2010

Implementation of spatial shift estimation approach for 3D profilometry based on digital fringe projection

Pu Cao
University of Wollongong

IMPLEMENTATION OF SPATIAL SHIFT ESTIMATION APPROACH FOR 3D PROFILOMETRY BASED ON DIGITAL FRINGE PROJECTION

A thesis submitted in fulfilment of the requirement for the award of the degree of

Master of Engineering Research

from

University of Wollongong

By

PU CAO

B.E, Huazhong University of Science and Technology of China

Master of Internet Technology, University of Wollongong

School of Electrical, Computer and Telecommunications Engineering

2010

CERTIFICATION

I, Pu Cao, declare that this thesis, submitted in partial fulfilment of the requirements for the award of Master of Engineering Research, in the School of Electrical, Computer & Telecommunications Engineering, University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualifications at any other academic institution.

Pu Cao

24 March 2010

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

FPP	Fringe Pattern Profilometry
3D	Three-Dimensional
DFP	Digital Fringe Projection
PDE	Phase Difference Estimation
SSE	Spatial Shift Estimation
FTP	Fourier Transform Profilometry
PSP	Phase Shifting Profilometry
PMP	Phase Measuring Profilometry
MMP	Modulation Measurement Profilometry
SPD	Spatial Phase Detection
PLL	Phase Lock Loop
MT	Moiré Technique
FFT	fast Fourier Transform
IFT	inverse Fourier Transform
IFSE	Inverse Function based Shift Estimation
PC	Personal Computer
FIR	Finite Impulse Response
SDE	Screen Door Effect

ABSTRACT

Fringe Pattern Profilometry (FPP) based on Digital Fringe Projection (DFP) is a promising optical noncontact three-dimension (3D) profile measurement technologies due to its accuracy and flexibility. Popular FPP approaches retrieve the 3D profile information using the detection of phase difference, called the Phase Difference Estimation (PDE). Recently, a new kind of FPP approach, referred to as Spatial Shift Estimation (SSE) is introduced, which retrieves the 3D profile information using the detection of spatial shift instead of phase different. Compared with PDE approaches, SSE approaches are advantageous in that the projected fringe patterns do not need to be sinusoidal, and thus accurate reconstruction can be obtained even when nonlinear distortions exist on the fringe patterns. However, efficient implementation of SSE approaches is still an issue.

This thesis work aims to implement the SSE approach for 3D profile measurement based on digital fringe projection. Firstly, a DFP system is designed and adopted in our laboratory, which is utilized as an experiment platform for the work presented in this thesis. SSE approaches are implemented on the system. Some problems associated with the implementation are studied and solved, including elimination of noise and distortion in the fringe patterns. Furthermore, an improved Inverse Function based Shift Estimation (IFSE) method is proposed to improve the performance of SSE approaches.

Secondly, shift unwrapping problem associated with SSE is investigated. Through reviewing the phase unwrapping problem in PDE based FPP, we indicate that a similar shift unwrapping problem also exists in SSE approaches. A method for solving the problem has been proposed and the experiment results are presented to demonstrate the effectiveness of the proposed method.

Finally, the research is carried out to improve the efficiency of SSE approaches. SSE approaches have the advantages that the projected fringe patterns are no longer required to be sinusoidal nor periodic. Therefore, we can choose a fringe pattern which has strong counter-interference capability against the noise and nonlinear distortion with simple implementation. Based on analysis of the limitations of traditional sinusoidal fringe, we propose to use sawtooth fringe pattern. Theoretical analysis has been given to evaluate the complexity of the proposed sawtooth fringe pattern based algorithms, and practical experiment are performed at last to prove the efficiency of this proposed fringe pattern.

ACKNOWLEDGEMENT

I wish to express my deepest appreciation to all the people that have contributed to the completion of this thesis. First of all, I would like to express my genuine gratitude to Associate Professor Jiangtao Xi and Professor Joe Chicharo, my supervisors, for their invaluable guidance and encouragement in the research and preparation of this thesis. Without their patience, this work would not be possible.

I sincerely thank Dr. Matthew J. Baker and Dr. Yingsong Hu for sharing their knowledge and experiences in the area of three-dimensional profile measurement by digital video fringe projection. They gave me patient and valuable direction throughout the experiment process.

I am also indebted to all my best friends for their friendship and supports.

Finally, I would like to thank my grandmother and my parents for their endless love, which encourages me to overcome all problems.

LIST OF PUBLICATIONS

Pu Cao, Jiangtao Xi, Joe Chicharo and Yanguang Yu, “A Fringe Period Unwrapping Technique for Digital Fringe Profilometry based on Spatial Shift Estimation”, in *Optical Inspection and Metrology for Non-Optics Industries*, edited by Peisen S. Huang, Toru Yoshizawa, Kevin G. Harding, Proceedings of SPIE Vol. 7432 (SPIE, Bellingham, WA 2009) 743208.

Charpter 1 Introduction

In recent years, optical noncontact three-dimension (3D) profile measurement has attracted increasing research efforts due to its distinct advantages over contact methods. Among others, the Fringe Pattern Profilometry (FPP) based on Digital Fringe Projection (DFP) has been proven to be one of the most promising techniques. Compared with the other methods, it has the advantages of simple system structure and high accuracy. Hence it provides a much more flexible and practical approach for 3D profile measurement.

A number of FPP approaches have been introduced. The most widely used methods are the Phase Difference Estimation (PDE). In these approaches, the deformed fringe pattern is considered as the result of phase modulation of the original fringe pattern, and hence detection of phase maps from original and deformed fringe patterns enables the retrieval of the 3D shape. Although the PDE approaches have been considered as the most popular, they suffer from a number of disadvantages. A major restriction is that fringe patterns must be either sinusoidal or ideal periodic. However, such a requirement is hard to meet in practice due to some factors, such as the nonlinear distortion on inherent to digital video projections. In order to solve the problem, a new profilometry approach is proposed. Instead of detecting the differences between the phase maps, the technique is based on the estimation of spatial shift for corresponding pixels on the two fringe patterns, and hence is called Spatial Shift Estimation (SSE) profilometry approach.

This research aims to implement SSE approach. The proposed initiative is facilitated through the design of a DFP system, development of solutions for problems existing in SSE approach, and furthermore by the proposal of new fringe pattern projections to improve the efficiency.

This chapter gives the general idea and background knowledge for this thesis, which is organized as follows. Section 1 provides the introduction for 3D profile measurement and DFP. Section 2 presents a comprehensive literature review which introduces the background of DFP system, and FPP approaches, including the PDE approaches and SSE approaches. Section 3 discusses the existing issue associated with the implementation of SSE approaches. Section 4 points out the aim of the thesis work. Section 5 demonstrates the contribution of research work. Section 6 gives the structure of this thesis.

1.1 General Introduction

1.1.1 Three-Dimensional profile measurement

Non-contact technique for 3D profile measurement has attracted increasing research efforts due to many applications, such as machine vision, animation, intelligent robot control, virtual reality, industrial monitoring, biomedicine, dressmaking and ergonomics, etc. Over the past few decades, numbers of optical methods for 3D profilometry have been proposed [1]. Some significant approaches include: Photogrammetry methods [2], such as Stereovision [3, 4], Shape from Shading [5], Shape from Texture [6] and Shape from Focusing [7, 8]; Interferometric methods including Two or Multiple Wavelengh [9, 10, 11, 12], Reactive Index [13, 14, 15] and Illumination Direction/Number of sources [16, 17, 18]; Moiré methods [19, 20]; Time-of-Flight method [21, 22]; and Structured Light methods [23] such as Code Structure Light [24], Laser Scanning [25] and Fringe Projection approaches. Among others, structure light methods including fringe profilometry became one of the most popular techniques, especially since the recent progresses in DFP technology have provided the required attributes for the development of dynamic and more robust structured light methods.

1.1.2 Three-Dimensional profile measurement using Digital Fringe Projection

Compared with other structured light methods, Fringe Pattern Profilometry (FPP) based on Digital Fringe Projection (DFP) is particularly attractive due to the advantages of simple system structure and controllable fringe patterns. Figure 1.1 shows the system structure of a DFP, consisting of a digital video projector, a CCD camera and a reference plane. With the system, a frame of image with a particular fringe pattern is produced by the digital projector and projected onto the reference plane, and then onto the surface of the object when the reference plane is removed. The projected images from the reference plane and the object surface are captured by the CCD camera, with the later being a deformed version of the former by the variance of the height of the object surface. As the deformed fringe pattern carries the information of surface shape, 3D profile of the object can be retrieved from these two fringe patterns. The two most promising FPP approaches are Phase Difference Estimation (PDE) profilometry approach and Spatial Shift Estimation (SSE) profilometry approach.

1.2 Literature Review

1.2.1 Introduction

The literature review provides an overview of the background knowledge of the FPP approaches, including the PDE approaches and SSE approaches. In this section, we firstly introduce the fundamentals of FPP approaches. Then we give a review of the existing PDE approaches, especially the FTP, PSP approaches. Finally, we introduce the SSE approaches.

1.2.2 Fringe Projection Profilmetry based on Digital Fringe Projection

In recent years, fringe pattern profilometry (FPP) has attracted increasing research

efforts as an enabling technology for non-contact measurement of 3D object surfaces. Among various system implementation schemes for FPP, the one based on digital fringe projection is particularly attractive due to the advantages of simple system structure and controllable fringe patterns.

FPP is based on the triangulation principle described as follows. As the image produced by the projector has a fringe structure, without loss of generality we can assume that light intensity varies periodically along *x* direction, while keeping constant along *y* direction, as shown in Figure 1.1. We can use $s(x)$, $d(x)$ and $h(x)$ to denote the variance of light intensity of the fringe pattern on the reference plane, object surface, and the height distribution along *x*-coordinate respectively. We also assume that the reference plane and the object surface have the same reflective characteristics.

Let us consider what happens when a beam of light is projected onto the point D on the object. From Figure 1.1, we can see when the object is removed, the same light beam (hence with the same intensity) should be projected onto point H on the reference surface, which is reflected back to the camera through point C. As the triangles E_cE_PH and CDH are similar, we have the following relationship:

$$\frac{d_0}{l_0} = \frac{\overline{CD}}{h(x_d)} \tag{1.1}$$

Note that x_d denotes the coordination positions of point D. $h(x_d)$ denotes the distance between points C and the reference plane, given by:

$$h(x_d) = \frac{l_0 \overline{CD}}{d_0} \tag{1.2}$$

The above relationship gives the foundation for FPP.

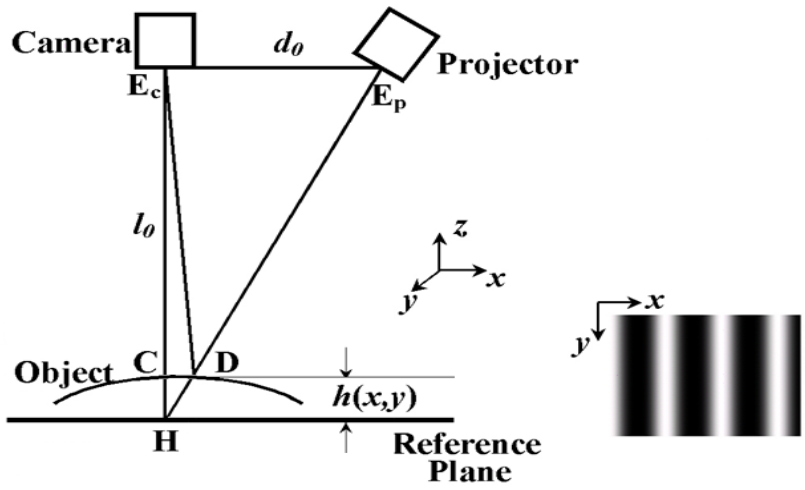

Figure 1.1 Schematic diagram of DFP system

1.2.3 Phase Difference Estimation Profilometry

A number of FPP approaches have been introduced. The most widely used approaches are the Phase Difference Estimation (PDE) profilometry approaches. In these approaches, the deformed fringe pattern is considered as the result of phase modulation of the original fringe pattern, and hence detection of phase maps from original and deformed fringe patterns enables the retrieval of the 3D shape.

In detail, these approaches utilize fringe patterns that are periodic and can be expressed as [26, 27]:

$$s(x) = \sum_{k=0}^{+\infty} b_k \cos(2\pi k f_0 x + \psi_k) \tag{1.3}$$

and the deformed fringe pattern can also be expressed as:

$$d(x) = \sum_{k=0}^{+\infty} b_k \cos(2\pi k f_0 x + \phi_k(x) + \psi_k) \tag{1.4}$$

In the above equations, f_0 is the spatial frequency of the fundamental component in the fringe patterns, and b_k is the amplitude of the *k*-th order harmonic component. ψ_k is the initial phase of the *k*-th order harmonic component, and $\phi_k(x)$ denotes the

phase difference between the *k*-th order harmonic components of these two fringe patterns.

Equations (1.3) and (1.4) show that $s(x)$ and $d(x)$ are related by the phase shift $\phi_k(x)$. Let us consider the light beam projected at point D on the object and H on the reference plane when the object is removed. The phase shift between C and D can be determined by the spatial distance $\overline{CD}$, and hence we have [26, 27]:

$$\phi_k(x_d) = 2\pi k f_0 \overline{CD} = k \cdot 2\pi f_0 \overline{CD} = k \cdot \phi(x_d) \tag{1.5}$$

where $\phi(x_d) = 2\pi f_0 \overline{CD}$ is the phase shift of the fundamental component.

Substituting Equation (1.5) to Equation (1.2) we have:

$$h(x_d) = \frac{l_0 \phi(x_d)}{2\pi f_0 d_0} \tag{1.6}$$

As points D and H are arbitrary, the above derivations should apply to all the points on the projected fringe pattern. Therefore, we have:

$$h(x) = \frac{l_0 \phi(x)}{2\pi f_0 d_0} \tag{1.7}$$

Equation (1.7) shows that as long as $\phi(x)$ can be detected, we are able to calculate the height distribution $h(x)$ of the object surface. This is the foundation of all PDE based approaches.

A number of fringe pattern analysis methods have been developed, such as Fourier Transform Profilometry (FTP) [28], Phase Shifting Profilometry (PSP), Phase Measuring Profilometry (PMP) [29, 30, 31], Modulation Measurement Profilometry (MMP) [32], Spatial Phase Detection (SPD) [33, 34], Phase Lock Loop (PLL) profilometry [35], Moiré Technique (MT) [36], laser triangulation measurement [37], colour-coded fringe projection [38, 39] and other methods [40, 41]. Among these

fringe pattern analysis methods, two of the most popular and typical algorithms for fringe pattern profilometry are FTP and PSP.

1.2.3.1 Fourier Transform Profilometry

Fourier transform profilometry (FTP) was firstly introduced by Takeda et al. [26, 27]. The concept of this approach is to analyse the fringe image by using the Fourier transformation.

From the Equation (1.3) and (1.4), we know that the $s(x)$ and $d(x)$ are composed of harmonic components. From Equation (1.7), it can be seen that the height distribution function $h(x)$ only associate with the fundamental component of the phase difference.

Hence the first step of FTP approach is applying a fast Fourier Transform (FFT) to the fringe image. A filter is then used to remove all signals except for the fundamental components in the spatial frequency domain. Finally, an inverse Fourier Transform (IFT) is applied to this filtered baseband signal.

After IFT, we have $\tilde{s}(x)$ and $\tilde{d}(x)$, the fundamental component of the $s(x)$ and $d(x)$, which can be expressed as:

$$\tilde{s}(x) = b_1 \cos(2\pi f_0 x + \psi_1) \tag{1.8}$$

$$\tilde{d}(x) = b_1 \cos[2\pi f_0 x + \phi(x) + \psi_1] \tag{1.9}$$

We use $\tilde{S}(x)$ and $\tilde{D}(x)$ to denote the complex signals of $\tilde{s}(x)$ and $\tilde{h}(x)$ respectively. Then we can retrieve the phase difference $\phi(x)$ by using the following equation:

$$\phi(x) = unwrap\ (\mathrm{Im}\{\ln[\tilde{D}(x) \cdot \tilde{S}^*(x)]\}) \tag{1.10}$$

where $\ln(\cdot)$ is the natural logarithm function, $\mathrm{Im}(\cdot)$ denotes the operation to get imaginary part of a complex number, $\tilde{S}^{*}(x)$ is the complex conjugate of $\tilde{S}(x)$. $unwrap\ (\cdot)$ is so-called phase unwrapping operation. In conventional PDE, the retrieved phase difference $\phi(x)$ is restricted to principle value ranging between $-\pi$ and π. Therefore, the estimated phase is a modulo 2π distribution. However, the true phase difference can be arbitrary. Thus a wrapping problem occurs. The phase unwrapping operation is used to solve the wrapping problem in PDE. The details of the phase unwrapping problem will be discussed in Chapter Three.

Accordingly, we can finally retrieve the height distribution function $h(x)$ by Equation (1.7). Figure 1.2 gives the flowchart of FTP.

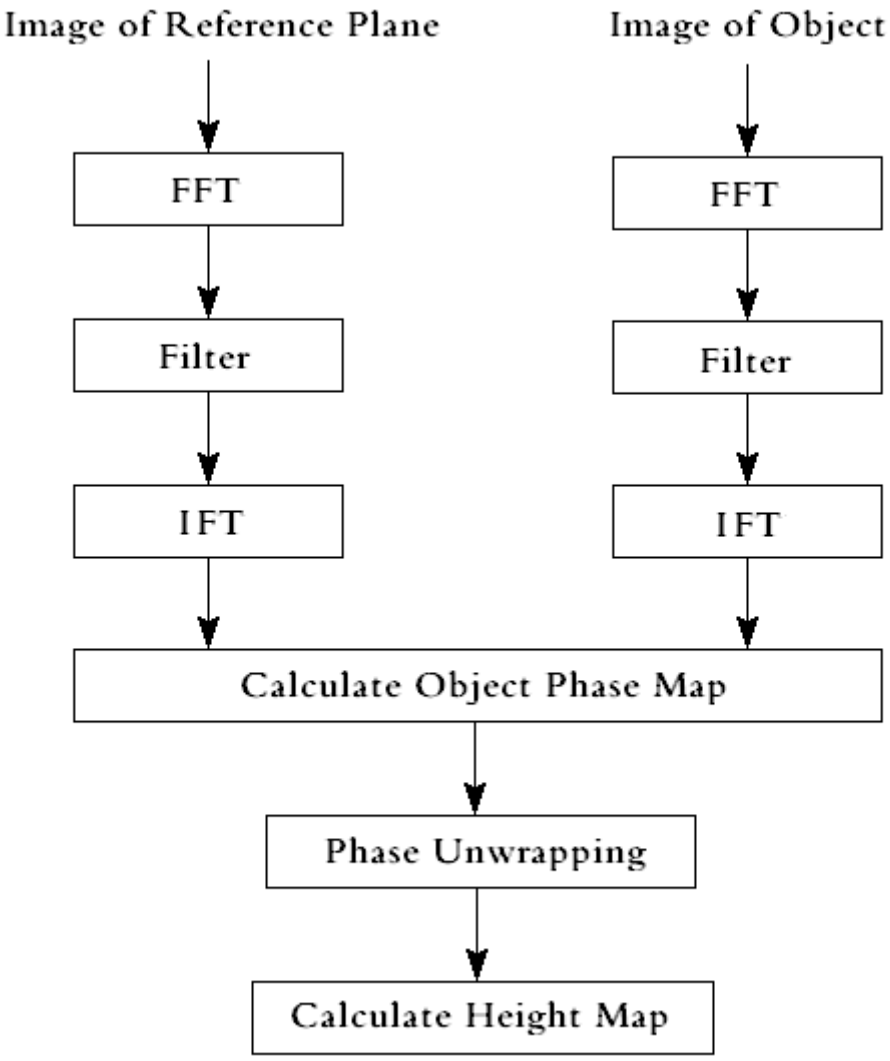

Figure 1.2 Flowchart of FTP processing

1.2.3.2 Phase Shifting Profilometry

Phase shifting profilometry is firstly introduced in [33]. In this approach, multiple frames of sinusoidal fringe patterns are projected onto the surface, and each frame shifted by a certain amount of phase angle. At least three images are needed but four or more are also popular for phase shifting profilometry.

The basic concept of this method is the use of triangle relationship. The first acquired image is sampled with the fringe pattern projected at a given position. The fringe

patterns are then shifted $2\pi / N$ from the previous frame of the patterns, where N is the number of phase shifting steps. Hence the fringe patterns on the reference plane and on the object surface can be respectively expressed as:

$$s_n(x) = \sum_{k=0}^{+\infty} b_k \cos(\, 2\pi k f_0 x + \psi_k + k\,\frac{2\pi n}{N}) \tag{1.11}$$

$$d_n(x) = \sum_{k=0}^{+\infty} b_k \cos[\, 2\pi k f_0 x + k\phi(x) + \psi_k + k\,\frac{2\pi n}{N}] \tag{1.12}$$

for n=0,1,2,…,N-1

Then we can calculate the phase map $\phi(x)$ by the following:

$$\phi(x) = unwrap\ (\arctan\ D_N) - unwrap\ (\arctan\ S_N) \tag{1.13}$$

where S_N, D_N are intermediate variables, given by:

$$S_N = -\left[\frac{\sum_{n=0}^{N-1} s_n(x)\sin(\, 2\pi n / N)}{\sum_{n=0}^{N-1} s_n(x)\cos(\, 2\pi n / N)}\right] \tag{1.14}$$

$$D_N = -\left[\frac{\sum_{n=0}^{N-1} d_n(x)\sin(\, 2\pi n / N)}{\sum_{n=0}^{N-1} d_n(x)\cos(\, 2\pi n / N)}\right] \tag{1.15}$$

Therefore, the height distribution function $h(x)$ can also be retrieved by using Equation (1.7). Figure 1.3 shows the flowchart of PSP processing.

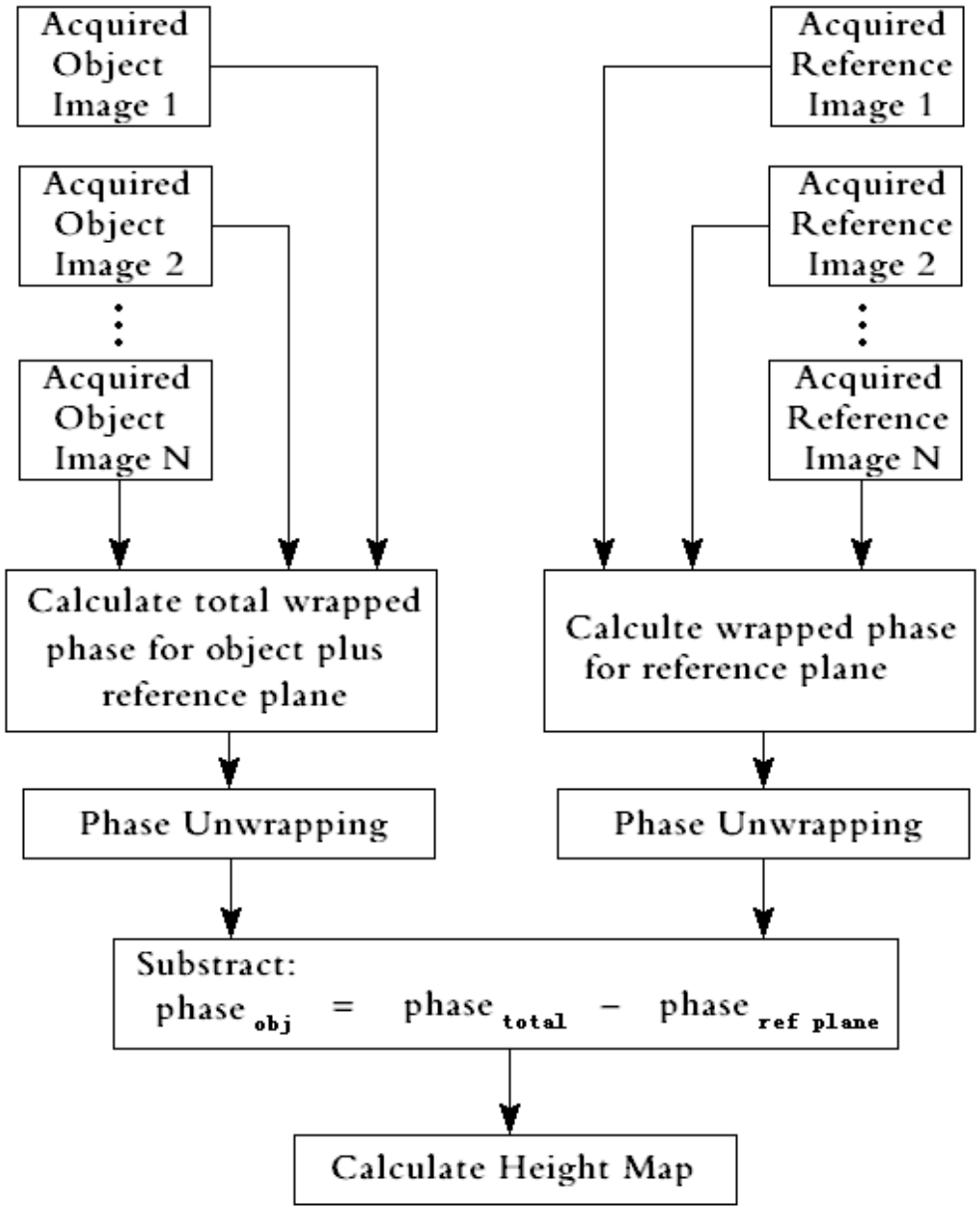

Figure 1.3 Flowchart of PSP processing

Compared with FTP, PSP has higher accuracy of measurement and the ability to combat random noise in the fringe patterns. Since this method uses multiple frames of images to calculate the height distribution, it can eliminate the noise by using many related images. This approach does not require FFT and IFT but only some simple calculations. However, the disadvantage of PSP method is also obvious. In PSP approach, at least three images are needed to retrieve the object height information, which takes longer time for 3D measurement.

1.2.4 Spatial Shift Estimation Profilometry

The PDE based FFP methods suffer from some limitations. In particular, the fringe pattern must be periodic so that the phase maps of $s(x)$ and $d(x)$ exist, and can be detected. However, due to many undesired factors inherent to digital projection, such as geometrical distortion and nonlinear intensity distortion, pure sinusoidal fringe patterns are hard to acquire. In order to solve the problem, a profilometry approach is proposed by Hu, *et al*. [42, 43], which is based on the estimation of spatial shift for corresponding pixels on the two fringe patterns instead of detecting the differences between the phase maps. This approach is referred to as Spatial Shift Estimation (SSE) profilometry approach [42, 43].

1.2.4.1 Spatial Shift Estimation

The idea of SSE based approach is rather simple and straight forward. Let us consider $\overline{CD}$, the distance between C and D again, which is obviously a function of the location of D (i.e. x_d ,), the location of H (or C, i.e., x_c) and the height of the object at point H $h(x_d)$. Therefore, we have the following:

$$\frac{d_0}{l_0} = \frac{u(x_d)}{h(x_d)} \tag{1.16}$$

where $u(x_d) = \overline{CD} = x_c - x_d$, which is the spatial distance between x_d and x_c. Note that x_d and x_c are the points on $d(x)$ and $s(x)$ with the same light intensity, that is $d(x_d) = s(x_c)$. As the above derivation is valid for any x_d and x_c, we can replace x_d by x, yielding the following:

$$h(x) = \frac{l_0 u(x)}{d_0} \tag{1.17}$$

Note that $u(x)$ is the spatial distance between a point x on $d(x)$ and the corresponding point on $s(x)$ with the same light intensity, that is:

$$d(x) = s(x - u(x)) \tag{1.18}$$

Equations (1.17) and (1.18) provide a straight forward way to obtain the 3D profile of the object surface. With $d(x)$ and $s(x)$ available, if we are able to calculate $u(x)$ to satisfy Equation (1.18), then we can utilize Equation (1.17) to yield $h(x)$, the height distribution of the object surface along *x*. By repeating the procedure for all *y* we are able to obtain the 3D profile of the object surface.

The spatial shift based approach has a particular advantage. Namely, the projected fringe patterns are no longer required to be sinusoidal nor periodic, which implies that even there are distortions with the fringe patterns, sufficient three-dimensional information on the object surface is contained in the variation between projected and deformed fringe patterns. Thus the profilometry can be achieved.

1.2.4.2 Direct Shift Estimation Algorithm

As discussed above, the key to reconstruct object surface using SSE is to obtain the shift distribution $u(x)$ from $d(x)$ and $s(x)$. Based on curve-fitting technique, Hu *et al.* [44] introduced a direct shift estimation method called Inverse Function based Shift Estimation (IFSE) Method. The idea of this method is the use of inverse function.

Suppose the projected signal function $r = s(x)$ is a monotonic function, or it is monotonic in intervals of *x*, in which $s(x)$ has a unique inverse function $s^{-1}(r)$, where:

$$s^{-1}(r) = x \tag{1.19}$$

Applying this inverse function $s^{-1}(r)$ to the deformed signal $d(x)$, we then have:

$$s^{-1}(d(x)) = s^{-1}\{s[x-u(x)]\} = x-u(x) \tag{1.20}$$

the shift distribution function $u(x)$ can be retrieved by:

$$u(x) = x - s^{-1}(d(x)) \tag{1.21}$$

Using Equation (1.21), the shift distribution function $u(x)$ can be easily calculated through the fringe patterns on the reference plane and object surface. Now the key problem is to obtain the inverse function $s^{-1}(r)$. In order to solve this problem, Hu introduced a method using the polynomial curve fitting. However, errors will be consequentially introduced during the fitting. This kind of errors is defined as a curve fitting error e_f, which is evaluated by the mean square error:

$$e_f = E[(y_f(x) - y(x))^2] \tag{1.22}$$

where $E(w)$ denotes the operation to calculate the mean value of w, $y_f(x)$ are the values of the curve fitting results calculated by the approximate polynomial and $y(x)$ are the data to be fitted.

To reduce the curve fitting error, one solution is to increase the degree of polynomial for fitting. However, the degree selected too high will result in heavy computation complexity. So we need to set up an upper bound of the fitting error e_f, and then calculate the minimum degree of polynomial which makes e_f less than the bound we have set. The procedure can be described as follows:

Step 1: Set e_m, which is an upper bound of curve fitting error, and *k*, the degree of polynomial used for curve fitting. We initialize the starting value of *k* equals to 1.

Step 2: Work out j_k, which is the polynomial of degree *k* to approximate the inverse function $s^{-1}(r)$ in least squares sense. In this step, we first obtain a symmetrical curve of $s(x)$ in each monotonic interval by using the straight line $x = r$ as a symmetry axis. Actually this curve is just the inverse function $s^{-1}(r)$, therefore we can calculate the curve fitting result j_k by making curve fit to the obtained symmetrical curve.

Step 3: Calculating curve fitting error e_f by the Equation (1.22), where $y_f(x)$ is approximated by using the curve fitting result j_k. If $e_f \le e_m$, go to Step 4; otherwise $k = k + 1$ and return Step 2.

Step 4: Using the curve fitting result $s^{-1}(r) \approx j_k$ and the value of deformed signal $d(x)$, calculate the shift distribution function $u(x)$ by Equation (1.21).

Note it has been proven that j_k which satisfies $E[(j_k(r) - s^{-1}(r))^2] < e_m$ always exists [45].

Figure 1.4 shows the flowchart of IFSE processing. The same as FTP and PSP, an unwrapping operation is needed before we retrieve the height information. However, one thing should be noticed is that different from FTP and PSP, the $unwrap(\cdot)$ is no more phase unwrapping operation but a new operation which is referred as shift unwrapping. This unwrapping is based on shift estimation, and will be detailed in Chapter Three.

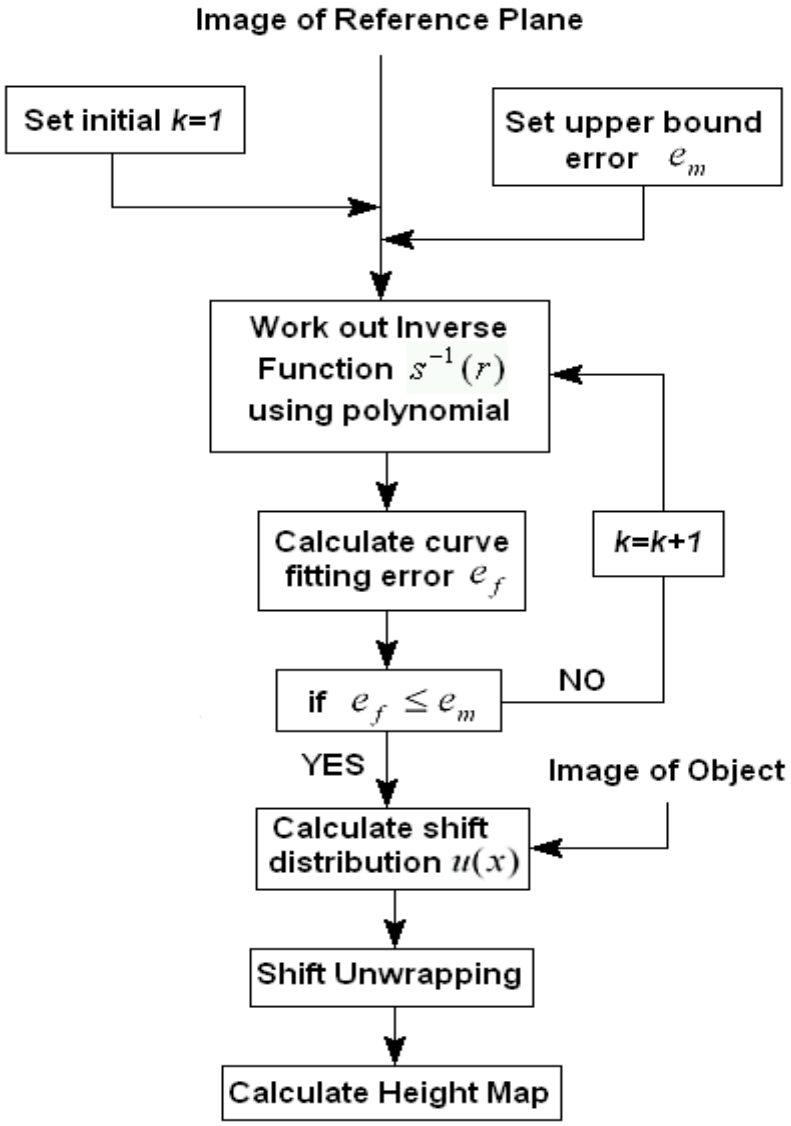

Figure 1.4 Flowchart of IFSE processing

The IFSE Method has two advantages. Firstly, since the curve fitting method is used to obtain the inverse function, the structure of the projected fringe pattern is no longer required to be sinusoidal. Secondly, because this method is based on the projected fringe patterns directly, it can be used even when the reference plane has been distorted by unknown effects.

1.3 Existing Issues

Let us now look at the existing research work of the area of the implementation of SSE approach. Through evaluating the strengths, weakness and the gaps of other researchers' work, the following existing issues have been noticed.

Implementation of SSE approach

According to the theory, SSE based FPP approach has been considered as a promising approach in the area of 3D profilometry. However, few practical systems have been demonstrated to implement this approach. Problems associated with the implementation still exist. Therefore, in order to study the performance of SSE approach and develop solutions of these problems, the implementation of SSE approach based on a DFP system is needed.

Shift unwrapping in SSE approach

For PDE-based approaches, the phase difference can only be detected within the main value range of $[-\pi, \pi]$, but the true phase difference can be arbitrary, hence a wrapping problem exists. In order to retrieve the actual surface shape of the object, phase unwrapping must be carried out to obtain the actual phase maps. Many methods have been proposed to solve this wrapping problem in PDE approach. In the SSE-based approaches, spatial shift between corresponding pixels on the two fringe patterns can also be arbitrary, but it only can be detected without ambiguity within the range of [0, T_0], where T_0 is the width of the individual fringe. Obviously, shift unwrapping is also required in order to restore the 3D shape of the object surface correctly. However, spatial shift unwrapping for SSE-based FPP is still an outstanding issue, which motivates the work presented in this thesis.

The efficiency of SSE approach

In most cases, the projected fringe pattern is selected to be sinusoidal. However, the

usage of sinusoidal fringe pattern faces many limitations, such as the vulnerability to geometry distortion and nonlinear intensity distortion. To solve the problems, additional filter is needed, which increases the computation complexity. Sinusoidal fringe pattern is also not fit for IFSE method in SSE approach for its long computation time. Hence improvement is needed to increase efficiency. Since the SSE approach has the advantage that projected fringe patterns are not required to be either sinusoidal or periodic, it is important to design a fringe which has strong counter-interference capability against the distortions and short computation time in IFSE method.

1.4 Aim of the thesis work

This thesis work aims to solve the following research issues associated with DFP systems:

- Design a DFP system, including hardware setting and software program designing. Then implement the SSE approach by using this system.
- Develop solutions for unwrapping problem in SSE approach: The unwrapping problem is a major problem in the retrieving of height distribution information. Many methods have been proposed to solve the unwrapping problem in PDE approach. However, these approaches are not able to solve the unwrapping problem in SSE approach.
- Increase the efficiency: The usage of sinusoidal fringe pattern in SSE approach suffers many limitations and all those results the long time in computation. Hence to find a solution to reduce the computation time and improve the efficiency is also important.

1.5 Contribution of the Thesis Work

In this thesis, we have made follow contributions:

A DFP system to implement the SSE approach

A DFP system model based on the principle of fringe pattern profilometry is proposed. This system consists of two sub-systems: Projection and Acquisition. A software interface is designed to control this system. Performance of SSE approach is then evaluated by implementation using this system. Solutions of the problems associated with implementation are also developed.

A solution to the unwrapping problem in SSE approach

The unwrapping problem is a key problem in FPP. It causes the discontinuity of the retrieving height information. After theoretical analysis, we indicated not only in conventional PDE approach but also in SSE approach, the unwrapping problem exists and prevents us to retrieve the correct height information. Therefore, this problem must be solved. Based on the theory of phase unwrapping methods, we introduce a spatial shift unwrapping method in order to solve the unwrapping problem in SSE approach. We also carry out experiments to test the performance. Results have been given to evaluate the effectiveness of the proposed unwrapping technique.

Use of new fringe pattern to improve the efficiency

In traditional FPP, the sinusoidal fringe pattern is selected as the projected fringe pattern. However, the usage of sinusoidal fringe pattern faces several limitations, such as the vulnerability to nonlinear intensity distortion and unsuitable for IFSE method. These limitations decrease the efficiency by introduced additional computations. In SSE approach, the projected fringe patterns are no longer required to be sinusoidal. Therefore, we develop a new fringe pattern, which is the sawtooth fringe. We analyse the advantages of this sawtooth fringe through several rules and compare it with the

sinusoidal fringe pattern in theory. Experiment is then performed. Detailed result data is presented to prove the improvement on efficiency, which introduced by using this new fringe pattern.

1.6 Organization of thesis

The remainder of this thesis is organized as follows: Chapter Two describes the procedure to implement the SSE approach using a DFP system, which includes an illustration of DFP system model and the implementation of SSE approach. Chapter Three presents the introduction of spatial shift unwrapping problem and a solution to address it. Chapter Four firstly discussed the disadvantages of traditional sinusoidal fringe pattern. Some rules of new fringe design such as simplicity, adjustable in period and monotonic are then introduced. Based on these rules, the sawtooth fringe pattern is chosen.

Chapter 2 Implemention of SSE Approach

2.1 Introduction

In Chapter One, we reviewed the existing FPP approaches. We find that the study of conventional PDE includes both theoretical research and applications. Compared with conventional PDE, the SSE approach is a new FPP which only has been introduced recently. The research about this method only contains theoretical studies. Hence a detailed instruction about the implementation of SSE approach using DFP system is required. In this chapter, the implementation of SSE approach is described, which contains the system introduction and the procedure of implementation, including problem encountered and the solutions.

This chapter is organized as follows: Section 2.2 presents the details of DFP system. This section includes both hardware and software. The hardware section gives the configuration of the system, and the software section presents the design of software interface for generating and controlling the fringes. Section 2.3 elaborates the implementation of SSE approach, which includes pre-processing of captured image, IFSE and object reconstruction. Section 2.4 provides the conclusion of this chapter.

2.2 DFP System Model

2.2.1 System Structure

The hardware of Digital Fringe Projection 3D sensing arrangement utilized for the implementation is depicted in the schematic diagram shown in Figure 2.1.

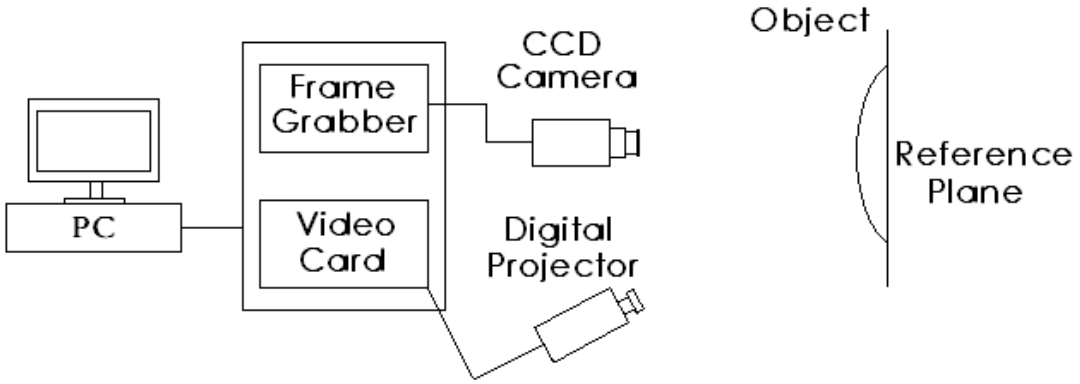

Figure 2.1 Digital Fringe Projection arrangement

From this figure, we can see a Personal Computer (PC) is used to manage both the projector and camera via the video and frame-grabber card respectively. This PC controls the projector and camera simultaneously using software, and further performs the processing of acquired data. The DFP system can be divided into two sub-systems: Projection and Acquisition. Each of these parts will now be further discussed.

Figure 2.2 shows the photo of our experimental setups. The distance of the digital camera to the projector and the distance between the camera lens and the reference plan are all adjustable. Hence the filed of vision for CCD camera can also be adjusted in order to adapt different sizes of objects.

Figure 2.2 The experimental system setup

2.2.1.1 Projection

In the projection sub-system, a digital projector connects the computer via a Matrox multiple head video card. We then use the standard operating system display setting to configure the projector as an independent display device. A “Coloreal Visual” software is accompanied with the Matrox card. This software allows precise calibration of each display output in terms of individual colour channel Display Gamma, Brightness and Contrast, hence to facilitate the appropriate configuration of the display outputs.

The projector we utilized in the system is a HITACHI CP-X260 3LCD projector. The specification for the projector is given in Table 2.1. A data sheet with details is given in [46].

Specification	HITACHI CP-X260
Technology	1.6 cm Poly-Si 3-LCD
Native Resolution	1024×768
Contrast Ratio	500:1
Brightness (ANSL Lumens)	2500

Table 2.1 Projector Specification

2.2.1.2 Acquisition

The acquisition sub-system should be specifically selected to accommodate for both single and multi-channel environments. In our system, a high resolution Duncan Tech MS3100 3-CCD camera is utilized for acquisition. This camera interfaces to the PC via a National Instruments IMAQ-1428 frame grabber card. A dichroic prism is used in this camera, and it contains three individual imaging channels with an independent resolution of 1392×1039 pixels at an 8 or 10 bit precision.

A specialized DTControl software developed by Duncan Tech is accompanied with this

camera. This software allows the control of various aspects of the device including:

- Individual channel gain
- Individual channel integration time and overall integration time
- Quantization Precision (8 or 10 bit)
- Triggering mode

The software also allows the display and record images and image data.

2.2.2 Software Interface

The software interface responsible for the integration of hardware components serves a distinct role: the generation and control of the fringes. In our system, the software is written in MATLAB. It produces the fringes in three individual colour channels: Red, Green and Blue. Each channel is controlled by three parameters:

- Contrast: This parameter defines the amplitude of generated sinusoidal fringe pattern. Usually we set 1.0 as the default value.
- Phase: This parameter controls the phase offset of generated sinusoidal fringe pattern. The default value is set to 0.
- Frequency: This value decides the frequency of produced sinusoidal fringe pattern.

The flowchart of the fringe generate software is given in Figure 2.3. In this software, we first generate three individual sinusoidal waveforms. The waveforms can be considerd as the colour intensity information of projecting fringe, and three waveforms stand for red, green and blue fringe controlled by different parameters which shown before, respectively. Then we combine them together and transfer the colour intensity information into a colour image.

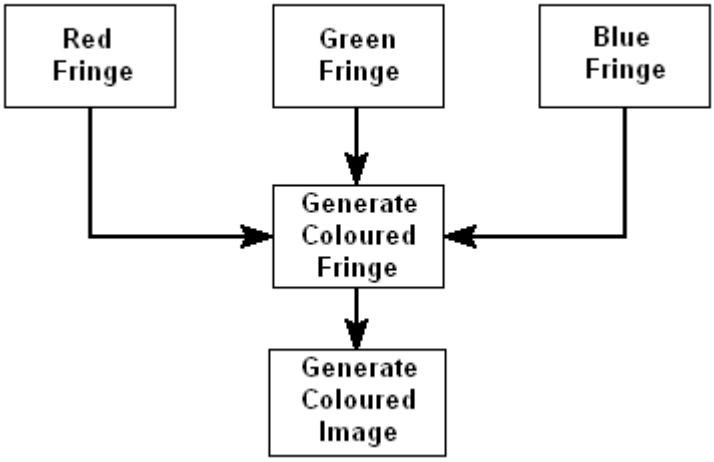

Figure 2.3 Flowchart of fringe generation

Figure 2.4 shows an example of a final produced colour fringe image. In this image, we set all the contrast, phase and frequency of the red, green and blue fringes at the same value. Hence a black-and-white image is produced.

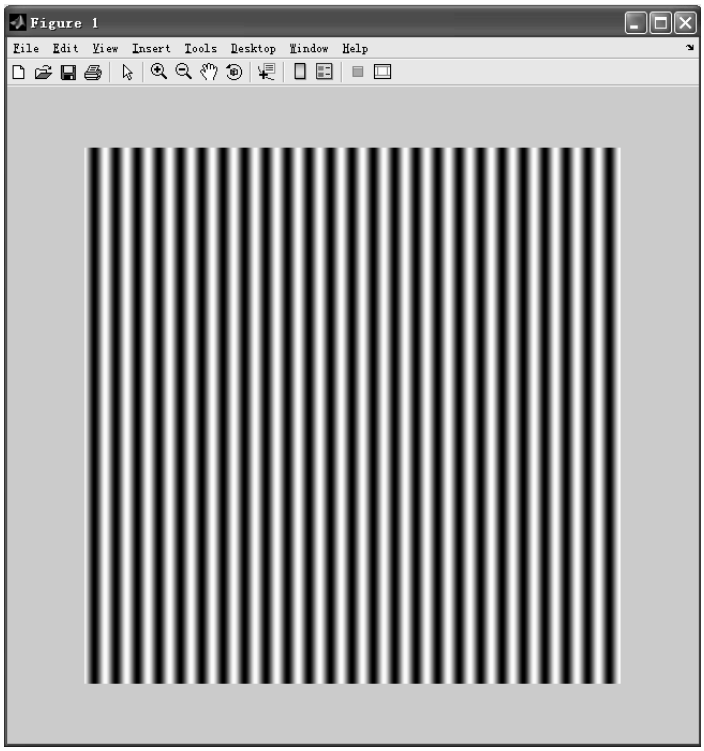

Figure 2.4 Fringe produced (Contrast = 1.0, Phase =0, Frequency = 500)

2.3 Captured Image Processing

As discussed in Chapter One, with SSE approach, we need to acquire two images, one on reference plane and one on object surface. In our experiment, a dome set on a flat board is used as the object, where the maximum height is 22.8mm. The diameter of the bottom surface of the dome is 99mm, and the thickness of the base board is 16mm.

Figure 2.5 shows the both images which on reference plane and one on object surface respectively. These photos are captured on this system setup: The digital camera is placed on top of the projector with a distance of 350 mm. The distance between the camera lens and the reference plan is 1295 mm. The resolution of the CCD camera is 1392×1039 pixels, and the filed of vision for CCD camera is 250mm×187mm. Hence, the equivalent spatial resolution is 0.1796 mm/pixel.

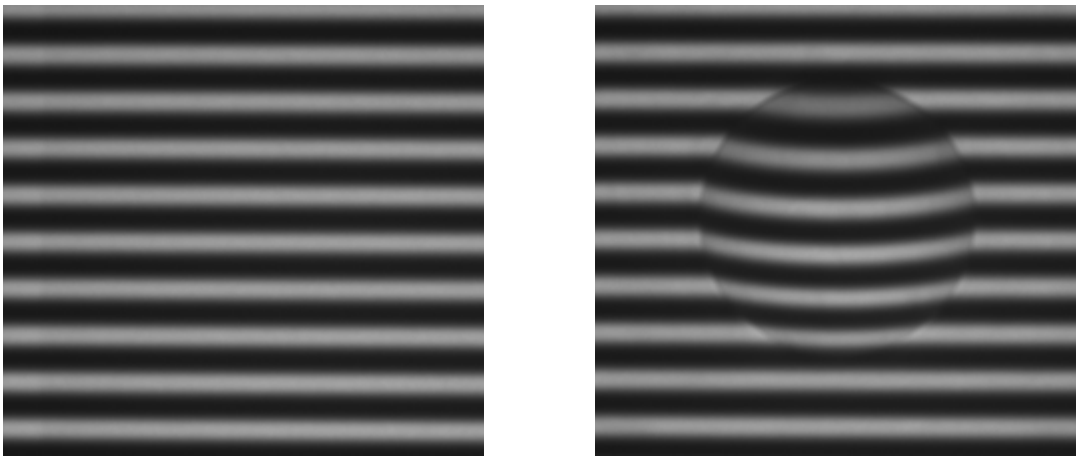

Figure 2.5 Captured image of reference plane (Left) and object surface (Right)

Image processing is performed to retrieve the height information. It is the main part of the whole SSE approach implementation and concerns the utilization of IFSE in application. The whole procedure includes pre-processing of captured image, height information retrieving and reconstruction of object surface plane.

2.3.1 Pre-processing of Captured Image

The first step of captured image process is retrieving the colour intensity information from the captured image. This can be easily performed by using MATLAB. However, the captured images suffer from distortion and noise introduced by the equipments and environment. Thus before we perform the IFSE procedure, a pre-processing to eliminate or reduce the distortion and noise for the captured image is required.

The first problem is the noise. As introduced in Section 1.2.4.2, the condition to perform IFSE requires the retrieved signal function $r = s(x)$ to be a monotonic function, or be monotonic in intervals of x. In theory, the fringe retrieved from image of reference plane should be an ideal sinusoidal. It is monotonic in each half of its period. However, in practice, with the effect of noise, the monotonic cannot be achieved.

Figure 2.6 shows the acquired fringe data from reference plane. For illustration, we select one line from each captured images, where $y1 = 750$, which is the middle of the dome object. From this figure, we can see the retrieved signal in each half of its period is not monotonic due to noise.

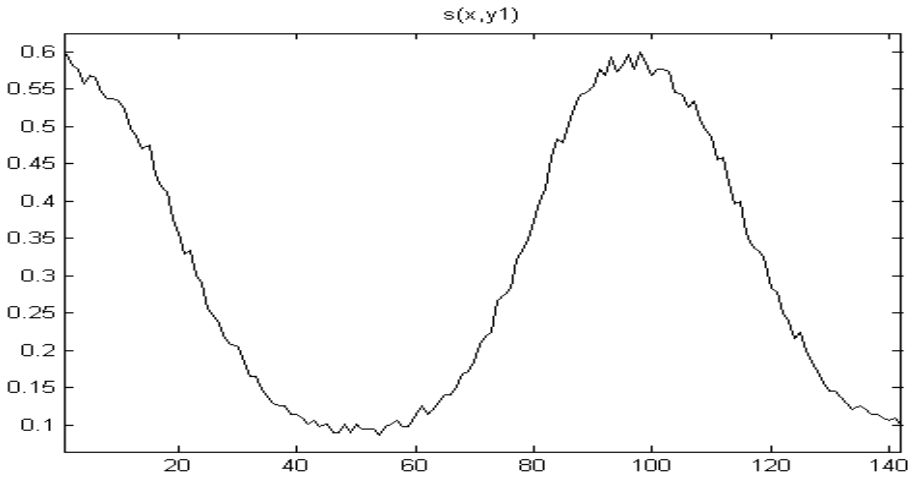

Figure 2.6 Acquired fringe data from reference plane

Besides the noise, there is another problem we need to solve. Figure 2.7 shows the acquired fringe data from the reference plane and the object. Compare with the reference fringe in this figure, it is easy to find the height of the object fringe is distorted.

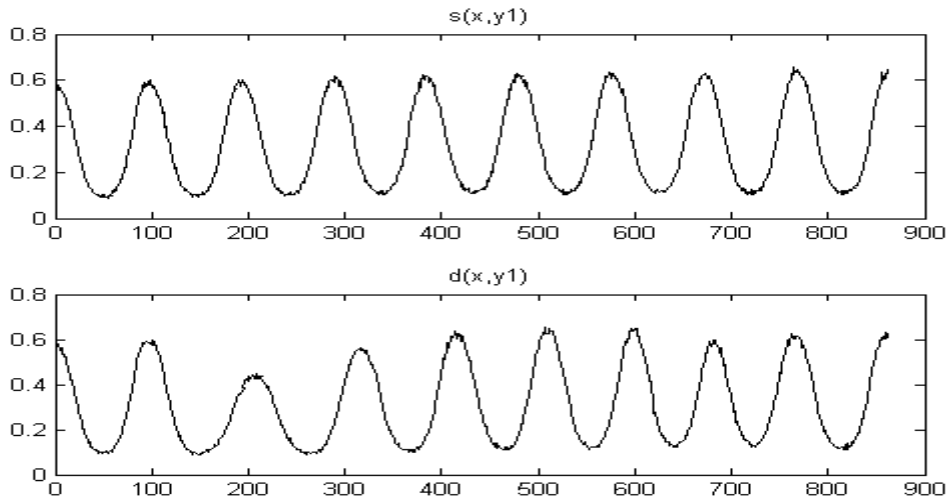

Figure 2.7 Acquired fringe data from captured image

After analysis, we find this distortion is caused by the shape of the object. From Chapter One, we know the fringes are projected onto the reference plane and object surface from a certain angle relative to the imaging optical axis. From Figure 2.8, it is easy to see, for the reference plane, the projection angles of each parallel ray are the same, hence reference plane have an equal intensity distribution when viewed from the camera. However, for object surface, the projection angles are different due to the shape of object. Therefore, the intensity distribution of object surface is unequal when viewed from the camera. This unequal intensity distribution causes the height distortion of object fringe.

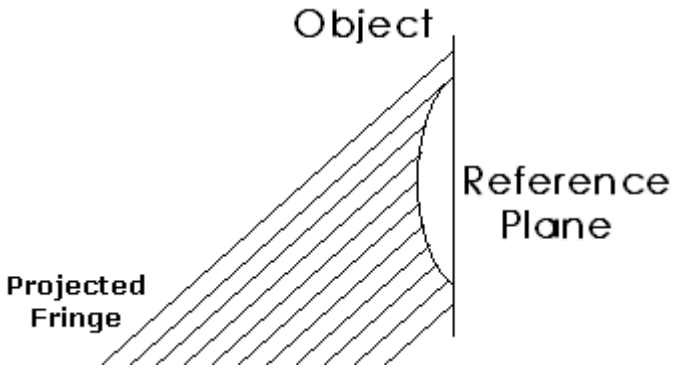

Figure 2.8 Projected fringes on object surface

In IFSE, we obtained the inverse function $s^{-1}(r)$ from the projected signal $s(x)$ and applied it to the deformed signal $d(x)$, hence the shift distribution function $u(x)$ can be retrieved using equation $u(x) = x - s^{-1}(d(x))$. However, this equation can be used only when $s(x)$ and $d(x)$ have the same height range. As the distortion changed the height range of the deformed signal $d(x)$, the IFSE cannot be performed until we eliminate the influence of distortion.

Thus in this part, we first design a filter to reduce the noise interference, and then propose a method to solve the distortion problem.

2.3.1.1 Filter Design

To eliminate the noise and distortion, we need to use a filter. From Figure 2.6, we find the noise is like a kind of high frequency signal. Hence a low-pass filter can be used to attenuate these high-frequency signals. The cutoff frequency should be precisely selected. Cutoff frequency too low will attenuate the fringe signal, cutoff frequency too high will result in the residua of noise.

According to Figure 2.7, it is obvious that the frequency of deformed signal $d(x)$ is

different from the frequency of the project signal $s(x)$ due to the height information of object surface. The frequency of $d(x)$ is higher than $s(x)$ when the slope of object is negative. Hence the cutoff frequency should be defined based on the frequency of deformed signal in order to prevent the lost of height information. Figure 2.9 presents the frequency spectrum of the deformed signal $d(x)$, where *x* label stands for the normalized coefficient with 1.0 corresponding to the half of sample rate, and *y* label is the normalized frequency response.

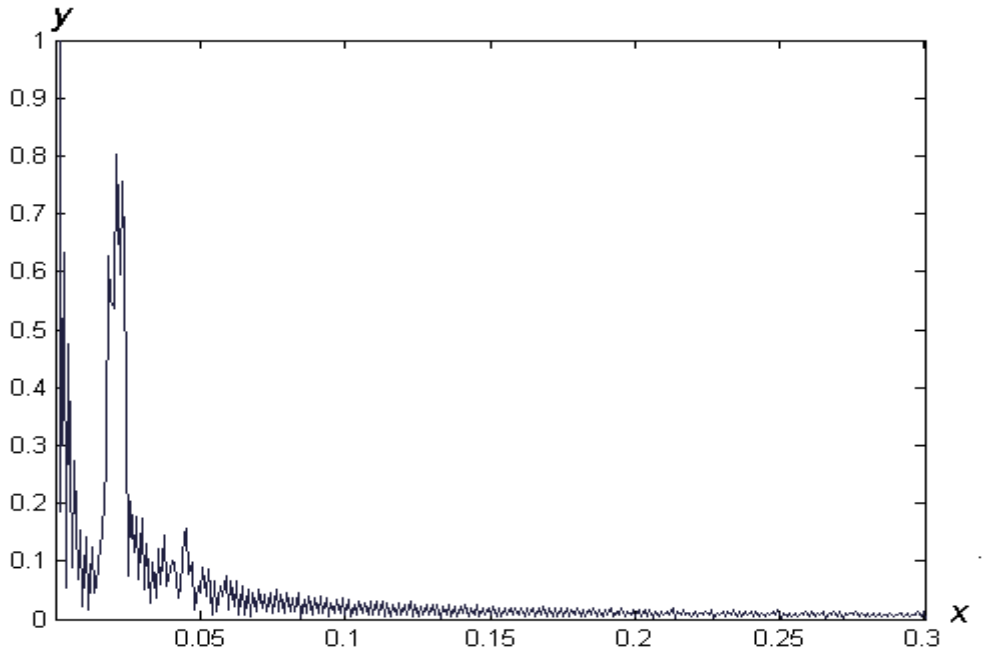

Figure 2.9 Frequency spectrum of deformed signal

It can be seen from this figure, when $x > 0.05$, the frequency response of deformed signal falls below 0.1 and keeps decreasing. It can be considered as the component of high frequency noise signal, to attenuate it, the cutoff frequency should be selected near $0.05 \times (SampleRate / 2)$.

In our experiment, a Finite Impulse Response (FIR) digital filter is used. Figure 2.10 shows the spectrum of the filter. After calculation and experiment, we finally set the

frequency breakpoints of this filter to $0.15\times(SampleRate/2)$. Then according to Figure 2.10, the cutoff frequency, which is the point where the frequency response at $\sqrt{1/2}$, is about $0.04688\times(SampleRate/2)$.

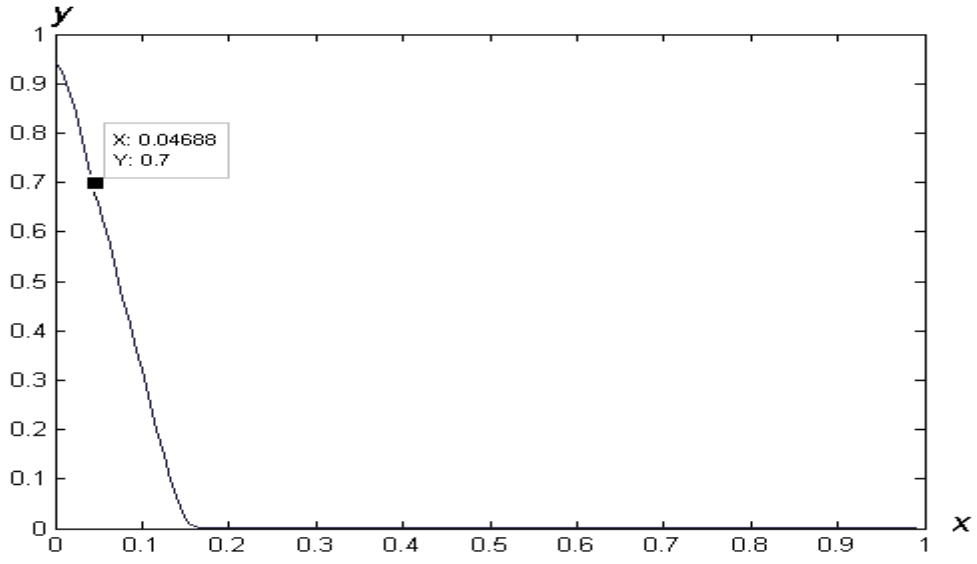

Figure 2.10 Frequency spectrum of designed filter

Figure 2.11 shows the fringe data after filter, and we can see the high frequency noise is removed successfully.

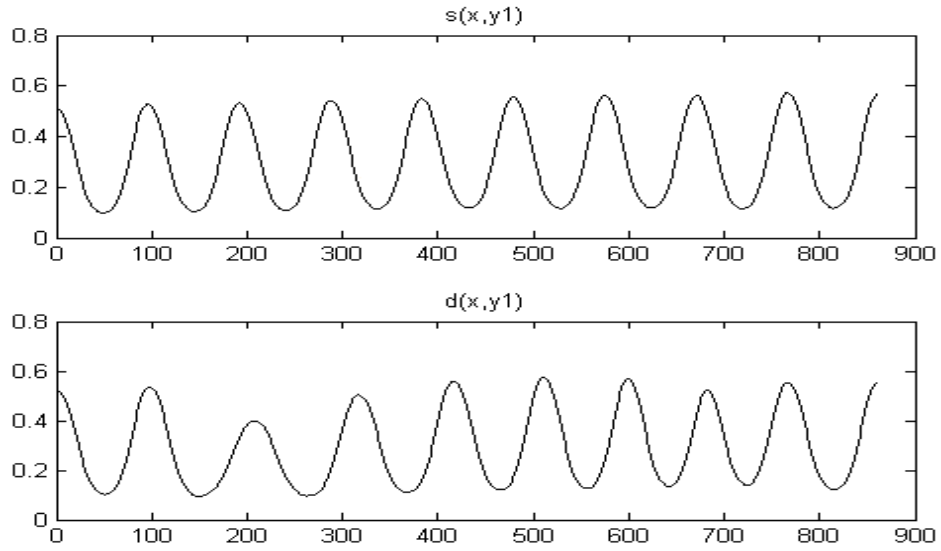

Figure 2.11 Acquired fringe data after filter

2.3.1.2 Normalization of Fringe

The distortion of deformed signal $d(x)$ is another problem we should solve. As mentioned before, this problem is introduced by the shape of object. After the analysis, we notice that the intensity distortion does no effect on the shift information, thus we only need to normalize the distortion in intensity to make the IFSE available for use.

The procedure of fringe normalization is very simple. First, we divide the projected signal $s(x)$ and deformed signal $d(x)$ into several monotonic intervals. Just like shown in Figure 2.12, we separate the $s(x)$ into the monotonic intervals: s_1, s_2, s_3… and $d(x)$ turns into d_1, d_2, d_3 and so on.

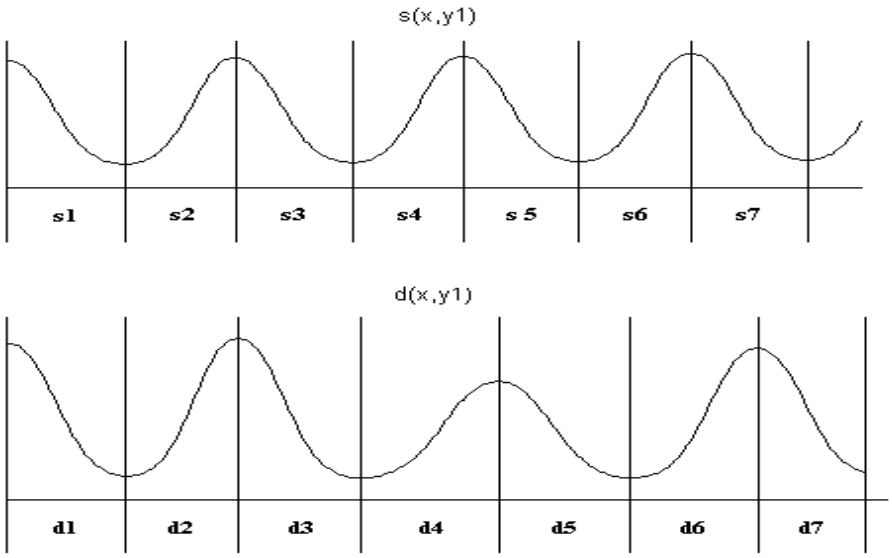

Figure 2.12 Fringe divided into monotonic intervals

Then we can perform the normalization processing on each monotonic interval by the following steps (suppose the intervals we selected are s_1 and d_1):

Step 1: Calculate the middle value of the selected monotonic interval in projected signal $s(x)$ and matched monotonic interval in deform signal $d(x)$ using these two equations:

$$s_1^{mid} = \frac{s_1^{\max} + s_1^{\min}}{2} \tag{2.1}$$

$$d_1^{mid} = \frac{d_1^{\max} + d_1^{\min}}{2} \tag{2.2}$$

where $s_1^{\max}$ and $d_1^{\max}$ are the maximum value in intervals s_1 and d_1 respectively, and $s_1^{\min}$ and $d_1^{\min}$ are the minimum value.

Step 2: Work out s_1^r and d_1^r, which are the height range of intervals s_1 and d_1, by using these equations:

$$s_1^r = s_1^{\max} - s_1^{\min} \tag{2.3}$$

$$d_1^r = d_1^{\max} - d_1^{\min} \tag{2.4}$$

Step 3: Normalize the height of d_1 use the follow equation:

$$d_1^{fix} = [(d_1 - d_1^{mid}) * (\frac{s_1^r}{d_1^r})] + s_1^{mid} \tag{2.5}$$

where d_1^{fix} is the normalized deformed fringe data.

The deformed fringe will be normalized after we perform these steps on each intervals of $d(x)$. Figure 2.13 shows the normalized fringe data.

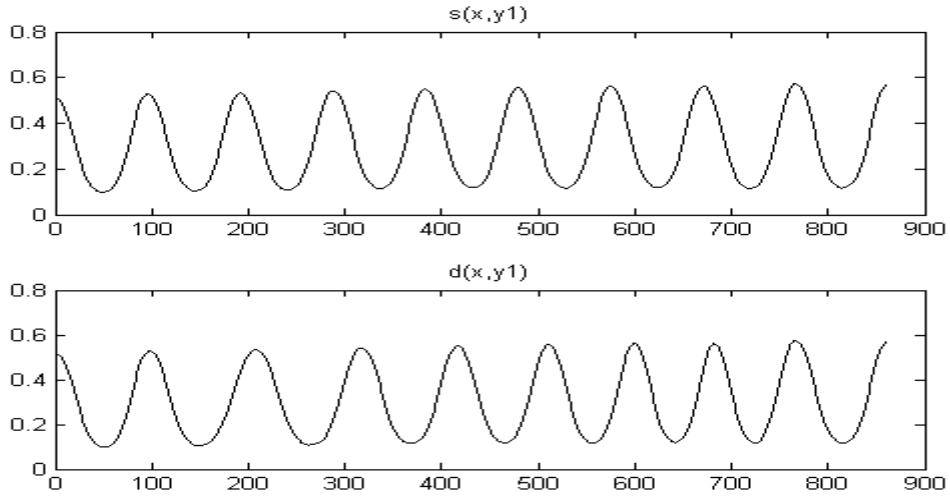

Figure 2.13 Normalized fringe data

2.3.2 Improved IFSE

As mentioned before, in SSE, we use the IFSE method to obtain the height information from the estimation of spatial shift. After the pre-processing on the acquired fringe data, all conditions for performing IFSE are satisfied. The whole procedure of IFSE is based on the theory introduced in Chapter One. However, during the implementation, problems still exist.

As described in Section 1.2.4.2, the first step of IFSE method is setting an upper bound of curve fitting error e_m, then repeat the curving fitting step until the curve fitting error e_f is smaller or equal to the bound. Then how to select the value of e_m becomes a problem. In our implementation, we find if we set this bound in a small value, it will consume too much time on calculation. However, when we increase the value, the time spend on calculating is reduced but the curve fitting error is increased. So we have to find a new solution.

One suggested solution is increasing the initial degree of polynomial used for curve fitting. As we know the reference fringe pattern is sinusoidal, it is a periodic pattern.

Hence the curve fitting polynomial for each monotonic interval should be the same in theory. However, in practice, there will be a slightly difference between each interval due to the noise and distortion. Thus we need to do the curve fitting for each interval, but degree of curve fitting polynomial can be considered varying at a small range. Then we can set the starting value of *k* near this range or just gives a fixed value of *k* which selected from the range. This method will reduce the computation time because we only need to calculate the range of *k* for one or two times instead of calculate it on every interval. Figure 2.14 shows the flow chart of this improved IFSE method.

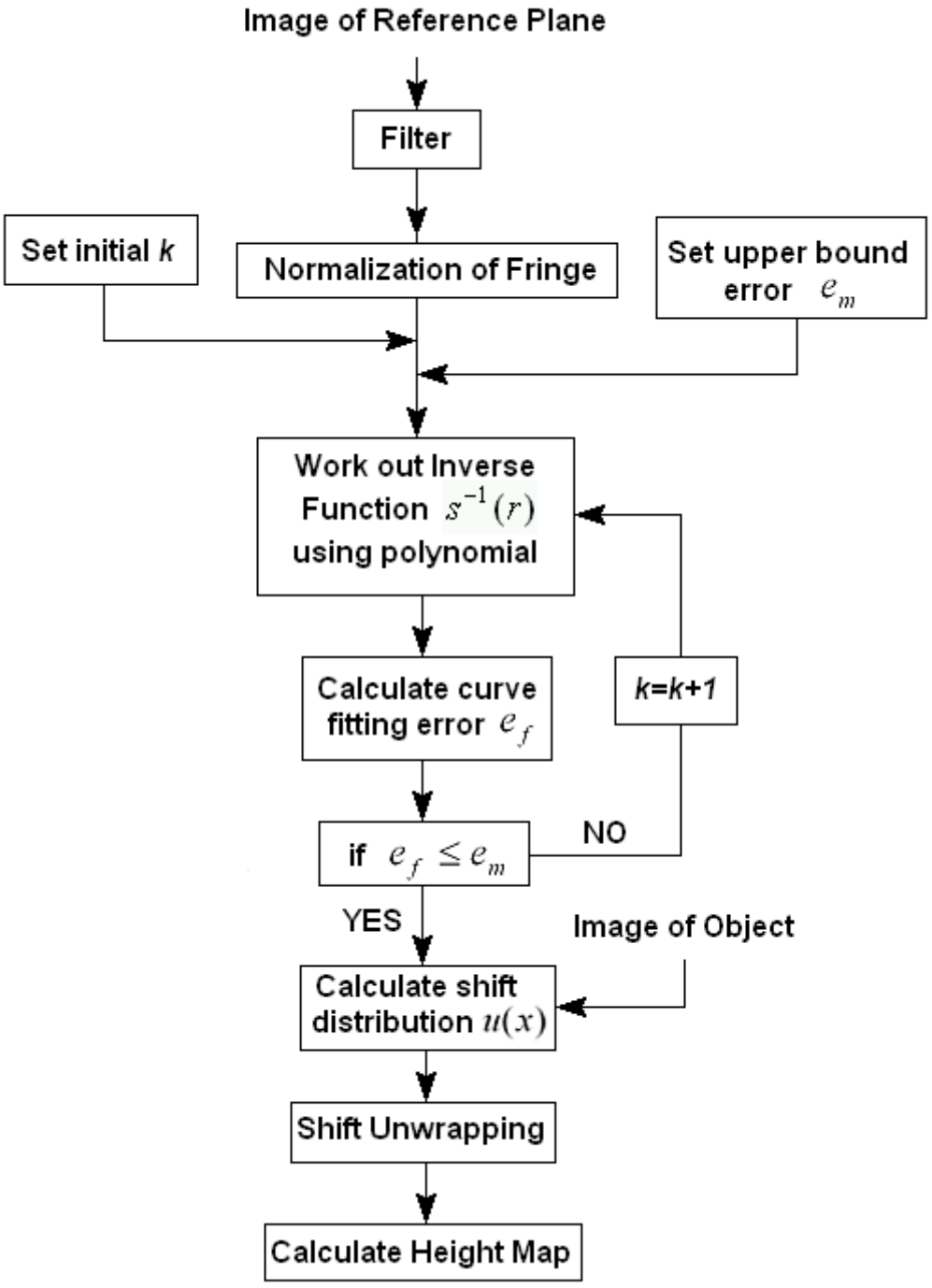

Figure 2.14 Flowchart of improved IFSE processing

Compared with the Figure 1.4 given in Chapter One, Figure 2.14 has two changes. First, before we perform the IFSE processing, two pre-processing parts are added. Second, the initial value of *k* is not limited to be 1, higher initial value can be used to

reduce the computation time. As shown in the flow chart, after unwrapping, we can retrieve the height distribution of the object.

2.3.3 Reconstruction of Object Surface

The final step of the implementation is the reconstruction of object surface. Using the improved IFSE method given before, we can retrieve the height distribution. We use Matlab to perform the reconstruct the 3D model of object surface. Figure 2.15 shows the final reconstructed object surface.

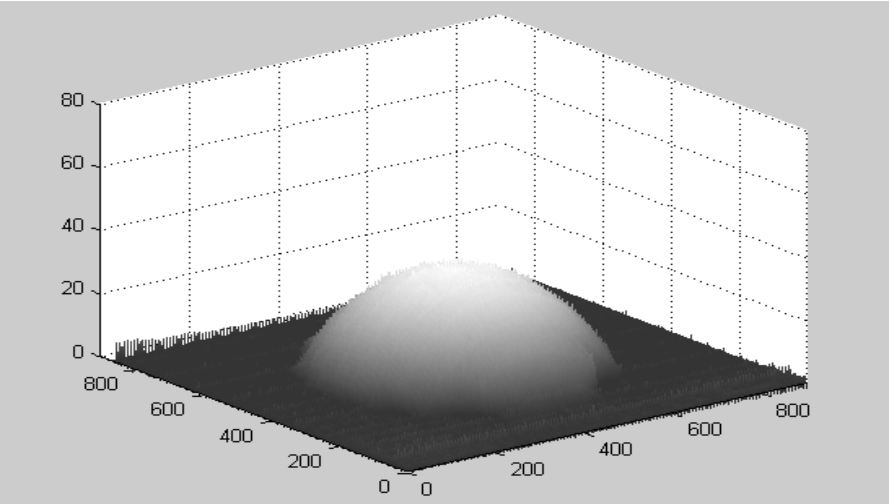

Figure 2.15 3D Reconstruct results

From Figure 2.15, we can see that dome surface is perfectly reconstructed and the result is satisfying. Thus we can say the implementation of SSE approach is successfully performed.

2.4 Conclusion

This chapter presented the implementation of SSE approach. First a DFP system is set up. Detailed information about this system is given, which includes the introduction of

system structure and software interface. Then the whole procedure of implementation is introduced. We emphasize the problems encountered in practice and propose the solutions, including pre-processing and improved IFSE method. At last, the result is shown to prove the contribution of our implementation.

Charpter 3 Shift Unwrapping in SSE Approach

3.1 Introduction

Both in PDE and SSE, the wrapping problem exists and becomes a major problem in the retrieving the height distribution information of object surfaces. For PDE-based approaches, the phases are limited within the main range from $[-\pi, \pi]$. In order to restore the actual shape of the object, phase unwrapping must be carried. In SSE approach, since most of the projected fringe patterns are also periodic, the shift can only be detected without ambiguity with the range of $[0, T_0]$, where T_0 is the width of the individual fringe. This will also result in discontinuity. Many methods have been proposed to solve the unwrapping problem in PDE approach. However, these approaches are not able to solve the unwrapping problem in SSE approach. Hence a solution to address the unwrapping problem existing in SSE approach is necessary.

This chapter is organized as follows: Section 3.2 presents a detail introduction of the wrapping problem, including the wrapping problem in conventional PDE and SSE approach. Section 3.3 discusses the solution of the wrapping problem, which is unwrapping. This section includes a review of the phase unwrapping problem, and based on which, a new theory of spatial shift unwrapping is given. Section 3.4 performs the experiment and uses the results to confirm the performance of the new spatial shift unwrapping theory. Section 3.5 provides the conclusion of this chapter.

3.2 The Wrapping Problem

As mentioned before, in conventional PDE approach, the phase shift information contains the height distribution of object surface. However, as we know the phase function is a periodic function with the period of 2π, the phase shift information

retrieved in most PDE based approaches can only be identified within the range of $[-\pi, \pi]$. Those phase shift which is bigger than π or smaller than $-\pi$ all wrapped into this range, thus the wrapping problem occurs. The same thing also happens in SSE approach, as the projected fringe we used is a fringe structure with a periodic fringe of width T_0, the spatial shift information retrieved will only be detected within the main value of $[0, T_0]$. Hence this can also be considered as a wrapping problem.

3.2.1 Wrapping Problem in Conventional PDE approach

From Equation (1.7) we have

$$\phi(x) = 2\pi \frac{d_0 h(x)}{T_0 l_0} \tag{3.1}$$

where $T_0 = 1/f_0$ is the width of an individual fringe. Obviously, $\phi(x)$ can take any value, depending on $h(x)$, d_0, l_0 and T_0. However, with most PDE based approaches, $\phi(x)$ can only be identified within the range of $[-\pi, \pi]$. In other words, the phase is wrapped into the main value range. In the following such a *wrapped* phase is denoted as $\phi_w(x)$. Figure 3.1 shows an example which demonstrates the difference between $\phi(x)$ and $\phi_w(x)$. Assuming we have an object with its height distribution given in Figure 3.1 (a), the phase map $\phi(x)$ should be the one shown in Figure 3.1 (b) based on Equation (3.1). However, most PDE based approaches are only able to yield a $\phi_w(x)$ shown in Figure 3.1 (c). If such a wrapped $\phi_w(x)$ is used in Equation (1.7), we will obtain a height distribution in Figure 3.1 (d), which obviously suffers from significant errors. Consequently, in order to correct $h(x)$, we must employ $\phi(x)$ instead of $\phi_w(x)$. Thus a phase unwrapping operation is needed.

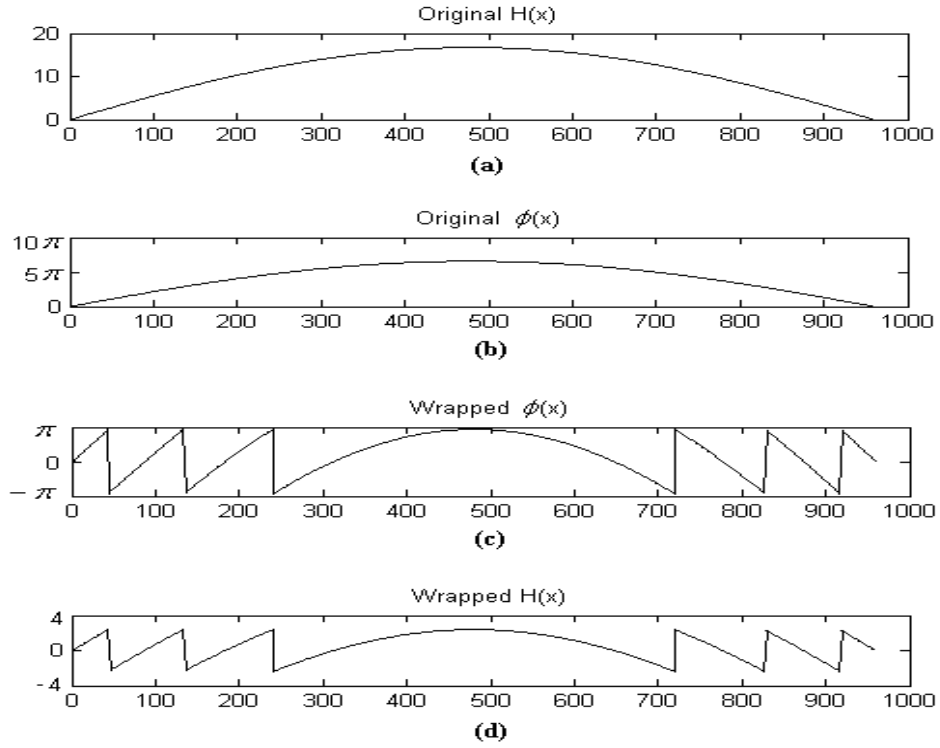

Figure 3.1 Original and wrapped phase maps

3.2.2 Wrapping Problem in SSE approach

The wrapping problem also exists in SSE approaches. From Equation (1.17) we have:

$$u(x) = \frac{d_0 h(x)}{l_0} \tag{3.2}$$

Depending on $h(x)$, d_0 and l_0, the shift function $u(x)$ may take any value as well. However, when $s(x)$ has a fringe structure with a periodic fringe of width T_0, $u(x)$ can only be detected within the main value of $[0, T_0]$. In other words, $u(x)$ is wrapped into $[0, T_0]$, which is denoted as $u_w(x)$ and given as follows:

$$u_w(x) = u(x) - kT_0, \text{ where } k = Integer\left[\frac{u(x)}{T_0}\right] \tag{3.3}$$

In order to demonstrate the relationship, we utilize the same example in Figure 3.1. With $h(x)$ shown in Figure 3.2 (a), we should have $u(x)$ in Figure 3.2 (b).

However, what we have is $u_w(x)$ as shown by Figure 3.2 (c). Use of $u_w(x)$ in Equation (1.17) will result in significant error in $h(x)$, as shown by Figure 3.2 (d). Therefore, we must work out a way to restore $u(x)$. The process is referred to as spatial shift unwrapping.

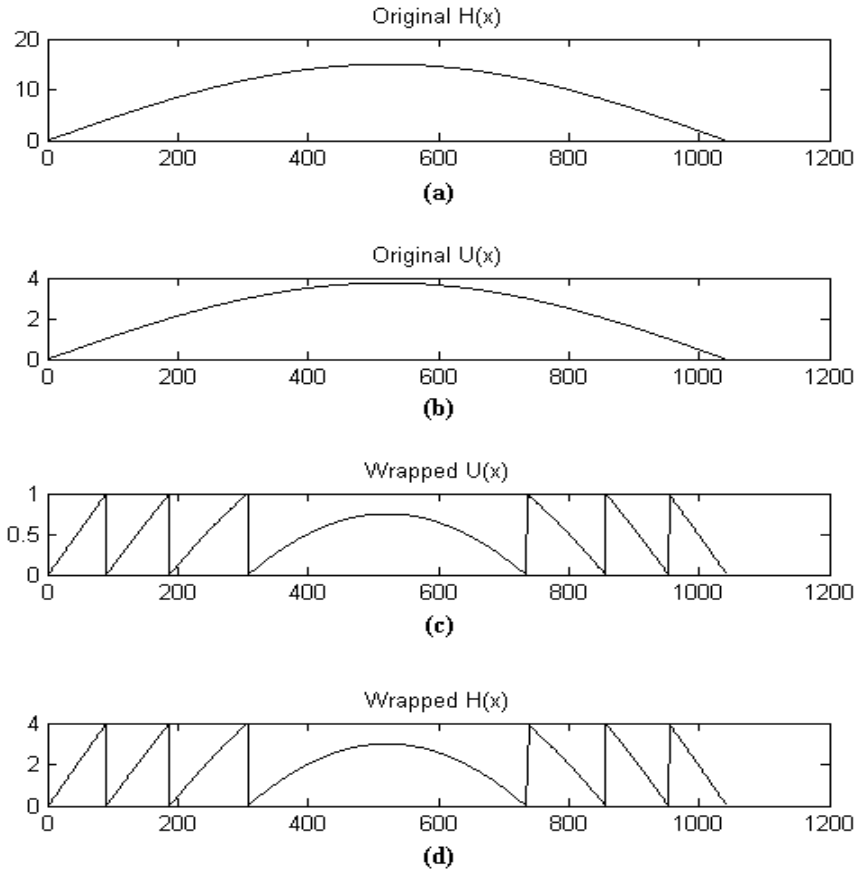

Figure 3.2 Original and wrapped shift maps

3.3 The Unwrapping

As presented before, the wrapping problem exists in both conventional PDE and SSE. Since it make the height distribution information suffer from the great distortion, it becomes a main obstacle in our 3D profilometry. To solve the wrapping problem, we need unwrapping. The unwrapping is a processing which tried to restore the original

phase or spatial shift maps before wrapped. Since the conventional PDE has attracted most of the research interests, the solutions for the wrapping problem in the phase shift based approaches have also been given. These solutions are referred to the phase unwrapping approaches [47][48].

In SSE-based FPP, the similar 'spatial shift unwrapping' is also required in order to restore the 3D shape of the object surface correctly. However, this spatial shift unwrapping for SSE-based FPP is still an outstanding issue. In this part, we first review the phase unwrapping problem in PDE based approaches, based on which we present a method for solving the spatial shift unwrapping problem.

3.3.1 Phase Unwrapping, a Review

Phase unwrapping refers to the process of restoring $\phi(x)$ from $\phi_w(x)$. As can be seen by Figure 3.1, the phase is wrapped in the following way. When $\phi(x)$ increases and reaches points $\pi, 3\pi, 5\pi,$ …(i.e., odd number multiples of π), $\phi_w(x)$ drops from π to $-\pi$. Similarly, when $\phi(x)$ varies decreasingly and reaches the same points, $\phi_w(x)$ jumps from $-\pi$ to π. The phase unwrapping should reverse the process. In other words, when we observe a phase drop from π to $-\pi$, we should add 2π to the unwrapped $\phi(x)$, and when we notice a phase jump from $-\pi$ to π, we should add -2π.

Generally, most existing phase unwrapping approaches starts from determine the differences of the neighbouring value. If a value suffers wrapped, the absolute value of the difference should more than π and can be determined as a breach of the continuity of the original phase map. Then this value is recorded as a discrete point. The compensation of 2π is performed based on these discrete points to reconstruct the

unwrapped phase maps. One thing we should notice is that these methods are based on the hypothesis that the differences of neighbouring value in original phase map are less than π.

3.3.2 Spatial Shift Unwrapping in SSE based FTP

In order to work out how to unwrap the spatial shift, we can see how $u(x)$ is wrapped into $u_w(x)$. From Figure 3.2 (b) and (c), we observed the following:

- When $u(x)$ varies increasingly and reaches points $T_0, 2T_0, 3T_0, \ldots$ (that is, integer multiples of T_0), $u_w(x)$ exhibits a drop of T_0 with its value dropping from T_0 to 0.
- When $u(x)$ varies decreasingly and reaches the same points, that is, the integer multiples of T_0, $u_w(x)$ will jump from 0 to T_0.

Spatial shift unwrapping should reverse process from $u_w(x)$ to $u(x)$. In other words, when we observe a drop from T_0 to 0, we should add T_0 to the wrapped result, and when we notice a jump from 0 to T_0, we should add $-T_0$. Assuming that the object has a continuous surface, and $u_w(x)$ is acquired in discrete form, that is, $u_w(x_i)$ ($i = 0, 2, 3, ..., N$), spatial shift unwrapping should be carried out by the following procedure:

Step 1: Initialization $u(x_0) = u_w(x_0) + kT_0$, where T_0 is determined by the height of the object at x_0. Because $u_w(x) << T_O$, from Equations (12) and (13) we have $u(x_0) = \dfrac{d_0 h(x_0)}{l_0}$ and $k = Integer\left[\dfrac{u(x_0)}{T_0}\right]$

Step 2: Starting from $u_w(x_0)$ and for $u_w(x_i)$, where $i = 0, 2, 3, ..., N$;

If $u(x_i)$ increases followed by a drop of T_0, that is, $u_w(x_i) - u_w(x_{i-1}) = -T_0$, we should increase *k* by 1, that is, $k = k + 1$;

else if $u(x_i)$ decreases followed by a jump of T_0, that is $u_w(x_i) - u_w(x_{i-1}) = T_0$, then decrease *k* by 1, that is, $k = k - 1$;

otherwise, keep *k* unchanged, that is, $k = k$;

Compute the unwrapped shift by $u(x) = u_w(x) + kT_0$

The above procedure is straight forward, but a number of issues must be resolved for its implementation in practice. Firstly, we should determine if $u_w(x_i)$ increases or decreases. Secondly we should be able to detect the sharp drop and jump. In ideal cases when surface is continuous, $s(x_i)$ and $d(x_i)$ are free of noise, these can be carried easily. However, in practice both $s(x_i)$ and $d(x_i)$ may contain noise, resulting in $u_w(x_i)$ corrupted by noise and disturbance. In order to eliminate the influence of noise and disturbance, we should make sure that $u_w(x_i)$ is as smooth as possible with respect to x_i. To this end we propose the following:

- Pre-processing of $s(x_i)$ and $d(x_i)$ by means of a digital filter in order to remove the noise and to smooth the waveform of $s(x_i)$ and $d(x_i)$. The challenges associated with pre-processing is to eliminate the unwanted noise and disturbance while keeping the original waveform of $s(x_i)$ and $d(x_i)$. Parameters of these filters should be selected with care.
- In order to determine the slop of $u_w(x_i)$, we firstly evaluate the difference of $u_w(x_i)$ over successive samples by $\Delta u_w(x_i) = u_w(x_i) - u_w(x_{i-1})$, and then take the

sign of the variance by $\delta_i = sign\{\Delta u_w(x_i)\}$ (where δ_i equals to 1, 0 and -1 for $\Delta u_w(x_i) > 0$, $\Delta u_w(x_i) = 0$ and $\Delta u_w(x_i) < 0$ respectively). We carry out the following:

If $\frac{1}{M}\sum_{j=i-M}^{i-1} \delta_i > 0$, then $u_w(x_i)$ increases, or

If $\frac{1}{M}\sum_{j=i-M}^{i-1} \delta_i < 0$, then $u_w(x_i)$ decreases.

In practice, a sharp jump or drop may transverse a few data samples on $u_w(x_i)$. Assuming that the jump or drop occurs within *L* samples, we should use the following to detect the drop:

If $u_w(x_i) - u_w(x_{i-L}) > T_2$, then $u_w(x_i)$ jumps, or

If $u_w(x_i) - u_w(x_{i-L}) < -T_2$, then $u_w(x_i)$ drops

where T_2 is the threshold which should be chosen based on the quality of $u_w(x_i)$. With the pre-processing of $s(x_i)$ and $d(x_i)$, $u_w(x_i)$ is smooth enough and hence we choose $T_2 = T_0/2$.

3.4 Experiments and Results

In order to test the performance of the approach proposed in Section 3.3.2, experiments are carried out in our laboratory. The experimental setup is the same as the last chapter.

After pre-processing and IFSE operation, we obtain the wrapped $u_w(x, y_1)$ in Figure 3.3 (a). With the proposed unwrapping approach we recovered $u(x, y_1)$ in Figure 3.3 (b), with which we obtain the height distribution $h(x, y_1)$ in Figure 3.3 (c). From

Figure 3.3 (c), the max height of $h(x, y_1)$ is 23.08mm and hence the error rate is 1.228%. Therefore we can say that shape of the object can be successfully retrieved with the proposed approach.

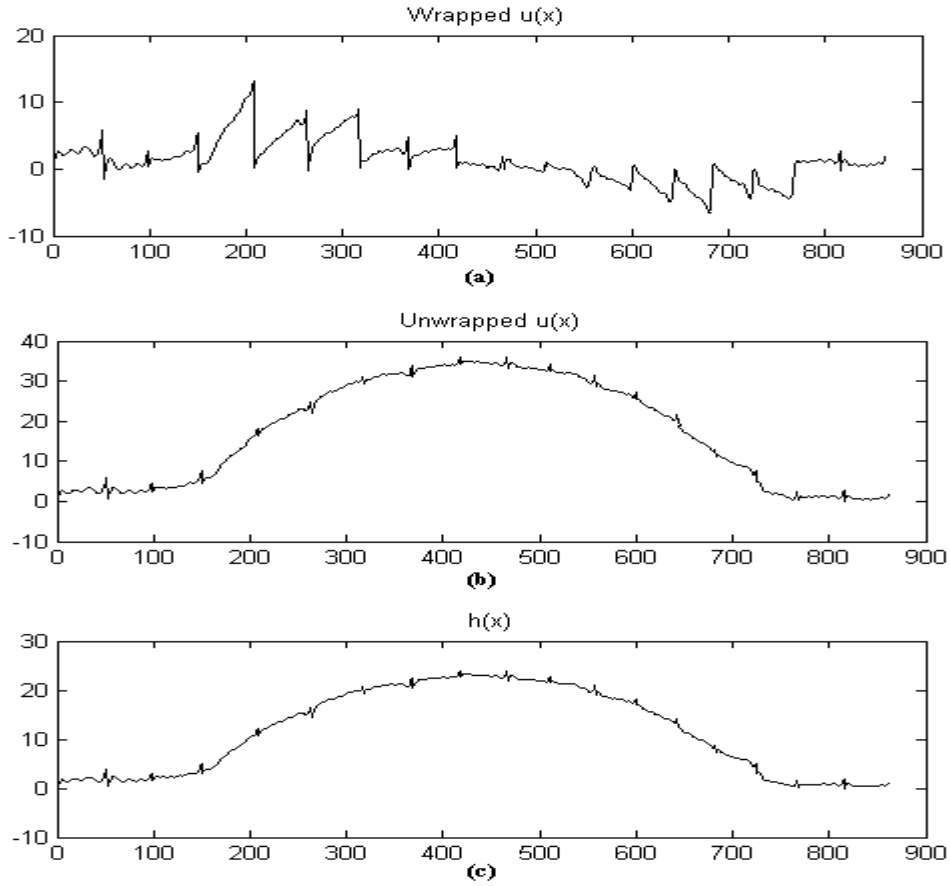

Figure 3.3 Spatial shift unwrapping and high distribution estimation.

3.5 Conclusion

In this chapter, we study the wrapping problem in FPP. Based on the phase wrapping problem in conventional PDE, we indicate the spatial shift wrapping problem associated with SSE-based FPP. The problem arises as the result of fringe reuse (that is, fringes periodic light intensity variance), and the spatial shift can only be identified without ambiguity with the range of a fringe width. We present a technique to carry out spatial shift unwrapping to remedy the problem which referred the phase unwrapping method

in conventional PDE-based FPP. In order to test the performance, we also carry out experiments on an object with simple hemisphere surface shape. The results have shown the effectiveness of the proposed unwrapping technique.

Charpter 4 Improvement of SSE

4.1 Introduction

In conventional PDE, the projected fringe pattern is required to be sinusoidal because they need to retrieve the height distribution from the phase map of the projected and deformed fringes. That means the phase maps of $s(x)$ and $d(x)$ must exist and can be detected. However, using of sinusoidal pattern faces these limitations: One thing is the pure sinusoidal fringe patterns cannot be obtained in practice due to many undesired factors inherent to digital projection. Another thing is even the pure sinusoidal fringe patterns can be produced, they are also vulnerable to geometry distortion and nonlinear intensity distortion. Since SSE approach retrieves the height distribution by estimating the spatial shift directly, it has the advantage that the projected fringe patterns are not required to be either sinusoidal or periodic. Thus we can design a fringe which has strong counter-interference capability against the noise and nonlinear distortion to improve our works.

This chapter is organized as follows: Section 4.2 discusses the limitations of traditional sinusoidal fringe. Section 4.3 presents the improvement of traditional SSE, including designs the rules of fringe selection and introduces a sawtooth fringe pattern instead of sinusoidal based on theoretic analysis. Section 4.4 carries out the experiment and uses the results to illustrate the advantage of the new sawtooth fringe patterns. Section 4.5 provides the conclusion of this chapter.

4.2 The Limitation of Traditional Sinusoidal Fringe

As mentioned before, the conventional PDE uses the detection of phase shift from original and deformed fringe patterns enables the retrieval of the 3D shape. Hence the projected fringes are limited to be sinusoidal. However, in practice, problems occur

when we use the sinusoidal fringe pattern as projected fringe, which severely impacted the accuracy of measurement.

The sinusoidal fringe pattern is also unsuitable for IFSE method in SSE approach. As shown in Chapter Two, the inverse function of sinusoidal requires a very high degree of curve fitting polynomial, which greatly increases the time of computation. Even though the curve fitting error still exists and affects the accuracy of measurement. Increasing the degree of curve fitting polynomial may reduce the curve fitting error, however, it cannot completely eliminate due to the characteristic of sinusoidal function.

In this section, we first discuss the disadvantages of using sinusoidal fringe pattern.

4.2.1 Disadvantage of sinusoidal fringe

According to the theoretical analyse in Chapter One, the conventional PDE approaches, like FTP and PSP, can retrieve the phase shift information precisely if the projected fringe pattern is pure sinusoidal. However, in most practice case, it is very hard to realize due to the undesired factors inherent to digital projection.

4.2.1.1 Finite Projection and Screen Door Effect

One factor is called finite projection. As we know, the projected fringe pattern is a digital signal, the intensity distribution is defined by pixels, and each pixel is confined to a finite set of intensity values. Hence the spatial resolution of the digital projector is limited to the spatial size of a projected pixel. This characteristic of digital projector is then referred to as finite projection, or pixelisation. The finite spatial resolution issue was referred by Coggrave and Huntley in [49].

This finite projection characteristic of projected signal introduces another problem: due to the physical limitations of digital projector, tiny discontinuities exist between the

pixels, which will make the intensity distribution look discrete. This phenomenon is referred as the “Screen Door Effect” (SDE).

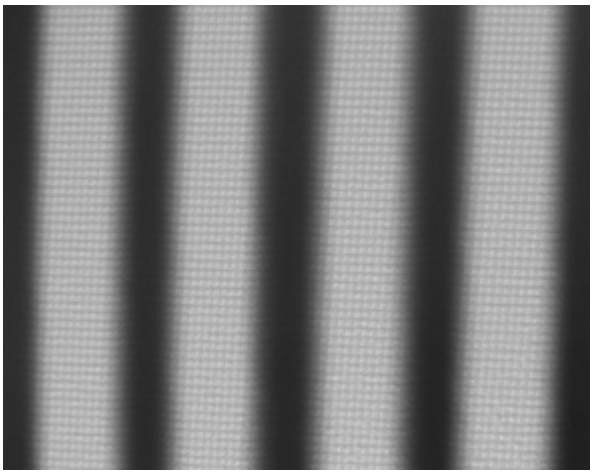

Figure 4.1 Screen Door Effect

Figure 4.1 shows the “Screen Door Effect”. From this figure, the discontinuity of intensity distribution can be seen clearly. This SDE characteristic ultimately influences the quality of projected and captured fringe pattern. The pure sinusoidal fringe pattern is impossible to produce due to the discontinuity of projected signal. As the intensity of captured images becomes discrete, it is hard for us to detect the precise phase shift information. Because these discontinuities cannot offer any information about object surface, errors will occur during the retrieving of height information. Thus the accuracy of measurement will be severely affected.

Since this SDE characteristic is introduced by the physical limitation of digital project, we can alleviate it by utilizing a high resolution projection source if we want to improve the accuracy. However, high resolution demands high accuracy of the projector, which will increase the cost of equipment, it is unsuitable for application. Thus we should find another way to solve this problem. Introducing a new fringe pattern which suffers less effect of SDE should be a solution.

4.2.1.2 Gamma Distortion

Despite the influence of SDE characteristic, there has another systematic limitation existing in the DFP system which imposed on the projected fringe. This limitation is called non-linear camera / projection luminous response, which can also be referred as Gamma distortion [50, 51]. This distortion is introduced by visual display systems in order to enhance human perception of the sensation of lightness. However, it compromises the geometrical structure of the projected signal.

The gamma distortion can be considered as a power function of intensity. In general, it can be modelled in Equation 4.1 [50, 51]:

$$w(x, y) = u(x, y)^{\gamma} \tag{4.1}$$

$$\text{for } u \in [0,1]$$

where $u(x, y)$ is the image intensity function delivered to projector, $w(x, y)$ is the actual output image intensity distribution and γ is typically a fractional value $1 < \gamma < 3$ specific to the display system.

Then we suppose the image intensity function which delivered to projector is a pure sinusoidal intensity distribution, which means

$$u(x, y) = a + b\cos(2\pi f_0 x) \tag{4.2}$$

where a and b are the constants referring to the projected fringe offset and contrast respectively, and f_0 is the spatial carrier frequency of the projected fringe.

Substituting Equation (4.2) to Equation (4.1), we have:

$$w(x, y) = [a + b\cos(2\pi f_0 x)]^{\gamma} \tag{4.3}$$

As we know γ is a fractional value, we can represent Equation (4.3) in Fourier Series like:

$$w(x, y) = c_0 + c_1 \cos(2\pi f_0 x) + \sum_{m=2}^{\infty} c_m \cos(m[2\pi f_0 x]) \qquad (4.4)$$

where m is the order of harmonic components and c_m is the corresponding amplitudes, and

$$c_m = \frac{2}{T}\int_0^T w(x, y)\cos(m[2\pi f_0 x])\,dx \qquad (4.5)$$

where T is the spatial period of the fringe image.

According to Equation (4.3) and (4.4), we can see the projected pure sinusoidal fringe is geometrical distorted and higher harmonic components are introduced due to the gamma distortion.

From Equation (1.7), we can find the height distribution $h(x)$ only associate with $\phi(x)$, which is the phase shift of the fundamental component. Hence those higher harmonic components must be eliminated if we want to retrieve the accuracy height distribution. That is why we said pure sinusoidal is better. However, because of the gamma distortion, the higher harmonic components always exist and severely impact the accuracy of measurement.

To solve the problem, one possible way is to use filters to eliminate the higher harmonic components and pick up the fundamental component of fringe pattern. However, in practical case, it is difficult to design an ideal filter which can completely eliminate the higher harmonic components and make the filtered fringe pure sinusoidal. Moreover, if the deformed fringe pattern has an overlapped spectrum, the bandpass filtering will also be unusable. Therefore, errors will arise if we use sinusoidal fringe pattern as projected fringe pattern.

4.2.1.3 Influence of Environment

Besides the undesired factors inherent to digital projection system, the projection environment also influences the quality of projected signal. In this part, we discuss the influence introduced by environment.

In Chapter Two, we have already mentioned the influences of environment on the projected fringe. The first thing is the interference introduced from the light source which did not belong to the projector but from the environment. This interference can be reduced by carrying out the experiment in a dim environment.

However, the distortion which introduced by the projection angle should become a key problem for sinusoidal fringe pattern. Although we solved the intensity distortion in Section 2.3.1.2, there is another distortion exist, which is the length change in each period of the sinusoidal fringe.

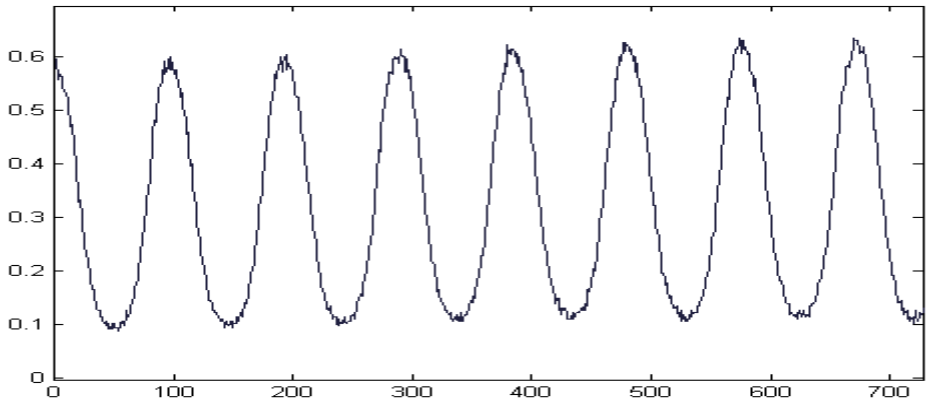

Figure 4.2 Acquired fringe data from captured image of reference plane

Figure 4.2 shows the retrieved intensity data from the captured image of reference plane. From this figure, we can easily see the decrease in length of each period of the sinusoidal fringe.

Like the intensity distortion, this length distortion is also introduced by the projection

angle. We suppose that the projector is adequately focused over the range of interest. The angle subtend by each period is the same, which is assumed as α. Then the projected length of each period on reference plane is then shown in Figure 4.3. According to the triangle principle, obviously we will have $T_1 > T_2 > T_3$.

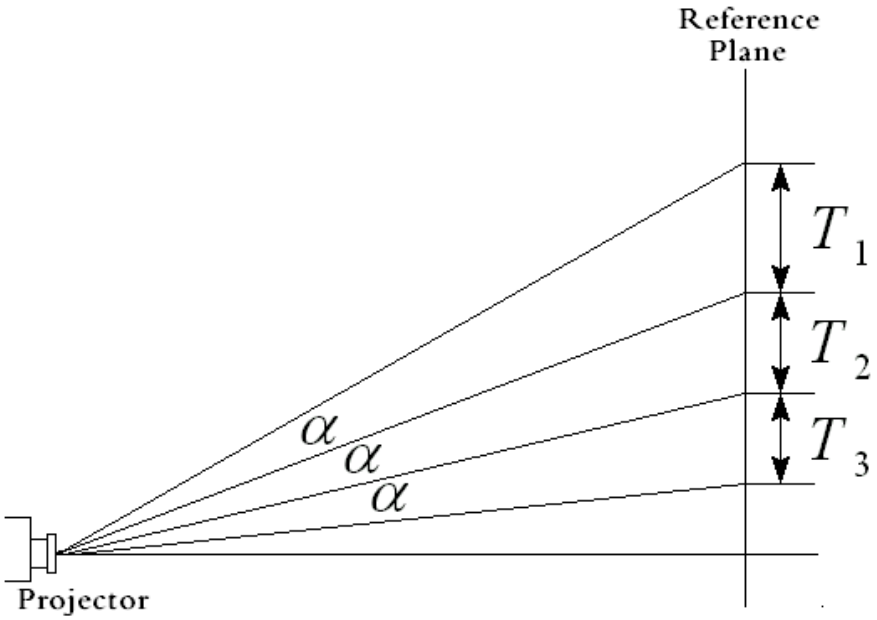

Figure 4.3 Length change introduced by projection angle

From Figure 4.2, we can see the period distortion caused by projection angle severely damage the geometric structure of captured sinusoidal fringe pattern. This influence will impact the accuracy of 3D surface measurement, especially for conventional PDE. The change of the period means the variety of the frequency f_0. Hence errors will arise in the retrieving of height distribution $h(x)$ if we apply the frequency of projected fringe pattern on the captured fringe pattern directly.

Since this period distortion is introduced by the projection angle, it cannot be avoided unless we change the basic principle of DFP system. The only way to solve this problem is to use the triangle principle to do compensation. However it will greatly

increase the total computation time if we want the result precisely.

In conclusion, the captured fringe pattern suffers many unavoidable distortions introduced by the limitation of digital projection system and the environment. The sinusoidal fringe pattern is vulnerable to these factors. Hence the inevitable errors caused by the selection of sinusoidal fringe will become an obstacle for us to pursue the high accuracy of measurement.

4.2.2 Limitation in IFSE

Since the conventional PDE retrieves the height distribution of object surface by using the phase shift $\phi(x)$, it is vulnerable to the geometric change brought by these distortions, such as gamma distortion and period distortion. Fortunately, as the SSE approach use the spatial shift distribution $u(x)$ to retrieve height distribution, it has a strong counter-interference capability against geometric distortion. However, the selection of sinusoidal fringe as projected fringe pattern in SSE approach faces other limitations which also block us to regain the measurement results precisely.

As mentioned before, the IFSE method use the inverse function of projected signal to obtain the spatial shift distribution $u(x)$, then retrieve the height information of object surface by using Equation (1.17). Hence the first step for IFSE processing is to find the inverse function of projected fringe $s(x)$. Because the sinusoidal function is not a monotonic function, it needs to be divided into several monotonic intervals. Then inverse functions are found for each interval. As the sinusoidal fringe pattern is used as projected fringe $s(x)$, the inverse function $s^{-1}(r)$ for each interval should be supposed to an arc sinusoidal function, like arcsine or arccosine. However, we cannot use the mathematical arcsine or arccosine function as the inverse function directly, because they are not mapping to the sine or cosine function one-by-one. For example,

we have both $\sin\left(\frac{\pi}{6}\right)=\frac{1}{2}$ and $\sin\left(\frac{5\pi}{6}\right)=\left(\frac{1}{2}\right)$, but in arcsine function, it only has $\arcsin\left(\frac{1}{2}\right)=\left(\frac{\pi}{6}\right)$. Same thing happens in arccosine function. Both $\cos\left(\frac{\pi}{3}\right)$ and $\cos\left(\frac{5\pi}{3}\right)$ equals $\frac{1}{2}$, however, the function $\arccos\left(\frac{1}{2}\right)$ can only have the result of $\frac{\pi}{3}$ but not $\frac{5\pi}{3}$.

Thus we should find another function which is mapping to the projected signal function one-by-one by using the polynomial. In theory, the arc sinusoidal function can be transformed into polynomial, however, problems occurs in practice:

$$\begin{aligned}\arcsin\ z &= z+\left(\frac{1}{2}\right)\frac{z^3}{3}+\left(\frac{1\cdot 3}{2\cdot 4}\right)\frac{z^5}{5}+\left(\frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\right)\frac{z^7}{7}+\ldots \\ &= \sum_{n=0}^{\infty}\left(\frac{(2n)!}{2^{2n}(n!)^2}\right)\frac{z^{2n+1}}{(2n+1)}; \qquad |z|\le 1 \end{aligned} \tag{4.6}$$

$$\begin{aligned}\arccos\ z &= \frac{\pi}{2}-\arcsin\ z \\ &= \frac{\pi}{2}-\left(z+\left(\frac{1}{2}\right)\frac{z^3}{3}+\left(\frac{1\cdot 3}{2\cdot 4}\right)\frac{z^5}{5}+\left(\frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\right)\frac{z^7}{7}+\ldots\right) \\ &= \frac{\pi}{2}-\sum_{n=0}^{\infty}\left(\frac{(2n)!}{2^{2n}(n!)^2}\right)\frac{z^{2n+1}}{(2n+1)}; \qquad |z|\le 1 \end{aligned} \tag{4.7}$$

Equation (4.6) and (4.7) shows the polynomial transform of the arcsine and arccosine function. From these functions, it can be seen the polynomials are infinite series, which means the degrees of polynomial should be infinite. However, in practical case, the degree of polynomial used for curve fitting is limited. As it was discussed in Chapter Two, the higher degree of polynomial uses in curve fitting, the longer

computation time takes. Too much time on computation is unbearable in real-time application. Errors also exist since limited degrees of polynomial cannot perfectly fitting the arc sinusoidal function, which block us to perform the high accuracy measurement.

Moreover, Equation (4.6) and (4.7) is based on the ideal condition. Due to the gamma distortion and other kinds of interference, the polynomial used for curve-fitting should be more complex than these equations. Thereby the system load is further increased.

Thus in consideration of improving the SSE, the selection of sinusoidal fringe pattern as projected fringe pattern is not properly due to these limitations in IFSE method.

4.3 The Improvement of SSE

In SSE approach, since the height distribution is retrieved from the spatial shift distribution $u(x)$ directly, the projected fringe is not required to have any particular structures. Since there has so many disadvantages of using sinusoidal fringe pattern as projected fringe pattern we discussed in last part, it should be a good choice for us to design a new kind of fringe pattern as projected fringe pattern.

In this part, we discuss the improvement of SSE by designing a different fringe as projected fringe pattern instead of the traditional sinusoidal fringe. First the rules of the fringe designing are given. Based on these rules, a sawtooth fringe pattern is introduced. Then theoretical analysis is followed to give a detail illustration about the applicability of this fringe pattern.

4.3.1 Rules of Fringe Design

From Chapter One, we know the SSE approach has many advantages in comparison to the conventional PDE approach. One of these advantages is it does not depend on the structure of the projected fringe pattern. Thus the tradition sinusoidal fringe can be replaced by some other kind of fringe in theory. However, it does not mean any fringe pattern can be used in SSE. Some fringes may be worse than sinusoidal. Hence we need to set some rules in order to find a proper fringe which can overcome the disadvantages in sinusoidal fringe.

- Simplicity

The first rule for fringe design is the designed fringe should be simple and concise in function.

As discussed in last part, the projected fringe suffers geometrical and non-linear distortion which mainly caused by Screen Door Effect and gamma distortion. Under the influence of SDE, the intensity distribution becomes discrete, and the discontinuities cannot offer any information. Since this influence is unavoidable, what we can do is try to alleviate its influence. Then we find fringes with simple function suffer less effect than the complex ones. This is because the structure of simple function is also very concise, thus it can be easily restored once damaged by SDE. Most information which lost on the discontinuities can also be retrieved.

Simple functions also show its strong counter-interference capability against gamma distortion. As we know the gamma distortion can be modelled as Equation (4.1). Then if we use a linear function as the projected fringe function $u(x, y)$, after gamma distortion, the actual output function $w(x, y)$ will be three-degree at max because γ is typically a fractional value $1 < \gamma < 3$. Hence in IFSE, a three-degree polynomial is enough to curve fitting the inverse function perfectly. The computation time will be

greatly reduced.

On the other side, high-degree function means more complexity in computation, which will increase the system load and computation time. This also not fit for the real-time application.

- Adjustable in period

Although spatial shift based approach allows the projected fringe patterns can be neither sinusoidal nor periodic. However, non-periodic fringe patterns require to design every time for difference objects. Therefore, we still select the periodic fringe pattern as the projected fringe pattern.

In Chapter Three, the unwrapping problem in SSE approach is studied. At last of that chapter, we referred a special condition in practice where a sharp jump or drop may transverse a few data samples on $u_w(x_i)$. The solution of this problem is based on the assumption where $u_w(x_i)$ is smooth enough. However, in application, $u_w(x_i)$ is not always smooth if the object has sharp change. To keep this solution working, what we should do is to increase the fringe period T_0. Hence the threshold T_2 to fit the sharp change of object surface is increased.

On the other side, large value of the fringe period T_0 will cause the impairment of the accuracy in our measurement. Thus a fringe designed with an adjustable period should be most suitable.

- Monotonic

This rule is in consideration of IFSE method. As mentioned before, the condition to perform IFSE requires the retrieved signal function $r = s(x)$ is a monotonic function, or it is monotonic in intervals of *x*. For sinusoidal function, we have to divide it into two

monotonic intervals to ensure the processing of IFSE, which introduced additional job. However, if we use a function which monotonic in each period, this additional job can be avoided.

In conclusion, the design of new fringe should follow these three rules: Simplicity in function, adjustable in period and monotonic.

4.3.2 The Selection of Sawtooth

Based on the rules we indicated, a new kind of fringe pattern which satisfied all the rules is introduced in this part. This fringe is sawtooth fringe. We will first give an introduction of this fringe pattern, and then discuss the advantages on the selection of sawtooth fringe in theory.

4.3.2.1 Introduction of sawtooth fringe

The characteristic of a sawtooth wave is its structure ramps upward and then sharply drops or ramps downward and then sharply rises. The traditional sawtooth function is a piecewise linear function which defined as:

$$s(x) = x - \mathit{floor}\ (x)\ ; \qquad x \geq 0 \qquad (4.7)$$

where the $\mathit{floor}\ (\cdot)$ is a function which maps a real number to the next smallest or next largest integer.

Equation (4.7) presents a typical sawtooth function in the range of $[0,1]$, and with period 1. According to the rules given in last part, the designed function should have an adjust period T_0, then we modify this function:

$$s(x) = a \cdot (\frac{x}{T_0} - \mathit{floor}\ (\frac{x}{T_0}))\ ; \qquad x \geq 0 \qquad (4.8)$$

where a is the contrast of the projected fringe, which usually in the range of [0,1]. T_0 is the period of fringe.

Hence we can generate the sawtooth waveforms by using the Equation (4.8), and Figure 4.4 shows the generated sawtooth fringe pattern with contrast a = 1.0 and period T_0 = 25.

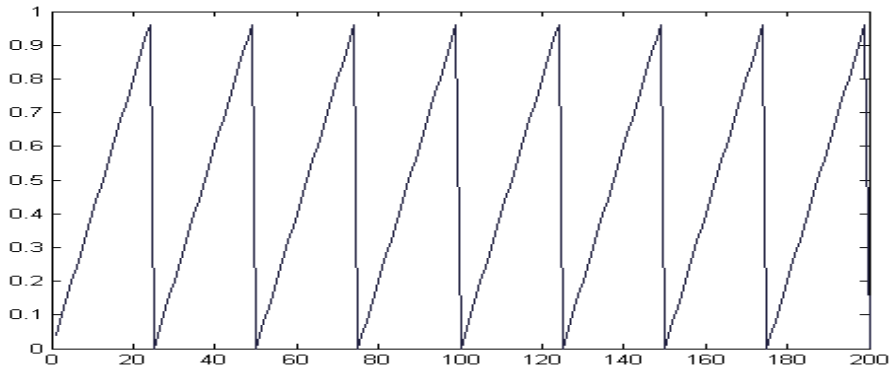

Figure 4.4 Sawtooth waveforms

Then we convert the function of colour intensity into the image using the same system which introduced in chapter two:

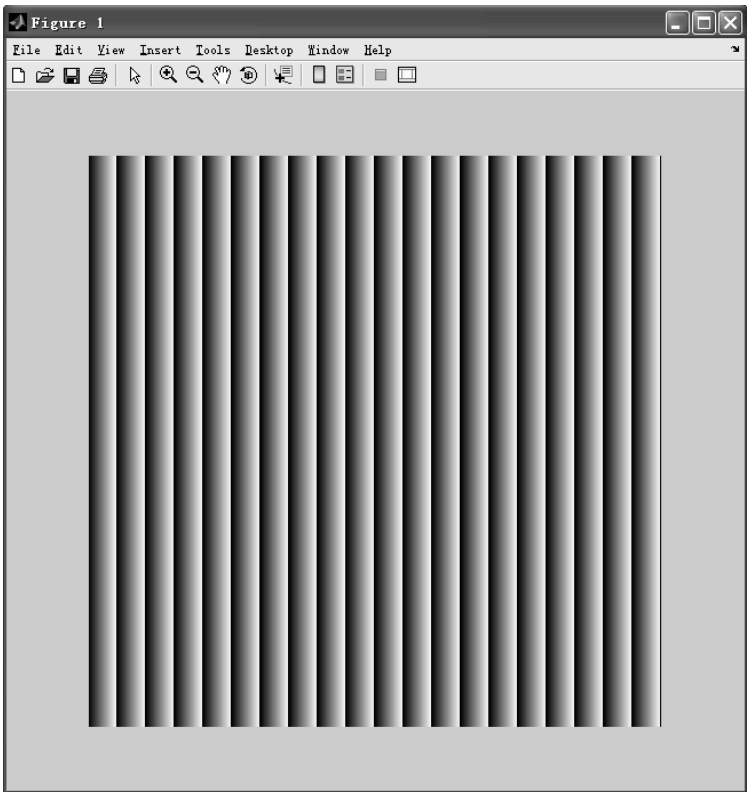

Figure 4.5 Sawtooth fringe produced

Figure 4.5 shows the projected image with sawtooth fringe pattern. The parameters of the red, green and blue fringes are all set at same value. Therefore it is a black-and-white image. It can be seen from this figure, the intensity ramps upward and then sharply drops when it reaches the max of luminance.

4.3.2.2 Advantage of sawtooth fringe

The design of this sawtooth fringe completely satisfies the rules given before: simplicity, adjustable in period and monotonic.

From Equation (4.8), it can be seen the sawtooth function we designed is a one-degree piecewise linear function. Hence both in function and structure, it satisfies the simplicity.

Because we modified Equation (4.7) into Equation (4.8), the period of fringe can be

controlled by adjusting the period value T_0. Thus the fringe function we designed can fulfil this rule of adjustable in period.

From Figure 4.4, it is easy to see the structure of this sawtooth function is monotonic increasing in each period.

Since the design completely follows these rules, the sawtooth fringe shows its great advantages in comparison to sinusoidal fringe:

The first advantage is reduced system load. As the designed function is a piecewise linear function, the inverse function is easier to calculate in IFSE.

The second advantage is this sawtooth fringe has strong counter-interference capability against geometrical and non-linear distortion, especially in solving the SDE and gamma distortion problems:

As we know the original sawtooth function is a linear function, the missing information in discontinuities brought by the SDE characteristic can be retrieved from its neighbouring values by using the linear relationship.

For gamma distortion, according to the Equation (4.7), the distorted fringe function will have no more than three degrees since γ is typically a fractional value $1 < \gamma < 3$. Thus the inverse function of the distorted fringe will also have no more than 3 degrees. Compare with sinusoidal fringe have to use at least 10 degrees to receive an acceptable result, sawtooth fringe still shows it great advantage in saving the computation time.

In conclusion, the sawtooth fringe we designed shows its advantages against traditional sinusoidal in sparing system load and counter-distortion capability, especially in facing the SDE and gamma distortion problem.

4.4 Experiments and Results

In order to test the performance of the sawtooth fringe pattern introduced in Section 4.3, experiments were carried out in the laboratory. The experimental system is same to Chapter Two. Figure 4.6 shows the captured two images, one on reference plane and one on object surface.

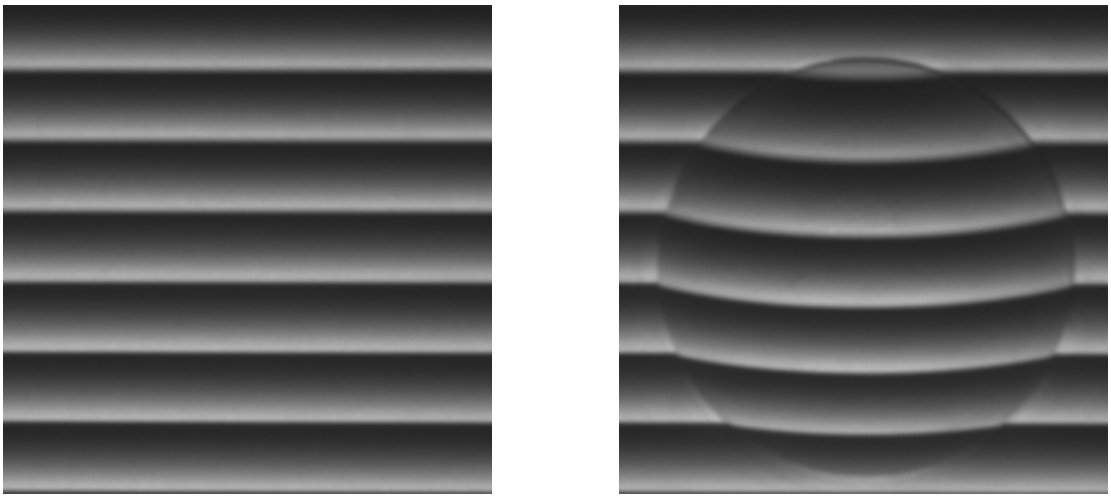

Figure 4.6 Captured sawtooth image of reference plane (Left) and object surface (Right)

The colour intensity information for these two images is retrieved. The acquired sawtooth fringe data from reference plane and the object plane can be seen in Figure 4.7.

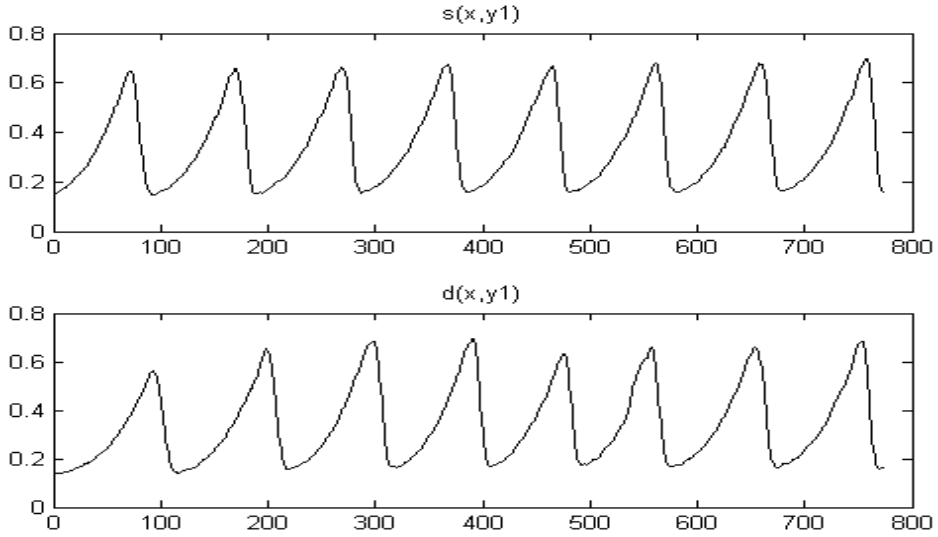

Figure 4.7 Acquired sawtooth fringe data from captured image (y1=750)

From Figure 4.7, we can see the structure of captured sawtooth fringe is mainly distorted due to non-linear distortion, especially the gamma distortion and height distortion in deformed fringe. Hence we still need to perform the pre-processing before using IFSE method.

As mentioned in chapter two, the captured sinusoidal fringe pattern first needs go through a filter to solve the non-linear distortion. However, the using of filter will bring in the distortion of deformed fringe pattern hence causes the height information lost. Since the sawtooth fringe patterns have strong counter-interference capability against non-linear distortion, there is no need for using the filter.

From Figure 4.7, we can find the deformed sawtooth fringe pattern suffers evident distortion in height as same as sinusoidal fringe. This is also caused by the unequal intensity distribution due to the shape of object surface. Therefore, same fringe normalization processing which referred in Section 2.3.1.2 can be used to solve the intensity distortion problem. After the pre-processing, the normalized sawtooth fringe data is shown in Figure 4.8.

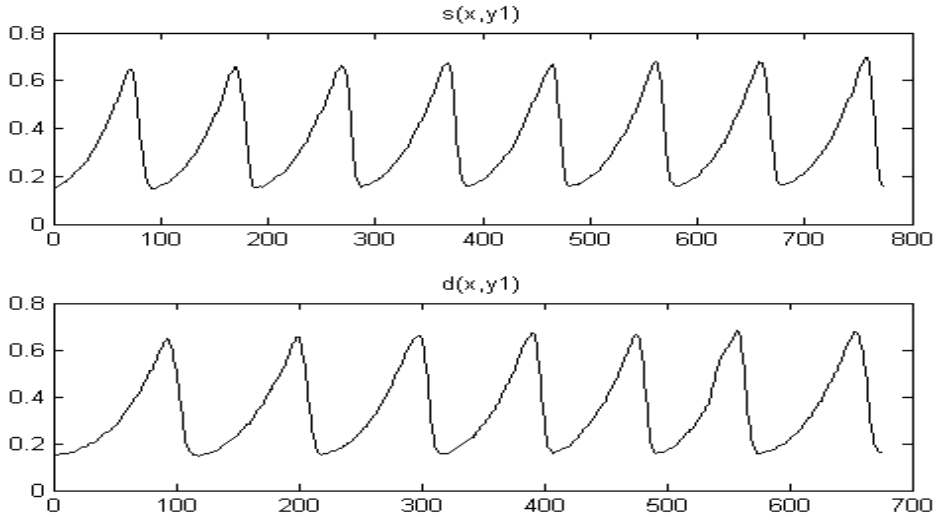

Figure 4.8 Normalized sawtooth fringe data

After pre-processing, the next step is the IFSE processing. Then we find a new problem occurred in the real practice. As mentioned before, the sawtooth fringe pattern we select should be monotonic increasing in each of its period. However, as shown in Figure 4.8, the structure of the fringe is distorted. At the end of each period, the fringe structure which supposes to be a sharp drop is distorted into a downward slope.

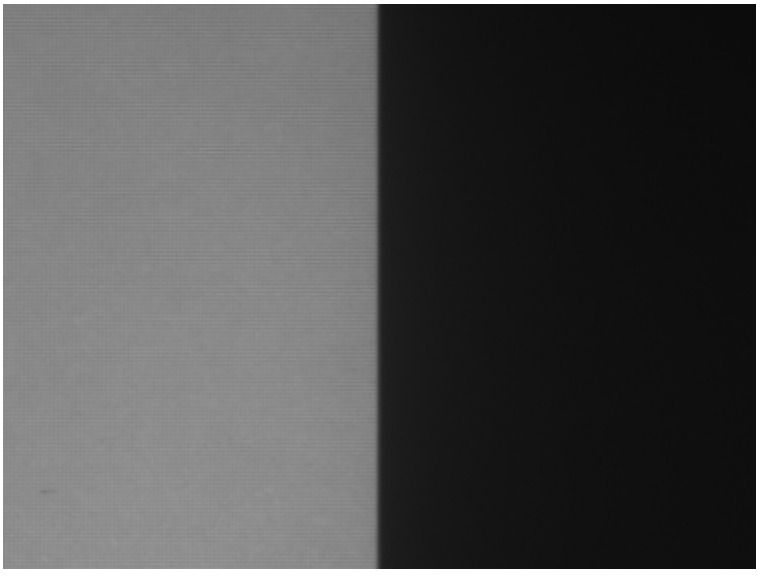

Figure 4.9 Test image with white and black

We projected a test image which shown in Figure 4.9 to analyse this distortion. In this image, we only use two colours: the white and black, which stand the maximum and minimum of the intensity. Hence the curve of the intensity data for this image should be shown like Figure 4.10, which only has a sharp drop in the middle of the curve.

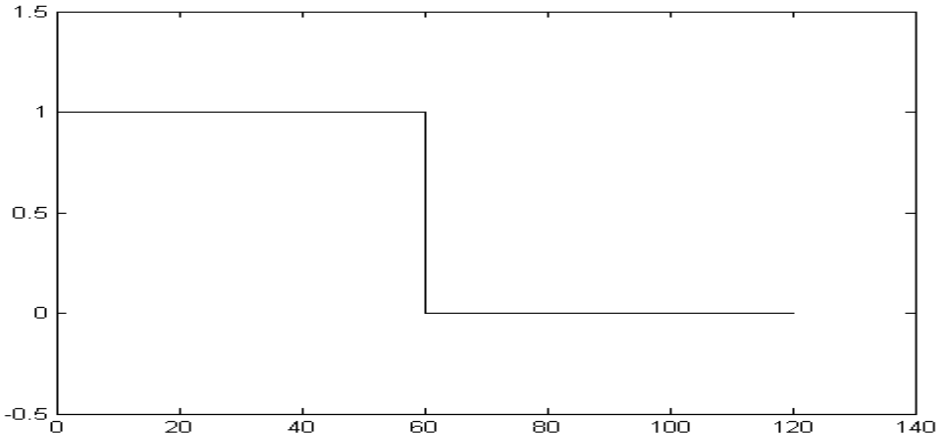

Figure 4.10 Intensity data of test image in theory

However, after projection, the intensity data we retrieved from the test image in Figure 4.9 has been distorted. The retrieved intensity data curve is shown in Figure 4.11.

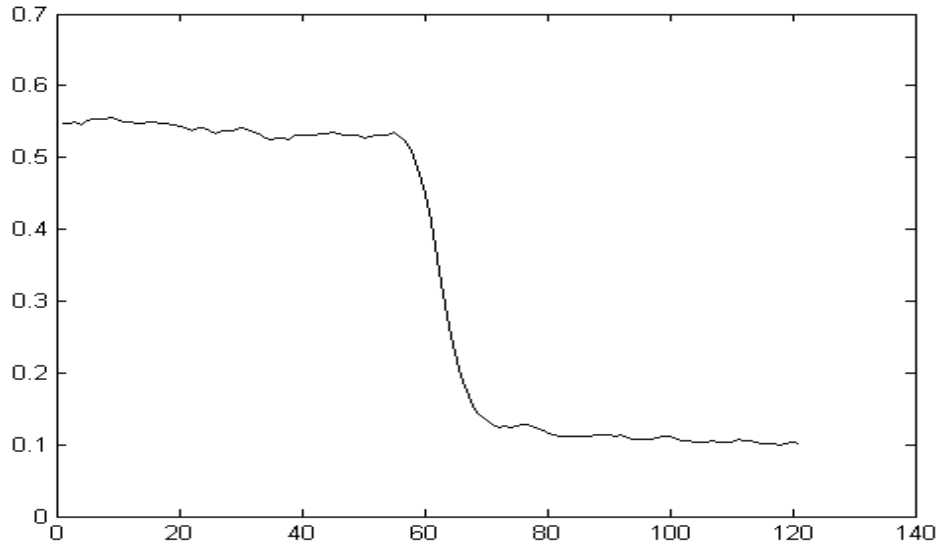

Figure 4.11 Intensity data retrieved from projected test image

From Figure 4.10, we find it is same to Figure 4.8, the curve structures which is

supposed to be a sharp drop are distorted into a downward slope. After looking into this kind of distortion, we find each decreasing interval generated has the same length alone *x* direction. Hence we infer that this distortion is introduced by another system limitation of digital projector, where the light intensity produced by the digital projector cannot adapt to the sharp change.

To solve this problem, one solution is increasing the sawtooth fringe period value T_0. As we know the distortion only affects fixed length alone *x* direction in each period, we can reduce the influence of this distortion by increasing the period. If the length of each period is long enough, the ratio of distorted part could be very small and can be able to ignore.

Another solution is performing the IFSE processing on the decreasing interval. This is much like what we do in sinusoidal fringe, where different inverse functions are used on the increasing and decreasing intervals respectively. As the length of the distorted interval is fixed, it is easy for us to separate the increasing and decreasing interval. Although an additional inverse function is needed for the decreasing interval, the computation time will not increase too much since the max degree of the inverse function will not exceed three.

After IFSE processing and spatial shift unwrapping, the final result of $u(x, y_1)$ and the height distribution $h(x, y_1)$ is obtained in Figure 4.12. From this figure, the max height of $h(x, y_1)$ is also 23.08 mm and the error rate is 1.228%.

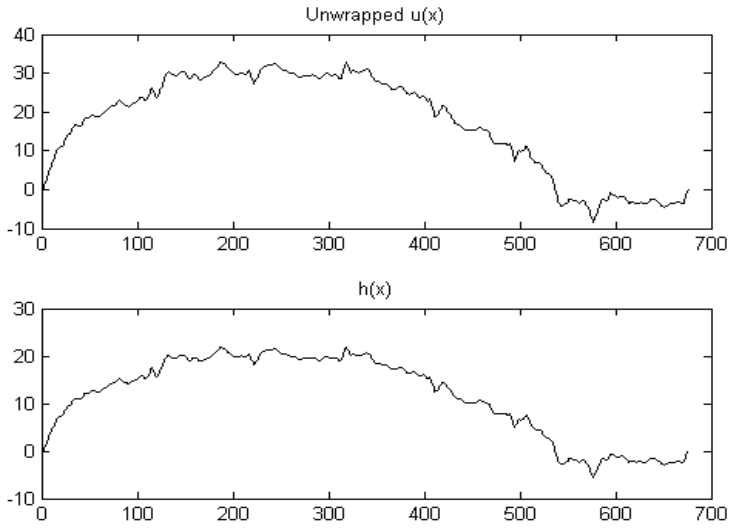

Figure 4.12 Retrieved spatial shift and high distribution result

Then we find the selection of sawtooth fringe pattern as projected pattern achieves the same accuracy as sinusoidal fringe pattern. However, compare with the sawtooth fringe, using traditional sinusoidal requires more additional computation time. Firstly, the captured sinusoidal fringe pattern needs to go through a filter to fix the non-linear distortion, which will cost too much time on the filter design and computation, and the filter itself will bring distortion to the deformed fringe pattern hence affect the measurement accuracy.

Secondly, the selection of sinusoidal fringe pattern needs a 10-degree inverse function in IFSE processing to reach such measurement accuracy. However, if we select sawtooth fringe, only 3-degree is enough. As we know, the inverse function is approximated in least squares sense. In the method of least squares, if we want to calculate the curve fitting polynomial with degree *k*, the following equation will be used:

$$\begin{bmatrix} \sum_{i=1}^{N} w_i & \sum_{i=1}^{N} w_i x_i & \cdots & \sum_{i=1}^{N} w_i x_i^k \\ \sum_{i=1}^{N} w_i x_i & \sum_{i=1}^{N} w_i x_i^2 & \cdots & \sum_{i=1}^{N} w_i x_i^{k+1} \\ \vdots & \vdots & & \vdots \\ \sum_{i=1}^{N} w_i x_i^k & \sum_{i=1}^{N} w_i x_i^{k+1} & \cdots & \sum_{i=1}^{N} w_i x_i^{2k} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} w_i y_i \\ \sum_{i=1}^{N} w_i x_i y_i \\ \vdots \\ \sum_{i=1}^{N} w_i x_i^k y_i \end{bmatrix}. \quad (4.9)$$

where N is the number of given data, w_i is the weight of given data (x_i, y_i), in this condition we have $w_i \equiv 1$, and $a_0, a_1, \cdots, a_k$ is the coefficients of fitting polynomial in ascending powers.

Hence if we calculate $a_0, a_1, \cdots, a_k$, which is the solution of Equation (4.9), we can have the curve fitting polynomial $\varphi(x)$:

$$\varphi(x) = a_0 + a_1 x + \cdots a_k x^k, \quad (4.10)$$

From Equation (4.9), it is easy to find the computation times is in proportion to $(k+1)^2$. Therefore, The ratio of time which spend on calculating the inverse function for sinusoidal function and inverse function for sawtooth function is $(10+1)^2 : (3+1)^2$, that is 121 : 16. Obviously, the computation time is greatly shortened, and the efficiency is significantly improved.

4.5 Conclusion

In this chapter, we discussed the efficiency improvement for the implementation of SSE approach. In most of the conventional PDE and SSE approaches, the projected fringe pattern is selected to be sinusoidal. However, the selection of sinusoidal fringe pattern faces many problems: The sinusoidal waveform is vulnerable to geometry distortion and

nonlinear intensity distortion, and it also not suitable for IFSE method in SSE approach. All these problems result in additional computation. Since SSE approach allows us to use another kind of fringe pattern instead of sinusoidal, we can improve the efficiency by design a new fringe pattern according to some rules. After selection, we find the sawtooth fringe pattern is very suitable for our designing rules in theoretical analyse. The advantage of this selection is then proven after we perform the practical experiment.

Charpter 5 Conclusions and Future Work

5.1 Conclusions

The main aim of this thesis is to implement the spatial shift estimation approach for 3D profile measurement base on digital fringe projection. The research work includes reviewing the existing theories, performing the implementation based on these theories, and solving the problems encountered in practical experiment. In particular three issues are considered in this thesis.

The first issue demonstrated in Chapter 2 provides details of implementation of SSE approach for 3D profile measurement base on digital fringe projection. We present a DFP system model based on the principle of FPP. Using such a system, the implementation of SSE approach based on the SSE and IFSE theory is performed. However, problems which did not be mentioned in reference have occurred during the implementation. Solutions then have been developed to the corresponding problems, such as using a filter and fringe normalization processing. We also introduce an improved IFSE method to reduce the computation time. Finally, the 3D model of object surface is reconstructed to show the performance of this implementation.

The second aspect considered in this thesis is to solve the shift unwrapping problem. As demonstrated in Chapter 3, we present a review of the phase unwrapping problem in conventional PDE based Fringe Pattern Profilometry, based on which we indicate that a similar unwrapping problem exists in SSE approach, which is the spatial shift unwrapping problem. A description of a method for solving this problem using graphs and mathematical expressions is then proposed. Finally, experiment is carried out, and results are given to show the effectiveness of the proposed spatial shift unwrapping technique.

The last issue of this thesis is the improvement of efficiency in SSE approach. In Chapter 4, we analyse the limitations of using traditional sinusoidal fringe pattern as the projected fringe pattern, which includes vulnerability to geometry distortion and nonlinear intensity distortion, such as screen door effect, gamma distortion and influence of environment and high computational complexity in IFSE method. To improve the efficiency, these limitations must be overcome. Since the SSE approach does not depend on the structure of the projected fringe pattern, we propose an improved SSE approach by using sawtooth fringe patterns instead of sinusoidal. The performance of sawtooth fringe pattern is then demonstrated through the analysis and experiments.

5.2 Suggestions for Future Research Work

Although we successfully implement the spatial shift detection approach for 3D profile measurement base on digital fringe projection, some issues still exist and require further attention in our future research work.

The most important yet challenging issue in our future work is the application of 3D profile measurement for those objects with complex surface. In our implementation, a dome set on a flat board is used as the object. This is a simple object with continuous surface. However, our final goal is the application of 3D profile measurement for any kinds of objects. Hence implementation of SSE approach on complex objects is required. This is a challenging issue since different objects may induce new problems in implementation. Further works are needed to solve these problems.

In Chapter 3, we presented a spatial shift unwrapping method. This method is based on the continuity of object surface. However, in practical application, objects with ideal continuous surface do not always exist. Since our shift unwrapping method cannot apply to those objects with discrete surface, to develop an improved shift

unwrapping method that can be used in this condition is also an important issue.

As demonstrated in Chapter 4, the sawtooth fringe pattern is introduced to overcome the limitations of traditional sinusoidal fringe pattern. Compared with sinusoidal fringe, the sawtooth fringe shows its strong counter-interference capability against the geometry and nonlinear distortion and improves the efficiency. Hence as a promising new developed method, further research attention on this kind of new fringe pattern should be considered in future work.

Another issue in highlight is the usage of colour fringe. The colour fringe projection is a significant promising technique in Fringe Pattern Profilometry. Compared with the black-and-white image we used in implementation, the coloured image contains more object height information and thereby greatly increasing data acquisition speed. As the fringe projection system we designed support colour fringe projection, effectively utilize the colour in Fringe Pattern Profilometry is also an attractive but challenging issue.

Finally, our experiment system should also be improved, thereby reduce the influence of environment hence to retrieve more accurate results.

REFERENCES

[1] F. Chen, G.. M. Brown and M. Song, “Overview of three-dimensional shape measurement using optical methods,” *Optical Engineering*, vol. 39, no. 1, pp.10-22, January 2000.

[2] *Mannual of Photogrammetry*, 4th ed. America Society of Photogrammetry, 1980.

[3] U. R. Dhond and J. K. Aggarwal, “Structure from stereo – a review,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 6, November/December 1989.

[4] M. Z. Brown, D. Burschka and G.. D. Hager, “Advances in computational stereo,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no.8, pp. 993-1008, August 2003.

[5] R. Zhang, P. Tsai, J. E. Cryer and M. Shah, “Shape from shading: A survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 8, pp. 690-706, August 1999.

[6] K. Kanatani and T. Chou, “Shape from texture: General principle,” *Artificical Intelligence*, vol. 38, pp. 1-48, 1989.

[7] S. K. Nayar and Y. Nakagawa, “Shape from focus,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no.8, pp.824-831, August 1994.

[8] M. Subbarao, T. Choi, “Accurate recovery of three dimensional shape from image focus,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 3, pp. 266-274, March 1995.

[9] C. Polhemus, “Two-wavelength interferometry,” *Applied Optics*, vol. 12, no. 9, pp. 2071-2074, September 1973.

[10] K. Creath, Y. Cheng and J. C. Wyant, “Contouring aspheric surfaces using two-wavelength phase-shifting interferometry”, *Optica Acta*, vol. 32, no. 12, pp. 1455-1464, May 1985.

[11] Y. Cheng and J. C. Wyant, “Multiple-wavelength phase-shift interferometry,” *Applied Optics*, vol. 24, no. 6, pp. 804-807, March 1985.

[12] K. Haines and B. P. Hildebrand, “Contour generation by wavefront reconstruction”, *Physics Letters*, vol. 19, no. 1, pp.10-11, September 1965.

[13] J. S. Zelenka and J. R. Varner, “Multiple-index holographic contouring,” *Applied Optics*, vol. 8, pp. 1431-1434, 1969.

[14] Y. Y. Hung, J. L. Turner, M. Tafralian, J. D. Hovanesian and C. E. Taylor, “Optical method for measuring contour slopes of an object,” *Applied Optics*, vol. 17, no. 1, pp. 128-131, January 1978.

[15] H. El-Ghandoor, “Tomographic investigation of the refractive index profiling using speckle photography technique,” *Optics Communications*, vol. 133, pp. 33-38, January 1997.

[16] N. Abramson, “Holographic contouring by translation,” *Applied Optics*, vol. 15, pp. 1018-1022, 1976.

[17] R. Rodriguez-Vera, D.Kerr and F. Mendoza-Santoyo, “Electronic speckle contouring,” *Journal of the Optical Society of America A*, vol. 9, no. 11, pp. 2000-2008, November 1992.

[18] R. S. Sirohi and F. S. Chau, *Optical Methods of Measurement Wholefield Techniques*, ser. Optical Engineering. Marcel Dekker, 1999, ch. 6, pp. 127-182.

[19] H. Takasaki, “Moiré topography”, *Applied Optics*, vol. 9, no. 6, pp. 1467-1472, June 1970.

[20] R. Harding and R. Tair, “Moiré techniques applied to automated inspection of machined parts,” in *Proceedings of SME Vision '86 conference*, Detroit, MI, 1986

[21] I. Moring, H. Ailisto, “Active 3-D vision system for automatic model-based shape inspection,” *Optics and Lasers in Engineering*, vol. 10, pp. 149-160, 1989.

[22] J. S. Massa, G. S. Buller, A. C. Walker, S. Cova, M. Umasuthan, and A. M. Wallace, “Time-of-flight optical ranging system based on time correlated single-photon counting,” *Applied Optics*, vol. 37, no. 31, pp. 7298-7304, November 1998.

[23] J. Batlle, E. Mouaddib and J. Salvi, “Recent progress in coded structured light as a technique to solve the correspondence problem: A survey,” *Pattern Recoginition*, vol. 31, no. 7, pp. 963-982, 1998.

[24] J. Salvi, J. Pages and J. Batlle, “Pattern codification strategies in structured light systems,” *Pattern Recoginition*, vol. 37, pp. 827-849, 2004.

[25] Z. Ji and M. C. Leu, “Design of optical triangulation devices,” *Optics and Laser Technology*, vol. 21, no. 5, pp. 335-338, 1989.

[26] M. Takeda, H. Ina, and S. Kobayashi, “Fourier-transform method of fringe-pattern analysis for computer-based topography and interferometry,” *Journal of the Optical Society of America A*, vol. 72, pp. 156–160, 1982.

[27] M. Takeda and K. Mutoh, “Fourier transform profilometry for the automatic measurement of 3-D object shapes,” *Applied Optics*, vol. 22, pp. 3977–3982, 1983.

[28] X. Su and W. Chen, “Fourier transform profilometry: a review,” *Optics and Lasers in Engineering*, vol. 35, pp. 263-284, 2001.

[29] H. Zhang, M. J. Lalor, and D. R. Burton, “Spatiotemporal phase unwrapping for the measurement of discontinuous objects in dynamic fringe-projection phase-shifting profilometry,” *Applied Optics*, vol. 38, pp. 3534–3541, June 1999.

[30] M. Halioua and H. C. Liu, “Optical three-dimensional sensing by phase measuring profilometry,” *Optics and Lasers in Engineering*, vol. 11, pp.185-215, 1989.

[31] J. Li, H. Su, and X. Su, “Two-frequency grating used in phase-measuring profilometry,” *Applied Optics*, vol. 36, pp. 277–280, January 1997.

[32] X. Su, L. Su, W. Li, and L. Xiang, “New 3D profilometry based on modulation measurement,” *Proceedings of SPIE*, Vol. 3853, pp. 1–7, 1998.

[33] S. Toyooka and M. Tominga, “Spatial fringe scanning for optical phase measurement,” *Optics Communications*, Vol. 51, pp. 68–70, 1984.

[34] S. Toyooka and Y. Iwaasa, “Automatic profilometry of 3-D diffuse objects by spatial phase detection,” *Applied Optics*, vol. 25, no. 10, pp. 1630–1633, May 1986.

[35] R. Rodriguez-Vera and M. Servin, “Phase locked loop profilometry,” *Optics and Lasers Technology*, vol. 26, no. 6, pp. 393–398, 1994.

[36] D. M. Meadows, W. O. Johnson, and J. B. Allen, “Generation of surface contours by moiré patterns,” *Applied Optics*, vol. 9, no. 4, pp. 942–947, April 1970.

[37] A. Asundi and Z. Wensen, “Unified calibration technique and its applications in

optical triangular profilometry," *Applied Optics*, vol. 38, no. 16, pp. 3556–3561, June 1999.

[38] C. Wust and D. W. Capson, "Surface profile measurement using color fringe projection," *MVA*, vol. 4, pp. 193–203, 1991.

[39] P. Huang, Q. Ho, F. Jin, and F. Chiang, "Colour-enhanced digitial fringe projection technique for high-speed 3-D surface contouring," *Optics Engineering*, vol. 38, pp. 1065–1071, 1999.

[40] A. J. Moore and F. Mendoza-Santoyo, "Phase demodulation in the space domain without a fringe carrier," *Optics and Lasers in Engineering*, vol. 23, pp. 319–330, 1995.

[41] J. Villa, M. Servin, and L. Castillo, "Profilometry for the measurement of 3-D object shapes based on regularized filters," *Optics Communication*, vol. 161, pp. 13–18, 1999.

[42] Y. Hu, J. Xi, E. Li, J. Chicharo, and Z. Yang, "Three-dimensional profilometry based on shift estimation of projected fringe patterns," *Applied Optics*, vol. 45, no. 4, pp. 678–687, February 2006.

[43] Y. Hu, J. Xi, Z. Yang, E. Li, and J. Chicharo, "Generalized analysis model for fringe pattern profilometry," in *IEEE Instrumentation and Measurement Conference (IMTC)*, Ottawa, Canada, May 2005, pp. 1951–1955.

[44] Y. Hu, J. Xi, J. Chicharo, W. Cheng, Z. Yang, "Inverse Function Analysis Method for Fringe Pattern Profilometry", *IEEE Transactions on Instrumentation and Measurement*, vol. 58, pp. 3305–3314, 2009.

[45] C. F. Gerald and P. O. Wheatley, *Applied Numerical Analysis*, 5th ed. Addison-Wesley publishing company, 1994.

[46] "HITACHI CP-X260 series", http://use.www.projectorcentral.com/pdf/projector_spec _3254.pdf.

[47] K. Itoh, "Analysis of the phase unwrapping algorithm", *Applied Optics*, vol. 21, Issue 14, pp. 2470-2470, July 1982

[48] D. C. Ghiglia, M. D. Pritt, *Two-Dimensional Phase Unwrapping: Theory, Algorithms, and Software*. John Wiley & Sons, 1998

[49] C. R. Coggrave, J. M. Huntley, “Optimization of a shape measurement system based on spatial light modulators”, *Optical Engineering*, vol. 39, no. 1, pp. 91-98, January 2000.

[50] M. J. Baker, J. F. Chicharo and J. Xi, “An Investigation into Temporal Gamma Luminance for Digital Fringe Fourier Transform Profilometers”, *IEEE International symposium on Intelligent Signal Processing, WISP 2007,* Madrid, Spain, October 3-5, 2007

[51] M. J. Baker, J. Xi and J. F. Chicharo, “Elimination of γ Non-linear Luminance Effects for Digital Video Projection Phase Measuring Profilometers”, *4th IEEE International Symposium on Electronic Design, Test & Applications, DELTA 2008,* Hong Kong, January 23-25 2008.

APPENDIX-Program Codes

1. Sinusoidal Fringe Pattern Generate Program (FringeSin.m):

```
clear

% ------------------ Red Fringe -----------------------------------------
RC = 1;          % Red Contrast (Between 0 - 1)
RF = 500;      % Red Frequency
RP = 0;          % Red Phase (Only can be 0, 1, 2)
t = 500;
x = 1:t;
T = 1;
Red = 0.5 + 0.5*RC*cos(2*pi*(RF/10000)*x + RP*(1/3));

RFringe = zeros(t,t);
for n=1:t
      RFringe(n,:) = Red;
end

% ------------------ Green Fringe ----------------------------------------
GC = 1;          % Green Contrast (Between 0 - 1)
GF = 500;      % Green Frequency
GP = 0;          % Green Phase (Only can be 0, 1, 2)
t = 500;
x = 1:t;
Green = 0.5 + 0.5*GC*cos(2*pi*(GF/10000)*x + GP*(1/3));

GFringe = zeros(t,t);
for n=1:t
      GFringe(n,:) = Green;
end

% ------------------ Blue Fringe -----------------------------------------
BC = 1;          % Blue Contrast (Between 0 - 1)
BF = 500;     % Blue Frequency
BP = 0;          % Blue Phase (Only can be 0, 1, 2)
t = 500;
x = 1:t;
Blue = 0.5 + 0.5*BC*cos(2*pi*(BF/10000)*x + BP*(1/3));
```

```
BFringe = zeros(t,t);
for n=1:t
      BFringe(n,:) = Blue;
end

% ------------------ Generate Colour Fringe -----------------------------
FRINGE(:,:,1) = RFringe;
FRINGE(:,:,2) = GFringe;
FRINGE(:,:,3) = BFringe;

% ------------------ Generate Colour Image -----------------------------
imshow(FRINGE)
```

2. Implementation of SSE using Sinusoidal Pattern (SinIFSE.m):

```
clear

% ------------------ Reading Image Data ---------------------------------
[img1] = imread('SinImage.bmp');
[img2] = imread('SinRef.bmp');

% ------------------ Filter Design ----------------------------------------
b=fir2(128,[0.0 0.15 0.85 1.0],[1,0,0,0]);
ff=fft(b);
ff=abs(ff);

% ------------------ Pre-processing ---------------------------------------
y1 = 750;                                          % Select y1 = 750
RRfringes = double(img2(:,y1,1))/255;      % Reading Reference Red fringe
Rfringes = double(img1(:,y1,1))/255;       % Reading Object Red fringe

RRfringe = filter(b,1,RRfringes);                % Filters Reference Fringe Data
Rfringe = filter(b,1,Rfringes);                  % Filters Object Fringe Data

RRfringe = RRfringe([121:982]);
Rfringe =    Rfringe([144:1005]);

% Finding Monotonic Intervals for Reference Fringe
RRCounter = 1;
while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
```

```
while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx1 = RRCounter;

while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx2 = RRCounter;

while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx3 = RRCounter;

while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx4 = RRCounter;

while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx5 = RRCounter;

while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx6 = RRCounter;
```

```
while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx7 = RRCounter;

while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx8 = RRCounter;

while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx9 = RRCounter;

while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx10 = RRCounter;

while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx11 = RRCounter;

while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
```

```
while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx12 = RRCounter;

while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx13 = RRCounter;

while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx14 = RRCounter;

while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx15 = RRCounter;

while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx16 = RRCounter;

while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx17 = RRCounter;
```

```
RRx18 = length(RRfringe);

RRf1 = RRfringe([1:RRx1]);
RRf2 = RRfringe([RRx1+1:RRx2]);
RRf3 = RRfringe([RRx2+1:RRx3]);
RRf4 = RRfringe([RRx3+1:RRx4]);
RRf5 = RRfringe([RRx4+1:RRx5]);
RRf6 = RRfringe([RRx5+1:RRx6]);
RRf7 = RRfringe([RRx6+1:RRx7]);
RRf8 = RRfringe([RRx7+1:RRx8]);
RRf9 = RRfringe([RRx8+1:RRx9]);
RRf10 = RRfringe([RRx9+1:RRx10]);
RRf11 = RRfringe([RRx10+1:RRx11]);
RRf12 = RRfringe([RRx11+1:RRx12]);
RRf13 = RRfringe([RRx12+1:RRx13]);
RRf14 = RRfringe([RRx13+1:RRx14]);
RRf15 = RRfringe([RRx14+1:RRx15]);
RRf16 = RRfringe([RRx15+1:RRx16]);
RRf17 = RRfringe([RRx16+1:RRx17]);
RRf18 = RRfringe([RRx17+1:RRx18]);

RRfs1 = RRfringe([2:RRx1-1]);
RRfs2 = RRfringe([RRx1+2:RRx2-1]);
RRfs3 = RRfringe([RRx2+2:RRx3-1]);
RRfs4 = RRfringe([RRx3+2:RRx4-1]);
RRfs5 = RRfringe([RRx4+2:RRx5-1]);
RRfs6 = RRfringe([RRx5+2:RRx6-1]);
RRfs7 = RRfringe([RRx6+2:RRx7-1]);
RRfs8 = RRfringe([RRx7+2:RRx8-1]);
RRfs9 = RRfringe([RRx8+2:RRx9-1]);
RRfs10 = RRfringe([RRx9+2:RRx10-1]);
RRfs11 = RRfringe([RRx10+2:RRx11-1]);
RRfs12 = RRfringe([RRx11+2:RRx12-1]);
RRfs13 = RRfringe([RRx12+2:RRx13-1]);
RRfs14 = RRfringe([RRx13+2:RRx14-1]);
RRfs15 = RRfringe([RRx14+2:RRx15-1]);
RRfs16 = RRfringe([RRx15+2:RRx16-1]);
RRfs17 = RRfringe([RRx16+2:RRx17-1]);
RRfs18 = RRfringe([RRx17+2:RRx18-1]);


% Finding Monotonic Intervals for Object Fringe
RCounter = 1;
while Rfringe(RCounter + 1) > Rfringe(RCounter);
```

```
    RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) < Rfringe(RCounter);
    RCounter = RCounter + 1;
end
Rx1 = RCounter;

while Rfringe(RCounter + 1) < Rfringe(RCounter);
    RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) > Rfringe(RCounter);
    RCounter = RCounter + 1;
end
Rx2 = RCounter;

while Rfringe(RCounter + 1) > Rfringe(RCounter);
    RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) < Rfringe(RCounter);
    RCounter = RCounter + 1;
end
Rx3 = RCounter;

while Rfringe(RCounter + 1) < Rfringe(RCounter);
    RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) > Rfringe(RCounter);
    RCounter = RCounter + 1;
end
Rx4 = RCounter;

while Rfringe(RCounter + 1) > Rfringe(RCounter);
    RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) < Rfringe(RCounter);
    RCounter = RCounter + 1;
end
Rx5 = RCounter;

while Rfringe(RCounter + 1) < Rfringe(RCounter);
    RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) > Rfringe(RCounter);
    RCounter = RCounter + 1;
```

```
end
Rx6 = RCounter;

while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx7 = RCounter;

while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx8 = RCounter;

while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx9 = RCounter;

while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx10 = RCounter;

while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx11 = RCounter;

while Rfringe(RCounter + 1) < Rfringe(RCounter);
```

```
    RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) > Rfringe(RCounter);
    RCounter = RCounter + 1;
end
Rx12 = RCounter;

while Rfringe(RCounter + 1) > Rfringe(RCounter);
    RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) < Rfringe(RCounter);
    RCounter = RCounter + 1;
end
Rx13 = RCounter;

while Rfringe(RCounter + 1) < Rfringe(RCounter);
    RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) > Rfringe(RCounter);
    RCounter = RCounter + 1;
end
Rx14 = RCounter;

while Rfringe(RCounter + 1) > Rfringe(RCounter);
    RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) < Rfringe(RCounter);
    RCounter = RCounter + 1;
end
Rx15 = RCounter;

while Rfringe(RCounter + 1) < Rfringe(RCounter);
    RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) > Rfringe(RCounter);
    RCounter = RCounter + 1;
end
Rx16 = RCounter;

while Rfringe(RCounter + 1) > Rfringe(RCounter);
    RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) < Rfringe(RCounter);
    RCounter = RCounter + 1;
```

```
end
Rx17 = RCounter;
Rx18 = length(Rfringe);

Rf1 = Rfringe([1:Rx1]);
Rf2 = Rfringe([Rx1+1:Rx2]);
Rf3 = Rfringe([Rx2+1:Rx3]);
Rf4 = Rfringe([Rx3+1:Rx4]);
Rf5 = Rfringe([Rx4+1:Rx5]);
Rf6 = Rfringe([Rx5+1:Rx6]);
Rf7 = Rfringe([Rx6+1:Rx7]);
Rf8 = Rfringe([Rx7+1:Rx8]);
Rf9 = Rfringe([Rx8+1:Rx9]);
Rf10 = Rfringe([Rx9+1:Rx10]);
Rf11 = Rfringe([Rx10+1:Rx11]);
Rf12 = Rfringe([Rx11+1:Rx12]);
Rf13 = Rfringe([Rx12+1:Rx13]);
Rf14 = Rfringe([Rx13+1:Rx14]);
Rf15 = Rfringe([Rx14+1:Rx15]);
Rf16 = Rfringe([Rx15+1:Rx16]);
Rf17 = Rfringe([Rx16+1:Rx17]);
Rf18 = Rfringe([Rx17+1:Rx18]);


% ----- Normalization of Fringe and IFSE Processing(including Shift Unwrapping) --------
k = 10;                    % Set initial degree of polynomial

% Section 1
x = 1:length(RRfs1);

% Normalization of Fringe
RefH = (max(RRf1) - min(RRf1));
ObjH = (max(Rf1) - min(Rf1));
Hadj = RefH / ObjH ;
Objmid = (max(Rf1) + min(Rf1))/2;
Refmid = (max(RRf1) + min(RRf1))/2;
Rf1 = ((Rf1 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs1,x',k);
Rfx1 = polyval(j,Rf1);


% Section 2
```

```
x = 1:length(RRfs2);

% Normalization of Fringe
RefH = (max(RRf2) - min(RRf2));
ObjH = (max(Rf2) - min(Rf2));
Hadj = RefH / ObjH ;
Objmid = (max(Rf2) + min(Rf2))/2;
Refmid = (max(RRf2) + min(RRf2))/2;
Rf2 = ((Rf2 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs2,x',k);
Rfx2 = polyval(j,Rf2) + RRx1;


% Section 3
x = 1:length(RRfs3);

% Normalization of Fringe
RefH = (max(RRf3) - min(RRf3));
ObjH = (max(Rf3) - min(Rf3));
Hadj = RefH / ObjH ;
Objmid = (max(Rf3) + min(Rf3))/2;
Refmid = (max(RRf3) + min(RRf3))/2;
Rf3 = ((Rf3 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs3,x',k);
Rfx3 = polyval(j,Rf3) + RRx2;


% Section 4
x = 1:length(RRfs4);

% Normalization of Fringe
RefH = (max(RRf4) - min(RRf4));
ObjH = (max(Rf4) - min(Rf4));
Hadj = RefH / ObjH ;
Objmid = (max(Rf4) + min(Rf4))/2;
Refmid = (max(RRf4) + min(RRf4))/2;
Rf4 = ((Rf4 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs4,x',k);
```

```
Rfx4 = polyval(j,Rf4) + RRx3;


% Section 5
x = 1:length(RRfs5);

% Normalization of Fringe
RefH = (max(RRf5) - min(RRf5));
ObjH = (max(Rf5) - min(Rf5));
Hadj = RefH / ObjH ;
Objmid = (max(Rf5) + min(Rf5))/2;
Refmid = (max(RRf5) + min(RRf5))/2;
Rf5 = ((Rf5 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs5,x',k);
Rfx5 = polyval(j,Rf5) + RRx4;


% Section 6
x = 1:length(RRfs6);

% Normalization of Fringe
RefH = (max(RRf6) - min(RRf6));
ObjH = (max(Rf6) - min(Rf6));
Hadj = RefH / ObjH ;
Objmid = (max(Rf6) + min(Rf6))/2;
Refmid = (max(RRf6) + min(RRf6))/2;
Rf6 = ((Rf6 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs6,x',k);
Rfx6 = polyval(j,Rf6) + RRx5;


% Section 7
x = 1:length(RRfs7);

% Normalization of Fringe
RefH = (max(RRf7) - min(RRf7));
ObjH = (max(Rf7) - min(Rf7));
Hadj = RefH / ObjH ;
Objmid = (max(Rf7) + min(Rf7))/2;
Refmid = (max(RRf7) + min(RRf7))/2;
```

```
Rf7 = ((Rf7 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs7,x',k);
Rfx7 = polyval(j,Rf7) + RRx6;


% Section 8
x = 1:length(RRfs8);

% Normalization of Fringe
RefH = (max(RRf8) - min(RRf8));
ObjH = (max(Rf8) - min(Rf8));
Hadj = RefH / ObjH ;
Objmid = (max(Rf8) + min(Rf8))/2;
Refmid = (max(RRf8) + min(RRf8))/2;
Rf8 = ((Rf8 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs8,x',k);
Rfx8 = polyval(j,Rf8) + RRx7;


% Section 9
x = 1:length(RRfs9);

% Normalization of Fringe
RefH = (max(RRf9) - min(RRf9));
ObjH = (max(Rf9) - min(Rf9));
Hadj = RefH / ObjH ;
Objmid = (max(Rf9) + min(Rf9))/2;
Refmid = (max(RRf9) + min(RRf9))/2;
Rf9 = ((Rf9 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs9,x',k);
Rfx9 = polyval(j,Rf9) + RRx8;


% Section 10
x = 1:length(RRfs10);

% Normalization of Fringe
RefH = (max(RRf10) - min(RRf10));
```

```
ObjH = (max(Rf10) - min(Rf10));
Hadj = RefH / ObjH ;
Objmid = (max(Rf10) + min(Rf10))/2;
Refmid = (max(RRf10) + min(RRf10))/2;
Rf10 = ((Rf10 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs10,x',k);
Rfx10 = polyval(j,Rf10) + RRx9;


% Section 11
x = 1:length(RRfs11);

% Normalization of Fringe
RefH = (max(RRf11) - min(RRf11));
ObjH = (max(Rf11) - min(Rf11));
Hadj = RefH / ObjH ;
Objmid = (max(Rf11) + min(Rf11))/2;
Refmid = (max(RRf11) + min(RRf11))/2;
Rf11 = ((Rf11 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs11,x',k);
Rfx11 = polyval(j,Rf11) + RRx10;


% Section 12
x = 1:length(RRfs12);

% Normalization of Fringe
RefH = (max(RRf12) - min(RRf12));
ObjH = (max(Rf12) - min(Rf12));
Hadj = RefH / ObjH ;
Objmid = (max(Rf12) + min(Rf12))/2;
Refmid = (max(RRf12) + min(RRf12))/2;
Rf12 = ((Rf12 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs12,x',k);
Rfx12 = polyval(j,Rf12) + RRx11;


% Section 13
```

```
x = 1:length(RRfs13);

% Normalization of Fringe
RefH = (max(RRf13) - min(RRf13));
ObjH = (max(Rf13) - min(Rf13));
Hadj = RefH / ObjH ;
Objmid = (max(Rf13) + min(Rf13))/2;
Refmid = (max(RRf13) + min(RRf13))/2;
Rf13 = ((Rf13 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs13,x',k);
Rfx13 = polyval(j,Rf13) + RRx12;


% Section 14
x = 1:length(RRfs14);

% Normalization of Fringe
RefH = (max(RRf14) - min(RRf14));
ObjH = (max(Rf14) - min(Rf14));
Hadj = RefH / ObjH ;
Objmid = (max(Rf14) + min(Rf14))/2;
Refmid = (max(RRf14) + min(RRf14))/2;
Rf14 = ((Rf14 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs14,x',k);
Rfx14 = polyval(j,Rf14) + RRx13;


% Section 15
x = 1:length(RRfs15);

% Normalization of Fringe
RefH = (max(RRf15) - min(RRf15));
ObjH = (max(Rf15) - min(Rf15));
Hadj = RefH / ObjH ;
Objmid = (max(Rf15) + min(Rf15))/2;
Refmid = (max(RRf15) + min(RRf15))/2;
Rf15 = ((Rf15 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs15,x',k);
```

```
Rfx15 = polyval(j,Rf15) + RRx14;


% Section 16
x = 1:length(RRfs16);

% Normalization of Fringe
RefH = (max(RRf16) - min(RRf16));
ObjH = (max(Rf16) - min(Rf16));
Hadj = RefH / ObjH ;
Objmid = (max(Rf16) + min(Rf16))/2;
Refmid = (max(RRf16) + min(RRf16))/2;
Rf16 = ((Rf16 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs16,x',k);
Rfx16 = polyval(j,Rf16) + RRx15;


% Section 17
x = 1:length(RRfs17);

% Normalization of Fringe
RefH = (max(RRf17) - min(RRf17));
ObjH = (max(Rf17) - min(Rf17));
Hadj = RefH / ObjH ;
Objmid = (max(Rf17) + min(Rf17))/2;
Refmid = (max(RRf17) + min(RRf17))/2;
Rf17 = ((Rf17 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs17,x',k);
Rfx17 = polyval(j,Rf17) + RRx16;


% Section 18
x = 1:length(RRfs18);

% Normalization of Fringe
RefH = (max(RRf18) - min(RRf18));
ObjH = (max(Rf18) - min(Rf18));
Hadj = RefH / ObjH ;
Objmid = (max(Rf18) + min(Rf18))/2;
Refmid = (max(RRf18) + min(RRf18))/2;
```

```
Rf18 = ((Rf18 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs18,x',k);
Rfx18 = polyval(j,Rf18) + RRx17;

% Retrieve u(x)
Rfx = [Rfx1',Rfx2',Rfx3',Rfx4',Rfx5',Rfx6',Rfx7',Rfx8',Rfx9',Rfx10',Rfx11',Rfx12',Rfx13',Rfx14',
Rfx15',Rfx16',Rfx17',Rfx18']';
x = 1:862;
U = x' - Rfx;

% Calculate Height Distribution
l0 = 1295;
d0 = 350;
h = (U*l0)/d0*0.1796;

% Plot Unwrapped u(x) and Height Distribution
subplot(2,1,1);plot(U,'k');title('Unwrapped u(x)');
subplot(2,1,2);plot(h,'k');title('h(x)');
```

3. Implementation of SSE using Sinusoidal Pattern with Object Surface Reconstruction (SinIFSEReconstruct.m):

```
clear

% ------------------ Reading Image Data ----------------------------------
[img1] = imread('SinImage.bmp');
[img2] = imread('SinRef.bmp');

% ------------------ Filter Design ----------------------------------------
b=fir2(128,[0.0 0.15 0.85 1.0],[1,0,0,0]);
ff=fft(b);
ff=abs(ff);

% ------------------ Pre-processing ---------------------------------------
for y = 4:20:900;
RRfringes = double(img2(:,y+300,1))/255;      % Reading Reference Red fringe
Rfringes = double(img1(:,y+300,1))/255;       % Reading Object Red fringe

RRfringe = filter(b,1,RRfringes);                % Filters Reference Fringe Data
```

```
Rfringe = filter(b,1,Rfringes);                    % Filters Object Fringe Data

RRfringe = RRfringe([121:982]);
Rfringe =    Rfringe([144:1005]);

% Finding Monotonic Intervals for Reference Fringe
RRCounter = 1;
while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx1 = RRCounter;

while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx2 = RRCounter;

while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx3 = RRCounter;

while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx4 = RRCounter;

while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
```

```
end
RRx5 = RRCounter;

while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx6 = RRCounter;

while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx7 = RRCounter;

while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx8 = RRCounter;

while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx9 = RRCounter;

while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx10 = RRCounter;

while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
```

```
    RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
    RRCounter = RRCounter + 1;
end
RRx11 = RRCounter;

while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
    RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
    RRCounter = RRCounter + 1;
end
RRx12 = RRCounter;

while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
    RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
    RRCounter = RRCounter + 1;
end
RRx13 = RRCounter;

while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
    RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
    RRCounter = RRCounter + 1;
end
RRx14 = RRCounter;

while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
    RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
    RRCounter = RRCounter + 1;
end
RRx15 = RRCounter;

while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
    RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
    RRCounter = RRCounter + 1;
```

```
end
RRx16 = RRCounter;

while RRfringe(RRCounter + 1) > RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
while RRfringe(RRCounter + 1) < RRfringe(RRCounter);
      RRCounter = RRCounter + 1;
end
RRx17 = RRCounter;
RRx18 = length(RRfringe);

RRf1 = RRfringe([1:RRx1]);
RRf2 = RRfringe([RRx1+1:RRx2]);
RRf3 = RRfringe([RRx2+1:RRx3]);
RRf4 = RRfringe([RRx3+1:RRx4]);
RRf5 = RRfringe([RRx4+1:RRx5]);
RRf6 = RRfringe([RRx5+1:RRx6]);
RRf7 = RRfringe([RRx6+1:RRx7]);
RRf8 = RRfringe([RRx7+1:RRx8]);
RRf9 = RRfringe([RRx8+1:RRx9]);
RRf10 = RRfringe([RRx9+1:RRx10]);
RRf11 = RRfringe([RRx10+1:RRx11]);
RRf12 = RRfringe([RRx11+1:RRx12]);
RRf13 = RRfringe([RRx12+1:RRx13]);
RRf14 = RRfringe([RRx13+1:RRx14]);
RRf15 = RRfringe([RRx14+1:RRx15]);
RRf16 = RRfringe([RRx15+1:RRx16]);
RRf17 = RRfringe([RRx16+1:RRx17]);
RRf18 = RRfringe([RRx17+1:RRx18]);

RRfs1 = RRfringe([2:RRx1-1]);
RRfs2 = RRfringe([RRx1+2:RRx2-1]);
RRfs3 = RRfringe([RRx2+2:RRx3-1]);
RRfs4 = RRfringe([RRx3+2:RRx4-1]);
RRfs5 = RRfringe([RRx4+2:RRx5-1]);
RRfs6 = RRfringe([RRx5+2:RRx6-1]);
RRfs7 = RRfringe([RRx6+2:RRx7-1]);
RRfs8 = RRfringe([RRx7+2:RRx8-1]);
RRfs9 = RRfringe([RRx8+2:RRx9-1]);
RRfs10 = RRfringe([RRx9+2:RRx10-1]);
RRfs11 = RRfringe([RRx10+2:RRx11-1]);
RRfs12 = RRfringe([RRx11+2:RRx12-1]);
RRfs13 = RRfringe([RRx12+2:RRx13-1]);
```

```
RRfs14 = RRfringe([RRx13+2:RRx14-1]);
RRfs15 = RRfringe([RRx14+2:RRx15-1]);
RRfs16 = RRfringe([RRx15+2:RRx16-1]);
RRfs17 = RRfringe([RRx16+2:RRx17-1]);
RRfs18 = RRfringe([RRx17+2:RRx18-1]);


% Finding Monotonic Intervals for Object Fringe
RCounter = 1;
while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx1 = RCounter;

while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx2 = RCounter;

while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx3 = RCounter;

while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx4 = RCounter;

while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
```

```
while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx5 = RCounter;

while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx6 = RCounter;

while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx7 = RCounter;

while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx8 = RCounter;

while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx9 = RCounter;

while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx10 = RCounter;
```

```
while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx11 = RCounter;

while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx12 = RCounter;

while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx13 = RCounter;

while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx14 = RCounter;

while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx15 = RCounter;

while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
```

```
while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx16 = RCounter;

while Rfringe(RCounter + 1) > Rfringe(RCounter);
      RCounter = RCounter + 1;
end
while Rfringe(RCounter + 1) < Rfringe(RCounter);
      RCounter = RCounter + 1;
end
Rx17 = RCounter;
Rx18 = length(Rfringe);

Rf1 = Rfringe([1:Rx1]);
Rf2 = Rfringe([Rx1+1:Rx2]);
Rf3 = Rfringe([Rx2+1:Rx3]);
Rf4 = Rfringe([Rx3+1:Rx4]);
Rf5 = Rfringe([Rx4+1:Rx5]);
Rf6 = Rfringe([Rx5+1:Rx6]);
Rf7 = Rfringe([Rx6+1:Rx7]);
Rf8 = Rfringe([Rx7+1:Rx8]);
Rf9 = Rfringe([Rx8+1:Rx9]);
Rf10 = Rfringe([Rx9+1:Rx10]);
Rf11 = Rfringe([Rx10+1:Rx11]);
Rf12 = Rfringe([Rx11+1:Rx12]);
Rf13 = Rfringe([Rx12+1:Rx13]);
Rf14 = Rfringe([Rx13+1:Rx14]);
Rf15 = Rfringe([Rx14+1:Rx15]);
Rf16 = Rfringe([Rx15+1:Rx16]);
Rf17 = Rfringe([Rx16+1:Rx17]);
Rf18 = Rfringe([Rx17+1:Rx18]);


% ----- Normalization of Fringe and IFSE Processing(including Shift Unwrapping) --------
k = 10;                    % Set initial degree of polynomial

% Section 1
x = 1:length(RRfs1);

% Normalization of Fringe
RefH = (max(RRf1) - min(RRf1));
ObjH = (max(Rf1) - min(Rf1));
Hadj = RefH / ObjH ;
```

```
Objmid = (max(Rf1) + min(Rf1))/2;
Refmid = (max(RRf1) + min(RRf1))/2;
Rf1 = ((Rf1 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs1,x',k);
Rfx1 = polyval(j,Rf1);


% Section 2
x = 1:length(RRfs2);

% Normalization of Fringe
RefH = (max(RRf2) - min(RRf2));
ObjH = (max(Rf2) - min(Rf2));
Hadj = RefH / ObjH ;
Objmid = (max(Rf2) + min(Rf2))/2;
Refmid = (max(RRf2) + min(RRf2))/2;
Rf2 = ((Rf2 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs2,x',k);
Rfx2 = polyval(j,Rf2) + RRx1;


% Section 3
x = 1:length(RRfs3);

% Normalization of Fringe
RefH = (max(RRf3) - min(RRf3));
ObjH = (max(Rf3) - min(Rf3));
Hadj = RefH / ObjH ;
Objmid = (max(Rf3) + min(Rf3))/2;
Refmid = (max(RRf3) + min(RRf3))/2;
Rf3 = ((Rf3 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs3,x',k);
Rfx3 = polyval(j,Rf3) + RRx2;


% Section 4
x = 1:length(RRfs4);
```

```
% Normalization of Fringe
RefH = (max(RRf4) - min(RRf4));
ObjH = (max(Rf4) - min(Rf4));
Hadj = RefH / ObjH ;
Objmid = (max(Rf4) + min(Rf4))/2;
Refmid = (max(RRf4) + min(RRf4))/2;
Rf4 = ((Rf4 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs4,x',k);
Rfx4 = polyval(j,Rf4) + RRx3;


% Section 5
x = 1:length(RRfs5);

% Normalization of Fringe
RefH = (max(RRf5) - min(RRf5));
ObjH = (max(Rf5) - min(Rf5));
Hadj = RefH / ObjH ;
Objmid = (max(Rf5) + min(Rf5))/2;
Refmid = (max(RRf5) + min(RRf5))/2;
Rf5 = ((Rf5 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs5,x',k);
Rfx5 = polyval(j,Rf5) + RRx4;


% Section 6
x = 1:length(RRfs6);

% Normalization of Fringe
RefH = (max(RRf6) - min(RRf6));
ObjH = (max(Rf6) - min(Rf6));
Hadj = RefH / ObjH ;
Objmid = (max(Rf6) + min(Rf6))/2;
Refmid = (max(RRf6) + min(RRf6))/2;
Rf6 = ((Rf6 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs6,x',k);
Rfx6 = polyval(j,Rf6) + RRx5;
```

```
% Section 7
x = 1:length(RRfs7);

% Normalization of Fringe
RefH = (max(RRf7) - min(RRf7));
ObjH = (max(Rf7) - min(Rf7));
Hadj = RefH / ObjH ;
Objmid = (max(Rf7) + min(Rf7))/2;
Refmid = (max(RRf7) + min(RRf7))/2;
Rf7 = ((Rf7 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs7,x',k);
Rfx7 = polyval(j,Rf7) + RRx6;


% Section 8
x = 1:length(RRfs8);

% Normalization of Fringe
RefH = (max(RRf8) - min(RRf8));
ObjH = (max(Rf8) - min(Rf8));
Hadj = RefH / ObjH ;
Objmid = (max(Rf8) + min(Rf8))/2;
Refmid = (max(RRf8) + min(RRf8))/2;
Rf8 = ((Rf8 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs8,x',k);
Rfx8 = polyval(j,Rf8) + RRx7;


% Section 9
x = 1:length(RRfs9);

% Normalization of Fringe
RefH = (max(RRf9) - min(RRf9));
ObjH = (max(Rf9) - min(Rf9));
Hadj = RefH / ObjH ;
Objmid = (max(Rf9) + min(Rf9))/2;
Refmid = (max(RRf9) + min(RRf9))/2;
Rf9 = ((Rf9 - Objmid) * Hadj) + Refmid;
```

```
% IFSE processing with Shift Unwrapping
j = polyfit(RRfs9,x',k);
Rfx9 = polyval(j,Rf9) + RRx8;


% Section 10
x = 1:length(RRfs10);

% Normalization of Fringe
RefH = (max(RRf10) - min(RRf10));
ObjH = (max(Rf10) - min(Rf10));
Hadj = RefH / ObjH ;
Objmid = (max(Rf10) + min(Rf10))/2;
Refmid = (max(RRf10) + min(RRf10))/2;
Rf10 = ((Rf10 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs10,x',k);
Rfx10 = polyval(j,Rf10) + RRx9;


% Section 11
x = 1:length(RRfs11);

% Normalization of Fringe
RefH = (max(RRf11) - min(RRf11));
ObjH = (max(Rf11) - min(Rf11));
Hadj = RefH / ObjH ;
Objmid = (max(Rf11) + min(Rf11))/2;
Refmid = (max(RRf11) + min(RRf11))/2;
Rf11 = ((Rf11 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs11,x',k);
Rfx11 = polyval(j,Rf11) + RRx10;


% Section 12
x = 1:length(RRfs12);

% Normalization of Fringe
RefH = (max(RRf12) - min(RRf12));
ObjH = (max(Rf12) - min(Rf12));
Hadj = RefH / ObjH ;
```

```
Objmid = (max(Rf12) + min(Rf12))/2;
Refmid = (max(RRf12) + min(RRf12))/2;
Rf12 = ((Rf12 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs12,x',k);
Rfx12 = polyval(j,Rf12) + RRx11;


% Section 13
x = 1:length(RRfs13);

% Normalization of Fringe
RefH = (max(RRf13) - min(RRf13));
ObjH = (max(Rf13) - min(Rf13));
Hadj = RefH / ObjH ;
Objmid = (max(Rf13) + min(Rf13))/2;
Refmid = (max(RRf13) + min(RRf13))/2;
Rf13 = ((Rf13 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs13,x',k);
Rfx13 = polyval(j,Rf13) + RRx12;


% Section 14
x = 1:length(RRfs14);

% Normalization of Fringe
RefH = (max(RRf14) - min(RRf14));
ObjH = (max(Rf14) - min(Rf14));
Hadj = RefH / ObjH ;
Objmid = (max(Rf14) + min(Rf14))/2;
Refmid = (max(RRf14) + min(RRf14))/2;
Rf14 = ((Rf14 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs14,x',k);
Rfx14 = polyval(j,Rf14) + RRx13;


% Section 15
x = 1:length(RRfs15);
```

```
% Normalization of Fringe
RefH = (max(RRf15) - min(RRf15));
ObjH = (max(Rf15) - min(Rf15));
Hadj = RefH / ObjH ;
Objmid = (max(Rf15) + min(Rf15))/2;
Refmid = (max(RRf15) + min(RRf15))/2;
Rf15 = ((Rf15 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs15,x',k);
Rfx15 = polyval(j,Rf15) + RRx14;


% Section 16
x = 1:length(RRfs16);

% Normalization of Fringe
RefH = (max(RRf16) - min(RRf16));
ObjH = (max(Rf16) - min(Rf16));
Hadj = RefH / ObjH ;
Objmid = (max(Rf16) + min(Rf16))/2;
Refmid = (max(RRf16) + min(RRf16))/2;
Rf16 = ((Rf16 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs16,x',k);
Rfx16 = polyval(j,Rf16) + RRx15;


% Section 17
x = 1:length(RRfs17);

% Normalization of Fringe
RefH = (max(RRf17) - min(RRf17));
ObjH = (max(Rf17) - min(Rf17));
Hadj = RefH / ObjH ;
Objmid = (max(Rf17) + min(Rf17))/2;
Refmid = (max(RRf17) + min(RRf17))/2;
Rf17 = ((Rf17 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs17,x',k);
Rfx17 = polyval(j,Rf17) + RRx16;
```

```
% Section 18
x = 1:length(RRfs18);

% Normalization of Fringe
RefH = (max(RRf18) - min(RRf18));
ObjH = (max(Rf18) - min(Rf18));
Hadj = RefH / ObjH ;
Objmid = (max(Rf18) + min(Rf18))/2;
Refmid = (max(RRf18) + min(RRf18))/2;
Rf18 = ((Rf18 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs18,x',k);
Rfx18 = polyval(j,Rf18) + RRx17;

% Retrieve u(x)
Rfx = [Rfx1',Rfx2',Rfx3',Rfx4',Rfx5',Rfx6',Rfx7',Rfx8',Rfx9',Rfx10',Rfx11',Rfx12',Rfx13',Rfx14',
Rfx15',Rfx16',Rfx17',Rfx18']';
x = 1:862;
U = x' - Rfx;
Obj(:,y) = U;
end

% Reconstruct Object Surface
C = [0,900,0,900,0,80];
mesh(Obj),axis(C);colormap(gray);shading interp;
material shiny;
```

4. Sawtooth Fringe Pattern Generate Program (FringeSaw.m):

```
clear

% ------------------ Red Fringe -----------------------------------------
RC = 1;          % Red Contrast (Between 0 - 1)
RT = 25;         % Red Period
t = 500;
x = 1:t;
T = 1;
Red = RC*(x/RT - floor(x/RT));

RFringe = zeros(t,t);
```

```
for n=1:t
      RFringe(n,:) = Red;
end

% ------------------ Green Fringe ----------------------------------------
GC = 1;         % Green Contrast (Between 0 - 1)
GT = 25;        % Green Period
t = 500;
x = 1:t;
Green = GC*(x/GT - floor(x/GT));

GFringe = zeros(t,t);
for n=1:t
      GFringe(n,:) = Green;
end

% ------------------ Blue Fringe -----------------------------------------
BC = 1;         % Blue Contrast (Between 0 - 1)
BT = 25;        % Blue Period
t = 500;
x = 1:t;
Blue = BC*(x/BT - floor(x/BT));

BFringe = zeros(t,t);
for n=1:t
      BFringe(n,:) = Blue;
end

% ------------------ Generate Colour Fringe ------------------------------
FRINGE(:,:,1) = RFringe;
FRINGE(:,:,2) = GFringe;
FRINGE(:,:,3) = BFringe;

% ------------------ Generate Colour Image -------------------------------
imshow(FRINGE)
```

5. Implementation of SSE using Sawtooth Pattern (SawIFSE.m):

```
clear

% ------------------ Reading Image Data ----------------------------------
[img1] = imread('SawObj.bmp');
```

```
[img2] = imread('SawRef.bmp');

% ------------------ Pre-processing ----------------------------------------
y1 = 750;                                                    % Select y1 = 750
RRfringes = smooth(double(img2(:,y1,1))/255);      % Reference Red fringe
Rfringes = smooth(double(img1(:,y1,1))/255);        % Red fringe
RRfringe = smooth(RRfringes([234:1008]));
Rfringe =    smooth(Rfringes([262:1036]));

% Finding Monotonic Intervals for Reference Fringe
fix = 0.001;
RRCounter = 1;
while RRfringe(RRCounter + 1) - RRfringe(RRCounter) >    - fix;
      RRCounter = RRCounter + 1;
end
RRx1 = RRCounter;

while    RRfringe(RRCounter + 1) -    RRfringe(RRCounter) < + fix;
      RRCounter = RRCounter + 1;
end
RRx2 = RRCounter;

while RRfringe(RRCounter + 1) - RRfringe(RRCounter) >    - fix;
      RRCounter = RRCounter + 1;
end
RRx3 = RRCounter;

while    RRfringe(RRCounter + 1) -    RRfringe(RRCounter) < + fix;
      RRCounter = RRCounter + 1;
end
RRx4 = RRCounter;

while RRfringe(RRCounter + 1) - RRfringe(RRCounter) >    - fix;
      RRCounter = RRCounter + 1;
end
RRx5 = RRCounter;

while    RRfringe(RRCounter + 1) -    RRfringe(RRCounter) < + fix;
      RRCounter = RRCounter + 1;
end
RRx6 = RRCounter;

while RRfringe(RRCounter + 1) - RRfringe(RRCounter) >    - fix;
      RRCounter = RRCounter + 1;
```

```
end
RRx7 = RRCounter;

while    RRfringe(RRCounter + 1) -    RRfringe(RRCounter) < + fix;
      RRCounter = RRCounter + 1;
end
RRx8 = RRCounter;

while RRfringe(RRCounter + 1) - RRfringe(RRCounter) >    - fix;
      RRCounter = RRCounter + 1;
end
RRx9 = RRCounter;

while    RRfringe(RRCounter + 1) -    RRfringe(RRCounter) < + fix;
      RRCounter = RRCounter + 1;
end
RRx10 = RRCounter;

while RRfringe(RRCounter + 1) - RRfringe(RRCounter) >    - fix;
      RRCounter = RRCounter + 1;
end
RRx11 = RRCounter;

while    RRfringe(RRCounter + 1) -    RRfringe(RRCounter) < + fix;
      RRCounter = RRCounter + 1;
end
RRx12 = RRCounter;

while RRfringe(RRCounter + 1) - RRfringe(RRCounter) >    - fix;
      RRCounter = RRCounter + 1;
end
RRx13 = RRCounter;

while    RRfringe(RRCounter + 1) -    RRfringe(RRCounter) < + fix;
      RRCounter = RRCounter + 1;
end
RRx14 = RRCounter;

while RRfringe(RRCounter + 1) - RRfringe(RRCounter) >    - fix;
      RRCounter = RRCounter + 1;
end
RRx15 = RRCounter;
RRx16 = length(RRfringe);
```

```
RRf1 = RRfringe([1:RRx1]);
RRf2 = RRfringe([RRx1+1:RRx2]);
RRf3 = RRfringe([RRx2+1:RRx3]);
RRf4 = RRfringe([RRx3+1:RRx4]);
RRf5 = RRfringe([RRx4+1:RRx5]);
RRf6 = RRfringe([RRx5+1:RRx6]);
RRf7 = RRfringe([RRx6+1:RRx7]);
RRf8 = RRfringe([RRx7+1:RRx8]);
RRf9 = RRfringe([RRx8+1:RRx9]);
RRf10 = RRfringe([RRx9+1:RRx10]);
RRf11 = RRfringe([RRx10+1:RRx11]);
RRf12 = RRfringe([RRx11+1:RRx12]);
RRf13 = RRfringe([RRx12+1:RRx13]);
RRf14 = RRfringe([RRx13+1:RRx14]);
RRf15 = RRfringe([RRx14+1:RRx15]);
RRf16 = RRfringe([RRx15+1:RRx16]);

RRfs1 = RRfringe([2:RRx1-1]);
RRfs2 = RRfringe([RRx1+2:RRx2-1]);
RRfs3 = RRfringe([RRx2+2:RRx3-1]);
RRfs4 = RRfringe([RRx3+2:RRx4-1]);
RRfs5 = RRfringe([RRx4+2:RRx5-1]);
RRfs6 = RRfringe([RRx5+2:RRx6-1]);
RRfs7 = RRfringe([RRx6+2:RRx7-1]);
RRfs8 = RRfringe([RRx7+2:RRx8-1]);
RRfs9 = RRfringe([RRx8+2:RRx9-1]);
RRfs10 = RRfringe([RRx9+2:RRx10-1]);
RRfs11 = RRfringe([RRx10+2:RRx11-1]);
RRfs12 = RRfringe([RRx11+2:RRx12-1]);
RRfs13 = RRfringe([RRx12+2:RRx13-1]);
RRfs14 = RRfringe([RRx13+2:RRx14-1]);
RRfs15 = RRfringe([RRx14+2:RRx15-1]);
RRfs16 = RRfringe([RRx15+2:RRx16-1]);


% Finding Monotonic Intervals for Object Fringe
RCounter = 1;
while Rfringe(RCounter + 1) - Rfringe(RCounter) > - fix;
      RCounter = RCounter + 1;
end
Rx1 = RCounter;

while Rfringe(RCounter + 1) - Rfringe(RCounter) <    + fix;
      RCounter = RCounter + 1;
```

```
end
Rx2 = RCounter;

while Rfringe(RCounter + 1) - Rfringe(RCounter) >    - fix;
      RCounter = RCounter + 1;
end
Rx3 = RCounter;

while Rfringe(RCounter + 1) - Rfringe(RCounter) <    + fix;
      RCounter = RCounter + 1;
end
Rx4 = RCounter;

while Rfringe(RCounter + 1) - Rfringe(RCounter) >    - fix;
      RCounter = RCounter + 1;
end
Rx5 = RCounter;

while Rfringe(RCounter + 1) - Rfringe(RCounter) <    + fix;
      RCounter = RCounter + 1;
end
Rx6 = RCounter;

while Rfringe(RCounter + 1) - Rfringe(RCounter) >    - fix;
      RCounter = RCounter + 1;
end
Rx7 = RCounter;

while Rfringe(RCounter + 1) - Rfringe(RCounter) <    + fix;
      RCounter = RCounter + 1;
end
Rx8 = RCounter;

while Rfringe(RCounter + 1) - Rfringe(RCounter) >    - fix;
      RCounter = RCounter + 1;
end
Rx9 = RCounter;

while Rfringe(RCounter + 1) - Rfringe(RCounter) <    + fix;
      RCounter = RCounter + 1;
end
Rx10 = RCounter;

while Rfringe(RCounter + 1) - Rfringe(RCounter) >    - fix;
```

```
    RCounter = RCounter + 1;
end
Rx11 = RCounter;

while Rfringe(RCounter + 1) - Rfringe(RCounter) <    + fix;
    RCounter = RCounter + 1;
end
Rx12 = RCounter;

while Rfringe(RCounter + 1) - Rfringe(RCounter) >    - fix;
    RCounter = RCounter + 1;
end
Rx13 = RCounter;

while Rfringe(RCounter + 1) - Rfringe(RCounter) <    + fix;
    RCounter = RCounter + 1;
end
Rx14 = RCounter;

while Rfringe(RCounter + 1) - Rfringe(RCounter) >    - fix;
    RCounter = RCounter + 1;
end
Rx15 = RCounter;
Rx16 = length(Rfringe);

Rf1 = Rfringe([1:Rx1]);
Rf2 = Rfringe([Rx1+1:Rx2]);
Rf3 = Rfringe([Rx2+1:Rx3]);
Rf4 = Rfringe([Rx3+1:Rx4]);
Rf5 = Rfringe([Rx4+1:Rx5]);
Rf6 = Rfringe([Rx5+1:Rx6]);
Rf7 = Rfringe([Rx6+1:Rx7]);
Rf8 = Rfringe([Rx7+1:Rx8]);
Rf9 = Rfringe([Rx8+1:Rx9]);
Rf10 = Rfringe([Rx9+1:Rx10]);
Rf11 = Rfringe([Rx10+1:Rx11]);
Rf12 = Rfringe([Rx11+1:Rx12]);
Rf13 = Rfringe([Rx12+1:Rx13]);
Rf14 = Rfringe([Rx13+1:Rx14]);
Rf15 = Rfringe([Rx14+1:Rx15]);
Rf16 = Rfringe([Rx15+1:Rx16]);
```

```
% ----- Normalization of Fringe and IFSE Processing(including Shift Unwrapping) --------
k = 3;                    % Set initial degree of polynomial

% Section 1
x = 1:length(RRfs1);

% Normalization of Fringe
RefH = (max(RRf1) - min(RRf1));
ObjH = (max(Rf1) - min(Rf1));
Hadj = RefH / ObjH ;
Objmid = (max(Rf1) + min(Rf1))/2;
Refmid = (max(RRf1) + min(RRf1))/2;
Rf1 = ((Rf1 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs1,x',k);
Rfx1 = polyval(j,Rf1);

% Section 2
x = 1:length(RRfs2);

% Normalization of Fringe
RefH = (max(RRf2) - min(RRf2));
ObjH = (max(Rf2) - min(Rf2));
Hadj = RefH / ObjH ;
Objmid = (max(Rf2) + min(Rf2))/2;
Refmid = (max(RRf2) + min(RRf2))/2;
Rf2 = ((Rf2 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs2,x',k);
Rfx2 = polyval(j,Rf2) + RRx1;

% Section 3
x = 1:length(RRfs3);

% Normalization of Fringe
RefH = (max(RRf3) - min(RRf3));
ObjH = (max(Rf3) - min(Rf3));
Hadj = RefH / ObjH ;
Objmid = (max(Rf3) + min(Rf3))/2;
Refmid = (max(RRf3) + min(RRf3))/2;
Rf3 = ((Rf3 - Objmid) * Hadj) + Refmid;
```

```
% IFSE processing with Shift Unwrapping
j = polyfit(RRfs3,x',k);
Rfx3 = polyval(j,Rf3) + RRx2;

% Section 4
x = 1:length(RRfs4);

% Normalization of Fringe
RefH = (max(RRf4) - min(RRf4));
ObjH = (max(Rf4) - min(Rf4));
Hadj = RefH / ObjH ;
Objmid = (max(Rf4) + min(Rf4))/2;
Refmid = (max(RRf4) + min(RRf4))/2;
Rf4 = ((Rf4 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs4,x',k);
Rfx4 = polyval(j,Rf4) + RRx3;

% Section 5
x = 1:length(RRfs5);

% Normalization of Fringe
RefH = (max(RRf5) - min(RRf5));
ObjH = (max(Rf5) - min(Rf5));
Hadj = RefH / ObjH ;
Objmid = (max(Rf5) + min(Rf5))/2;
Refmid = (max(RRf5) + min(RRf5))/2;
Rf5 = ((Rf5 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs5,x',k);
Rfx5 = polyval(j,Rf5) + RRx4;

% Section 6
x = 1:length(RRfs6);

% Normalization of Fringe
RefH = (max(RRf6) - min(RRf6));
ObjH = (max(Rf6) - min(Rf6));
Hadj = RefH / ObjH ;
Objmid = (max(Rf6) + min(Rf6))/2;
Refmid = (max(RRf6) + min(RRf6))/2;
Rf6 = ((Rf6 - Objmid) * Hadj) + Refmid;
```

```
% IFSE processing with Shift Unwrapping
j = polyfit(RRfs6,x',k);
Rfx6 = polyval(j,Rf6) + RRx5;

% Section 7
x = 1:length(RRfs7);

% Normalization of Fringe
RefH = (max(RRf7) - min(RRf7));
ObjH = (max(Rf7) - min(Rf7));
Hadj = RefH / ObjH ;
Objmid = (max(Rf7) + min(Rf7))/2;
Refmid = (max(RRf7) + min(RRf7))/2;
Rf7 = ((Rf7 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs7,x',k);
Rfx7 = polyval(j,Rf7) + RRx6;

% Section 8
x = 1:length(RRfs8);

% Normalization of Fringe
RefH = (max(RRf8) - min(RRf8));
ObjH = (max(Rf8) - min(Rf8));
Hadj = RefH / ObjH ;
Objmid = (max(Rf8) + min(Rf8))/2;
Refmid = (max(RRf8) + min(RRf8))/2;
Rf8 = ((Rf8 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs8,x',k);
Rfx8 = polyval(j,Rf8) + RRx7;

% Section 9
x = 1:length(RRfs9);

% Normalization of Fringe
RefH = (max(RRf9) - min(RRf9));
ObjH = (max(Rf9) - min(Rf9));
Hadj = RefH / ObjH ;
Objmid = (max(Rf9) + min(Rf9))/2;
Refmid = (max(RRf9) + min(RRf9))/2;
```

```
Rf9 = ((Rf9 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs9,x',k);
Rfx9 = polyval(j,Rf9) + RRx8;

% Section 10
x = 1:length(RRfs10);

% Normalization of Fringe
RefH = (max(RRf10) - min(RRf10));
ObjH = (max(Rf10) - min(Rf10));
Hadj = RefH / ObjH ;
Objmid = (max(Rf10) + min(Rf10))/2;
Refmid = (max(RRf10) + min(RRf10))/2;
Rf10 = ((Rf10 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs10,x',k);
Rfx10 = polyval(j,Rf10) + RRx9;

% Section 11
x = 1:length(RRfs11);

% Normalization of Fringe
RefH = (max(RRf11) - min(RRf11));
ObjH = (max(Rf11) - min(Rf11));
Hadj = RefH / ObjH ;
Objmid = (max(Rf11) + min(Rf11))/2;
Refmid = (max(RRf11) + min(RRf11))/2;
Rf11 = ((Rf11 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs11,x',k);
Rfx11 = polyval(j,Rf11) + RRx10;

% Section 12
x = 1:length(RRfs12);

% Normalization of Fringe
RefH = (max(RRf12) - min(RRf12));
ObjH = (max(Rf12) - min(Rf12));
Hadj = RefH / ObjH ;
Objmid = (max(Rf12) + min(Rf12))/2;
```

```
Refmid = (max(RRf12) + min(RRf12))/2;
Rf12 = ((Rf12 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs12,x',k);
Rfx12 = polyval(j,Rf12) + RRx11;

% Section 13
x = 1:length(RRfs13);

% Normalization of Fringe
RefH = (max(RRf13) - min(RRf13));
ObjH = (max(Rf13) - min(Rf13));
Hadj = RefH / ObjH ;
Objmid = (max(Rf13) + min(Rf13))/2;
Refmid = (max(RRf13) + min(RRf13))/2;
Rf13 = ((Rf13 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs13,x',k);
Rfx13 = polyval(j,Rf13) + RRx12;

% Section 14
x = 1:length(RRfs14);

% Normalization of Fringe
RefH = (max(RRf14) - min(RRf14));
ObjH = (max(Rf14) - min(Rf14));
Hadj = RefH / ObjH ;
Objmid = (max(Rf14) + min(Rf14))/2;
Refmid = (max(RRf14) + min(RRf14))/2;
Rf14 = ((Rf14 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs14,x',k);
Rfx14 = polyval(j,Rf14) + RRx13;

% Section 15
x = 1:length(RRfs15);

% Normalization of Fringe
RefH = (max(RRf15) - min(RRf15));
ObjH = (max(Rf15) - min(Rf15));
Hadj = RefH / ObjH ;
```

```
Objmid = (max(Rf15) + min(Rf15))/2;
Refmid = (max(RRf15) + min(RRf15))/2;
Rf15 = ((Rf15 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs15,x',k);
Rfx15 = polyval(j,Rf15) + RRx14;

% Section 16
x = 1:length(RRfs16);

% Normalization of Fringe
RefH = (max(RRf16) - min(RRf16));
ObjH = (max(Rf16) - min(Rf16));
Hadj = RefH / ObjH ;
Objmid = (max(Rf16) + min(Rf16))/2;
Refmid = (max(RRf16) + min(RRf16))/2;
Rf16 = ((Rf16 - Objmid) * Hadj) + Refmid;

% IFSE processing with Shift Unwrapping
j = polyfit(RRfs16,x',k);
Rfx16 = polyval(j,Rf16) + RRx15;

% Retrieve u(x)
Rfx =
[Rfx1',Rfx2',Rfx3',Rfx4',Rfx5',Rfx6',Rfx7',Rfx8',Rfx9',Rfx10',Rfx11',Rfx12',Rfx13',Rfx14',Rfx15',R
fx16']';
x = 1:775;
U = x' - Rfx;

% Calculate Height Distribution
l0 = 1295;
d0 = 350;
h = (U*l0)/d0*0.1796;

% Plot Unwrapped u(x) and Height Distribution
V =[0,900,-10,20];
subplot(2,1,1);plot(U,'k');title('Unwrapped u(x)');
subplot(2,1,2);plot(h,'k');title('h(x)');
```