

An Automated Data Extraction Algorithm for a Bubble Tracking Camera System

S. C. Reid¹, A. J. Wilkinson¹, M. C. Harris², E. W. Randall²

¹Department of Electrical Engineering
University of Cape Town

²Department of Chemical Engineering
University of Cape Town

stewart@chemeng.uct.ac.za

Abstract

The velocity profiles of bubbles rising in water are of interest to the minerals processing industry. The Centre for Minerals Research at University of Cape Town has an apparatus which, using a mobile camera with a control loop, can automatically track a bubble as it rises in a column of water. Images of the rising bubble are then post-processed to obtain accurate bubble positional data using image processing techniques. From the positional data, velocity profiles are estimated. This paper includes a discussion of the post-processing algorithm, as well as an example of the results that were obtained in the application of the algorithm to experimental data.

1. Introduction

1.1. Background

Many processes in the minerals processing industry, use dispersed air (fine bubbles) to separate out particles from water-based mixtures. Such systems feature complex dynamic environments, which makes fundamental understanding difficult [1, 2].

Research by Sam et al [1] has shown that study of bubble velocity profiles as they rise in varying solutions can be used to investigate bubble interactions in these environments. As a result, the Centre for Minerals Research at the University of Cape Town (UCT) has designed a system which uses a camera to automatically track bubbles in a column. The UCT system is similar to the one used by Sam at the University of Montreal, Canada.

Unfortunately, the manual processing of the large number of images generated during the tracking of bubbles constitutes a laborious and dull task. As a result, steps have been taken to automate the data extraction procedure for the system at UCT.

Since no off-the-shelf solution to the automation problem existed, one was commissioned by the Departments of Chemical and Electrical Engineering.

1.2. Apparatus Description

A schematic of the bubble tracking camera system operating at UCT can be seen in Figure 1. It consists of a transparent perspex column with a cross section measuring 20cm by 20cm, and a height of approximately 4m. Bubbles are released into the

column via a glass capillary. The bubbles are approximately spherical, and have diameters in the range $0.4 < d_e < 1.4$ mm, which is small enough that they do not exhibit helical motion when rising [1].

Standing next to the column is a track on which is mounted an IEEE 1394 (FireWire) camera, such that the camera is approximately 8 cm from the column. The camera is configured for 640×480 8-bit grayscale operation at 30 fps, and is mounted on its side, such that the long axis of its view frame is parallel to the direction of the bubble's motion. The top left pixel is

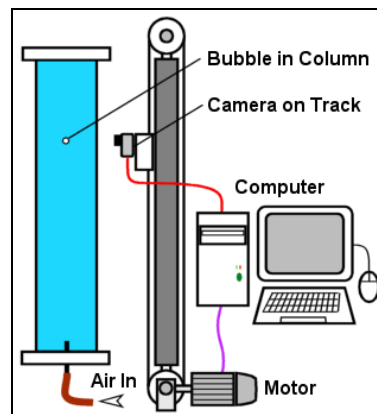


Figure 1: Reduced schematic of the bubble tracking apparatus at UCT.

considered the position (0,0), the standard for top-down Bitmap and Portable Grayscale Map (PGM) images [4, 5]. A tape measure is suspended inside the column, next to the path of the rising bubble. The camera's position allows about 11cm of the tape measure to be seen in the frame. An example of an image taken from the camera can be seen in Figure 2. It should be noted that if the bubble in the image is closer to or further from the camera than the tape measure, error of parallax can come into effect, though in practice this effect is small. This paper assumes that the tape measure and bubble are coplanar.

During tracking, a PC adjusts the position of the camera on the track with a motor, using images from the camera to provide feedback. These images are then saved for post-processing,

where accurate bubble positional data is extracted, and the velocity profile estimated. A more detailed description of the apparatus can be found in [3].

The program containing the algorithm was developed in Matlab and Visual C++, and finally written in Visual C++.

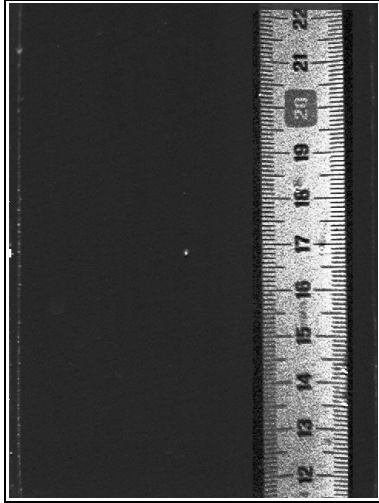


Figure 2: View from FireWire camera. The contrast around the ruler has been increased for clarity.

1.2.1. Bubble Height Measurement Method

A number of methods were considered to determine the absolute height of the bubble in each frame. They are discussed here:

A prototype of the system made use of an optical counter mechanism instead of a tape measure, which gave the position of the camera on the track. Frames were stamped with the position of the camera when taken.

Given the camera's height, the bubble's height had to be estimated, rather than calculated from the bubble's position in the frame, since no absolute measuring device was present in the frame. This uncertainty was compounded by the fact that there was a significant hardware time lag found in the counter mechanism. Accuracy using this method could not be guaranteed.

While characterizing and then compensating for these uncertainties was considered, placing an absolute measuring device in the column had the advantage of eliminating them completely. Additionally, using such a device condensed all input into the algorithm into one source – the set of images.

A tape measure was chosen for a number of reasons. First, the tape measure provides a human-readable format, so that the height of the bubble in any frame can be read easily by a human should the need arise, such as in debugging and error checking results. Secondly, the tape measure is not susceptible to aliasing where a less differentiated repeating pattern might be. Finally, the initial investigation into the feasibility of using a measuring device in the column was done using a tape measure. Results from that investigation indicated that a high degree of accuracy could be obtained from it.

1.3. Problem Specifications

The set of images generated by a track run consists of a couple

hundred frames similar to Figure 2, though the vertical position of the bubble in the frame tends to slowly vary, as do the numbers on the tape measure.

A plot of bubble velocity versus time is required from such a set of images. The velocity plot is estimated from a plot of bubble position versus time, using a discrete derivative:

$$V(t) = \frac{y_i - y_{i-1}}{t_i - t_{i-1}} \quad (1)$$

2. Algorithm Structure

The post-processing algorithm which is used to obtain the positional plot of the bubble, and hence the desired velocity plot, is fairly complex; an overview is given below:

For each frame:

1. Determine bubble position:
 - (a) Predict likely bubble position based on positions in previous frames.
 - (b) Obtain a region around the prediction to be searched for the bubble.
 - (c) Search for the bubble using correlation.
2. Read Tape Measure:
 - (a) Find the tape measure in the frame.
 - (b) Estimate the positions of the numbers on it.
 - (c) For each estimated number position:
 - i. Perform several correlations in the surrounding region, using templates shaped like all the numbers that could occupy that position.
 - ii. For each template, obtain a normalized "fitness" describing how well that template matched the region.
 - (d) Use the fitness values to determine what number sequence is the best fit for the tape measure in this image.
3. Calculate absolute bubble height based on position of bubble and position of numbers.
4. Calculate bubble velocity from position data.

Each step in the above overview is described in proceeding sections.

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.

3.1. Initial Prediction for Bubble Position

The algorithm uses information from previous frames to estimate where the bubble will be in the frame that it is currently processing. (In the initial condition, the estimate is simply set to near the bottom of the frame.)

The vertical prediction value is given by:

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

where Y_n^p is the predicted vertical position of the bubble in the

frame, and the Y_{n-i} terms are the actual positions of the bubble in the previous two frames. This method effectively predicts that the bubble will move the same amount from frame to frame.

The horizontal component of the bubble's position is predicted to remain unchanged between frames.

3.2. Search Region

The region searched is confined to a rectangular region centered on the predicted bubble position. It is kept a constant 160 pixels in height (25% of the image height of 640 pixels) by 48 pixels in width (10% of the image width of 480 pixels). This was found to be a sufficiently large region around the prediction to ensure that the bubble remained in the search area.

The disadvantage of the static region is that it does not take into account the fact that, later in the image set, the bubble remains centered and relatively stationary between frames. A dynamically resizing region would save time and further reduce the possibility of spurious detections. Development is still underway to exploit this fact.

3.3. Searching for the Bubble

The region is searched using a series of slightly modified normalized cross-correlations, using a series kernels shaped like circles of various radii. While a wide range of radius values is used in the initial condition, the range is reduced once the bubble size becomes apparent in order to save time.

Regular two dimensional normalized cross-correlation, using a size $N \times M$ kernel can be written as:

$$C_{x,y} = \frac{\sum_i \sum_j (I_{i,j} - \bar{I}) \cdot (K_{x+i,y+j} - \bar{K})}{\sigma_I \cdot \sigma_K \cdot N \cdot M} \quad (3)$$

where K refers to the kernel and I refers to the image region. Before correlation, each element in the kernel and image region arrays has the mean of the array subtracted, and is divided by the array's standard deviation. The output is also divided by the size of the region.

The actual correlation equation used to find the bubble is based on equation (3):

$$C_{x,y} = \frac{\sum_i \sum_j (I_{i,j}) \cdot (K_{x+i,y+j} - \bar{K})}{\sigma_K \cdot N \cdot M} \quad (4)$$

The only difference is that the mean and standard deviation of the image region have been removed from the equation. The bubble usually appears as an extra bright region in the region searched, and this modification allows that fact to be exploited to aid in reducing spurious detections.

Thus, the bubble's position and size is given by the best fitting kernel in terms of its position and radius.

4. Tape Measure Processing

To find the absolute height of the bubble, the positions of the numbers on the tape measure in the image must be found. This part of the algorithm has several steps, which are described here.

4.1. Tape Measure Position

First of all, the tape measure itself must be found in the image. This is done automatically, so that the tape measure can be moved without the need for human input.

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

Figure 3 shows the plot obtained from averaging every column of the image in Figure 2. The tape measure's presence can clearly be seen in the set of higher values on the right.

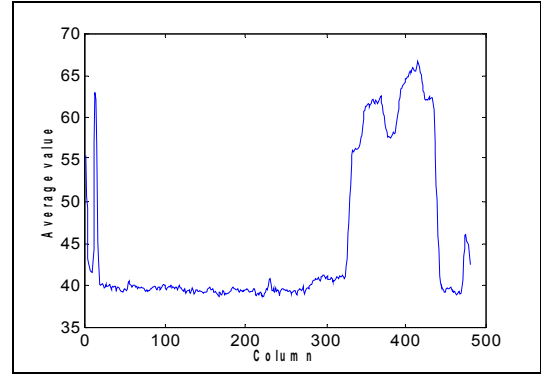


Figure 3: Graph showing column average for Figure 2.

A number of one dimensional 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 ruler edges.

The apparent width of the ruler in each frame does not change much through the set of images, so the range of width values is kept small to avoid spurious ruler detections – such as from the spike on the left of Figure 3.

4.2. Digit Kernels

In order to test for the presence of numbers on the tape measure in the frame, correlation kernels resembling the numbers had to be constructed.

Examination of Figure 2 shows some important practical facts about the tape measure. First, the numbers are of a consistent font. Secondly, each number, while it varies in width, is exactly centered on the marker for its position. Finally, it should be noted that multiples of ten are printed as light on a dark background, opposite to regular numbers. Provision is made for this by negating the kernel for such numbers before correlation.

A set of 10 kernels, one for each digit, was created. The kernels store images of the digits as they appear in the track images – on their sides. Thus, all kernels have the same width (26 pixels in this case), but vary in height according to the “width” of the digit. For instance, the digit “1”, being very thin, results in a short kernel (8 pixels), whereas the broader “2” has one of the tallest (13 pixels).

These kernels can be “stacked” on top of each other giving a composite number, which can be used in a correlation operation.

4.3. Initial Number Position Estimation

At this point in the algorithm, the image is negated in order to make the numbers appear as light features on the dark background of the tape measure.

Before correlation for actual numbers can take place, an estimate of the position of each number is needed. The horizontal positions of the numbers is known to be exactly between the edges of the tape measure. The vertical period and phase of the numbers must still be determined.

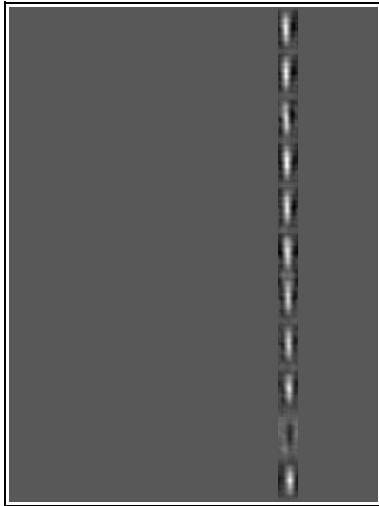


Figure 4: Segmented and blurred image of tape measure numbers.

A blurring operation is performed around the center of the tape measure, using a Fast Fourier Transform (FFT – the FFTW library in particular [7]). The size of the blurring kernel is chosen based on the size of the digit kernels. The resulting image is shown in Figure 4.

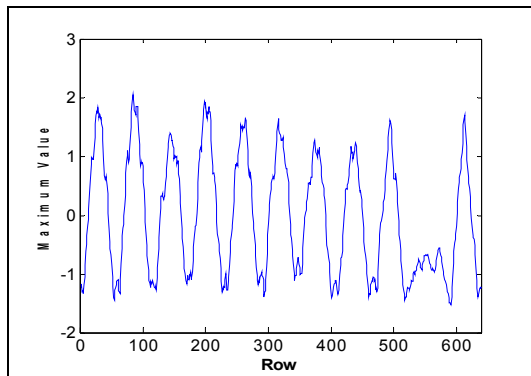


Figure 5: Graph showing plot of row maximums from blurred image. The data set has been normalized.

For each of the rows in the result, the maximum value is extracted. A plot of these values can be seen in Figure 5, clearly showing the periodicity of the numbers. This dataset is padded with zeros (to increase frequency resolution), and then an “FFT-

shift” operation is performed, swapping the first and second halves of the array. The FFT is then performed, the result of which can be seen in Figure 6.

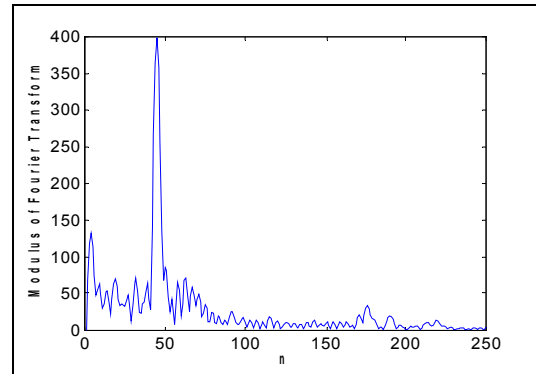


Figure 6: Graph showing zoom of spectrum of row maximums data set.

The FFT array is then searched for the maximum absolute value. This value is visible in the graph around element 50. The estimate for number periodicity is obtained by dividing the length of the array by the element index.

Phase information is then extracted by taking the arctangent of the argument of the maximum element.

The period and phase information gives a good estimate of the positions of the numbers on the tape measure. These positions are then enumerated. For example, the 0th position is the estimated position for the top number on the tape measure – this would be roughly around the number 22 in Figure 2. The exact positions of the numbers, as well as *which* numbers are visible, can now be determined.

4.4. Final Number Position Determination

The algorithm requires a set of consecutive numbers known to be visible in the given frame. This is not a problem. The algorithm is initialized with a constant set of numbers, and then a new set is generated for the next image. Known number values are handed off from frame to frame as each iteration of the algorithm takes place.

The set of known numbers generally needs to be fairly long – six or seven numbers was found to be robust.

Table 1 best demonstrates possible arrangements for such a set of numbers, by giving an example based on Figure 2. Assume that the known set consists of the numbers [16...21].

Sequence 2 is, of course, the correct one. The algorithm determines this by a series of normalized cross-correlations.

Normalized cross-correlation (mentioned in section 3.3) has two advantages over regular cross-correlation:

- It is independent of changing light intensity across an image, which is important in this case, since correlation results across the height of the image are being compared to one another [6].
- Every element of the output has a value of between -1 and 1, 1 being a perfect match. This uniformity makes comparisons between various correlations possible.

Est. Position	Actual #	Seq 1	Seq 2	Seq 3	etc
0	22	21	22	23	...
1	21	20	21	22	...
2	20	19	20	21	...
3	19	18	19	20	...
4	18	17	18	19	...
5	17	16	17	18	...
6	16	15	16	17	...
7	15	14	15	16	...
8	14	13	14	15	...
9	13	12	13	14	...
10	12	11	12	13	...

Table 1: Examples of possible arrangements for a set of consecutive numbers.

For each possible sequence, a kernel is created for each number (as explained in section 4.2), and a normalized cross-correlation is performed around the corresponding estimated position. The maximum correlation value returned and the coordinates at which it occurred are stored.

The size of the correlation region around the estimated position can be kept very small – the period, phase, and edge information provides a surprisingly good approximation.

The maximum correlation values for each of the numbers in the sequence are then added up, and the sequence value with the highest result is considered the “correct” sequence. From this sequence, the other numbers (the non-highlighted numbers in Table 1) are inferred, and then correlated for around their estimated positions, giving a full model of the tape measure in the image.

4.4.1. FFT Version of Correlation

A frequency-domain version of the correlation for the numbers was tested to determine if a speed improvement was possible.

A number of different methods were tested. The different results achieved are shown in Table 2. The method worked as follows: the region selected was normalized and then, if necessary, padded with zeros (it was always padded such that its dimensions were powers of two). The kernel was normalized, and then padded up to the same size. An FFT-based correlation was then performed in the usual way, using the FFTW library [7]:

$$R = F^{-1}[F(I) \cdot F(k)^*] \quad (5)$$

where R is the resultant image, I is the image array and k is the kernel array. F denotes the FFT operation.

Normalization can not happen with every iteration of the correlation process in a frequency based method. An estimate, based on the entire region of the image must be used, resulting in loss of accuracy in the value of the correlation result which is used for the sequence comparison. This results in incorrect sequences being chosen.

This was corrected by finding the coordinates of the number with the FFT version, and then performing a single normalized cross-correlation centered on that pixel for the comparison.

	Mean Time [ms]	Std Dev
spatial domain operation:	93.4	21.8
FFT, no padding, EST:	87.6	8.32
FFT, no padding, MEAS:	88.4	11.1
FFT, padding, EST:	130	21.1
FFT, padding, MEAS:	132	33.8

Table 2: Table showing times taken to perform correlation operations using different methods. Sample size is 300.

- “No padding” means that no zero padding was done: the FFT size was the same size as the region.
- “Padding” means the FFT dimensions were increased to the next highest power of two, giving better frequency resolution.
- “EST” refers to the fact that the FFTW_ESTIMATE option of the FFTW library was used for the transform.
- “MEAS” means that the FFTW_MEASURE option was used instead.

The options EST and MEAS involve the efficiency of the FFTW library architecture. MEAS executes several FFTs, measuring the time taken for each to determine the best way to perform the FFT. EST simply estimates the best way, usually sub-optimally [7]. As can be seen, EST performs marginally better in this case, most likely because any advantages of MEAS are offset by the small FFT size.

As can be seen, the spatial domain correlation performs better than the padded FFT version, but slower than the non-padded version.

It is somewhat surprising that the FFT correlations are not greatly faster than the spatial domain version. This is most likely because of the very small regions used.

The program allows the user to select which correlation method should be used in the post-processing algorithm.

5. Bubble Height Calculation

The bubble's height is calculated in two ways. The first involves straight linear inference. The two numbers closest to the bubble are found and the following equation is used:

$$h_b = \frac{(N_j - N_i) \cdot (Y_b - Y_i)}{(Y_j - Y_i)} + N_i \quad (6)$$

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; Y_i and Y_j are their respective heights in the frame (pixels); and Y_b is the bubble's height in the frame (pixels).

The second method involves a ray undistortion algorithm [8]. A checkerboard calibration pattern was placed in the column in front of the camera, and several images were taken. These were used to calibrate the camera with the Matlab Camera Calibration Toolbox [9].

Though the toolbox is designed only to compensate for optical distortion from a camera lens, the calibration method involved a general polynomial approximation to the distortion, which could be used to compensate for errors due to refraction

from the column as well. A function for making use of the parameters from the toolbox was created to determine if performance could be enhanced.

The method involved undistorting the coordinates of the bubble and two numbers before applying equation (6). A slight increase in performance was noted for certain runs (velocity readings had fractionally less spread), but by and large, the effect was negligible.

6. Results

Once the position data have been calculated, equation (1) is used to estimate velocity. A plot of the results from a run is given in Figure 7.

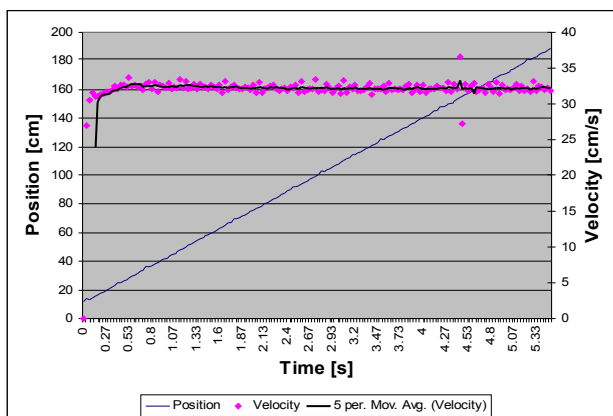


Figure 7: Graph showing position and velocity results from a tracking run. A trendline has been added.

This graph is consistent with data in research published by Zhang, Gomez and Finch [10].

A single spurious data point can be clearly seen – the result of an error in the bubble position determination, where some development still needs to be done. However, points such as this are easy to discern for the operator of the apparatus and can be removed or smoothed over with filtering operations, should the need arise.

Actual time taken to process a data set such as the one in Figure 7 is in the region of half an hour. Provision has been made in the program for several data sets to be processed, one after the other, so that the PC can be left overnight.

7. Conclusions

Conventional image processing techniques have successfully been used to provide a solution to a complex practical problem. While there are still minor modifications to be made, the algorithm provides a generally robust and accurate output from the set of images. The algorithm should prove useful in automating a mechanical task that would otherwise have cost a human time and energy to complete.

8. References

- [1] A. Sam, C. O. Gomez, J. A. Finch (1996), "Axial Velocity Profiles of Single Bubbles in Water/Frother Solutions." *Int. J. Miner. Process.*, no. 47, pp. 177-196.
- [2] Prof. J. S. Laskowski (2001), "Effect of Frothers on Bubble Coalescence, Bubble Rising Velocity and Induction Time in Bubble-Particle Attachment." Research Proposal. Department of Mining and Mineral Process Engineering, University of British Columbia.
- [3] S. Reid (2003), "Implementation of a Bubble Tracking Camera System." Undergraduate thesis. Department of Electrical Engineering, University of Cape Town.
- [4] MSDN BITMAPINFOHEADER Description [Online] (2006). Available: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdi/bitmaps_1rv2.asp. Microsoft Corporation.
- [5] PGM Format Specification [Online] (2003). Available: <http://netpbm.sourceforge.net/doc/pgm.html>. Jeff Poskanzer.
- [6] J. P. Lewis, "Fast Normalized Cross-Correlation" [Online]. Available: <http://www.idiom.com/~zilla/Work/nvisionInterface/nip.html>.
- [7] [Online]. Available: <http://www.fft.w.org>.
- [8] J. Hekkiliä, O. Silvén (2002), "A Four-step Camera Calibration Procedure with Implicit Image Correction." Infotech Oulu and Department of Electrical Engineering, University of Oulu.
- [9] J. Bouguet, "Camera Calibration Toolbox for Matlab" [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/
- [10] Y. Zhang, C. O. Gomez, J. A. Finch (1996), Terminal Velocity of Bubbles: Approach and Preliminary Investigations." *Proceedings of the International Symposium on Column Flotation*.