CLUSTER TRACKING ALGORITHMS FOR A
DIGITAL IMAGE-BASED ELASTO-TOMOGRAPHY SYSTEM

A thesis

submitted in partial fulfilment

of the requirements for the Degree

of

Master of Mechanical Engineering

in the

University of Canterbury

By

Anthony J. H. Hii

University of Canterbury

2005
Digital Image-based Elasto-tomography (DIET) is an emerging method for non-invasive breast cancer screening. Effective clinical application of the DIET system requires highly accurate motion tracking of the surface of an actuated breast with minimal computation. Normalized Cross-Correlation (NCC) is the most robust correlation measure for determining similarity between points in two or more images providing an accurate foundation for motion tracking. A motion tracking technique using Fast Fourier Transform (FFT) based cross-correlation (FFTC) is initially investigated to measure the motion of human skin, chicken skin and computer-generated fluid particle images. However, although motion was successfully tracked, FFTC is found to be too computationally intense for rapidly managing sequences of large images.

A significantly faster method of calculating the NCC is presented that uses rectangular approximations in place of randomly place landmark points or the natural marks on the breast. These approximations serve as an optimal set of basis functions that are automatically detected, dramatically reducing computational requirements. To prove the concept, the method is shown to be 37-150 times faster than the FFT-based NCC with the same accuracy for simulated data, a visco-elastic breast phantom experiment and human skin. Clinically, this approach enables thousands of randomly placed points to be rapidly and accurately tracked, providing high resolution for the DIET system.
ACKNOWLEDGEMENTS

Foremost, I would like to express my gratitude to my supervisor, Assoc. Prof. J. Geoffrey Chase for his guidance and enlightenment that has been invaluable to me, not just in overcoming many problems encountered during this project, but also in shaping up my research skills. I am grateful to my associate supervisor, Dr. Christopher E. Hann, for his invaluable suggestions and support, particularly with the development of the algorithms and the completion of my thesis.

Also, many thanks are due to:

   Dr. Eli Van Houten for his technical assistance during the commencement of the DIET project.
   Dr. Lawrence A. Ray for his enthusiasm and advice, particularly in thinking outside the square to get the algorithms to work more efficiently.
   My fellow postgraduate colleagues, particularly Andrew Comerford, Samara Aizaidi and Sanitta Thongpang, as knowing you all has made this postgraduate life more cherished and memorable.
   My mum and dad for their precious emotional and financial support.
   My brothers Michael and Vincent for their treasured help and constant care.
   Those whom I have not mentioned.

Great appreciation also goes to Prof. Tim David for providing the facilities, as the work for this thesis was carried out in the Centre for Bioengineering at the Canterbury University.
# Table of Contents

Abstract ........................................................................................................ i
Acknowledgments ......................................................................................... ii
List of Figures ............................................................................................... v
List of Tables ................................................................................................. viii

Chapter 1  INTRODUCTION

1.1 Motivation .............................................................................................. 1
1.2 DIET System Overview .................................................................... 5
1.3 Motion Sensing ................................................................................. 8
1.4 Other Tracking Methods ................................................................. 11
1.5 Summary ............................................................................................ 13

Chapter 2  CROSS-CORRELATION THEORY

2.1 One-Dimensional Cross-Correlation .............................................. 15
2.2 Fast Fourier Transformed Cross-Correlation .................................. 20
2.3 Two-Dimensional Cross-Correlation ............................................. 26
2.4 Standard NCC Method and Motion Tracking .................................. 28

Chapter 3  EFFICIENT 2-DIMENSIONAL CROSS-CORRELATION ALGORITHM

3.1 FFT and Sum-Table for Denominator of NCC .................................. 33
3.2 Numerator of NCC with Sum-Tables and Basis Functions .............. 39
3.3 Computational Efficiency Example .............................................. 43
3.4 Basis Functions .............................................................................. 46
3.5 Algorithm Summary ...................................................................... 48
Chapter 4  FFT BASED MOTION TRACKING
4.1 Tracking Procedure ................................................................. 51
4.2 Experimental Setup ............................................................... 56
4.3 Motion Vector Field for each Test Sample ............................... 58
4.4 Correlation Coefficient Threshold ............................................ 62
4.5 Accuracy and Computational Time for FFTC ......................... 66
4.6 Summary .............................................................................. 68

Chapter 5  MOTION TRACKING USING BASIS FUNCTIONS
5.1 Simulated Data ................................................................. 69
5.2 Visco-Elastic Breast Phantom Test Case .............................. 76
5.3 Human Skin ........................................................................ 80

Chapter 6
DISCUSSIONS AND CONCLUSIONS ........................................ 85

Chapter 7
FUTURE WORK ......................................................................... 89

REFERENCES ............................................................................ 91

Appendices A  MATLAB CODES
A1 Main Function for Fast-NCC Method .................................. A – 1
A2 Function for Generating the Basis Functions .................... A – 4
A3 Function for Calculating the NCC via Sum-Table ................ A – 5
A4 Function for Plotting the Motion Vector Field .................... A – 6
A5 Function for Comparing Circles and Basis Functions ........ A – 8
A6 Function for Producing Nth Densities of Circle ................ A – 9
A7 Function for Producing Nth Densities of Circle and Apply Motion to Each Circle

iv
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1</td>
<td>DIET system with its 4 major components.</td>
<td>7</td>
</tr>
<tr>
<td>Figure 1.2</td>
<td>Pictorial representation of DIET system.</td>
<td>8</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>Pictorial representation of evaluating cross-correlation.</td>
<td>18</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>Evaluating Equation (18) for $N = 3$.</td>
<td>23</td>
</tr>
<tr>
<td>Figure 2.3</td>
<td>Formation of a correlation plane by directly cross-correlating a $2 \times 2$ pixel template $T(x, y)$ with a $4 \times 4$ pixel search window $W(x, y)$, resulting in a $3 \times 3$ pixel correlation plane, where the brightest pixel, at the $(2, 2)$ location in this example, indicates $r_{max}$ corresponding to the best match.</td>
<td>30</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Implementation of the numerator of NCC by using FFT algorithm.</td>
<td>35</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>Pictorial representation of calculating $s(x, y)$.</td>
<td>39</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>Approximating a simple landmark point in a template by a rectangle.</td>
<td>41</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>Template with circles converted to basis functions.</td>
<td>47</td>
</tr>
<tr>
<td>Figure 3.5</td>
<td>Template with overlapped and partial circles converted to basis functions.</td>
<td>47</td>
</tr>
<tr>
<td>Figure 3.6</td>
<td>Motion tracking procedure.</td>
<td>49</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Locating the highest peak for a single ROI in (a) 3-dimensional and (b) 2-dimensional view of correlation plane.</td>
<td>53</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>An example of using the local median filter to remove a spurious vector and interpolate the filtered vector within a $3 \times 3$ kernel size.</td>
<td>55</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>Schematic diagram of the FFTC tracking process.</td>
<td>56</td>
</tr>
<tr>
<td>Figure 4.4</td>
<td>Experimental setup for capturing the images of the stretched human skin and actuated chicken skin.</td>
<td>57</td>
</tr>
</tbody>
</table>
Figure 4.5 Test images of (a) human skin, (b) actuated chicken skin, (c) computer generated fluid particle image and (d) distinctive skin pattern of a hand knuckle.
Figure 4.6 Motion fields plotted for human arm skin stretched in (a) left, (b) right, (c) up and (d) down direction.
Figure 4.7 Motion vector fields plotted before and after filtering and interpolation for (a1-a2) human skin on the arm, (b1-b2) actuated chicken skin and (c1-c2) fluid particle motion.
Figure 4.8 Window expansion for three iterations of template matching.
Figure 4.9 Poor motion field detected without the iterative method.
Figure 4.10 Motion vectors field produced by 0.45 correlation coefficient threshold.
Figure 4.11 Motion vector field from Figure 10 after filtering and interpolation.
Figure 4.12 Motion tracking accuracies for each test sample under 5 different window sizes.

Figure 5.1 Test images with circles' density of (a) 5, (b) 10, (c) 20 and (d) 30.
Figure 5.2 1M pixel image with randomly placed circles.
Figure 5.3 Non-uniform motion field example for 1M pixel image simulated motion, scaled for ease of viewing.
Figure 5.4 Two 1M pixel images of two different deformations of a visco-elastic breast phantom with randomly placed markings.
Figure 5.5 Template pattern from the first image of Figure 5.4 (a), thresholded (b), and converted into basis functions (c).
Figure 5.6 Template basis function pattern registered in the corresponding search window image.
Figure 5.7 Motion field representation of the first image in Figure 5.4.
Figure 5.8 One deformation of human skin.
Figure 5.9 230×230 template corresponding to Figure 5.8.
Figure 5.10 Thresholding template to obtain brighter pixels corresponding to speckles.
Figure 5.11 Representing natural patterns on the skin as rectangular basis functions.

Figure 5.12 Registration of speckles on Figure 5.8 to speckles on Figure 5.6.
List of Tables

Table 3.1  Number of computations required for calculating numerator of the NCC given by Equation (29). 46
Table 4.1  Detection performance of FFTC using the optimal window sizes. 67
Table 4.2  Detection performance of FFTC using the optimal window sizes in the iterative method. 67
Table 5.1  Performance time measured for template of size 150×150 inside a 200×200 search region. 71
Table 5.2  Performance time measured for template of size 180×180 inside a 200×200 search region. 72
Table 5.3  Performance time measured for template of size 200×200 inside a 250×250 search region. 72
Table 5.4  Performance time measured for template of size 230×230 inside a 250×250 search region. 72
Table 5.5  Performance time measured for template of size 150×150 inside a 200×200 search region. 73
Table 5.6  Performance time measured for template of size 180×180 inside a 200×200 search region. 73
Table 5.7  Performance time measured for template of size 200×200 inside a 250×250 search region. 73
Table 5.8  Performance time measured for template of size 230×230 inside a 250×250 search region. 74
INTRODUCTION
Chapter 1

Introduction

1.1 Motivation

Breast cancer is a significant health problem in both developed and developing countries. It is estimated that each year the disease is diagnosed in over one million women worldwide and is the cause of death in over 400,000 women [1]. In New Zealand, breast cancer accounts for the highest mortality rate of all cancers among women and it has the sixth highest death rate out of 173 developed countries. In general, it affects one in ten women in their lifetime and about 2300 women are diagnosed each year, where 600 women die from breast cancer each year in New Zealand alone [2], a value that is greater than the average annual road toll (500) over the past decade and 4 times more than the number that die from drink driving [3].

The risk of developing breast cancer increases with age, but the incidence is also currently increasing among younger women, as a result of the increasing size of the population in the younger generation worldwide. Other factors linked to a high risk of developing breast cancer include: ethnic group, not breastfeeding, late menopause and, most importantly, women with family histories of breast cancer. In some research, it was discovered that diet is another risk factor for the
development of breast cancer [4]. However, the direct causes and the reasons why one woman develops breast cancer and another does not, remain obscure.

There are many treatment options available, including surgery, chemotherapy, radiation therapy, and hormonal therapy. These treatments are significantly more effective in reducing the mortality of the disease with early detection through breast cancer screening programmes. Two of the most commonly used methods for early detection of breast cancer are mammography and manual palpation examinations performed by a healthcare professional or by self-exam.

For decades, medical specialists or physicians have used the palpation technique as part of the physical examination to identify any pathology in the human body. The presence of the stiffer or hardness of tissue associated with the cancerous pathology usually characterises an early warning sign of cancers, and breast cancer in particular. Thus, regular breast self-examination is considered an important preventive measure. However, the problem with physical examination is that a suspected cancerous lump in the breast has to be at least 1-2 cms to be located easily, and by this stage the cancerous cells may have spread beyond the breast.

Mammography can detect a much smaller cancer size of the order of 1-5 mm, and is thus a potentially more reliable method. During a mammogram, each breast of the patient is compressed between two clear plastic paddles attached to a specially designed X-Ray machine. The flattened breast tissue is imaged and examined later. However, reading and interpreting these images can lead to
variable results and inconsistencies. Hence, the average tumour size detected with mammography is approximately 1 cm.

A mammogram alone cannot completely confirm that an abnormal lump is cancerous. Hence, additional breast imaging such as ultrasound or biopsy is required. A breast biopsy is performed by removing a sample of tissue from the suspicious area for microscopic examination and is the most reliable option a pathologist has to accurately diagnose that the breast tumour is cancerous.

Some major drawbacks of mammography are that the procedure causes immense discomfort to a large majority of women patients due to pressure exerted on their breasts by plates. The radiation exposure during the mammogram is also a significant health concern. Finally, due to the specialized and subjective nature of reading a mammogram X-Ray to detect cancer, the final clinical test result can take from 24 hours to maybe a few days. For the patients, this period can be a very anxious moment, so it is desirable that such a lengthy period of waiting time is minimized, if possible.

Digital Image-based Elasto-Tomography (DIET) is an emerging technology for non-invasive breast cancer screening. The DIET system uses digital imaging of an actuated breast surface to determine tissue surface motion from a specified input. It then reconstructs the 3D internal tissue stiffness distribution from that motion. Regions of high stiffness suggest cancer since cancerous tissue is between 3 and 10 times stiffer than healthy tissue in the breast [5-7]. This approach eliminates the need for X-Rays and excessive, potentially painful
compression of the breast [8] as required in a mammogram. Hence, screening could start much younger and might enjoy greater compliance [9].

Presently, there are other Elasto-Tomographic methods based on magnetic resonance [10] and ultrasound [11] modalities. Both methods are capable of measuring the tissue elasticity and are undergoing rapid development across the globe. However, they are also costly in terms of equipment and take significant time to use. They are therefore limited for practical screening applications.

The DIET system, in contrast, is silicon based and is thus potentially low cost and portable, so the technology could be used in any medical centre, particularly in remote areas. In addition, the use of silicon technology ensures that as it improves and scales upward in capability so will the DIET system performance. This scalability of performance is not true for X-Ray or ultrasound based approaches.

The DIET system relies on a fast and accurate measurement of the actuated breast using multiple calibrated, high resolution digital cameras. This thesis deals with this digital imaging side of the DIET system. In particular, it focuses on developing a tissue motion tracking algorithm based on normalized cross-correlation (NCC) measures to determine similarity between image regions in consecutive time frames of an image sequence. This point registration and motion tracking represent one of the first critical steps in implementing any DIET type of system.
More specifically, a fast NCC motion tracking algorithm based on a sum-table scheme is implemented to rapidly measure the motion between pairs of images. The method is tested on simulated motion, as well as a visco-elastic phantom and human skin motion, to prove the concept. Substantial computational speed gains are achieved over the traditional NCC tracking method while still maintaining the same accuracy. These gains are important because any DIET implementation will require tracking large numbers (100 - 1000's) over several images to adequately capture the tissue surface motion in high accuracy and spatial resolution [8].

1.2 DIET System Overview

The objective of the DIET project is to design a low cost, portable, elastic property imaging technique for breast cancer screening. Using an array of high speed digital cameras in conjunction with image processing for motion sensing and finite element (FE) algorithms, DIET can reconstruct the internal elastic stiffness distribution of the breast using motion data measured at the breast surface. DIET is similar in concept to an X-Ray Computer Tomography (CT) scan. However, a CT scan requires the measurement of X-Ray particle wave patterns throughout the breast, while the DIET system only relies on the measurement of mechanical wave patterns at the surface. In addition, a CT scan requires no secondary FE, or other, analysis being based on radia-density detection, which has very low contrast for cancerous tissue [12, 13].
Overall, there are 4 major components associated with the DIET process:

1) An actuator is applied to part of the surface of the breast, inducing steady-state sinusoidal motion of the breast tissue.

2) High resolution spatially calibrated digital imaging cameras are placed in an array over the breast to capture a sequence of high quality two-dimensional images of clearly marked randomly placed reference points on the surface of the breast.

3) An image processing algorithm designed with motion sensing capability converts the two-dimensional image output data from the camera's into a three-dimensional motion vector for each reference point on the breast surface.

4) An inverse reconstruction algorithm based on FE calculations takes in the three-dimensional motion data for all reference points and generates an elastic modulus distribution within the three-dimensional breast volume.

The reconstructed stiffness identifies the carcinoma or malignant tumour together with its characteristics based on their high elastic property value, in contrast with the adjacent healthy tissues. Initial studies have proven the concept of mechanical property reconstruction based on surface displacement measurements [8]. The overall system process is shown schematically in Figure 1.1 and the pictorial representation of DIET breast screening is shown in Figure 1.2.
1. Motion Generate Actuator

Induces breast motion.

2. High Resolution Digital Camera

Captures the surface motion.

3. Digital Imaging with Motion Sensing Algorithm

Detects and measures the surface motion.

4. 3-D Map of Breast Tissue Stiffness

Reconstructs a 3-D map based on the motion data.

Breast Cancer Diagnosis

**Figure 1.1:** DIET system with its 4 major components.
1. A woman's breast is vibrated by an actuator and imaged with high-resolution digital cameras.

2. Spatially calibrated digital cameras combined with a motion sensor measures the surface motion of the breast.

3. Finite Element method converts the measured breast surface motion into a 3-D stiffness distribution, where regions of high stiffness suggest cancer.

Figure 1.2: Pictorial representation of DIET system.

1.3 Motion Sensing

A major component of the DIET breast cancer screening system is accurate measurement of breast surface motion due to harmonic excitation with high spatial resolution. The stiffness of the internal breast tissue can then be reconstructed from the measured 3D surface motion, similar to full volume elastographic methods using MRI [14, 15] or ultrasound [16]. To measure useful 3D surface motion requires accurate motion tracking of a large number of
randomly placed landmark points on the breast, or direct tracking of the natural marks and tone on the breast. In addition, clinical application for several hundred or thousand points per image, over several images and for several cameras, will require minimal computation to ensure clinical effectiveness. Hence, the DIET concept requires highly accurate motion tracking with minimal computation to be practicable.

The motion tracking system developed in this thesis enables the surface motion of the breast to be measured from calibrated cameras and involves analysing similarities or differences between consecutive frames in an image sequence. The standard way of tracking features between two images is by template matching that involves taking a given pattern in one image and shifting a template containing the same pattern in another image until the best comparison is found. Thus, points and clusters of points are tracked.

Currently, the most common and effective way of carrying out this task is by Normalized Cross-Correlation (NCC) methods. These methods have a significant advantage over standard cross-correlation (CC) methods, as they are robust to different lighting conditions across an image and less sensitive to noise [17]. However, both methods can be computationally intense, especially for large images. Therefore, this research focuses explicitly on methods and algorithms for dramatically reducing the computation required to calculate the NCC.

At present, a relatively efficient way of calculating the NCC is by using the Fast Fourier Transform (FFT) to compute the standard CC and then using a sum-
tables scheme to perform the normalization [18]. The corresponding method has already been applied for defect detection [19, 20] to greatly reduce computational requirements. However, there is no direct way of applying the sum-table formulation to increase the efficiency of the numerator of the NCC calculation, leaving significant computation involved [18]. However, it was shown in Briechle and Hanebeck [21] that if basis functions are used to approximate the matching template, then substantial computational gains could be obtained over the FFT-based methods.

The method of Briechle and Hanebeck [21] relies on choosing a suitable set of basis functions to approximate the template. However, the process of selecting or determining the best set of basis functions is non-trivial with no guarantee of finding the optimal solution in terms of the best approximation with the fewest basis functions. Additionally, a threshold value that describes how close the basis functions should approximate the template image has to be chosen and can significantly affect accuracy. In the case of tracking breast motion, there is no guarantee that this threshold value will stay constant, for a given patient or across many patients, as many complex surface motions can be produced if a tumour is present and there is significant inter-patient variability [8, 22].

The basis function generation concept introduced in this thesis lets every randomly placed landmark point on the breast correspond to a basis function. The problem is then reformulated in terms of tracking the landmarks. Thus, the basis function representation in this case is guaranteed to be the optimal and
there is physical control over the number and distribution of landmark points that are placed on the breast.

An automatic method of rapidly calculating the basis functions is also presented. The method is tested on simulated motion, as well as a visco-elastic phantom and human skin motion, to prove the concept. Note that unlike the method of Briechle and Hanebeck [21], the pre-calculation of the sum-tables are included in the time taken for computations presenting a more accurate comparison of computational costs. Overall, substantial computational speed gains are obtained over the NCC method of Lewis [18] while still maintaining the same high accuracy of their method.

1.4 Other Tracking Methods

There are other methods of tracking that do not use NCC, including Gradient Descent Search (GDS) and Active Contour Matching "Snakes". The GDS approach [23] is based on a first order approximation to image motion. It also has the significant restriction that feature translation is small and the inter-frame translation must be less than the radius of the basin surrounding the minimum of the matching error surface.

In the DIET system, there will be large numbers of landmark points to track, which are all close together [8]. Thus, any significant local perturbation on the surface due to a tumour [22] that causes a sudden local increase in amplitude
relative to other parts of the surface could cause an error in the motion measurement predicted by GDS, since GDS only uses a first order approximation to motion. Furthermore, GDS methods only require one occasion where a particular landmark point jumps to another landmark point nearby for the entire trajectory to be corrupted. Another drawback is that the image gradient is required, which like any numerical derivative is sensitive to noise. Thus, to compute a reliable estimate of the gradient, the image must be smoothed, which depending on the number of smoothings applied, can distort the image, introducing further potential error. Hence, GDS methods are not particularly well suited to the DIET application, although they may be more elegant than NCC-based methods.

The snake method [24] tracks individual feature contours. Similar to GDS, it is restricted to small changes in the contour’s shape and displacement. Furthermore, the snake method is sensitive to any intrinsic fuzziness or varying lighting conditions, as the image features must have clearly defined boundaries. In the DIET system, landmark points will be quite close together so any variation in lighting conditions could cause a contour to be placed around one point in one image and two points in another image, thus corrupting results.

An improvement to the robustness of snake tracking has been proposed [25], yet the method only tracks macro scale single contours, such as the shape of a hand. The DIET system would need thousands of contours to be tracked, which would require significantly large numbers of computations. Another feature based
method has been looked at involving thin plate splines [26], however the method is also very computationally heavy and thus not suitable for a DIET system.

Finally, an empirical study of five template matching algorithms in the presence of various image distortions [27] found that NCC provides the best performance in all image categories. Also the choice of correlation coefficient over alternative matching criteria, such as the sum of absolute differences [28], has also been justified as a maximum-likelihood estimation [29]. Thus, the NCC approach is potentially most suitable for the DIET system, which requires highly robust and accurate tissue surface motion tracking for a large number of closely located points. Hence, fast evaluation of the NCC so it is more practicable for a DIET implementation is the subject of this thesis.

1.5 Summary

The objective of this thesis is the development of efficient and robust approaches to motion tracking to provide accurate and high-resolution 3D surface motion. The goal is for the motion result to be input to a FE reconstruction algorithm to locate the areas of high stiffness that indicate potential cancerous lesions. Since NCC has been shown to be the most reliable method for such applications, it is the focus of this research.

Chapter 2 describes the traditional NCC tracking process, including FFT algorithms and the standard sum-table approach to compute the denominator of
the NCC. The basis function method based on the automatic detection and approximation of landmark points is then presented. Chapter 3 provides the results of applying the standard FFT-based NCC method on various examples of skin motion, describing the problems encountered and computational issues. The fast basis function method for calculating NCC is then applied on simulated motion, an actuated visco-elastic breast phantom, and human skin motion in Chapter 4. Chapter 4 also includes a detailed comparison to the standard NCC method in terms of computational time and accuracy. Conclusions and future work are discussed in Chapter 5.
METHODOLOGY
Chapter 2

Cross-Correlation Theory

2.1 One-Dimensional Cross-Correlation

Cross-correlation (CC) in 1-dimensional is a measure of similarity between two data sequences. In digital signal processing, CC is a standard mathematical approach that is applied to measure the similarities or shared properties between two signals [30]. More specifically, it is commonly used to find the features in an unknown signal by comparing it to a known signal, to detect a known signal in a noisy signal, or to search for cyclic data.

The motivation for CC was originally based on the covariance measure, which quantifies the degree to which two variables vary together. The use of CC for data analysis or as a measure of the degree of linear relationship between two variables is based on the covariance theorem, which is commonly used in statistics [31, 32]. The unnormalized or traditional CC coefficient [32, 33] corresponds to the sample covariance $\text{Cov}_{XY}$ of two random sequences $X$ and $Y$ with means $\mu_x$ and $\mu_y$ respectively and is defined:

$$\rho_{XY} = \text{Cov}_{XY} = \sum_{i=1}^{N} (X_i - \mu_x)(Y_i - \mu_y)$$  \hspace{1cm} (1)
where \( N \) is the length of the sequence and Equation (1) relates to how two variables \( X \) and \( Y \) vary about their means together [33]. The covariance has the symmetric property \( \text{Cov}_{xy} = \text{Cov}_{yx} \) and the magnitude of the covariance depends on the components of \( X \) and \( Y \).

To illustrate the evaluation of the traditional CC of Equation (1) to determine the similarity of two sequences \( X = [X_1, X_2, \ldots, X_N] \) and \( Y = [Y_1, Y_2, \ldots, Y_N] \), consider two random vectors \( X = [1, 4, 6, 9] \) and \( Y = [6, 1, 2, 7] \) with \( N = 4 \), consisting of numbers from 1 to 9. From Equation (1),

\[
\begin{align*}
\mu_X &= \frac{1+4+6+9}{4} = 5, \quad \mu_Y = \frac{6+1+2+7}{4} = 4 \\
\rho_{xy} &= (1-5)(6-4) + (4-5)(1-4) + (6-5)(2-4) + (9-5)(7-4) \\
&= -8 + 3 - 2 + 12 \\
&= 5
\end{align*}
\]

Also,

\[
\begin{align*}
\rho_{xx} &= (1-5)^2 + (4-5)^2 + (6-5)^2 + (9-5)^2 = 34 \\
\rho_{yy} &= (6-4)^2 + (1-4)^2 + (2-4)^2 + (7-4)^2 = 26
\end{align*}
\]

Comparing Equations (3) and (4) with Equation (2) show that the two uncorrelated vectors give a significantly smaller correlation coefficient than the two examples of exactly correlating vectors or auto-correlation \( \rho_{xx} \) and \( \rho_{yy} \).

This result is evident from the fact that for two uncorrelated vectors, the components of \( \rho_{xy} \) tend to cancel out as shown in Equation (2), where for the
two correlated vectors of Equations (3) and (4), there is no cancelling. Thus, high values of $\rho_{xy}$ suggest a high correlation between the vectors $X$ and $Y$.

In signal processing, the 1-dimensional CC given by Equation (1) can be used to find one signal inside another. For example, given a vector $X$ of length $N$, a vector $Y$ of length $M < N$ can be located in $X$ by shifting the $Y$ vector across all possible positions in $X$. The position that has the largest CC corresponds to the best match of $Y$ to $X$. Mathematically, this concept can be formulated as:

$$\rho_{xy}(u) = \sum_{i=1}^{N-M+u+1} (X_i - \bar{X}_u)(Y_{i-u} - \bar{Y})$$  \hspace{1cm} (5)$$

where

$$\bar{X}_u = \frac{1}{M} \sum_{i=u+1}^{N-M+u+1} X_i$$ \hspace{1cm} (6)$$

$$\bar{Y} = \frac{1}{M} \sum_{i=1}^{M} Y_i$$ \hspace{1cm} (7)$$

and $u = 0, \ldots, M$ is the amount that $Y$ is shifted across $X$.

Writing $Y'_{i-u} = Y_{i-u} - \bar{Y}$; $\sum_{i=u+1}^{N-M+u+1} Y'_{i-u} = 0$ since $Y'_{i-u}$ has zero mean, thus Equation (5) can be simplified:
Cross-Correlation Theory

\[
\rho_{XY}(u) = \sum_{i=1}^{N-M+u+1} X_i Y'_{i-u} - \bar{X} \sum_{i=1}^{N-M+u+1} Y'_{i-u}
\]

\[
= \sum_{i=1}^{N-M+u+1} X_i Y'_{i-u}
\]  

(8)

For example, consider two vectors \(X = [X_1, X_2, X_3, X_4, X_5]\) and \(Y = [Y_1, Y_2, Y_3]\) where \(Y\) lies somewhere inside \(X\). In this case, the value of \(u = 0, 1, 2\) that gives the largest value of \(\rho_{XY}(u)\) corresponds to the shift that best matches \(Y\) onto \(X\). The process of evaluating each \(\rho_{XY}(u)\) for \(u = 0, 1, 2\) is shown pictorially in Figure 2.1, where vector \(Y\) is transformed:

\[
Y' = [Y'_1, Y'_2, Y'_3]
= [Y_1 - \bar{Y}, Y_2 - \bar{Y}, Y_3 - \bar{Y}]
\]  

(9)

\[
\rho_{XY}(0) = X_1 Y'_1 + X_2 Y'_2 + X_3 Y'_3
\]

\[
\rho_{XY}(1) = X_2 Y'_1 + X_3 Y'_2 + X_4 Y'_3
\]

\[
\rho_{XY}(3) = X_3 Y'_1 + X_4 Y'_2 + X_5 Y'_3
\]

Figure 2.1: Pictorial representation of evaluating cross-correlation.
However, one problem with CC is how large should $\rho_{XY}$ be before two vectors are considered to be correlated? For example, if the two sequences represent the intensities on an image, then it would be difficult to place a threshold on $\rho_{XY}$ that determines when two sequences correspond to similar features between images, as $\rho_{XY}$ could change with different lighting conditions across an image sequence. Additionally, $\rho_{XY}$ would vary in size depending on the length of the signals and length of sequences being searched for.

This problem can be resolved by using the normalized CC formula defined as:

$$\rho_{XY} = \frac{\text{Cov}_{XY}}{\sigma_X \sigma_Y}$$  \hspace{1cm} (10)

where $\sigma_X \neq 0$ and $\sigma_Y \neq 0$ correspond to the standard deviation of variables $X$ and $Y$ respectively. For two sequences $X$ and $Y$ of length $N$, Equation (10) is written as:

$$\rho_{XY} = \frac{\sum_{i=1}^{N} (X_i - \mu_X)(Y_i - \mu_Y)}{\sqrt{\sum_{i=1}^{N} (X_i - \mu_X)^2 \cdot \sum_{i=1}^{N} (Y_i - \mu_Y)^2}}$$ \hspace{1cm} (11)

One of the primary properties of the normalized CC is that the coefficients are bounded between -1 and 1, as opposed to the covariance coefficient of Equation (1), which has no upper or lower limits. The positive coefficients indicate $X$ and
$Y$ correlate directly and negative coefficients indicate inverse correlation. This definition thus allows easy comparison of results.

Generally, the range $0 \leq |\rho_{xy}| \leq 0.3$ means weak correlation, $0.3 \leq |\rho_{xy}| \leq 0.7$ indicates moderate correlation and $0.7 \leq |\rho_{xy}| \leq 1$ means high correlation between vectors $X$ and $Y$ [34-36]. For independent variables $X$ and $Y$, $\rho_{xy}$ is zero but for linear dependence $Y = aX + b$, $\rho_{xy}$ becomes:

$$\rho_{xy} = \begin{cases} 1, & a > 0 \\ -1, & a < 0 \end{cases}$$  \hspace{1cm} (12)

The normalized CC also satisfies the symmetry property of covariance, $\rho_{xy} = \rho_{yx}$, but most importantly it is invariant to scale and addition. That is, for $\bar{X} = a_1X + b_1$ and $\bar{Y} = a_2\bar{Y} + b_2$:

$$\rho_{\bar{X}\bar{Y}} = \rho_{xy}$$  \hspace{1cm} (13)

### 2.2 Fast Fourier Transformed Cross-Correlation

The direct evaluation of Equation (8) for many shifts $u$ and large vectors $X$ and $Y$ can be very computationally expensive. A common technique used in image and signal processing is to compute the CC of Equation (8) using the discrete
Fast Fourier Transform (FFT). This dramatically reduces the number of computations required to calculate the CC and thus also the normalized CC of Equation (10). The FFT is the discrete analogy of the continuous Fourier transform.

For a given real valued function $x(t)$ the continuous Fourier transform of $x(t)$ is defined:

$$\mathcal{F}[x(t)] = \int_{-\infty}^{\infty} x(t) e^{-j2\pi f t} \, dt = X(f)$$  \hspace{1cm} (14)

To formulate the CC in terms of the discrete FFT, Equation (8) is first written in the form:

$$\rho_{xy}(u) = \sum_{n=0}^{N-1} x(n) h(u+n) \quad 0 \leq u \leq N - M$$  \hspace{1cm} (15)

where

$$x(n) = X_{n+1}, \quad 0 \leq n \leq N - 1$$  \hspace{1cm} (16)

$$h(n) = 0, \quad n < 0$$

$$= Y_{n+1}, \quad 0 \leq n \leq M - 1$$

$$= 0, \quad n > M - 1$$  \hspace{1cm} (17)
Equations (15) and (8) are completely equivalent. However, Equation (15) is not yet in a suitable form to utilize the discrete FFT. Thus, Equation (15) is extended to a more general form by defining:

\[ \hat{r}_{xy}(u) = \sum_{n=0}^{N-1} x(n) h(n + u), \quad -(N-1) \leq u \leq (N-1) \]  

(18)

where \( x(n) = x_{n+1}, \quad 0 \leq n \leq N - 1 \) and \( h(n) \) is defined:

\[
\begin{align*}
  h(n) &= 0, \quad n < 0 \\
  &= h_{n+1}, \quad 0 \leq n \leq N - 1 \\
  &= 0, \quad n > N - 1
\end{align*}
\]  

(19)

Equation (18) defines a new set of cross-correlation coefficients of two vectors \( x = [x_1, \ldots, x_N] \) and \( h = [h_1, \ldots, h_N] \) of the same length, which includes a new class of shifts of \( h \) across \( x \). The evaluation of Equation (18) for the case \( N = 3 \), is demonstrated pictorially in Figure 2.2.
Equation (18) is the discrete analogy of the continuous correlation $x \otimes h$ of two real valued functions $x(t)$ and $h(t)$ defined:

$$x \otimes h = \int_{-\infty}^{\infty} x(\tau) h(t + \tau) \, d\tau$$

$$= \int_{-\infty}^{\infty} x(-\tau) h(t - \tau) \, d\tau$$

$$= x(-t) * h(t)$$

(20)

where the * denotes the convolution operation [37]. In the frequency domain, Equation (20) becomes:
\[ \Im \{ x \otimes h \} = \Im \left[ \int_{-\infty}^{\infty} x(\tau) h(t+\tau) \, d\tau \right] \]
\[ = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x(\tau) h(t+\tau) \, d\tau \, e^{-j2\pi f t} \, dt \]
\[ = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x(\tau) h(t+\tau) \, e^{-j2\pi f t} \, d\tau \, dt \]
\[ = \int_{-\infty}^{\infty} \left[ \int_{-\infty}^{\infty} x(\tau) \, e^{j2\pi f t} \, d\tau \right] h(t) \, dt \]
\[ = \int_{-\infty}^{\infty} x(\tau) \, e^{j2\pi f t} \, \left[ \int_{-\infty}^{\infty} h(t) \, e^{-j2\pi f t} \, dt \right] d\tau \]
\[ = \left[ \int_{-\infty}^{\infty} x(-t) \, e^{-j2\pi f t} \, dt \right] H(f) \]
\[ = \Im \left[ x(-t) \right] \cdot H(f) \]
\[ = X^*(f) \cdot H(f) \]

where \( X^* \) is the complex conjugate of the FFT of \( x(t) \) and is defined:

\[ X^*(f) = \left( \int_{-\infty}^{\infty} x(t) \, e^{-j2\pi ft} \, dt \right)^* \]
\[ = \int_{-\infty}^{\infty} x(t) \, e^{j2\pi ft} \, dt \]
\[ = \Im \left[ x(-t) \right] \]

Thus, the convolution in the time domain given in Equation (20) is equivalent to multiplication in the frequency domain, as given in Equation (21).
Equation (21) is known as the correlation theorem and for two $N$-length sequences $x = [x_1, x_2, ..., x_N]$ and $h = [h_1, h_2, ..., h_N]$, it has the discrete analogy [37]:

$$\sum_{n=0}^{N-1} x(n) h(k+n) \Leftrightarrow X^*(n) \cdot H(n)$$

(23)

where $X^*(n)$ is the complex conjugate of the discrete FFT of the sequence $x(n)$ and $H(n)$ represents the discrete FFT of the sequence $h(n)$.

The evaluation of Equation (18) for two random sequences $x = [x_1, x_2, ..., x_N]$ and $h = [h_1, h_2, ..., h_N]$ using the FFT can be summarized in 4 steps:

- **Step 1**: Compute the FFT of $x(n)$ and $h(n)$ to produce $X(f)$ and $H(f)$ respectively.
- **Step 2**: Take the conjugate of $X(f)$ to form $X^*(f)$.
- **Step 3**: Multiply $X^*(f)$ and $H(f)$ together.
- **Step 4**: Inverse Fourier transform the product of $X^*(f)$ and $H(f)$ back to produce the desired CC coefficients given by Equation (18).

Note that for two random sequences $x = [x_1, x_2, ..., x_N]$ and $h = [h_1, h_2, ..., h_M]$, where $M < N$, the $h$ is padded with zeros in order to form the same length as $x$. Steps 1 to 4 are then applied to evaluate Equation (18) using FFT.
2.3 Two-Dimensional Cross-Correlation

The concepts in 1-dimensional cross-correlation can be readily extended to two dimensions. The traditional 2-dimensional CC is very similar to Equation (1) but involves comparing matrices \( X \) and \( Y \) rather than vectors. For an \( n \times n \) matrix \( X \) and \( n \times n \) matrix \( Y \), the 2-dimensional analogy of Equation (1) is defined:

\[
\rho_{XY} = \sum_{i=1}^{n} \sum_{j=1}^{n} (X_{i,j} - \bar{X})(Y_{i,j} - \bar{Y})
\]

(24)

where \( \bar{X} \) and \( \bar{Y} \) are the mean values of the \( X \) and \( Y \) matrices respectively.

Similar to Equation (1), for random uncorrelated matrices \( X \) and \( Y \), the terms in Equation (24) tend to cancel out giving significantly lower numbers than for highly correlated matrices \( X \) and \( Y \). This property can be used for template matching in motion tracking as a feature \( X \) in one image corresponding to an \( n \times n \) matrix of intensities can be compared with all possible \( n \times n \) matrices \( Y \) in a second image using \( \rho_{XY} \). The largest value of \( \rho_{XY} \) corresponds to the best match of feature \( X \) to the second image, and the translation required to match \( Y \) onto \( X \) gives the amount of pixel motion between the images.

For a given template function \( t(x, y) \) of size \( N_x \times N_y \) to be found in a larger sized image function \( f(x, y) \), the template matching process can be expressed mathematically by the formula:
where $u$ and $v$ correspond to the amount of pixel shifts in the $x$ and $y$ directions relating each comparison of $t(x, y)$ with the corresponding $N_x \times N_y$ image in $f(x, y)$. Figure 2.3 shows a pictorial representation of this process.

Note that an alternative motivation of the cross-correlation formula of Equation (25), as suggested by Lewis [18], is based on the squared Euclidean distance measure:

$$d^2_{f,t}(u, v) = \sum_{x=u}^{u+N_x-1} \sum_{y=v}^{v+N_y-1} \left[ f(x, y) - t(x-u, y-v) \right]^2$$  \hspace{1cm} (26)

Expanding Equation (26) results in:

$$d^2_{f,t}(u, v) = \sum_{x=u}^{u+N_x-1} \sum_{y=v}^{v+N_y-1} \left[ f^2(x, y) - 2f(x, y) t(x-u, y-v) + t^2(x-u, y-v) \right]$$  \hspace{1cm} (27)

where the expression $\sum_{x,y} t^2(x-u, y-v)$ is constant, since the template function $t(x, y)$ is constant and known. Assuming the term $\sum_{x,y} f^2(x, y)$ is nearly constant, the best match of $t$ onto $f$ occurs at the minimum value of $d^2_{f,t}(u, v)$, which corresponds to maximizing $\rho(u, v)$ of Equation (25).
However, there are several problems with using Equation (25) for template matching:

- Equation (25) will produce inaccurate CC coefficients if the lighting conditions vary non-homogeneously across the images.
- $\rho(u, v)$ is unbounded and the range of coefficients changes depending on the size of the feature template $t$. This makes it difficult to choose a reliable threshold to determine when two images are sufficiently correlated so that they can be said to contain the same image features.
- $\rho(u, v)$ assumes high correlation for only linear relationships between images. Thus, it can be sensitive to noise.

### 2.4 Standard NCC Method and Motion Tracking

To overcome the difficulties associated with the standard CC template matching approach for given $n \times n$ matrices $X$ and $Y$, the cross-correlation coefficient of Equation (25) is normalized by dividing by the 2-dimensional standard deviations $\sigma_X$ and $\sigma_Y$, analogous to Equations (11):

$$\rho_{xy} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} (X_{i,j} - \mu_X)(Y_{i,j} - \mu_Y)}{\sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} (X_{i,j} - \mu_X)^2 \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} (Y_{i,j} - \mu_Y)^2}}$$

(28)
where $\mu_X$ and $\mu_Y$ are the mean's of the entries in the $X$ and $Y$ matrices respectively. Thus, Equation (24) can be normalized to unit magnitude producing the 2-dimensional NCC definition in Equation (28).

The template matching method of the CC can then be applied in the same way for the NCC by determining the location of a desired pattern represented by a template function, $t$, inside a 2-dimensional image $f$. The template is shifted pixel-by-pixel across the image, forming what is now a normalized correlation plane consisting of NCC coefficients in the range between -1 and 1. The correlation plane provides information of where the template best matches the image by finding the entry which is the closest to 1.

Therefore, in correlation based motion tracking applications, a pair of time-dependent images $I_{m_{t}}$ and $I_{m_{t+\Delta t}}$ are compared in a pixel-by-pixel basis. For example, consider a $4 \times 4$ sub-image of $I_{m_{t+\Delta t}}$ denoted by $W(x, y)$ and a $2 \times 2$ feature template $T(x, y)$ of $I_{m_{t}}$ which is contained in $W(x, y)$. The template $T(x, y)$ is shifted into nine different positions, where at each position, intensities are multiplied and summed, producing a correlation coefficient matrix, $\gamma_{u,v}$, as shown in Figure 2.3. The pixel location $(u, v)$ corresponding to the maximum NCC $\gamma_{\text{max}}$ value corresponds to the best location of the template feature in the sub-image $W(x, y)$. This process is continued over all sub-images $W(x, y)$ contained in $I_{m_{t+\Delta t}}$ until $I_{m_{t}}$ is correlated to $I_{m_{t}}$ and motion tracking of features in the template is achieved [38].
Figure 2.3: Formation of a correlation plane by directly cross-correlating a $2 \times 2$ pixel template $T(x, y)$ with a $4 \times 4$ pixel search window $W(x, y)$, resulting in a $3 \times 3$ pixel correlation plane, where the brightest pixel, at the $(2, 2)$ location in this example, indicates $\gamma_{\text{max}}$ corresponding to the best match.

Let $f(x, y)$ be the intensity value of the $M_x \times M_y$ image $f$ at pixel $(x, y)$, $x \in \{0, \ldots, M_x - 1\}$, $y \in \{0, \ldots, M_y - 1\}$. Similarly, let $t(x, y)$ be the intensity value of the $N_x \times N_y$ template $t$ at pixel $(x, y)$, where $N_x \leq M_x$ and $N_y \leq M_y$. NCC is evaluated at every point $(u, v)$ for $f$ and $t$, which has been shifted over the original image $f(x, y)$ by $u$-steps in the $x$-direction and $v$-steps in the $y$-direction. All the NCC coefficients are stored in a correlation matrix $\gamma_{u,v}$ defined:
\[ \gamma_{u,v} = \frac{\sum_{x,y} (f(x,y) - \bar{f}_{u,v})(t(x-u,y-v) - \bar{t})}{\sqrt{\sum_{x,y} (f(x,y) - \bar{f}_{u,v})^2 \sum_{x,y} (t(x-u,y-v) - \bar{t})^2}} \]  

where \( u \in \{0,1,2,\ldots,M_x-N_x\} \) and \( v \in \{0,1,2,\ldots,M_y-N_y\} \), and \( \bar{f}_{u,v} \) denotes the mean value of \( f(x,y) \) within the area of the template \( t \) shifted by \( (u,v) \) steps and defined:

\[ \bar{f}_{u,v} = \frac{1}{N_x N_y} \sum_{x=u}^{u+N_x-1} \sum_{y=v}^{v+N_y-1} f(x,y) \]  

Finally, \( \bar{t} \) denotes the mean value of the template \( t \) defined in a similar way.

Direct computation of Equation (29) involves the order of \( N_x N_y (M_x-N_x)(M_y-N_y) \) calculations, which is very computationally expensive. It is thus not suitable for motion tracking of large numbers of points or features in each image. For example, to match a 200\times200 pixel template with a relatively small 250\times250 pixel image would require a total of approximately \( 10^8 \) calculations. This value would thus be much larger and intractable for realistic images of 4M pixels and greater with template of any size.
Chapter 3

Efficient 2-Dimensional Cross-Correlation Algorithm

The NCC of Equation (29) is very computationally expensive to directly evaluate. In this chapter, methods of dramatically reducing the computational time required to compute the NCC are developed. The numerator of the NCC is first formulated in terms of the FFT, and then the denominator is formulated in terms of sum-tables, which are pre-calculated look-up tables that can significantly reduce the computations. However, the FFT calculation is still computationally intense and there is no direct way of formulating the numerator of the NCC in terms of sum-tables to simplify the calculations as can be done for the denominator of the NCC. The rest of this chapter looks at approximating the feature template with basis functions, which allows sum-tables to be used and provides a dramatic improvement on the FFT-based NCC.

3.1 FFT and Sum-Table for Denominator of NCC

A significantly more efficient way of calculating the NCC is by computing the numerator of Equation (29) via FFT. More specifically, cross-correlation in the spatial domain as in Equation (29) is equivalent to multiplication in the frequency-domain:
\[ r(u, v) = \sum_{x,y} f(x, y) \cdot t(x-u, y-v) \]

\[ \Rightarrow R(u, v) = F^*(u, v) \cdot T(u, v) \]

\[ \Rightarrow r(u, v) = S^{-1}(R(u, v)) \]

Equation (31) corresponds to computing a 2D FFT on the template, \( t \), and the region of interest (ROI) window, \( f \), of the images followed by a complex-conjugate multiplication of the resulting Fourier coefficients. The final products are then inverse Fourier transformed to produce the actual coefficient cross-correlation plane, as demonstrated in Figure 3.1.

The process defined by Equation (31) is completely analogous to the FFT formulation of the 1-dimensional correlation given by Equation (23). That is the NCC of Equation (29) is generalized to include a new class of shifts \( u \) and \( v \), and \( t \) is padded with zeros to make it the same size as \( f \). The resulting formulation is analogous to Equation (18) and corresponds to shifting the zero padded template \( t \) across \( f \), and cutting off any non-overlapping parts in a similar way to Figure 2.2.

Effectively, this approach is equivalent to correlating smaller and smaller sub-template windows in the zero-padded template until eventually two single pixels are correlated. Note that, as was the case for the 1-dimensional CC, this process of including a new class of shifts in Equation (29) must be done so that the analogous 2-dimensional correlation theorem to Equation (23) can be applied. The details of this 2-dimensional construction are very similar to the 1-dimensional construction of Section 2.2, so the details are not included. Awcock and Thomas [39] have presented the 2-dimensional formulation in detail.
The use of the FFT to simplify the numerator calculations in Equation (29) reduces the number of NCC calculations to the order of $M_x M_y \log_2(M_x M_y)$ [40]. Thus, for the $200 \times 200$ template and $250 \times 250$ image example at the end of Section 2.4, the calculations are now of the order $6 \times 10^5$. However the denominator of the NCC in Equation (29) does not have a correspondingly efficient frequency domain expression [18].

A further significant reduction to the number of computations required to compute the NCC in Equation (29) can be made using the idea of a sum-table to simplify computation of the denominator in Equation (29). The sum-table is a pre-computed data structure that acts as a lookup table, dramatically reducing the number of multiplications and additions required to evaluate a given expression. More specifically, the sum-table is a discrete version of an integrated or integral image [41, 42].

Let $f(x, y)$ be an integrable 2-dimensional function with non-negative $x, y$. The integral image of $f$ is defined:
\[ I(u, v) = \int_{x=0}^{u} \int_{y=0}^{v} f(x, y)dx \]  
\hspace{1cm} (32)  

where \( u \) and \( v \) are defined over the domain of \( f \) and \( I(u, v) \) is an integral transformation of \( f(x, y) \) into the \( u, v \) domain.

Equation (32) can be used to compute an explicit integral:

\[ \int_{x=0}^{y_b} \int_{y=0}^{x_b} f(x, y)dydx = I(x_b, y_b) + I(x_a, y_a) - I(x_a, y_b) - I(x_b, y_a) \]  
\hspace{1cm} (33)  

The discrete versions of Equation (32) and (33) that would be applied to the discrete pixels of a given image are defined:

\[ I(x, y) = \sum_{x=0}^{X_b} \sum_{y=0}^{Y_b} f(x', y') \]  
\hspace{1cm} (34)  

\[ \sum_{x=x_a}^{x_b} \sum_{y=y_a}^{y_b} f(x, y) = I(x_b, y_b) + I(x_a - 1, y_a) - I(x_a - 1, y_b) - I(x_b, y_a - 1) \]  
\hspace{1cm} (35)  

where Equation (34) corresponds to a rectangular numerical approximation to the analytical integral of Equation (32). For a digital image, where \( f(x, y) \) is a pixel intensity, at the \( (x, y) \) pixel, Equation (34) represents a summed, or volume of, pixel intensity over that portion of the image.
Using the sum-table notation \( s(x, y) = I(x, y) \) as defined in Equation (34), the double sums \( \sum_{x,y} f(x, y) \) and \( \sum_{x,y} f^2(x, y) \) in the denominator of Equation (29) can be rewritten in a computationally efficient form:

\[
\sum_{x=u}^{u+N_x-1} \sum_{y=v}^{v+N_y-1} f(x, y) = s(u + N_x - 1, v + N_y - 1) + s(u - 1, v - 1) - s(u - 1, v + N_y - 1) - s(u + N_x - 1, v - 1)
\]

\[\text{(36)}\]

\[
\sum_{x=u}^{u+N_x-1} \sum_{y=v}^{v+N_y-1} f^2(x, y) = s^2(u + N_x - 1, v + N_y - 1) + s^2(u - 1, v - 1) - s^2(u - 1, v + N_y - 1) - s^2(u + N_x - 1, v - 1)
\]

\[\text{(37)}\]

where

\[
s(u, v) = f(u, v) + s(u - 1, v) + s(u, v - 1) - s(u - 1, v - 1)
\]

\[\text{(38)}\]

\[
s^2(u, v) = f^2(u, v) + s^2(u - 1, v) + s^2(u, v - 1) - s^2(u - 1, v - 1)
\]

\[\text{(39)}\]

Note that \( s(u, v) = s^2(u, v) = 0 \) if \( u, v = 0 \). The double sums in the left-hand side of Equations (36) and (37) are evaluated over the region of template \( t \) bounded by \( u < x < u + N_x - 1 \) and \( v < y < v + N_y - 1 \). Equation (38) is a reformulation of Equation (34) in terms of recursive relations enabling rapid calculation of the sum-table in one global sweep over the image, and similarly so for Equation (39) [18].

Once the sum-tables \( s(u, v) \) and \( s^2(u, v) \) are calculated over the whole image, they act as a pre-computed look-up table for Equations (36) and (37),
significantly reducing the number of computations required to calculate the NCC at each pixel shift. This approach therefore eliminates repeated multiplication and addition over the images as the template is moved.

To demonstrate the equivalence of Equation (38) with Equation (34), consider a $2 \times 2$ image $f$ and corresponding sum-table matrix $S$ defined:

$$
\begin{bmatrix}
  a & b \\
  c & d
\end{bmatrix}
$$

$$
\begin{bmatrix}
  s(0,0) & s(0,1) \\
  s(1,0) & s(1,1)
\end{bmatrix}
$$

Applying the recursive relation Equation (38) gives:

$$
\begin{align*}
  s(0,0) &= f(0,0) + s(-1,0) + s(0,-1) - s(-1,-1) = a + 0 + 0 - 0 = a \\
  s(0,1) &= f(0,1) + s(-1,1) + s(0,0) - s(-1,0) = b + 0 + a - 0 = b + a \\
  s(1,0) &= f(1,0) + s(0,0) + s(1,-1) - s(0,-1) = c + a + 0 - 0 = c + a \\
  s(1,1) &= f(1,1) + s(0,1) + s(1,0) - s(0,0) = d + (b + a) + (c + a) - a = a + b + c + d
\end{align*}
$$

Thus, geometrically, for a given $(x, y)$, the value of the sum-table $S(x, y)$ is the sum of all the pixels above and to the left of $x, y$ inclusively, corresponding to the double sum over the $x$ and $y$ directions that is equivalent to Equation (34).

This operation can be seen pictorially in Figure 3.2, where the sum-table value at location 1 is the sum of all the pixels in region A, and the sum table values at location 2, 3 and 4 correspond to $A + B$, $A + C$ and $A + B + C + D$ respectively.
More importantly, the double sums in Equation (29) are calculated using the simplified double sums of Equations (36) and (37) at every \((u, v)\). For the case of a \(200 \times 200\) template and a \(250 \times 250\) search window, each shift of the template amounts to 3 additions/subtractions for computing the double sum in Equation (36) or (37), compared to \(50^2 = 2500\) additions for a direct computation of \(\sum_{x,y} f(x, y)\) or \(\sum_{x,y} f^2(x, y)\). Thus, a reduction of \(\frac{2500}{3} \approx 800 \times\) is obtained for this very simple case.

### 3.2 Numerator of NCC with Sum-Tables and Basis Functions

In the frequency domain under the FFT, the number of computations required to evaluate the numerator of the NCC in Equation (29) is still relatively high. Furthermore, there is no direct way of reformulating the numerator in terms of
sum-tables to significantly reduce the number of computations, as was done for the denominator in Equations (36)-(39) [18]. However, if the template, \( t \), is approximated by a set of \( K \) basis functions, then an approximation to the NCC can be obtained, which enables the numerator to be written in terms of the sum-table \( s(u,v) \) given by Equation (38). Thus, further, potentially large, computational gains could be obtained over the FFT-based methods [21].

For the case of breast tissue motion tracking considered in this research, the set of \( K \) basis functions will correspond to a set of \( K \) landmark points on the breast. These points could be either randomly placed marker points or natural patterns and tone on the breast skin. The problem can then be thought of in terms of matching landmark points defined as simple basis functions rather than pixel templates. Thus, the basis functions will all be distinct, well defined, can be detected automatically, and are guaranteed to optimally approximate the template.

The template \( t(x,y) \) is rewritten as a weighted sum of \( K \) rectangular basis functions. The 2D compact shape of each landmark point is approximated by a rectangle, which is described by indices \( x^i_l, x^i_u \) and \( y^i_l, y^i_u \) corresponding to the lower and upper bounds of the rectangular areas in the \( x \) and \( y \) direction respectively. Figure 3.3 shows an example of a single arbitrary-shaped landmark point in a template, which is approximated by a rectangle. In practice, the template is described by a matrix of positive integers. Each integer is a value from 0 to 255 where 0 and 255 correspond to black and white respectively, with values in between corresponding to varying intensities of grey.
Figure 3.3: Approximating a simple landmark point in a template by a rectangle.

A typical description of Figure 3.3 would be to have high numbers, greater than 200 (light), describing the landmark with low numbers, less than 100, everywhere else describing the background. A simple one intensity approximation to the landmark would be to make every pixel in the rectangle equal to the average intensity of all the pixels in the landmark. This approach leads to an approximation \( \widetilde{t}(x, y) \) to the template function \( t(x, y) \) defined:

\[
\widetilde{t}(x, y) = \sum_{i=1}^{K} k_i t_i(x, y) 
\]  

where,

\[
t_i(x, y) = 1, \quad x_i^l \leq x \leq x_i^h \text{ and } y_i^l \leq y \leq y_i^h \quad \text{and} \quad y_i^l = 0, \text{ otherwise} 
\]
and $k_i$ is the average intensity of the $i^{th}$ landmark point, $i = 1, ..., K$. Thus, it follows from Equations (42) and (43) that for any $x_i', y_i'$, where $x_i' \leq x \leq x_i$ and $y_i' \leq y \leq y_i'$, corresponding to a pixel in coordinates in the rectangle surrounding the $i^{th}$ landmark, $\tilde{I}(x_i', y_i') = k_i$.

The numerator of Equation (29) can then be written:

$$N(u, v) = \sum_{x=\mu}^{u+u'-1} \sum_{y=\nu}^{v+v'-1} f(x, y) t'(x-u, y-v) - \tilde{I} \sum_{x=\mu}^{u+u'-1} \sum_{y=\nu}^{v+v'-1} t'(x-u, y-v)$$  \hspace{1cm} (44)$$

where $t'(x-u, y-v) = t(x-u, y-v) - \tilde{I}$. Since $t'(x-u, y-v)$ has zero mean, the term $\tilde{I} \sum_{x=\mu}^{u+u'-1} \sum_{y=\nu}^{v+v'-1} t'(x-u, y-v)$ is zero [18]. Replacing $t(x-u, y-v)$ with $\tilde{I}(x-u, y-v)$ from Equation (42) and substituting $t'(x-u, y-v) = \tilde{I}(x-u, y-v) - \tilde{I}$ into Equation (44) gives an approximation for the numerator of the cross correlation coefficient:

$$\tilde{N}(u, v) = \sum_{x=\mu}^{u+u'-1} \sum_{y=\nu}^{v+v'-1} f(x, y) \left( \sum_{i=1}^{K} k_i t_i(x-u, y-v) - \tilde{I} \right)$$  \hspace{1cm} (45)$$

$$= \sum_{i=1}^{K} \sum_{x=\mu}^{x_i+u} \sum_{y=\nu}^{y_i+v} f(x, y) t_i(x-u, y-v) - \tilde{I} \sum_{x=\mu}^{x_i+u} \sum_{y=\nu}^{y_i+v} f(x, y)$$  \hspace{1cm} (46)$$

$$= \sum_{i=1}^{K} k_i \left( \sum_{x=\mu}^{x_i+u} \sum_{y=\nu}^{y_i+v} f(x, y) \right) - \tilde{I} \sum_{x=\mu}^{x_i+u} \sum_{y=\nu}^{y_i+v} f(x, y)$$  \hspace{1cm} (47)$$
Equation (47) follows from Equation (43) where:

\[
t_i(x - u, y - v) = 1, \quad x_i^t + u \leq x \leq x_i^t + u \quad \text{and} \quad y_i^t + v \leq y \leq y_i^t + v
\]

\[
= 0, \quad \text{otherwise}
\]  

Equation (47) is now in a form that can utilize the sum-table formulation of Equation (36). Thus, \( \tilde{N}(u, v) \) is defined:

\[
\tilde{N}(u, v) = \sum_{i=1}^{K} k_i \left( s(x_i^t + u, y_i^t + v) - s(x_i^t + u, y_i^t + v - 1) - s(x_i^t + u - 1, y_i^t + v) 
+ s(x_i^t + u - 1, y_i^t + v - 1) \right) - \tilde{t} \sum_{x=u}^{u+N_x-1} \sum_{y=v}^{v+N_y-1} f(x, y)
\]

where \( \sum_{x=u}^{u+N_x-1} \sum_{y=v}^{v+N_y-1} f(x, y) \) is given by Equation (36), which has already been evaluated during the denominator calculation of Equation (29), and can be reused.

### 3.3 Computational Efficiency Example

To demonstrate the computational efficiency of using Equation (49) to calculate the 2-dimensional NCC compared with the FFT and the traditional CC formulation, consider a \( 4 \times 7 \) template \( t \) with two basis functions \( (K = 2) \) of average intensity 2 and a \( 6 \times 9 \) search window \( f \) matrix defined:
In this case, the NCC of the images \( t \) and \( f \) in Equations (50) and (51) will be a 3\( \times \)3 correlation matrix defined:

\[
\gamma = \begin{bmatrix}
\gamma_{0,0} & \gamma_{0,1} & \gamma_{0,2} \\
\gamma_{1,0} & \gamma_{1,1} & \gamma_{1,2} \\
\gamma_{2,0} & \gamma_{2,1} & \gamma_{2,2}
\end{bmatrix}
\]  

(52)

where \( \gamma_{u,v} \), \( u = 0, \ldots , 2 \), \( v = 0, \ldots , 2 \) are given by Equation (29). Using Equation (49) with \( K = 2 \), \( k_1 = 2 \) and \( k_2 = 2 \), the approximation \( \tilde{N}(0,0) \) to the numerator of the first entry \( \gamma_{0,0} \) in Equation (52) is given by:

\[
\gamma(0,0) = 2(s(x_u^1, y_u^1) - s(x_u^1, y_u^1 - 1) - s(x_u^1 - 1, y_u^1) + s(x_u^1 - 1, y_u^1 - 1)) \\
+ 2(s(x_u^2, y_u^2) - s(x_u^2, y_u^2 - 1) - s(x_u^2 - 1, y_u^2) + s(x_u^2 - 1, y_u^2 - 1)) \\
- i \sum_{x=u}^{u+N_u-1} \sum_{y=v}^{v+N_v-1} f(x, y)
\]

(53)
The expression \( \tilde{t} \sum_{i=u}^{u+N_x-1} \sum_{j=v}^{v+N_y-1} f(x, y) \), has already been calculated in the denominator of Equation (29). Thus, the number of computations required to evaluate Equation (53) is 2 multiplications and 7 additions/subtractions. Similar computations are required for the numerators of the other 8 entries of Equation (52), giving a total of \( 9 \times 2 = 18 \) multiplications and \( 9 \times 7 = 63 \) additions/subtractions, for a total of 81 computations. This total of 81 can be compared to 504 computations for the numerator of the direct calculation of the NCC.

This example can be readily generalized to a \( N_x \times N_y \) template \( t \) with \( K \) basis functions and a \( M_x \times M_y \) search window \( f \) producing a \((M_x - N_x + 1) \times (M_y - N_y + 1)\) correlation matrix requiring in total, \( K(M_x - N_x + 1) \times (M_y - N_y + 1) \) multiplications and \((3K + 1)(M_x - N_x + 1) \times (M_y - N_y + 1) \) additions/subtractions for the \((M_x - N_x + 1) \times (M_y - N_y + 1)\) entries of \( \tilde{N}(u, v) \), \( 0 \leq u \leq M_x - N_x, 0 \leq v \leq M_y - N_y \) given by Equation (49). The results are summarized in Table 3.1, which compares the sum-table formulation to the FFT and direct formulation of the numerator of Equation (29).
<table>
<thead>
<tr>
<th>Method</th>
<th>Number of Multiplications</th>
<th>Number of additions/subtractions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum-Table + basis functions</td>
<td>( K(M_x - N_x + 1)(M_y - N_y + 1) )</td>
<td>((3K+1)(M_x - N_x + 1)(M_y - N_y + 1))</td>
</tr>
<tr>
<td>FFT</td>
<td>( M_xM_y \log_2(M_xM_y) )</td>
<td>( M_xM_y \log_2(M_xM_y) )</td>
</tr>
<tr>
<td>Direct</td>
<td>( N_xN_y(M_x - N_x + 1)(M_y - N_y + 1) )</td>
<td>( N_xN_y(M_x - N_x + 1)(M_y - N_y + 1) )</td>
</tr>
</tbody>
</table>

**Table 3.1:** Number of computations required for calculating numerator of the NCC given by Equation (29).

### 3.4 Basis Functions

The process in Section 3.3 depends entirely on the use of basis functions. The automatic determination of the \( K \) basis functions is equivalent to automatically identifying the marker points in the image. This task is done by initially labelling each marker point region, so that each pixel in a particular region has the same number. In practice, this task is achieved by first thresholding the template image to get a matrix of 0’s and 1’s, then finding all regions inside the matrix that are connected by 1’s. The minimum and the maximum indices for each region are then utilized to form the best representation of a rectangular basis function.

For example, consider the case where the marker points are randomly placed circles, as shown in Figure 3.4 (a). These circles are first labelled and the rectangular basis functions are found as shown in Figure 3.4 (b). As the circles are randomly distributed in the test images, some templates may contain
overlapping circles. In practice, this overlap would correspond to two marker points that are sufficiently close together that thresholding does not separate them. However, this situation would have no significant effect on accuracy, as a single rectangular basis function would cover both circles. Partially formed circles due to the template boundary will similarly have no effect on accuracy as the corresponding rectangle will just have one of its sides on the boundary of the template image. Figure 3.5 illustrates these two examples when automatic basis detection is applied on randomly placed circles.

![Figure 3.4](a) Circle Image (b) Basis Image)

**Figure 3.4:** Template with circles converted to basis functions.

![Figure 3.5](a) Circle Image (b) Basis Image)

**Figure 3.5:** Template with overlapped and partial circles converted to basis functions.
3.5 Algorithm Summary

The fast-NCC based motion tracking algorithm presented, which uses the concepts of sum-table and basis function for efficient calculation of the NCC, operates in the five steps shown in Figure 3.6: image acquisition, template and search window arrangement, calculation of sum-tables and basis functions, NCC calculation and motion pattern representation:

- **Image acquisition** involves using a high-resolution digital camera to capture a pair of images describing the given motion that is required to be tracked.

- **Template and search window arrangement** involves dividing the first image into smaller template images and dividing the second image into search window sub-images, which are of a sufficient size to guarantee that the corresponding template images lie within the window sub-images. The sizes of the window sub-images relative to the template images can be determined from an upper-bound on expected motion. For a more accurate motion field, the template images in the first image are made to overlap by 50%.

- The **sum-tables** for the second image are calculated using the recursive formulas of Equation (38) and (39). The basis functions in each overlapping template sub-image are then calculated by thresholding the image and labelling and identifying the boundaries and centres of landmark points or natural speckle patterns on the skin.
The NCC is calculated using the sum-table representation given by Equations (36) and (37) to match every template image in first image onto the corresponding window sub-image in second image, thus producing a motion vector for each template centre.

The motion vectors for each template centre provide the overall motion pattern representation.

![Diagram](image)

**Figure 3.6:** Motion tracking procedure.
Efficient 2-Dimensional Cross-Correlation Algorithm
RESULTS
Chapter 4

FFT based Motion Tracking

In this chapter, the FFT formulation of Chapters 2 and 3 is applied to track human skin motion, chicken skin motion and computer-generated fluid particle images. The main purpose is to see how good the NCC algorithms developed are at tracking motion of skin from the natural patterns on the skin and to investigate the effect of template window size on accuracy and speed. The final goal is to evaluate the overall computational time involved with the FFT-based NCC. Filtering, interpolation and thresholding methods are also investigated to see if they improve accuracy.

4.1 Tracking Procedure

The technique of template matching explained in Chapter 3 is applied to track the motion induced from stretching skin samples and to track computer-generated fluid particle images, which are discussed later in this chapter. More specifically, a set of template images in the first image from an initial time-step, are matched to a corresponding Region of Interest (ROI) in the second image taken in the next time-step using NCC. However, note that, as discussed in Chapter 3, to compute the NCC using the FFT requires both the template image and the ROI to be the same. Therefore, if the template is smaller than the ROI to be searched, then the template is padded with zeros.
The FFT approach of evaluating the NCC combined with template matching is commonly used in Particle Image Velocimetry (PIV), which is an optical method for determining the instantaneous velocity vector field of air particles or liquid dynamic flow [1]. The method has also been utilized to estimate the motion of soft tissue and blood [2].

MATLAB was used to implement the Fast Fourier Transform-based Cross-Correlation (FFTC) algorithm, which, including the filtering and interpolation operations, can be summarized in five stages:

- **Stage 1:** Charge Coupled Device (CCD) digital camera attached with Single Lenses Reflex (SLR) at 35-136 mm range of focal lengths is utilized to capture the motion exerted on the skin samples. SLR is necessary for imaging close-up shots as it provides sharp focus and stronger contrast.

- **Stage 2:** Image processing is carried out to convert the RGB (24-bit) images into grey-scale images for computational simplicity. Additionally, the visual quality of the images is enhanced using Adobe Photoshop.

- **Stage 3:** The NCC of Equation (29) is calculated in the frequency domain via the relation in Equation (31). The calculation is carried out for every $N_x \times N_y$ template image in the first image with the corresponding $N_x \times N_y$ ROI to produce a $2N_x - 1 \times 2N_y - 1$ correlation plane. The displacement between images is then measured according to the position of the highest peak located within the correlation plane, as shown in Figure 4.1.
Figure 4.1: Locating the highest peak for a single ROI in (a) 3-dimensional and (b) 2-dimensional view of correlation plane.

- Stage 4: For vector validation, the local median method with a $3 \times 3$ kernel size [17] was used to filter any spurious vectors incurred in the motion data produced by FFTC. The local median filter is based on the local medians $m_u$ and $m_v$ and the standard deviations $\sigma_u$ and $\sigma_v$ of two $3 \times 3$ matrices:

$$U = \begin{bmatrix} u_1 & u_2 & u_3 \\ u_4 & u_5 & u_6 \\ u_7 & u_8 & u_9 \end{bmatrix}, \quad V = \begin{bmatrix} v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 \\ v_7 & v_8 & v_9 \end{bmatrix}$$

whose entries are the $U$ and $V$ components of the motion field. A vector $(u_i, v_i)$ for some $i = 1, \ldots, 9$ is considered invalid if, $|u_i - m_u| > 3\sigma_u$, or $|v_i - m_v| > 3\sigma_v$. 
For example, Figures 4.2 (a) and 4.2 (b) represent a $3 \times 3$ kernel size of the $U$ and $V$ components of the raw motion field, which requires filtering. By comparing each vector component in both the $U$ and $V$ matrices with their median values $m_u$ and $m_v$ respectively, the middle entry $(226, -205)$ is found to be invalid since:

$$m_u = -110, \quad \sigma_u = 8.24$$
$$m_v = -55, \quad \sigma_v = 3.93$$

$$|226 - m_u| = 336 > 3\sigma_u$$
$$|-205 - m_v| = 150 > 3\sigma_v$$

(54) (55)

The rest of the entries $U_i$ and $V_i$ satisfy $|u_i - m_u| < 3\sigma_u$ and $|v_i - m_v| < 3\sigma_v$ respectively, therefore only the middle entry $(226, -205)$ is invalid. The end result is a list of valid vectors and invalid vectors across the whole image.

- Stage 5: A linear cubic spline is formed across both the $U$ and $V$ components of the valid vectors and the invalid vectors are linearly interpolated. An example of this interpolation is shown in Figures 4.2 (c) and 4.2 (d).

Stages 1-5 are illustrated schematically in Figure 4.3.
Figure 4.2: An example of using the local median filter to remove a spurious vector and interpolate the filtered vector within a $3 \times 3$ kernel size.
4.2 Experimental Setup

Human and chicken skin are used first to test the algorithm's performance. Both skin samples were stretched and captured using the CCD camera. Figure 4.4 illustrates the experimental setup in the laboratory, where the CCD camera was attached on the adjustable sliding tripod together with two sets of table lamps to fully illuminate the test samples, so that clearer and brighter speckle patterns on the skin samples could be more optimally imaged.
Skin on the arm and knuckle were used for imaging and the chicken skin was extracted from the thigh portion as it contains a high contrast speckle pattern. Fluid particle images were then obtained from the Visualization Society of Japan [44] to further test the method. Figure 4.5 shows a collection of the test samples for the experiment.

**Figure 4.4:** Experimental setup for capturing the images of the stretched human skin and actuated chicken skin.
Figure 4.5: Test images of (a) human skin, (b) actuated chicken skin, (c) computer generated fluid particle image and (d) distinctive skin pattern of a hand knuckle.

4.3 Motion Vector Field for each Test Sample

For the knuckle skin experiment, four different directions of motion were recorded and the template and ROI image were chosen to be of size $256 \times 256$ pixels. This window size was the minimum window size that allowed sufficiently
accurate motion vector fields for this sample. Note that \(256 = 2^8\) which was chosen to be a power of 2 as the discrete FFT requires this property [37]. The motion on the skin was generated by physically stretching the skin in four directions and all detected motions are plotted in Figure 4.6. No vector filtering procedure was required in this situation as the skin texture in this region, shown in Figure 4.5 (d), has a large amount of structure with high contrast.

\[\text{(a)}\]  
\[\text{(b)}\]  
\[\text{(c)}\]  
\[\text{(d)}\]

**Figure 4.6**: Motion fields plotted for human skin on the arm stretched in (a) left, (b) right, (c) up and (d) down direction.
By inspection, Figure 4.6 shows that the resulting motion vector fields are globally non-uniform due to the overall stretch. However, there are also small local differences between vectors due to unevenness in the skin. Thus, the FFTC method successively tracked the knuckle skin motion regardless of which direction the skin was stretched.

The FFTC method was then applied to track the motion between two images of stretched human skin on the arm, and to track the fluid particle motion. The resulting motion fields are plotted in Figure 4.7, where the left and right columns correspond to the motion fields before and after filtering and interpolation respectively. Figure 4.7 (a1) shows that the motion of the human skin on the arm was detected reasonably accurately in the first iteration, apart from a few incorrect vectors directed sideways. The motion field is corrected in Figure 4.7 (a2), which shows that the NCC is capable of tracking skin motion from the natural skin patterns and speckles.

The FFTC method was next applied on the chicken skin. In this case, a $512 \times 512$ window size is used, as the speckle pattern of the chicken skin is approximately twice as big as the human skin. Figures 4.7 (b1) and (b2) show the resulting motion fields, which contain no spurious vectors, indicating a high accuracy in motion tracking over the whole image.

Finally, the FFTC method was applied to track the motion of the particle fluid image of Figure 4.5 (c). The particle fluid motion changes at a significantly smaller scale than the skin examples, so a small $32 \times 32$ window size was chosen. Figure 4.7 (c1) shows that the majority of the motion is detected and
Figure 4.7 (c2) shows that the filtering and interpolation step corrected all spurious vectors and also preserved the continuity of the particle fluid motion.

Figure 4.7: Motion vector fields plotted before and after filtering and interpolation for (a1-a2) human skin on the arm, (b1-b2) actuated chicken skin and (c1-c2) fluid particle motion.
4.4 Correlation Coefficient Threshold

The significant size differences between image patterns and features in the different skin samples gave rise to significantly different window sizes for accurate motion tracking using NCC. An experimentally defined correlation coefficient threshold $\delta$ is therefore used to allow a more automatic selection of window size that maximizes the accuracy of the motion detection. The window size is selected by iteratively expanding the $N_x \times N_y$ correlation search window by a factor $\beta$ until the threshold is reached. Specifically, the window is expanded until the relation $\rho_{xy}(u, v) < \delta$ is satisfied, where $0 < \delta < 1$ and $\rho_{xy}(u, v)$ is defined by Equation (29). This approach ensures a window size will be chosen that minimizes the amount of spurious vectors detected, and thus produce a sufficiently accurate overall motion vector field.

However, the method has the drawback that there could be potentially a large number of iterations required to reach the threshold depending on the size of the initial window and the size of $\beta$, all of which would significantly increase computation. Note that $\beta$ would have to be chosen to be sufficiently small, so that a single iteration does not jump too far away from the optimal window size. The optimal window size is the window that contains a sufficient amount of image contrast to allow accurate comparison between images, but not too large that small local motions are lost. The problem with choosing $\beta$ small however, is that it increases the number of iterations required to reach the optimal window size, leading to increasing computational cost.
Another drawback is that the threshold $\delta$ that produces the optimal window size could vary from sample to sample. Thus, fixing $\delta$ at some pre-decided value could give non-optimal results with a varying degree of loss in accuracy. Figure 8 demonstrates how the search window is expanded using the correlation coefficient threshold.

For the skin samples, the threshold of $\delta = 0.45$ was chosen based on analysing all the motion data gathered from the experiments. The value of $\beta = 2$ was also chosen. Figure 4.9 shows an example of a poor motion field detected for the human skin motion on the arm, where a window size is chosen that does not reach the threshold and the iterative method is not applied.

In Figure 4.9, the pixel area surrounding each motion vector is colour-coded according to the highest correlation coefficient evaluated at that region. Inspection of Figure 4.9 shows that the majority of the spurious vectors have a smaller correlation coefficient and are near the left edge of the image. The
iterative method is then applied and the new motion field is plotted in Figure 4.10 showing high correlation coefficients across the image, where the number of spurious vectors detected has reduced from 48 in Figure 4.9 to 21 in Figure 4.10. Thus, the iterative method has improved the accuracy of the motion vector field. Figure 4.11 shows the final motion field after filtering and interpolation of the motion field in Figure 4.10.

![Figure 4.9: Poor motion field detected without the iterative method.](image)

**Figure 4.9:** Poor motion field detected without the iterative method.
Figure 4.10: Motion Vectors field produced by 0.45 correlation coefficient threshold.

Figure 4.11: Motion vector field from Figure 10 after filtering and interpolation.
4.5 Accuracy and Computational Time for FFTC

The window sizes calculated using the iterative threshold method width \( \delta = 0.45 \) for each test sample varied significantly from 32 x 32 to 512 x 512.

To see the effect that the window size has on accuracy, window sizes of 32, 64, 128, 256 and 512 for the ROI window are tested and their corresponding tracking accuracies are plotted in Figure 4.12. The best window size producing the most accurate motion vector field for the detection of fluid particle motion was 32 x 32 pixels. For human skin samples, the best detection window size was 256 x 256 pixels, though as shown in Figure 4.12, a 512 x 512 window size gives a similar result and thus could also be used for motion detection in this case.

The chicken skin has significantly larger texture compared to the human skin samples, so a search window of 512 x 512 pixels gave the most accurate results. If smaller windows are applied in both cases, then poorer quality motion fields would result. Thus, window sizes have a significant impact on the overall detection accuracy and are dependent on the geometry of the patterns and structure in the images.

![Figure 4.12: Motion tracking accuracies for each test sample under 5 different window sizes.](image-url)
All tracking simulations were performed on a Pentium 4 processor with 1.6 GHz and 512 MB of RAM. Computation time for each example varies significantly since the dimensions of the images and the search window are different in each case. Table 4.1 shows the overall detection performance with optimal window sizes used in each case.

Table 4.2 shows the computational times for motion tracking for the human and chicken skin using the iterative method for determining window size with a starting ROI of $32 \times 32$ pixels, $\delta = 0.45$ and $\beta = 2$. The computational times are significantly greater than those of Table 4.1, which shows that an initial starting point that is far away from the best window size can result in significantly extra computational cost.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>1440 x 2160</td>
<td>201</td>
<td>256 x 256</td>
<td>0.92</td>
<td>89</td>
</tr>
<tr>
<td>Chicken</td>
<td>1440 x 2160</td>
<td>150</td>
<td>512 x 512</td>
<td>3.71</td>
<td>100</td>
</tr>
<tr>
<td>Fluid</td>
<td>256 x 256</td>
<td>4.4</td>
<td>32 x 32</td>
<td>0.07</td>
<td>96</td>
</tr>
</tbody>
</table>

**Table 4.1:** Detection performance of FFTC using the optimal window sizes.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>1440 x 2160</td>
<td>256 x 256</td>
<td>1745</td>
</tr>
<tr>
<td>Chicken</td>
<td>1440 x 2160</td>
<td>512 x 512</td>
<td>1569</td>
</tr>
</tbody>
</table>

**Table 4.2:** Detection performance of FFTC using the optimal window sizes in the iterative method.
4.6 Summary for FFTC

The FFTC tracking algorithm successfully identified the motion patterns within the test samples of human skin, chicken skin and computer-generated fluid particle images. An iterative method was developed to minimize the number of spurious vectors detected in the motion field, which involved expanding the window size until a preset correlation coefficient threshold was met. A $3 \times 3$ kernel size median filter was utilized to further improve motion tracking accuracy. However, the iterative method was found to significantly increase computational cost especially when an initial window size was chosen to be far away from the best window size.
Chapter 5

Motion Tracking using Basis Functions

In this chapter, the fast-NCC method of Chapter 3 is tested on simulated three motion data sets:

- Random circles on a binary image.
- Visco-elastic breast phantom.
- Human skin.

In terms of the DIET system, the concept is to formulate breast surface motion tracking in terms of the tracking of artificially, randomly placed landmark points. As discussed in Chapter 3, each landmark point corresponds to a basis function, which can be utilized to dramatically increase the computational efficiency of the NCC calculation. Hence, these three motion data sets represent tests of increasing realism with respect to the goal application.

5.1 Simulated Data

Another advantage of using artificially placed points is that the optimum window size that gives the highest accuracy in motion tracking, which can be pre-calculated, is dependent only on the density of the randomly placed points. This pre-calculated window size could be applied in all cases as long as the data point
density, which can be physically controlled, remains the same in each case. Thus, the computationally costly iterative method in Chapter 4 for determining the optimal window size is not required. The density of points would thus be chosen to satisfy the accuracy requirements of the DIET system.

Breast surface motion that occurs between two successive images of an actuation sequence is simulated by performing translational motion of up to 50 pixels on binary images of randomly placed circles. Note that allowing too much motion between the images risks missing potentially small perturbations on the surface that could arise due to a tumour [8, 22]. In addition, the motion between images must be sufficiently small to avoid any significant scaling or rotation that would affect the accuracy of the NCC methods, which in any form only examine average translations between images. However, these issues should be readily addressed in the specific DIET implementation that gathers the image data.

To test the speed and accuracy of the sum-table method versus the standard FFT-based NCC method, template images of size $150 \times 150$, $180 \times 180$ and $200 \times 200$, $230 \times 230$ are matched to images of size $200 \times 200$ and $250 \times 250$ respectively. In each case, 5, 10, 20 and 30 circles with a radius of 10 pixels, are randomly placed in each image similar to that shown in Figure 3.4 (a) and Figure 3.5 (a). Figure 5.1 shows examples of $250 \times 250$ images with circle densities of 5, 10, 20 and 30.

The sum-table method is applied to calculate the simulated motion for each image containing the four different circle densities and the computation times are
recorded. Tables 5.1-5.4 show the average CPU times from 100 random simulations of the sum-table method with and without accounting for the time for calculating the basis functions, as well as the CPU time of the FFT-based NCC. All simulations were done on Matlab using a Pentium 4 desktop with 3.0 GHz CPU and 1.0 GMb of RAM.

![Test images with circles' density of (a) 5, (b) 10, (c) 20 and (d) 30.](image)

**Figure 5.1:** Test images with circles' density of (a) 5, (b) 10, (c) 20 and (d) 30.

<table>
<thead>
<tr>
<th>Density of Circles</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT-based NCC</td>
<td>0.5892</td>
<td>0.5892</td>
<td>0.5892</td>
<td>0.5892</td>
</tr>
<tr>
<td>Sum-Table</td>
<td>0.0111</td>
<td>0.0092</td>
<td>0.0103</td>
<td>0.0104</td>
</tr>
<tr>
<td>Sum-Table (with basis)</td>
<td>0.0207</td>
<td>0.0226</td>
<td>0.0271</td>
<td>0.0298</td>
</tr>
</tbody>
</table>

**Table 5.1:** Performance time measured for template of size 150 150 inside a 200 200 search region.
Table 5.2: Performance time measured for template of size 180 × 180 inside a 200 × 200 search region.

<table>
<thead>
<tr>
<th>Density of Circles</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT-based NCC</td>
<td>0.4538</td>
<td>0.4538</td>
<td>0.4538</td>
<td>0.4538</td>
</tr>
<tr>
<td>Sum-Table</td>
<td>0.0107</td>
<td>0.0113</td>
<td>0.0101</td>
<td>0.0094</td>
</tr>
<tr>
<td>Sum-Table (with basis)</td>
<td>0.0256</td>
<td>0.0315</td>
<td>0.0399</td>
<td>0.0466</td>
</tr>
</tbody>
</table>

Table 5.3: Performance time measured for template of size 200 × 200 inside a 250 × 250 search region.

<table>
<thead>
<tr>
<th>Density of Circles</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT-based NCC</td>
<td>0.6286</td>
<td>0.6286</td>
<td>0.6286</td>
<td>0.6286</td>
</tr>
<tr>
<td>Sum-Table</td>
<td>0.0147</td>
<td>0.0163</td>
<td>0.0152</td>
<td>0.0149</td>
</tr>
<tr>
<td>Sum-Table (with basis)</td>
<td>0.0390</td>
<td>0.0395</td>
<td>0.0507</td>
<td>0.0573</td>
</tr>
</tbody>
</table>

Table 5.4: Performance time measured for template of size 230 × 230 inside a 250 × 250 search region.

<table>
<thead>
<tr>
<th>Density of Circles</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT-based NCC</td>
<td>1.7589</td>
<td>1.7589</td>
<td>1.7589</td>
<td>1.7589</td>
</tr>
<tr>
<td>Sum-Table</td>
<td>0.0136</td>
<td>0.0110</td>
<td>0.0116</td>
<td>0.0126</td>
</tr>
<tr>
<td>Sum-Table (with basis)</td>
<td>0.0370</td>
<td>0.0455</td>
<td>0.0635</td>
<td>0.0747</td>
</tr>
</tbody>
</table>

The mean pixel error between the two methods over all simulations in Tables 5.1-5.4 was ~ 0.1 pixels with an average standard deviation of ~ 0.2 pixels. Note that unlike Briechle and Hanebeck [21], here the CPU time for pre-calculating the sum-table is included in all cases.
Tables 5.5-5.8 show the comparison of CPU times for the sum-table method versus the FFT-based CC method, which only involves calculating the numerator of the NCC given by Equation (29). The results show in all cases that the sum-table method is just as accurate and at least an order of magnitude faster than both the FFT-based NCC and CC methods.

<table>
<thead>
<tr>
<th>Density of Circles</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT-based CC</td>
<td>0.3747</td>
<td>0.3747</td>
<td>0.3747</td>
<td>0.3747</td>
</tr>
<tr>
<td>Sum-Table</td>
<td>0.0005</td>
<td>0.0014</td>
<td>0.0020</td>
<td>0.0019</td>
</tr>
<tr>
<td>Sum-Table (with basis)</td>
<td>0.0100</td>
<td>0.0130</td>
<td>0.0192</td>
<td>0.0230</td>
</tr>
</tbody>
</table>

**Table 5.5:** Performance time measured for template of size 150 150 inside a 200 200 search region.

<table>
<thead>
<tr>
<th>Density of Circles</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT-based CC</td>
<td>0.3729</td>
<td>0.3729</td>
<td>0.3729</td>
<td>0.3729</td>
</tr>
<tr>
<td>Sum-Table</td>
<td>0.0008</td>
<td>0.0008</td>
<td>0.0016</td>
<td>0.0017</td>
</tr>
<tr>
<td>Sum-Table (with basis)</td>
<td>0.0164</td>
<td>0.0255</td>
<td>0.0306</td>
<td>0.0383</td>
</tr>
</tbody>
</table>

**Table 5.6:** Performance time measured for template of size 180 180 inside a 200 200 search region.

<table>
<thead>
<tr>
<th>Density of Circles</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT-based CC</td>
<td>0.4285</td>
<td>0.4285</td>
<td>0.4285</td>
<td>0.4285</td>
</tr>
<tr>
<td>Sum-Table</td>
<td>0.0027</td>
<td>0.0019</td>
<td>0.0016</td>
<td>0.0042</td>
</tr>
<tr>
<td>Sum-Table (with basis)</td>
<td>0.0171</td>
<td>0.0252</td>
<td>0.0375</td>
<td>0.0434</td>
</tr>
</tbody>
</table>

**Table 5.7:** Performance time measured for template of size 200 200 inside a 250 250 search region.
The method was then applied to 1M pixel images, where the first image is shown in Figure 5.2. Each image contains 500 circles of radius 10 pixels. Hence, on average, a 250 × 250 image would contain ~ 30 circles. A non-uniform motion field with a maximum displacement of 25 pixels was simulated, as shown in Figure 5.3, where the motion vectors are scaled for ease of viewing.

Note that it is assumed here that the maximum potential displacement between two images during a sinusoidal actuation is known, or estimated, in advance. In practice, this maximum displacement would be an upper bound on expected motion that could be fixed at a chosen value, as there is physical control on the time taken between images in a sequence. The time taken between images can then be used to limit non-translational changes, as well as search areas, but would be dependent on actuation frequencies and amplitudes [22].

<table>
<thead>
<tr>
<th>Density of Circles</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT-based CC</td>
<td>0.4332</td>
<td>0.4332</td>
<td>0.4332</td>
<td>0.4332</td>
</tr>
<tr>
<td>Sum-Table</td>
<td>0.0004</td>
<td>0.0007</td>
<td>0.0020</td>
<td>0.0016</td>
</tr>
<tr>
<td>Sum-Table (with basis)</td>
<td>0.0210</td>
<td>0.0325</td>
<td>0.0461</td>
<td>0.0492</td>
</tr>
</tbody>
</table>

Table 5.8: Performance time measured for template of size 230 × 230 inside a 250 × 250 search region.
The first image \( (I_1) \) and the second image \( (I_2) \) are both broken into 64 smaller 50% overlapping 250 × 250 sub-images. The 250 × 250 sub-images in \( I_1 \) are then each represented by a 200 × 200 template image by truncating each image by 25
pixels on all sides. Since a maximum displacement of 25 pixels is assumed, each
200 × 200 template in I₁ is guaranteed to lie somewhere inside the corresponding
250 × 250 image in I₂. Larger maximum displacements could be handled by
either decreasing the template sizes in I₁ or increasing the sub-images sizes in I₂.
Each 200 × 200 template image in I₁ was then matched to the corresponding 250
× 250 images in I₂ to calculate the motion using NCC.

The difference in motion between the FFT-based and sum-table methods was on
average 0.25 pixels with a standard deviation of 0.43 pixels. However, the total
CPU time for the FFT-based NCC was 37.67 s while the sum-table method took
2.84 s, including basis function calculation. Without including the basis function
calculation, the total CPU time was 1.01 s. In this case, the sum-table method is
approximately 13-37 times faster than the FFT method.

5.2 Visco-Elastic Breast Phantom Test Case

The sum-table and basis functions method presented is applied on two 1M pixel
images taken of two different deformations of a visco-elastic breast phantom
with randomly placed markings, as shown in Figure 5.4. For this example, 75
pixels were used as an upper-bound on the maximum motion between the
images. Image 1 was broken into 50% overlapping templates of size 250 × 250
with corresponding 400 × 400 search windows in Image 2 to account for up to 75
pixel movement in either the x or y directions of the 250 × 250 templates.
Figure 5.5 shows an example of the calculation of basis functions for a template
after thresholding. Figure 5.6 shows the basis functions from Image 1 in Figure 5.5, registered onto the corresponding search window of Image 2 after finding the best template match in Image 2.

![Image 1 and Image 2](image)

**Figure 5.4:** Two 1M pixel images of two different deformations of a visco-elastic breast phantom with randomly placed markings.

The basis function approximated template images of Image 1 in Figure 5.4 (a) are then compared to the search windows in Image 2 in Figure 5.4 (b). This produces the overall motion field between Image 1 and Image 2, and the result is shown in Figure 5.7. The total mean error between the FFT-based NCC motion field and the basis function and sum-table based NCC motion field is 0.12 pixels, with a standard deviation of 0.11 pixels. This result shows that approximating the template with basis functions for computational efficiency does not affect accuracy.
Figure 5.5: Template pattern from the first image of Figure 5.4 (a), thresholded (b), and converted into basis functions (c).
Figure 5.6: Template basis function pattern registered in the corresponding search window image.

Figure 5.7: Motion field representation of the first image in Figure 5.4.
5.3 Human Skin

The third test on the human skin in this chapter is to show that the basis function idea could also be extended to significantly reduce the computation of the NCC in the case where the natural speckle patterns of the breast are tracked rather than artificially placed points.

Figures 5.8 and 5.9 show images taken of two different deformations of human skin. The first image in Figure 5.8 is the reference image and is of size 250 × 250. The second image in Figure 5.9 is the template and is of size 230 × 230. In this case, the natural patterns on the skin are used as basis functions in the sum-table method. To calculate the basis functions to represent the template, the image is first thresholded to detect the brighter pixels corresponding to speckles in the natural skin tone, as shown in Figure 5.10. All regions connected by 1 pixel are then labelled and features that have less than 10 pixels are removed. This process produces the 30 basis functions shown in Figure 5.11.

The sum-table method is then used to approximate the NCC and match the template of Figure 5.9 onto Figure 5.8. Figure 5.12 shows a close up of the registration of speckles on the template of Figure 5.10 to speckles on the target image in Figure 5.8, showing a close match. The motion vectors between the template and search window images for the FFT-based NCC and sum-table based NCC were the same with the vector value (12, 19). This result reaffirms that approximating the template with basis functions does not lead to reduced accuracy.
Figure 5.8: One deformation of human skin.

Figure 5.9: Template corresponding to Figure 5.8.
Figure 5.10: Thresholding template to obtain brighter pixels corresponding to speckles.

Figure 5.11: Representing natural patterns on the skin as rectangular basis functions.
Figure 5.12: Registration of speckles on Figure 5.8 to speckles on Figure 5.6.

In summary, the fast-NCC motion tracking method presented in this chapter is tested on three sets of data, simulated circles, visco-elastic breast phantom and human skin, mimicking the types of applications, which are likely to be seen in the actual DIET system, so they are realistic. It is also shown to have the potential to track the natural marks or tones on human skin. The results are summarized as follows:

Method works accurately for all three sets of data shown by the very low errors.

The sum-table and basis functions formulations for the NCC calculations are significantly faster (13-37×) than the FFT-based NCC or CC methods.

The use of basis functions does not reduce accuracy.
PART IV

CONCLUSIONS
Chapter 6

Discussions and Conclusions

A fast-NCC motion tracking algorithm using basis function approximations to the template and a sum-table scheme has been presented. The purpose is to overcome the heavy computational cost associated with standard FFT-based NCC motion tracking and, more specifically, to enable rapid and accurate breast surface motion tracking in a DIET system. Efficient algorithms are required, given the large images and high density of tracked points expected in a DIET application.

It has been demonstrated that the NCC computation on several hundred or thousand randomly placed landmark points on the breast surface can be made highly efficient through the use of pre-calculated sum-tables, which act as a look-up table dramatically reducing the number of computations. Furthermore, the sum-tables over the image function require very little extra computational effort as they only involve cumulative sums along the columns or rows of the image matrix, which can be computed very quickly in one global sweep.

The formation of basis functions on each template effectively reduces the amount of pixel information to be dealt with, by only concentrating on the important or essential features of the ROI. The detection of basis functions is equivalent to identifying or locating the boundaries and centre points of each landmark point.
or speckle pattern on the skin surface. Thus, the fast-NCC motion tracking essentially matches large numbers of landmark points approximated by basis functions between images.

The FFT formulation is unable to make use of the structures and patterns in the image to simplify the NCC calculation, as it is dependent only on the size of the images, not the actual distribution of intensities. Thus, the FFT formulation is very inefficient because it has to deal with a large amount of redundant information on the image even though FFT is more efficient than the direct evaluation of NCC. As shown in the FFTC experiment of Chapter 4, the texture pattern of the skin on the human arm and the size of the detection windows were seen to have a significant impact on the overall motion detection accuracy. An iterative method for finding the optimum window size was developed in Chapter 4, but it was very computationally expensive as shown by the CPU times in Table 4.2.

The use of artificially placed points on the breast surface avoids having to use the iterative method during motion tracking as the optimum window size can be pre-calculated, dependent only on the density of the randomly placed points, which can be physically controlled. However, if required, the fast-NCC motion tracking algorithm could also potentially be applied to track natural patterns on the skin, as shown by the results in Chapter 5.

The motion calculated using the fast sum-table based NCC of Chapter 3 was always within 1 pixel of the FFT-based NCC for both simulated motion and
human skin motion. This result shows that there is no compromise on accuracy when using the sum-table method. However, the sum-table method is at least an order of magnitude faster than the FFT for both NCC and CC in all cases. Depending on the size of the images, the sum-table method is $\sim$11-28 (9-20) times faster than FFT for NCC (CC) including basis function calculation and $\sim$37-150 (187-271) times faster than FFT for NCC (CC) not including basis function calculation.

For the case of breast surface motion tracking, given the high accuracy required to detect small perturbations on the surface, every landmark point must be precisely identified between images for accurate interpolated motion and camera calibration. Thus, both the FFT and sum-table based methods require the known position of the landmarks in every image. The process of finding the position of a landmark point is effectively the same as calculating the basis functions. Thus, the sum-table based NCC is effectively 37-150 times faster than FFT-based NCC depending on the sizes of the images considered. Such a large computational saving for calculating the NCC is important for real time clinical application of the DIET system in breast cancer screening, or any other application tracking large numbers of points.
Chapter 7

Future Work

The idea of the rectangular basis function approximations used in this thesis is designed to only use the dominating image features to represent each template image, rather than including the background noise. For the case of artificially placed landmark points, as shown by the visco-elastic breast phantom example in Chapter 5, the landmark points heavily dominate the image and are of a significantly different intensity than the background noise, ensuring an accurate detection of basis functions. However, currently the images are thresholded globally with a constant threshold value that is determined experimentally depending on the nature or the characteristics of the images.

A more robust basis function detection method that would take into account, potentially significant changes in light through an image sequence would be to normalize each template image into a pre-defined range based on the given range of intensities. For example, an image enhancement routine like linear contrast stretch could be used on each template before thresholding that stretches the range of pixel intensities into the full range of grey-values from 0 to 255. Thus, the image's contrast is enhanced, where light toned areas will appear lighter and dark areas will become darker, readily enabling a consistent threshold to be used over all images.
The computational savings of the basis function and sum-table formulation of NCC does not utilize the fact that motion vectors between two 50% overlapping sub-images do not change significantly. To find the motion vector for the current template centre, the closest motion vector of a previously computed template centre could be used as an approximation. Since the motion fields of the previous and current template centres may not differ significantly, this approach would dramatically reduce the number of shifts required to match the current template to the first image and thus reduce $M_x \times N_x$ and $M_y \times N_y$ in Table 3.1. This bootstrapping idea could be applied over the whole image further significantly reducing computational requirements. Note that the FFT approach is not able to take advantage of such bootstrapping as the computational time is largely dependent on the size of the template, not the number of the shifts required to match the template to the image.
References


[44] Visualization Society of Japan: [www.vsj.or.jp](http://www.vsj.or.jp).
Appendices A MATLAB CODE

A1 Main Function for Fast-NCC Method ................................ A - 1
A2 Function for Generating the Basis Functions ......................... A - 4
A3 Function for Calculating the NCC via Sum-Table .................. A - 5
A4 Function for Plotting the Motion Vector Field ..................... A - 6
A5 Function for Comparing Circles and Basis Functions ............. A - 8
A6 Function for Producing N\textsuperscript{th} Density of Circles .......... A - 9
A7 Function for Producing N\textsuperscript{th} Density of Circles .......... A - 11
and Apply Motion to Each Circle
A1: Main Function for Fast-NCC Method

function [xx,yy,uu, vv, dtime] = fast_NCC_sum(imtype, im1, im2, fwinsize, ...
swinsize, overlap, thres, plotshow)

% The function [XX,YY,UU,VV,DTIME] = FAST_NCC_SUM(IMTYPE,IM1,IM2,FWIN_SIZE,
% SWINSIZE,OVERLAP,THRES,PLOTSHOW) perform the normalized cross-
% correlation (NCC) calculation by using the Sum-Table scheme, which is
% substantially faster than the traditional approach as well as FFT method.
% The function works by following five main steps:
% (1) Loading the images and convert to 256 grey-scales intensities.
% (2) Subdividing the image into sub-windows and templates
% (3) Determine the basis functions in each template.
% (4) Form the sum-tables to evaluate the NCC calculations.
% (5) Obtain the motion data and illustrate the motion vector flow pattern.
% Example of executing the function in MATLAB.
% [xx,yy,uu,vv,dtime] = fast_NCC_sum(2, im1,im2,250,75,0.5,120,'Y');
% Input: IMTYPE = 1, Load the digital images.
% 2, Load the matrices.
% IM1, IM2 = Input image pair for NCC tracking.
% FWIN_SIZE = Size of Template.
% SWINSIZE = Dimension required to form the Search Window.
% OVERLAP = Overlapping ratio between the windows.
% THRES = Grey-scale value for thresholding.
% PLOTSHOW = Y, illustrate the test images.
% N, no images are shown.
% Output: XX, YY = Locations of the motion vectors.
% UU, VV = Detected displacement vector components.
% DTIME = CPU time required for completing the routine.
% Author: Anthony J. H. Hii, Dr. Christopher E. Hann &
% Assoc. Prof. J. Geoffrey Chase
% Date: 27th May 2005

% 1. Image Processing
% Check all input arguments.
if nargin < 8
  error('Insufficient input parameters!!!');
disp('input(imtype,im1,im2,fwinsize,swinsize,overlap,thres,plotshow'));
end

% Routine for loading image files into matrices.
if (imtype == 'IMAGES') | (imtype == 1)
  [filename1,pathname1]=uigetfile('*.bmp; *.tiff; *.jpg', 'Choose file1');
  [filename2,pathname2]=uigetfile('*.bmp; *.tiff; *.jpg', 'Choose file2');
  im1 = imread(fullfile(pathname1,filename1));
  im2 = imread(fullfile(pathname2,filename2));
else if (imtype == 'MATRIX') | (imtype == 2)
  im1 = im1;
  im2 = im2;
else
  error('Specify the type of data for analysis: IMAGES/ MATRIX')
end

% Cast into real numbers instead of integers.
A = double(im1);  B = double(im2);
fwinsize = round(fwinsize);
swinsize = round(swinsize);

% Check the overlap ratio is within the range 0-1.
% Amount of overlap (pixels) is relative to the window size.
if overlap > 1
    error('Overlap ratio should be in the range 0-1.');
else
    winoverlap = floor((1-overlap)*fwinsize);
end

% Check all images are in gray-scale: Grey value = (R+G+B)/3
% Alternatively, use the RGB2GRAY function from the Image tool-box.
if (length(size(A)) > 2) | ndims(A) == 3
    A = (A(:,:,1) + A(:,:,2) + A(:,:,3))/3;
    B = (B(:,:,1) + B(:,:,2) + B(:,:,3))/3;
end

% The matrix entries have been linearly stretched between
% the initial minimum and maximum entries, in order to
% occupy the whole gray scale, from 0 to 255.
M1 = max(max(A));  M2 = max(max(B));
ml = min(min(A));  m2 = min(min(B));
A = 255/(M1-ml)*(A-ml);
B = 255/(M2-m2)*(B-m2);

% Thresholding the images if necessary.
if thres == 0
    A = (A < thres);
    B = (B < thres);
end

% Output the two sequential images before and after pre-process.
if plotshow == 'Y'
    figure(1),
    subplot(221), imagesc(im1), title('Original 1st'), axis image
    subplot(222), imagesc(A), colormap(gray(256)),
    title('Converted 1st'), axis image
    elseif plotshow == 'N'
        disp('Sequential images are not shown.')
end

% Check that test images must have identical sizes.
if size(A) ~= size(B)
    disp('Size of test images are DIFFERENT!!!')
    [ax, ay] = size(A);
    B = B(1:ax, 1:ay);
    fprintf(1,'PRE-PROCESSING finished. \n' )
    disp('.... .... ....')
else
    disp('.... .... ....')
    fprintf(1,'PRE-PROCESSING finished. \n'
    disp('.... .... ....')
end
Main loop for template and search window arrangement as well as performing the NCC motion tracking procedures.

```matlab
m = 1;
n = 1;
tt = cputime;
gui_active(1);
h = progressbar([],0,'Executing two-dim NCC ...', 'DIEP Motion Imaging');
for ii = 1+swinsize:winoverlap:size(A,1)-fwinsize-swinsize+1
    for jj = 1+swinsize:winoverlap:size(A,2)-fwinsize-swinsize+1
        % 2. Subdivide the images into Template and Search Window.
        subim = A(ii:ii+fwinsize-1, jj:jj+fwinsize-1);
        subsm = B(ii-swinsize:ii+fwinsize+swinsize-1,...
                    jj-swinsize:jj+fwinsize+swinsize-1);
        % 3. Determine the basis functions in each template.
        [imlim, ptime, imbasis] = fastbasis(subim, 2);
        % 4. Form the sum-tables to evaluate the NCC calculations
        [correlation] = sumNCC_fast(subim, subsm, imlim);
        % Locate the maximum coefficient value in the correlation plane and
        % Measure the displacement detected.
        [xd, yd] = find(correlation == max(max(correlation)));
        cplane(m,n) = max(max(correlation));
        xdf = -(xd-size(correlation,1));
        ydf = -(yd-size(correlation,2));
        % Assigning output parameters.
        xx(m,n) = ii;
        yy(m,n) = jj;
        if (length(xdf) > 1) | (length(ydf) > 1)
            uu(m,n) = mean2(xdf)-swinsize;
            vv(m,n) = mean2(ydf)-swinsize;
        else
            uu(m,n) = xdf-swinsize;
            vv(m,n) = ydf-swinsize;
        end
        % Update the counter.
        n = n + 1;
    end
    n = 1;
    m = m + 1;
    h = progressbar(h, winoverlap/(size(A,1)-fwinsize-swinsize+1));
    if ~gui_active
        break;
    end
end
progressbar(h,-1);
```

% 5. Motion Field Presentation.

dtime = cputime-tt;
[xx,yy,uu,vv] = motion_plot(xx,yy,uu,vv,fwinsize,swinsize,im2,dtime);

%---------------------------------------- End of fast_NCC_sum.m ----------------------------------------
A2: Function for Generating the Basis Functions

function [blim, ptime, imbasis] = fastbasis(imin, plott)
% The function [BLIM, PTIME, IMBASIS] = FASTBASIS(IMIN, PLOTT)
% transforms the input image IMIN into rectangular basis
% function IMBASIS. The vertices detected for each basis function
% are stored in the array BLIM for NCC calculation via Sum-Table.
% Input: IMIN = Input binary image (template).
% PLOTT = 1, illustrate both circle and basis function images.
% Otherwise, no images are presented.
% Output: BLIM = Upper and Lower bound of the basis functions.
% PTIME = CPU times for basis function formation.
% IMBASIS = Basis function image.
% Author: Anthony J. H. Hii, Dr. Christopher E. Hann &
% Assoc. Prof. J. Geoffrey Chase
% Date: 12th June 2005

% Label each ROI and determine the vertices of each ROI.
tt = cputime;
imbw = bwlabel(imin, 8);
troi = max(max(imbw));
LL = 1;
while LL <= troi
    [r, c] = find(imbw == LL);
    rr(LL) = r;
    cc(LL) = c;
    row1(LL) = min(r);
    row2(LL) = max(r);
    col1(LL) = min(c);
    col2(LL) = max(c);
    LL = LL + 1;
end
% Store the vertices according to each ROI located.
imbasis = imin;
if LL == 1
    error('Template pattern too small-!')
else
    blim(1,:) = row1;
    blim(2,:) = row2;
    blim(3,:) = col1;
    blim(4,:) = col2;
end
% Total time taken for basis function formation.
ptime = cputime-tt;
% Form the approximated basis function image.
if plott == 1
    imbasis = imin;
    for ii = 1:size(blim, 2)
        imbasis(blim(1,ii):blim(2,ii), blim(3,ii):blim(4,ii)) = 1;
    end
% Illustrate the circle and basis images.
A3: Function for Calculating the NCC via Sum-Table

function [correlation] = sumNCC_fast(subim, subsm, ilim)
% % The function [CORRELATION] = SUMNCC_FAST(SUBIM, SUBSM, IMLIM) uses the % Sum-Table schema to evaluate the 2-dimensional NCC between the search % window SUBIM and template SUBSM according to the upper and lower bound % limits stored in the matrix IMLIM. In addition, the calculation is % expected to be much faster than the conventional or traditional NCC in % MATLAB or FFT because the 'for' loop is not utilized to get extra % speed gain. % % Input: SUBIM = Search Window % SUBSM = Template % IMLIM = Upper and Lower bound of the basis functions. % % Output: CORRELATION = Correlation plane. % % Author: Anthony J. H. Hii, Dr. Christopher E. Hann & % Assoc. Prof. J. Geoffrey Chase % Date: 27th May 2005 % % Generate the abale via using the % cummulative sum method.
BB1 = subsm;
BB2 = BB1.^2;
st1 = cumsum(cumsum(BB1))';
st2 = cumsum(cumsum(BB2))';

% Check parameters for NCC evaluation.
[m,n] = size(subsm);
[Nx,Ny] = size(subim);
tempm = mean2(subim);
tempsum1 = sum(sum((subim-tempm)));
tempsum2 = sum(sum((subim-tempm).^2));

% Assign the limits of the rectangular basis function.
xl = imlim(1,:); xu = imlim(2,:);
yl = imlim(3,:); yu = imlim(4,:);
% Main loop for evaluating the Numerator of the NCC.
st10 = zeros(m+1,n+1);
st10(2:m+1, 2:n+1) = st1;
Numers = 0;
for kk = 1:size(imlim,2)
    ul = x(kk) + [0:m-Nx];
    vl = y(kk) + [0:n-Ny];
    uu = x(kk) + [0:m-Nx]+1;
    vu = y(kk) + [0:n-Ny]+1;
    Numers = Numers + st10(uu,vu)-st10(ul,vu)-st10(uu,vl)+st10(ul,vl);
end

% Algorithm for evaluating the Denominator of the NCC.
st100 = zeros(m+1,n+1);
st200 = zeros(m+1,n+1);
st100(2:m+1, 2:n+1) = st1;
st200(2:m+1, 2:n+1) = st2;
ft1 = st100(Nx+[2:m-Nx+2]-1, [2:n-Ny+2]+Ny-1) - ...
     st100([2:m-Nx+2]-1, [2:n-Ny+2]+Ny-1) - ...
     st100([2:m-Nx+2]+Nx-1, [2:n-Ny+2]-1) + ...
     st100([2:m-Nx+2]-1, [2:n-Ny+2]-1);
ft2 = st200(Nx+[2:m-Nx+2]-1, [2:n-Ny+2]+Ny-1) - ...
     st200([2:m-Nx+2]-1, [2:n-Ny+2]+Ny-1) - ...
     st200([2:m-Nx+2]+Nx-1, [2:n-Ny+2]-1) + ...
     st200([2:m-Nx+2]-1, [2:n-Ny+2]-1);

% Final NCC coefficient plane.
Denoms = (ft2 - (ft1.^2)/(Nx*Ny)) * tempsum2;
Numersd = Numers - (tempm * ft1);
correlation = imrotate(Numersd ./ sqrt(Denoms),180);

%------------------------------------------------ End of sumNCC_fast.m -------------------------------

\textbf{A4: Function for Plotting the Motion Vector Field}

function [xx, yy, uu, vv] = motion_plot(xx, yy, uu, vv, fwinsize, swinsize, iml, dtime)
% The function \texttt{[XX, YY, UU, VV]=MOTION\_PLOT(XX, YY, UU, VV, FWINSIZE, SWINSIZE, IM1, DTIME)} illustrates the final motion vectors field together with the reference image. Scaled vectors field are also shown for ease of viewing.
% Input: Xx, Yy = Vector locations.
% UU, VV = Displacement vectors.
% FWINSIZE = Template size.
% SWINSIZE = Dimension required to form the Search Window.
% IM1 = Reference image.
% DTME = Total CPU time.
% Output: XXX, YYYY = Vector locations.
% UU, VV = Displacement vectors.

% Author: Anthony J. H. Hii, Dr. Christopher E. Hann &
% Assoc. Prof. J. Geoffrey Chase
% Date: 27th May 2005

% Determine the average displacement incurred.
mdistance_x = mean(mean(abs(uu)));
mdistance_y = mean(mean(abs(vv)));

% Statistical results: displacements & time-elapsed.
result1 = strcat('> Mean displacement in x and y (pixel) = ', num2str(mdistance_x, '%8.4f'), ', ', num2str(mdistance_y, '%8.4f'));
result2 = strcat('> Maximum displacement in u and v (pixel) = ', num2str(max(max(abs(uu))), '%8.4f'), ', ', num2str(max(max(abs(vv))), '%8.4f'));
result3 = strcat('> Elapsed time [sec] = ', num2str(dtime));
disp(result1)
disp(result2)
disp(result3)
disp('============================================================')

% Plot the motion vectors field.
figure, quiver(yyyy, xxx, vvv, uuu, 0); axis image, axis ij
title('Motion vectors for detected speckles motion (SUM-TABLE)');
xlabel('x (pixel)'), ylabel('y (pixel)');

% Plot the motion field together with the test image.
figure, A = iml;
[nxx, nyy, nnn] = size(A);
ccc = floor(length(A)/length(yyy));
[xxx, yyy] = meshgrid(ccc:ccc:length(A)-ccc, ccc:ccc:length(A)-ccc);
imagesc(1:nxx, 1:nyy, A), axis image, title('Image with Detected Motion Field');
hold on
quiver(yyyy, xxx, vvv(1:length(yyy), 1:length(yyy)), uuu(1:length(xxx), 1:length(xxx)), 0),
axis image, axis ij, colormap(gray(256)), hold off

%------------------ End of motion_plot.m ------------------

Published with MATLAB 7.0
A5: Function for Comparing Circles and Basis Functions

function [imc, imb] = circle_n_basins(nx, ny, radi, ndensity)

% The function [IMC, IMB] = CIRCLE_N_BASIS(NX, NY, RADI, NDENSITY) generates desired NDENSITY number of circles with specified radius RADI, which are randomly placed inside the NX x NY image. Additionally, each circle is transformed into a basis function.

% Input: NX, NY = Row and Column size of the image.
% RADI = Radius of each circle.
% NDENSITY = Number of circles defined.
% Output: IMC = Image with circles.
% IMB = Image with basis functions.

% Author: Anthony J. H. Hii, Dr. Christopher E. Hann & Assoc. Prof. J. Geoffrey Chase
% Date: 12th July 2005

% Define the size of a blank image.
bm = zeros(nx, ny);

% Morphological structuring element function for specifying the circle formation.
dot = strel('disk', radi, 6);

% Main loop for randomly distributing the dots.
i = 1;
while i <= ndensity
    x = floor(rand*nx); y = floor(rand*ny);
    if x == 0, x = 1; end
    if x < 0, x = -1*x; end
    if y == 0, y = 1; end
    if y < 0, y = -1*y; end
    bm(x, y) = 1;
    i = i + 1;
end

% Dilation procedure for generating the circles from the dots.
blackml = imdilate(bm, dot);
imc = double(blackml > 0);

% Main loop for identifying the vertices of each ROI, where circles is the Region of Interest.
imbw = bwlable(imc, 8);
troi = max(max(imbw));
LL = 1;
while LL <= troi
    [r, c] = find(imbw == LL);
    rr[LL] = r;
    cc[LL] = c;
    row1(LL) = min(r);
    row2(LL) = max(r);
    col1(LL) = min(c);
    col2(LL) = max(c);
    LL = LL + 1;
end
% Main loop for creating the basis functions based on
% the vertices measurements obtained previously.
imbasis = imc;
blim(1,:) = row1;
blim(2,:) = row2;
blim(3,:) = col1;
blim(4,:) = col2;
for ii = 1:size(blim,2)
    imbasis(blim(1,ii):blim(2,ii), blim(3,ii):blim(4,ii)) = 1;
end

% Illustrate both circle and basis function images.
figure,
imc = imc;
imb = imbasis;
subplot(121), imagesc(imc), axis image, colormap(gray), title('Circle Image')
subplot(122), imagesc(imb), axis image, colormap(gray), title('Basis Image')

%------------------ End of circle_n_basis.m ----------------------

A6: Function for Producing $N^{th}$ Density of Circles

function [im1, im2] = small_circles(nx, ny, mx, my, radi, ndensity)
% % The function [IM1, IM2] = SMALL_CIRCLES(NX, NY, MX, MY, RADI, NDENSITY)
% generates two NX x NY images IM1 & IM2 that contains NDENSITY numbers of
% circles with radius RADI. The 1st image IM1 is treated as the reference
% image, while the 2nd image IM2 has been applied with constant motion
% components MX & MY. The output images represent the simple test samples
% for the experiment, since the motion-involved are linear and constant.
% % Input: NX, NY = Row and Column size of the image.
% % MX, MY = X-across and Y-down motion components.
% % RADI = Radius of each circle.
% % NDENSITY = Number of circles defined.
% % Output: IM1 = Reference or 1st Image.
% % IM2 = 2nd Image.
% % Author: Anthony J. H. Hii, Dr. Christopher B. Hann &
% % Assoc. Prof. J. Geoffrey Chase
% % Date: 20th July 2005
% %

% Form the two blank images.
bm1 = zeros(nx,ny);
bm2 = zeros(nx,ny);
% Main loop for forming the circles and randomly distributing
% circles. Shifting all the circles in the 2nd image based on the
% motion components defined.
i = 1;
dot = strel('disk', radi, 6);
while i <= ndensity
    x = floor(rand*nx); y = floor(rand*ny);
    % Original image with dots.
    if x == 0, x = 1; end
    if x < 0, x = -1*x; end
    if y == 0, y = 1; end
    if y < 0, y = -1*y; end
    bm1(x,y) = 1;

    % Constant motion applied to each dots.
    xx = x + mx;
    yy = y + my;
    if xx == 0, xx = 1; end
    if yy == 0, yy = 1; end
    if xx < 0, x = -1*xx; end
    if yy < 0, yy = -1*yy; end
    bm2(xx,yy) = 1;

    % Set the boundary for each circle placed.
    if (xx > nx) | (yy > ny)
        bm2(xx,yy) = 0;
    end
    i = i + 1;
end

% Generate final images with circles via dilation process.
blackm1 = imdilate(bm1, dot);
blackm2 = imdilate(bm2, dot);
blackm1 = double(blackm1 > 0);
blackm2 = double(blackm2 > 0);

% Check the final test images have the same dimension.
imi = blackm1;
if size(blackm2) == size(imi)
    blackm22 = blackm2;
    blackm22(nx,ny) = 0;
    im2 = blackm22(1:nx, 1:ny);
else
    im2 = blackm2(1:nx, 1:ny);
end

% Illustrate the output test images.
figure,
subplot(121), imagesc(imi), axis image, colormap(gray), title('1st Image')
subplot(122), imagesc(im2), axis image, colormap(gray), title('2nd Image')
%------------------ End of small_circles.m ------------------
A7: Function for Producing N\textsuperscript{th} Density of Circles and

Apply Motion to Each Circle

function [imb1,imb2,maxdx,maxdy] = big_circles(k,mx,my,radi,ndensity)
%
The function [IMB1,IMB2,MAXDX,MAXDY] = BIG_CIRCLES(K,MX,MY,RAD1,NDENSITY)
generates two images of 1000 x 1000 pixels that contain circles of
radius RADI and randomly distributed within the two images. The input
K controls the number of pair test images to be formed.
%
The motion involved in the 2nd image is based on the function:
%
u-direction: x = x + mx*x
v-direction: y = y + my*y
%
Alternatively, this function can be modified or changed to any function
preferred by the user.
%
Input: k = Nth pair of images for testing.
MX, MY = X-across and Y-down motion components.
RADI = Radius of each circle.
NDENSITY = Number of circles defined.
%
Output: IMB1 = An array that contains the 1st image.
IMB2 = An array that contains the 2nd image.
maxdx, maxdy = Maximum displacement incurred.
%
Author: Anthony J. H. Hii, Dr. Christopher E. Hann &
Assoc. Prof. J. Geoffrey Chase
Date: 20th July 2005
%
% Main loop for generating kth pair of test images.
for jj = 1:k
nx = 1000;
ny = 1000;
blackml = zeros(nx,ny);
%
% Main loop for forming the circles and randomly distributing
% circles. Shifting all the circles in the 2nd image based on the
% motion components defined.
i = 1;
dot = strel('disk',radi,6);
while i <= ndensity
x = floor(rand*nx); y = floor(rand*ny);
%
% Original image with dots.
if x == 0, x = 1; end
if x < 0, x = -1*x; end
if y == 0, y = 1; end
if y < 0, y = -1*y; end
blackml(x,y) = 1;
%
% Function-based motion.
xx = floor(x + mx*x);
yy = floor(y + my*y);
%
% 2nd image with dots shifted.
if xx == 0, xx = 1; end
if yy == 0, yy = 1; end
if xx < 0, x = -1*xx; end
if yy < 0, yy = -1*yy; end
blackm2(xx,yy) = 1;

% Set the image boundary for the circles generation.
if (xx > nx) | (yy > ny)
    blackm2(xx,yy) = 0;
end

% Determine the maximum displacement incurred.
xf(i) = x;
yf(i) = y;
xff(i) = xx;
yff(i) = yy;
i = i + 1;

% Generate final images with circles via dilation process.
blackm1 = imdilate(blackm1,dot);
blackm2 = imdilate(blackm2,dot);
blackm1 = double(blackm1 > 0);
blackm2 = double(blackm2 > 0);

% Form the two testing images.
im1 = blackm1;
if size(blackm2) == size(blackm1)
    blackm2(nx,ny) = 0;
    im2 = blackm2(1:nx, 1:ny);
else
    im2 = blackm2(1:nx, 1:ny);
end

% Store the image pairs.
imb1{jj} = im1;
imb2{jj} = im2;

% Determine the maximum displacement incurred.
a = max(max(xf));
b = max(max(xff));
c = max(max(yf));
d = max(max(yff));
maxdx{jj} = (b-a) + 1;
maxdy{jj} = (d-c) + 1;

figure(jj),
subplot(121), imagesc(imb1{jj}), colormap(gray),
axis image, title('1st Image')
subplot(122), imagesc(imb2{jj}), colormap(gray),
axis image, title('2nd Image')

% Clear the previous test images.
clear blackm1;
clear blackm2;
end

%------------------------------------------------- End of big_circles.m --------------------------------------------------

Published with MATLAB 7.0