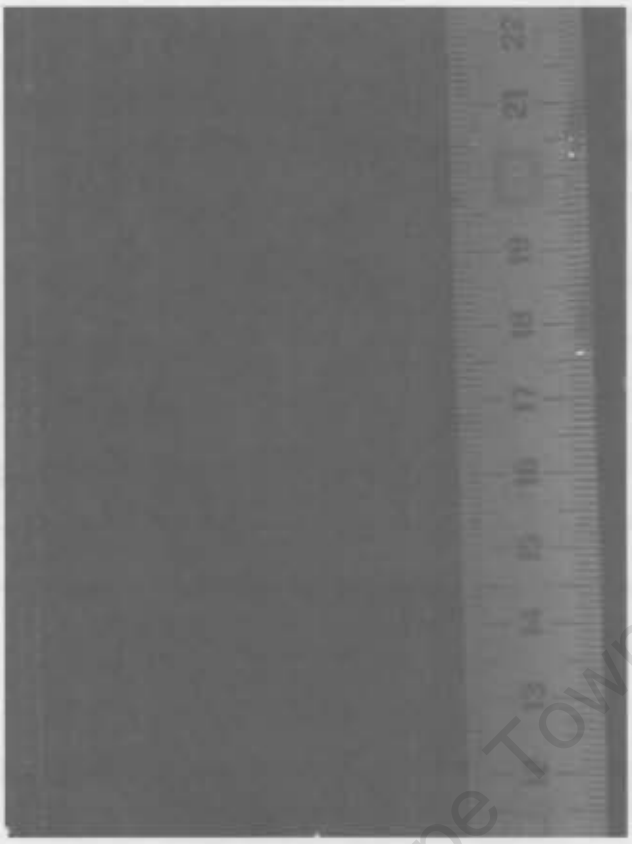

Figure 5.7: First image from tracking run shown in Figure 5.6.

the run. The irregularities in the run in Figure 5.6 were included to demonstrate this. Following the flange, the usual scatter is visible, but there is also a very large pair of spurious velocity points. Figure 5.8 is a zoom of a frame from the run from around those points.

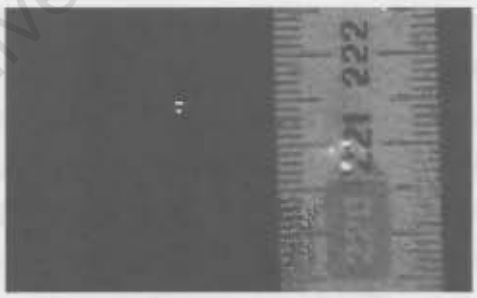

Figure 5.8: Zoom of image from run shown in Figure 5.6, near to spurious velocity points. Contrast and brightness have been enhanced for clarity.

A bubble attached to the tape measure is obscuring the number 221. This caused incorrect sequences to be chosen for a number of frames, giving rise to the spurious data points. This is similar to the processing error shown in Figure 5.3, except in this case, the extra bubble has caused a tape measure processing error, rather than a bubble processing error. The spurious data point is clearly visible on the graph, and can be easily identified, and if necessary, removed from the plot by a human operator.

Figure 5.9 shows a plot for a run in a solution of Dowfroth 250 at a concentration of 30

PPM.

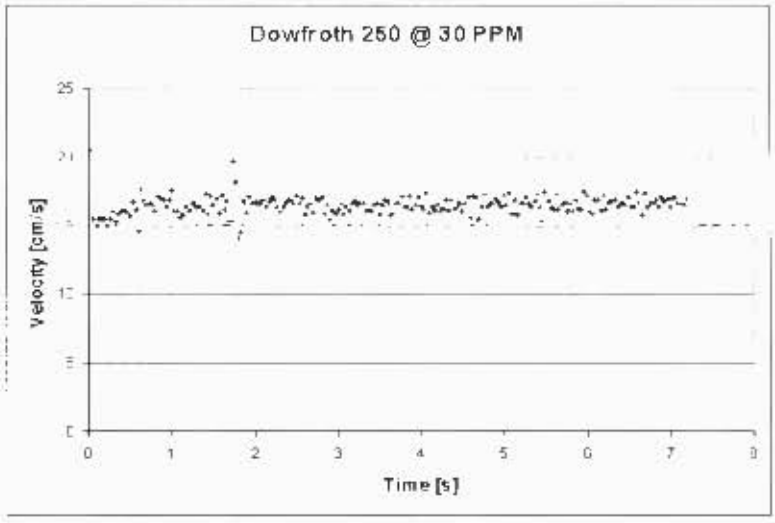

Figure 5.9: Velocity profile for Dowfroth 250 @ 30 PPM, $d_e = 1.5$ mm. Tracked December 15, 2006, 11:26 am. Total processing time was 20:16 for 216 images.

Once again, the similarities to the plot obtained by Sam [17] are clearly visible (Figure A.7, 30 PPM graph). Note that there is no break in the data. This is because the bubble is moving so slowly that it does not reach the flange by the end of the run. This is common for runs in high surfactant concentrations. If a longer run were desired, it would simply be a case of increasing the maximum number of frames used, and requiring more computer memory to be set aside[1].

Of interest is the first non-zero velocity data point, which seems abnormally high. However, there is no second, abnormally low velocity point that would make up the pair that accompanies a position error. Investigation of the output file indicated that there were no processing errors. This high velocity value actually corresponds to the expected behaviour for high surfactant concentrations. Initially, there is a higher velocity as the bubble is released, but the abundant surfactant molecules are rapidly adsorbed onto the bubble's surface, preventing it from acquiring its maximum possible velocity (U_{max} - see Section 1.4).

The other seemingly spurious data point of interest for investigation is the high velocity reading just before 2 s into the run. Figure 5.10 is the frame corresponding to this point.

The crosshairs in the image show the calculated positions used for the two numbers used to determine the bubble's heights. As can be seen, the position used for the number 40 is incorrect. This is caused by a large number of bubbles on the tape measure - they interfere with the accuracy of the normalized cross-correlations.

There are two possible methods to fix this. The first is to increase the threshold for the correlation result of a number that it must satisfy to be used to calculate the bubble's

[1] As it stands, a run requires memory for 300 images of 640×480 bytes, working out to over 92 Mb of RAM.

Figure 5.10: Track image corresponding to spurious data point in Figure 5.9.

position (Section 4.5). In this case, it would result in the numbers 41 and 39 being used to calculate the bubble's height[2].

The second solution is to make use of a bubble release mechanism that does not force a stream of bubbles to flow during set up. This problem was mentioned in Section 2.2.1, and the image in Figure 5.10 demonstrates the extent of this effect. The elimination of the bubbles stuck to the tape measure and wall of the column would be a byproduct of implementing such a bubble release mechanism. In the meantime, the algorithm performs quite well and relatively accurately, despite the noise.

5.2.4 MIBC

Figure 5.11 shows a run in a surfactant solution of MIBC at a concentration of 0.06 PPM. Also included is a plot of the bubble's horizontal frame position throughout the run, in pixels, plotted such that higher in the graph is to the left in the frame.

Most of the spurious points in this profile result from phenomena that have already been covered. The first, minor scatter occurring at around 0.5 s is due to a bubble obscuring numbers used to calculate bubble position, such as in Figure 5.10. The major errors in the points on either side of the flange are caused by numbers being obscured by bubbles

[2]The two numbers closest to the bubble are used to decrease the effects refraction that could increase height error.

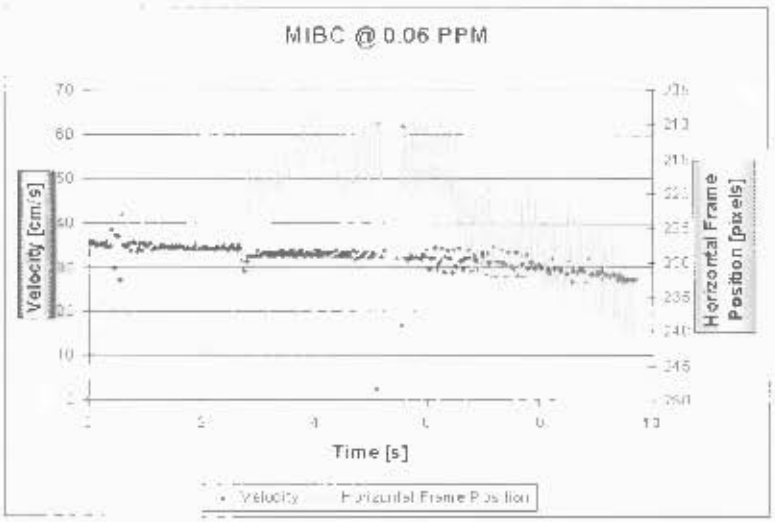

Figure 5.11: Velocity profile and horizontal frame position for MIBC @ 0.06 PPM, d_c = 1.5 mm. Tracked December 15, 2006, 12:19 pm. Total processing time was 36:52 for 291 images.

or smudges on the column, causing the wrong sequences to be chosen. In both cases, the system managed to recover within one frame.

It is the velocity deviation at around 3 seconds into the run that is of interest in this chart. At first, the spontaneous dip in the velocity appeared to be an error in the processing, but investigation of the output file and video footage proved this was not the case. As can be seen from the horizontal frame position, there is a real perturbation of the bubble at that point in the run. The bubble is knocked to the left, and oscillations, visible as the jagged lines in the horizontal position, start as a result.

The video footage shows no visible obstructions contacting the bubble at this point. The cause could be a sudden change of temperature in the solution in the column, or the size of the bubble, increasing as the bubble rises and the pressure decreases, reaches a critical level where oscillatory bubble motion can set in. A combination of these two is also very likely.

The important point for this dissertation, is that the behaviour was picked up clearly and easily in the processing. The sudden, subtle change in velocity is a part of the bubble behaviour that this apparatus was designed for, and the format of the data output allows plots of different aspects of the run (such as velocity and horizontal frame position, in this case) to be shown and correlated.

The final test run presented here, performed with a bubble rising in a solution of MIBC at 30 PPM, is shown in Figure 5.12.

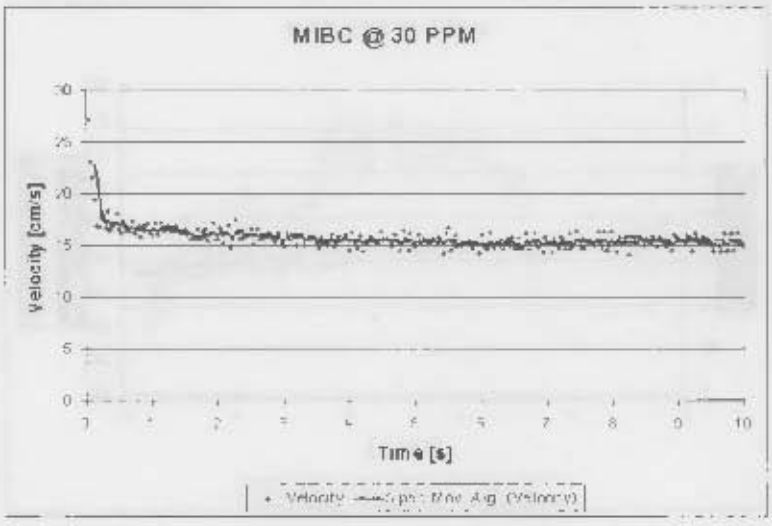

Figure 5.12: Velocity profile, with trend line, for MIBC @ 30 PPM, $d_c = 1.5$ mm. Tracked December 15, 2006, 12:40 pm. Total processing time was 32:39 for 299 images.

There are no processing errors visible in the graph (despite the presence of a large number of bubbles stuck to the tape measure), the scatter is relatively well contained, and the graph conforms to the expected behaviour for bubbles rising in high surfactant concentrations. A trend line is added to show the effect of a smoothing kernel applied to the data.

The initially high velocity decays rapidly towards the terminal velocity - this is more clear than in the case of the high concentration Dowfroth 250 run (Figure 5.9).

5.3 Error of Parallax

The calculations performed by the algorithm to obtain the results make the assumption that the tape measure numbers and the center of the bubble are coplanar (they lie the same distance from the wall of the column facing the camera). The 2D images generated in tracking do not show lateral bubble motion towards and away from the camera.

Examination of the graph in Figure 5.11 shows that there is a significant amount of horizontal bubble motion, even before the oscillations begin. It can be assumed that a similar motion pattern would be evident perpendicular to the camera.

If the bubble and the tape measure are not in the same plane, error of parallax can come into effect. This section contains an investigation of the uncertainties associated with error of parallax, and its effect on the results.

The column was filled with water, and two rulers were set up in front of and behind the column, relative to the camera. Figure 5.13 is an image from the camera showing the rulers, and Figure 5.14 is an illustration of the side view geometry.

The lines on the image show the places where readings were taken from the two rulers. The slight sloping visible in some of the markings on the front ruler is due to its beveled

Figure 5.13: Image from camera showing two rulers and lines used for ray data points. The front ruler is on the right, and slightly out of focus, though the numbers are still visible. The rear ruler, behind the column is, visible on the left.

edge. Readings were taken from the leftmost edge, for consistency, and as that edge was closest to the perspex of the column.

Error of parallax, in this case, affects the bubble's height reading, associated with the y co-ordinates of the bubble and numbers[3]. Thus, the horizontal (x) components of the co-ordinates will be neglected for this analysis, allowing the two ruler readings in Figure 5.13 to be represented as rays in the profile image shown in Figure 5.14.

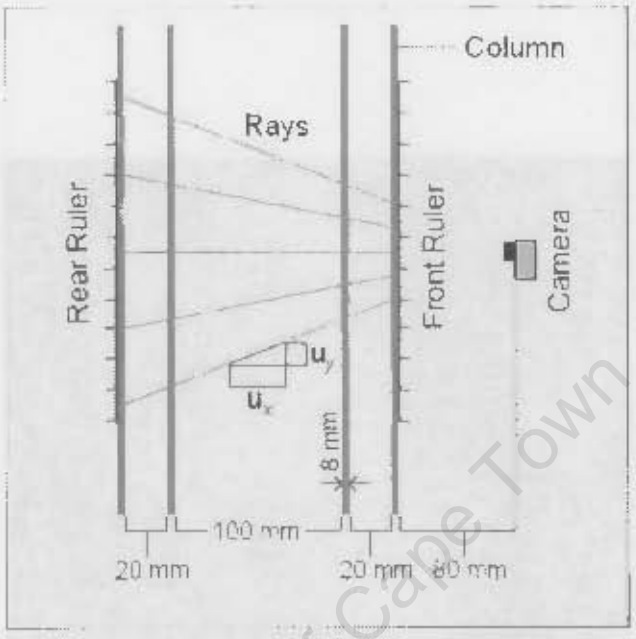

Figure 5.14: Rays through the column between the two rulers.

The profile shows various paths through the column that rays can take, associating a measurement on the rear ruler with one on the front ruler. There would be a slight refraction as the rays pass through the perspex walls, but this has been neglected as the perspex makes up only a small part of the total distance.

For a given ray entering the camera, if the horizontal position of the target (bubble) has an uncertainty (shown as u_x in Figure 5.14), there will be a corresponding vertical uncertainty in the reading (shown as u_y). The vertical uncertainty is dependent on the gradient of the ray, and hence, the screen position of the bubble.

The ray gradient as a function of screen position was determined as follows: A set of gradient values was determined using the readings from the two rulers from Figure 5.13, and the knowledge that the column width was 176 mm. A linear approximation to this data set was generated by least squares. Figure 5.15 shows the data points (crosses), and the fitted curve (line).

[3]The x co-ordinate will only affect the height reading in the case of optical distortion in the image, from the camera or refraction in the column. The purpose of the ray undistort algorithm (Section 4.5) was to compensate for this. It showed that the effect was minimal.

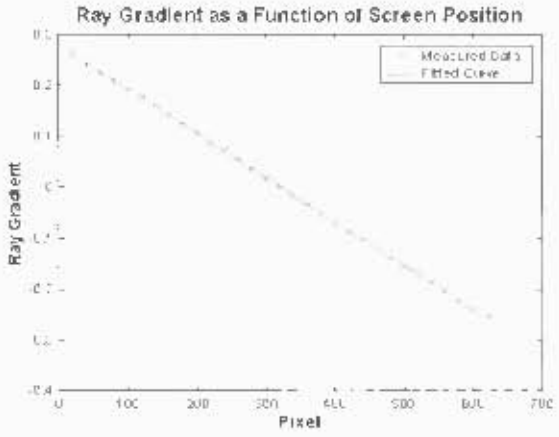

Figure 5.15: Measured gradient values and linear (least squares) approximation to data set.

As can be seen, the approximation fits the data well. This graph can then be used to estimate a vertical uncertainty from the screen position and a given horizontal uncertainty according to the following equations:

$$g = \frac{u_y}{u_x} \tag{5.2}$$

$$\therefore u_y = g \cdot u_x \tag{5.3}$$

Figure 5.16 shows the magnitude of u_y for various values of u_x, using the vertical frame position of the bubble from the tap water tracking run, shown in Figure 5.1.

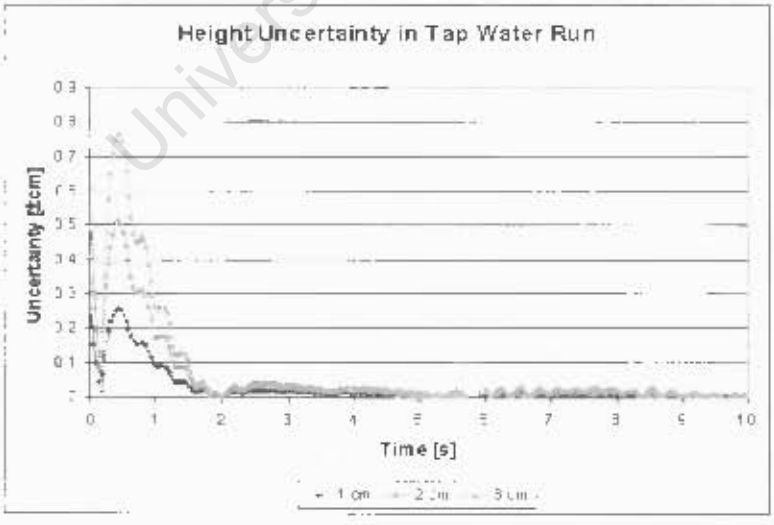

Figure 5.16: Height uncertainty in tap water run, for various assumed values for u_x.

Thus if the bubble is 1 cm closer to or further from the camera than the plane containing the tape measure, this results in a maximum uncertainty of roughly 0.25 cm in the position reading. If the bubble deviates a greater distance from the tape measure plane, the

maximum positional uncertainty increases. However, once the bubble is centered, error of parallax disappears, as does the uncertainty associated with it.

The effect of error of parallax on the velocity readings is more complex, as it is *changes* in the error of the height reading that affect velocity. A continuous position error would not affect the velocity reading. An impression of the velocity uncertainty can be obtained by estimating the horizontal motion of the bubble (towards and away from the camera) between frames.

The bubble's lateral motion in the frame (perpendicular to the camera) is assumed to be similar to, and give an indication of, the bubble's motion towards and away from the camera. Therefore, the bubble center's x co-ordinate values are converted to a measurement in centimeters, and then the mean *change* in frame position is used to plot velocity uncertainty. The plots of velocity and horizontal frame position against time for the tap water run are shown in Figure 5.17. Horizontal frame position has been converted from pixels to centimeters (using the tape measure as an estimate, 50 pixels/cm), so the horizontal frame position is given as a distance from the center of the screen. Once again, upwards in the graph represents leftwards in the screen.

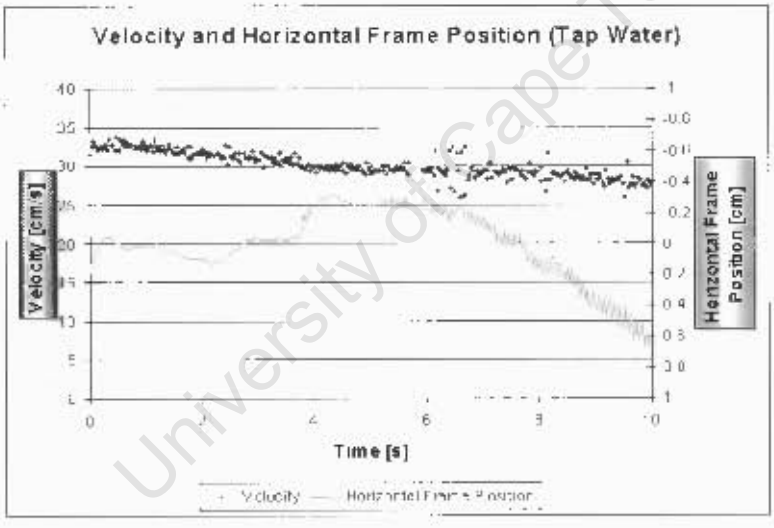

Figure 5.17: Velocity and horizontal frame position (tap water run).

From the horizontal frame position data in Figure 5.17, the average change in frame position was determined to be 36.23×10^{-3} cm, with a standard deviation of 38.90×10^{-3}. The maximum jump in frame position was found to be 0.16 cm, quite near the end of the run, where oscillations began to get large.

Note that the standard deviation is larger than the mean value, indicating a broad scatter in the data set. This is evident in the plot of horizontal frame position, as the bubble has minimal lateral motion in the beginning of the run, but this increases once oscillations begin.

Assuming that the horizontal motion towards and away from the camera would be similar to the plot in Figure 5.17, its mean and maximum values can be used to estimate the velocity uncertainty due to error of parallax. These plots can be seen in Figure 5.18.

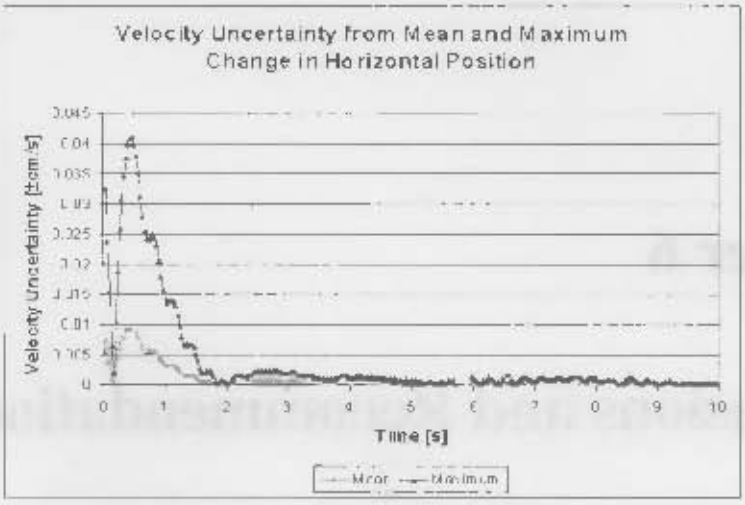

Figure 5.18: Velocity uncertainty, assuming change in horizontal bubble position between frames equivalent to mean and maximum values for lateral motion (perpendicular to camera) in tap water run.

As can be seen, the uncertainties from error of parallax are very small in comparison to the velocities that the bubbles achieve. The peak value for the mean is one hundredth of a cm/s, and the maximum plot peaks at only four times that. Note also that, since the bubble initially does not move much laterally, the uncertainty value near the beginning is likely to be low - the mean value gives a better estimate. Further into the run, once the oscillations begin, the maximum value gives the better uncertainty estimate, but since the bubble is centered in the frame by this time, error of parallax fades away into insignificance, as does the uncertainty associated with it.

It should be noted that the statistical data used in this analysis is from a single run. For a full error of parallax uncertainty analysis, much more data would need to be collected and processed. Nonetheless, the shortened analysis presented here indicates that error of parallax in the readings does not have a significant influence on the velocity readings obtained by this apparatus. Uncertainties and noise from other sources play a much more significant role in the velocity profiles.

Chapter 6

Conclusions and Recommendations

This chapter summarizes the conclusions reached during the course of this dissertation. This is followed by recommendations for continuing work on the apparatus.

6.1 Conclusions

An apparatus has been developed to track a single bubble rising in an aqueous solution using a video technique, and process the resulting footage in order to obtain the bubble's velocity profile, amongst other data.

The system's automation in terms of the tracking control, as well as in terms of processing, is a novel application to this problem. Though some human review of the frames before the processing is usually required, it nonetheless saves operators time and effort, as large amounts of data need to be collected and processed in the experimental investigation of bubble velocity profiles. Other systems performing the same work required operators to perform the tracking and processing.

The tracking mechanism developed performs well enough to track bubbles rising at 40 cm/s, though there is some minor oscillatory behaviour evident in its setpoint response.

The post-processing image analysis can provide accurate bubble velocity profiles, though some modifications will be required to reduce errors in determining the bubble's position. The algorithm takes a significant amount of time to complete the processing for a full tracking run (30 to 40 minutes for 250 to 300 images). A number of methods were considered to reduce the processing time. Decreasing the correlation region size and reducing the number of sequences tested were methods that performed well and were included in the algorithm, but the FFT version of the normalized cross-correlation was not found to be robust enough.

In both the tracking and the post-processing, the flange halfway up the column has been compensated for in software and poses no significant problem to the system's performance.

The system was tested by tracking bubbles rising in various conditions, including tap wa-

ter, distilled water, the surfactant Dowfroth 250 (at concentrations of 0.06 and 30 PPM), and the surfactant MIBC (again, at 0.06 and 30 PPM). The results obtained coincided with results obtained previously with similar systems, as well as with the current models for bubble behaviour in solution. Spurious data points are easily identifiable in the velocity plot, and can thus be isolated, and if necessary, removed.

An investigation into the effect of error of parallax on the results indicated that its uncertainty is less significant than other sources of error in the system.

6.2 Recommendations

The bubble generating hardware currently does not allow for the generation of bubbles of particular sizes. In addition, it forces a stream of bubbles to flow in the column during set up, which can stick to the inside of the column and to the tape measure. A new bubble generating method, which avoids both of these problems, should be developed.

The tape measure currently used as a reference in the post-processing algorithm does not make use of any formatted patterns suitable for machine vision. It is also difficult to position adequately as it is flexible and suspended in the column. While adequate, and sometimes excellent, results could be obtained, a tape measure specifically fabricated to the system would enhance performance. Such a tape measure should have a repetitive pattern suitable for image processing, such as a checkerboard or bar-coding, integrated with a human-readable number system. Any pattern should have a spatial frequency of greater than 1.67 cm, to avoid the problem of aliasing. This tape measure should also be firmly attached to the side of the column for its entire length, so that it remains stationary, flat, and coplanar with the bubble's point of release.

The post-processing algorithm's method for finding the center and radius of the bubble, though sufficient for development, is inadequate for experimental use. A better method should be found by investigation, and then implemented.

Before post-processing, human review of the images is required to spot potential problem areas. Further automation in this area could result in the dropping of this requirement.

In terms of the extendability of the apparatus, there are a number of research paths which could be followed.

A multiple camera version of the system could use several cameras positioned along the height of the column. With each camera capturing video while the bubble is in its field of view, a velocity profile could be constructed by concatenating the footage from all of the cameras. The advantages of this method would include the fact that mechanical parts (such as the motor, gearbox and conveyor belt) would not be required, which could reduce the costs of constructing such an apparatus. The stationary camera system would also lend itself to calibration (allowing the bubble's position to be determined by its frame position only), thereby removing the need for the intense image processing of a tape measure or other reference. The main disadvantage, however, is that the apparatus could

require a large number of high resolution cameras. This could increase the cost of such an apparatus, potentially eliminating any benefits of the multiple-stationary camera version. An investigation would need to take place to determine first if such a system is required in addition to the one presented in this dissertation, and second, if it is feasible in terms of costs.

Using two mobile cameras to track the bubble would allow a complete 3D model for the rise path of the bubble to be constructed as a function of time, giving velocity as well as much additional information about bubble behaviour (lateral motion and oscillations, in particular). This would require the redesign of some of hardware components of the apparatus in addition to much of the software.

Performing some reworking of the tracking and processing algorithms could allow for the tracking of a cloud of bubbles, although this would again require the redesign of hardware (particularly the bubble release mechanism, to give a cloud of bubbles). The ability to track and plot the paths of multiple bubbles would allow bubble interactions to be studied.

Appendix A

Previous Results

This appendix includes data from the McGill University apparatus, tracked and processed by hand. These graphs are provided for comparison with the results obtained with the automated UCT apparatus.

Note that in some cases, the x-axis of the graphs has been set to distance, rather than time, which is the standard used primarily in this dissertation. The difference is subtle, so comparisons are still possible. Distance can be used as the independent variable in the results from the UCT apparatus, as the bubble's vertical position is provided in the output file.

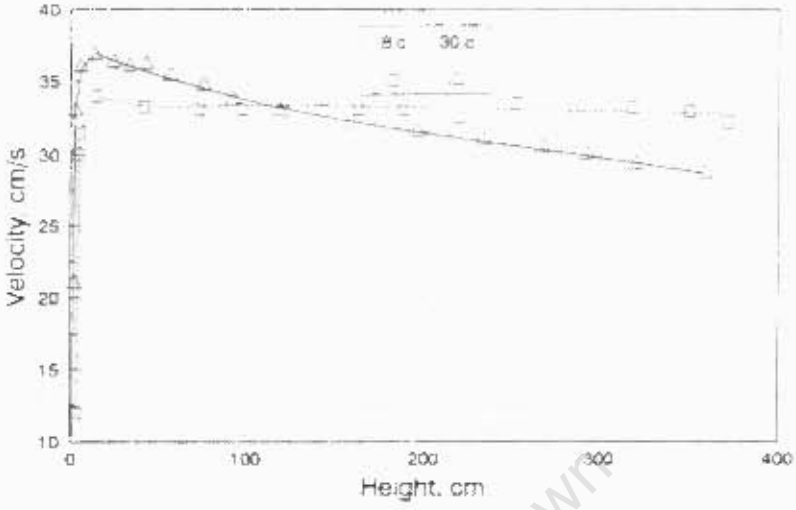

Figure 2- Velocity profiles in water alone at 8 and 30° C

Figure A.1: Tap water for different temperatures [25].

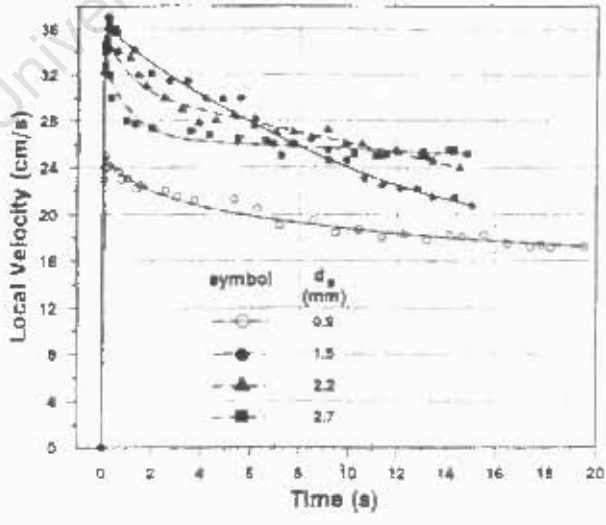

Fig. 6 Bubble velocity profiles in the same tap water for different bubble sizes (April '94)

Figure A.2: Tap water for different bubble sizes [17].

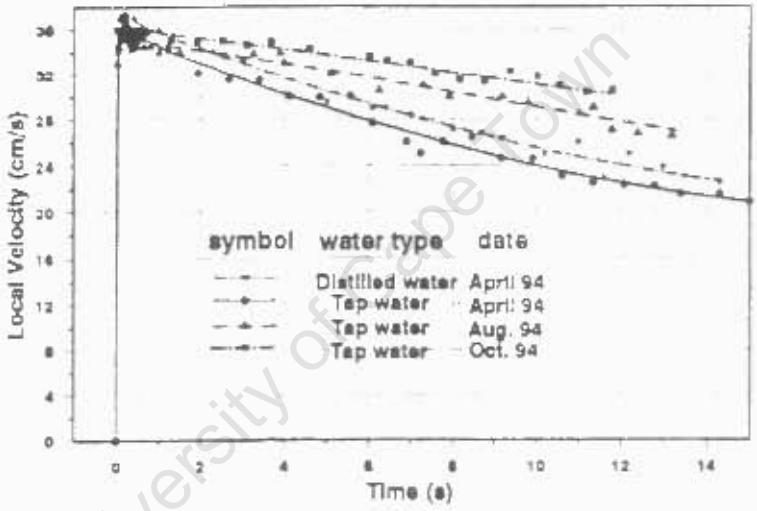

Fig. 7. Bubble velocity profiles in distilled water and tap water on different dates (d_e = 1.5 mm)

Figure A.3: Distilled and tap water from different dates [17].

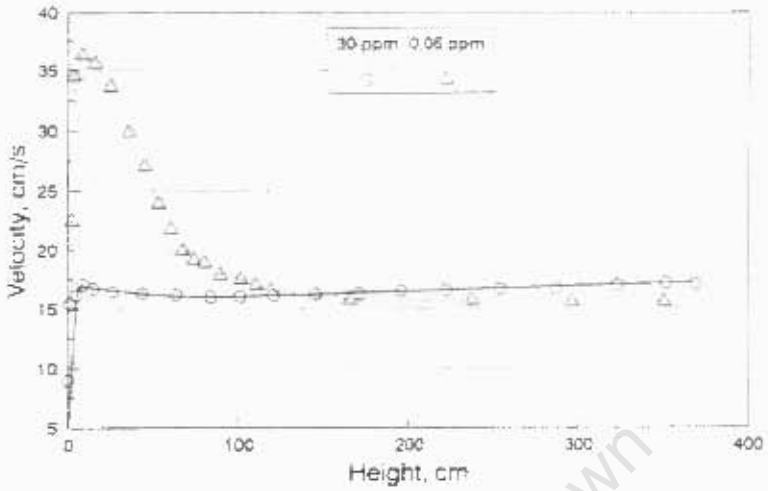

Figure 3- Velocity profile at a two concentrations of frother DF 250

Figure A.4: Dowfroth 250 for different concentrations [25].

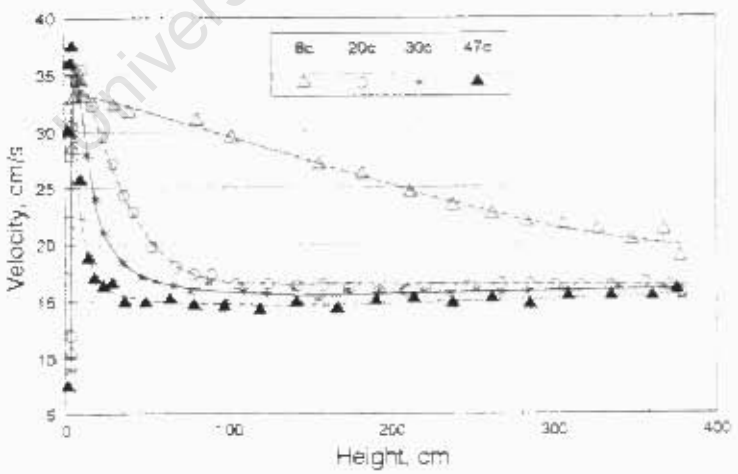

Figure 4- Velocity profile as a function of temperature for 0.06 ppm DF 250

Figure A.5: Dowfroth 250 for different temperatures [25].

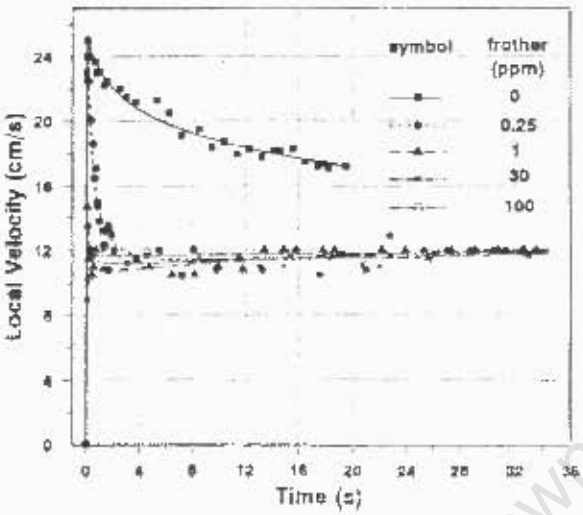

Fig. 8. Bubble velocity profiles (d_e = 0.9 mm, frother = Dowfroth 250)

Figure A.6: Dowfroth 250 at different concentrations for bubbles with $d_e = 0.9$ mm [17].

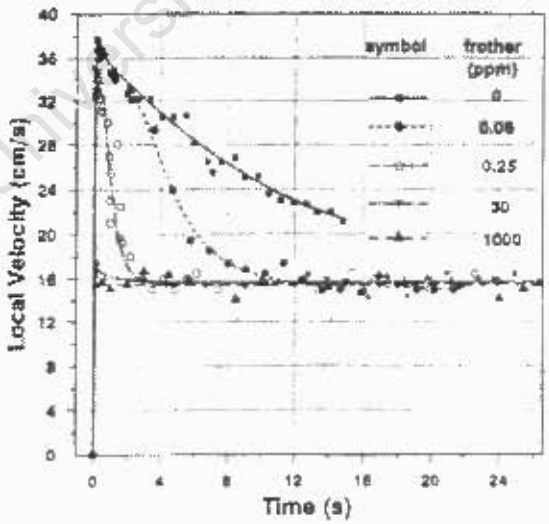

Fig. 9. Bubble velocity profiles (d_e = 1.5 mm, frother = Dowfroth 250)

Figure A.7: Dowfroth 250 at different concentrations for bubbles with $d_e = 1.5$ mm [17].

Bibliography

[1] M. Braae. *Control Theory for Electrical Engineers*. UCT Press, 1994.

[2] M. Braae. *Control Engineering - 2*. UCT Press, 1996.

[3] D. Clarke. Pid algorithms and their computer implementation. *Transactions of the Institute of Measurement and Control*, 6(6):305–316, 1984.

[4] R. Fdhila and P. Duineveld. The effect of surfactant on the rise of a spherical bubble at high reynolds and peclet numbers. *Physics of Fluids*, 8:310–320, 1996.

[5] M. Frigo and S. Johnson. The design and implementation of fftw3. *Proceedings of the IEEE*, 93:216–231, 2005.

[6] I. Gyongy and D. Clarke. On the automatic tuning and adaptation of pid controllers. *Control Engineering Practice*, 14:149–163, 2006.

[7] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. Technical report, Infotech Oulu and Department of Electrical Engineering, University of Oulu, 1997.

[8] J. Laskowski. Effect of frothers on bubble coalescence, bubble rising velocity and induction time in bubble-particle attachment. Research Proposal, March 2001.

[9] J. Lewis. Fast normalized cross-correlation. http://www.idiom.com/ zilla/Work/nvisionInterface/nip.html.

[10] Z. Mathe, M. Harris, C. O'Connor, and J.-P. Franzidis. Review of froth modelling in steady state flotation systems. *Minerals Engineering*, 11(5):397–421, 1998.

[11] K. Matis and P. Mavros. Recovery of metals by ion flotation from dilute aqueous solutions. *Separation and Purification Methods*, 20:1–48, 1991.

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

[13] A. Oppenheim, R. Schafer, and J. Buck. *Discrete-Time Signal Processing*. Prentice Hall, 2nd edition, 1999.

[14] S. Reid. Implementation of a bubble tracking camera system. Technical report, University of Cape Town, 2003.

[15] S. Reid, A. Wilkinson, M. Harris, and E. Randall. An automated data extraction algorithm for a bubble tracking camera system. In *The Seventeenth International Symposium of the Pattern Recognition Association of South Africa*, 2006.

[16] A. Sam. *Single Bubble Behaviour Study in a Flotation Column*. PhD thesis, Department of Mining and Metallurgical Engineering, McGill University, Montreal, Canada, 1995.

[17] A. Sam, C. Gomez, and J. Finch. Axial velocity profiles of single bubbles in water/frother solutions. *International Journal of Mineral Processing*, 47:177–196, 1995.

[18] F. Takemura and A. Yabe. Rising speed and dissolution rate of a carbon dioxide bubble in slightly contaminated water. *Journal of Fluid Mechanics*, 378:319–334, 1999.

[19] D. Tao. Role of bubble size in flotation of coarse and fine particles - a review. *Separation Science and Technology*, 39(4):741–760, 2004.

[20] I. Ulrich and I. Nourbakhsh. Firewire untethered: High-quality images for notebook computers. Advanced Imaging Magazine, January 2000. 69-70.

[21] C. Vigneault, B. Panneton, and G. Raghavan. Real time image digitizing system for measurement of air bubbles. *Canadian Agricultural Engineering*, 34(2):151–155, 1992.

[22] M. Wu and M. Gharib. Experimental studies on the shape and path of small air bubbles rising in clean water. *Physics of Fluids*, 14(7):49–52, July 2002.

[23] J. Zhang and L.-S. Fan. On the rise velocity of an interactive bubble in liquids. *Chemical Engineering Journal*, 92:169–176, 2003.

[24] Y. Zhang and J. Finch. A note on single bubble motion in surfactant solutions. *Journal of Fluid Mechanics*, 429:63–66, 2001.

[25] Y. Zhang, C. Gomez, and J. Finch. Terminal velocity of bubbles: Approach and preliminary investigations. In *Proceedings of the International Symposium on Column Flotation*, pages 63–69, 1996.