Windowed fourier transform based fringe pattern analysis techniques

Zhao, Ming

2016

Zhao, M. (2016). Windowed fourier transform based fringe pattern analysis techniques. Doctoral thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/68569

https://doi.org/10.32657/10356/68569

Nanyang Technological University

Windowed Fourier Transform Based Fringe Pattern Analysis Techniques

A thesis submitted to the School of Computer Engineering

of Nanyang Technological University

by

Zhao Ming

in partial fulfillment of the requirement for

the Degree of Doctor of Philosophy

2016

Acknowledgments

First, I would like to express my gratitude to my supervisor, Prof. Qian Kemao, for his insightful advice and serious-minded guidance during my study period. Without him, I would not have been able to finished my research work.

I would like to thank Dr. Huang Lei for his help in the past years. To date, I can still remember the scenes that we were discussing phase unwrapping algorithms in lab.

Also, I would like to thank Dr. Wang Haixia and Dr. Gao Wenjing for sharing their invaluable expertise and experiences with me.

Meanwhile, I want to thank Nanyang Technological University and the School of Computer Engineering for providing resources and support for my research.

Finally, I would like to thank my families and my friends, especially my girlfriend Yang Fang. Your love, patience and encouragement support me throughout the entire study period.

Contents

Summary

Measurement techniques are powerful tools for obtaining properties of objects and recording their changes under different conditions. The demand of those valuable information promotes the development of measurement techniques. Fringe pattern based techniques are an important family of measurement techniques. They cover a measurement range from nanometers to kilometers. Due to the importance of fringe patterns, fringe pattern analysis techniques are widely used and studied in academia and industry. However, as the requirements of measurement techniques are increasing, people still need to seek more reliable, more accurate and faster analysis techniques.

Windowed Fourier transform (WFT) algorithms, including windowed Fourier ridges (WFR2) and windowed Fourier filtering (WFF2) methods, are effective and automatic fringe pattern analysis techniques. They can be applied for fringe pattern denoising, demodulation and providing local fringe properties for other analyses. One drawback of the WFR2/WFF2 algorithms is that they are very time-consuming. In this dissertation, factors for an efficient implementation, such as the selections of Fourier transform libraries and parallel techniques, are explored in detail, and a multi-threaded implementation is presented.

Fringe patterns are special images, so they can be denoised by general image denosing methods. To understand the performance of general image denoising methods on fringe patterns, the block-matching and 3D filtering (BM3D) method

is selected and compared with the WFF2 method. The BM3D method does not outperform the WFF2 method in general, but its performance in discontinuous areas is better than the WFF2 method. This finding inspires us to combine them together. The hybrid method is proposed and it achieves better denoising results than using those two algorithms individually.

Besides fringe pattern denoising, phase unwrapping is another important task of fringe pattern analysis. In this dissertation, quality guided phase unwrapping (QGPU) is reviewed by comparisons of quality maps and guiding strategies. New data structures are also proposed for accelerating the QGPU process.

The QGPU process is vulnerable to discontinuity. A snake assisted QGPU (sQGPU) method is proposed to solve the discontinuity problem. With the aid of the snake model, piece-wise phase unwrapping, which is immune to the discontinuity problem, is finally achieved.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Measurement and fringe patterns

When studying an object, properties of the object, such as three dimensional (3D) shape and internal structure, and changes under different conditions, such as deformation and displacement, are very valuable information. Measurement techniques are thus used to capture those information upon inspection. Due to their importance, people are attracted to develop advanced measurement techniques.

Traditionally, gauging tools, including calipers, hand-held micrometers and touch probes, are used to measure objects. However, non-contact methods gain more attentions, due to following advantages [1, 2]:

- Non-contact methods will not cause damage on the object surface, like scratches and dents.
- Non-contact methods are suitable to measure soft and liquid materials.
- Non-contact methods can be easily implemented in a automatic mode.

In the last decades, interferometric techniques, a major category of the non-contact measurement methods, have achieved significant growth [3]. These tech-

niques can fulfill demands of measuring different samples. For example, interferometric synthetic aperture radar (InSAR) is often used to observe terrain deformations [4–6]; shearography and Moiré interferometry are widely employed to measure strains and deformations [7–11]; digital holographic interferometry can be used to examine small samples, such as microlens arrays [12] or cells [13,14].

All the techniques above are based on the well-known interference phenomenon [3]. Suppose two light waves propagating in the same direction and polarizing in the same plane are superposed at a point. The complex amplitude at the point is the sum of the complex amplitudes of the two light waves, which can be described as follows

$$A = A_1 + A_2, \tag{1.1}$$

where $A_1 = a_1 exp(-j\varphi_1)$ and $A_2 = a_1 exp(-j\varphi_2)$ are the complex amplitudes of the two light waves, j is the imaginary unit, and φ_1 and φ_2 are the phase terms. The intensity at the point can be further written as

$$\begin{aligned} I &= |A_1 + A_2|^2 \\ &= |A_1|^2 + |A_2|^2 + A_1 A_2^* + A_1^* A_2 \\ &= I_1 + I_2 + 2(I_1 I_2)^{1/2} cos(\varphi_1 - \varphi_2) \\ &= I_1 + I_2 + 2(I_1 I_2)^{1/2} cos(\Delta\varphi), \end{aligned} \tag{1.2}$$

where I_1 and I_2 are intensities of the two light waves, respectively; $\Delta\varphi$ is the phase difference. The phase difference term reflects the difference of the optical paths of the two light waves, and it is usually spatially continuous. Therefore, the recorded intensity changes cosinusoidally, and generates a wave structure. This wave structure is called fringe pattern, and it is the output of interferometric techniques and some non-interferometric techniques [15–18].

1.2 Fringe pattern analysis

1.2.1 Model of fringe pattern

Generally, a fringe pattern is modelled as follows [19],

$$f(x,y) = a(x,y) + b(x,y)cos[\varphi(x,y)] + n(x,y), \tag{1.3}$$

where $f(x,y)$ is the fringe pattern, $a(x,y)$ is the background intensity, $b(x,y)$ is the fringe amplitude, $\varphi(x,y)$ is the phase distribution, and $n(x,y)$ is the noise term.

The purpose of fringe pattern analysis is to extract phase from fringe patterns which are obtained from front-end measurement techniques. However, it is a very arduous task. First, directly extracting the phase from a fringe pattern is a ill-posed problem. This problem can be easily seen in Eq. 1.3. Only $f(x,y)$ is known and others are all unknown. It also suffers the sign ambiguity problem and the order ambiguity problem. The sign ambiguity problem is that $\varphi(x,y)$ and $-\varphi(x,y)$ can be the solution concurrently in a single measurement, due to $cos[\varphi(x,y)] = cos[-\varphi(x,y)]$. The order ambiguity problem is that all $\varphi(x,y) + 2k\pi, k \in \mathbb{Z}$ are the solution when $\varphi(x,y)$ is the solution, due to $cos[\varphi(x,y)] = cos[\varphi(x,y) + 2k\pi]$. These ambiguities prevent us from obtaining a unique solution. In addition, noises and discontinuities in fringe patterns make the phase extracting process even harder.

1.2.2 Types of fringe patterns

Equation 1.3 provides a general model of fringe patterns, and fringe patterns can be further classified into four types based on their sources [19]:

Exponential phase fields (EPF)

An EPF without noise can be modeled as follows,

$$f_1(x,y) = b(x,y)\exp\left[j\varphi(x,y)\right]. \tag{1.4}$$

Since results of phase shifting techniques [20–22] are in the form of Eq. 1.3, EPFs can be easily obtained from those techniques. With EPFs, phase can be extracted via following equation,

$$\varphi(x,y) = atan2\{Imag[f_1(x,y)], Real[f_1(x,y)]\}, \tag{1.5}$$

where $Imag()$ and $Real()$ are the imaginary part and the real part of the given EPF; $atan2()$ is the arctangent function with two arguments that directly indicates the quadrant of the obtained angle [23]. We can find that the obtained phase $\varphi(x,y)$ is actually wrapped in the range of $(-\pi,\pi]$ [$\varphi_w(x,y)$ is used to distinguish wrapped phase from continuous phase]. The obtained phase is discontinuous, while its corresponding EPF is continuous. Therefore EPFs are preferred for analysis.

Wrapped phase

Once an EPF is given, its corresponding wrapped phase map can be obtained as well,

$$f_2(x,y) = \varphi_w(x,y) = angle[f_1(x,y)], \tag{1.6}$$

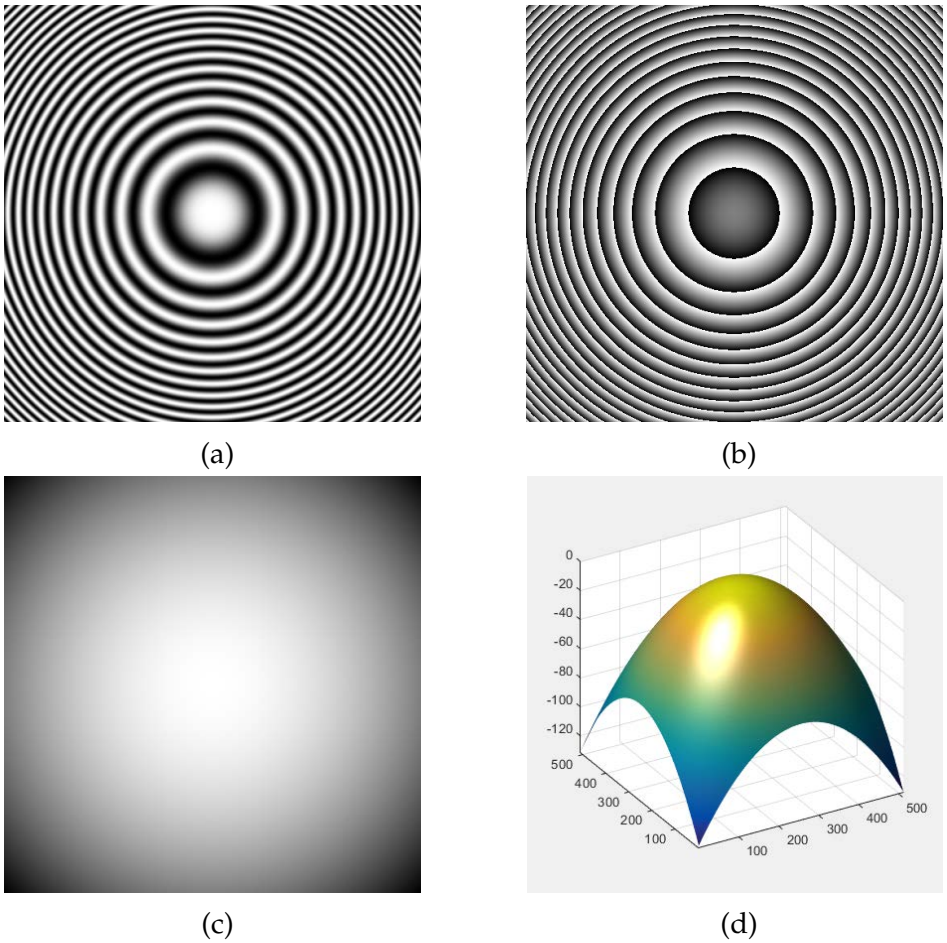
where $angle()$ represents the angle of a complex number. A wrapped phase map can be easily convert to an EPF with $b(x,y) = 1$.

The final continuous phase map can be computed from a wrapped phase map.

Their relationship is described as follows,

$$\varphi(x,y) = \varphi_w(x,y) + 2k\pi, \tag{1.7}$$

where $k \in \mathbb{Z}$. The process of recovering $\varphi(x,y)$ from $\varphi_w(x,y)$ is called phase unwrapping. This process seems simple, but it becomes very challenging when noise or discontinuity occurs in the wrapped phase map.



(a) (b)

(c) (d)

Figure 1.1: A fringe pattern example. (a) A simulated fringe pattern; (b) wrapped phase map; (c) unwrapped phase map; (d) 3D profile of the unwrapped phase map.

Carrier fringe patterns

A carrier fringe patten is defined as

$$f_3(x,y) = a(x,y) + b(x,y)cos[\varphi(x,y) + \omega_{cx}x + \omega_{cy}y], \tag{1.8}$$

where ω_{cx} and ω_{cy} are the spatial carrier frequencies in x and y directions, respectively. The carrier is known, and provides a prior information of the fringe pattern. This information can be used to solve the sign ambiguity problem. A carrier fringe pattern can be easily converted to an EPF by Fourier transform techniques.

Single closed fringe patterns

If a fringe pattern does not have carriers, it is closed and harder to convert to an EPF than a wrapped phase map or a carrier fringe pattern. Demodulation techniques are necessary to extract the phase information. A single closed fringe pattern is defined as:

$$f_4(x,y) = a(x,y) + b(x,y)cos[\varphi(x,y)]. \tag{1.9}$$

1.2.3 Framework of fringe pattern analysis

To conquer the obstacles of fringe pattern analysis, many techniques have been developed, and the framework of fringe pattern analysis is shown in Fig. 1.2.

Phase shifting techniques

Phase shifting is a fundamental technique to solve the ill-posed problem and the sign ambiguity problem of fringe pattern analysis. Because the number of unknowns in Eq. 1.3 are more than the number of equations, one natural solution is

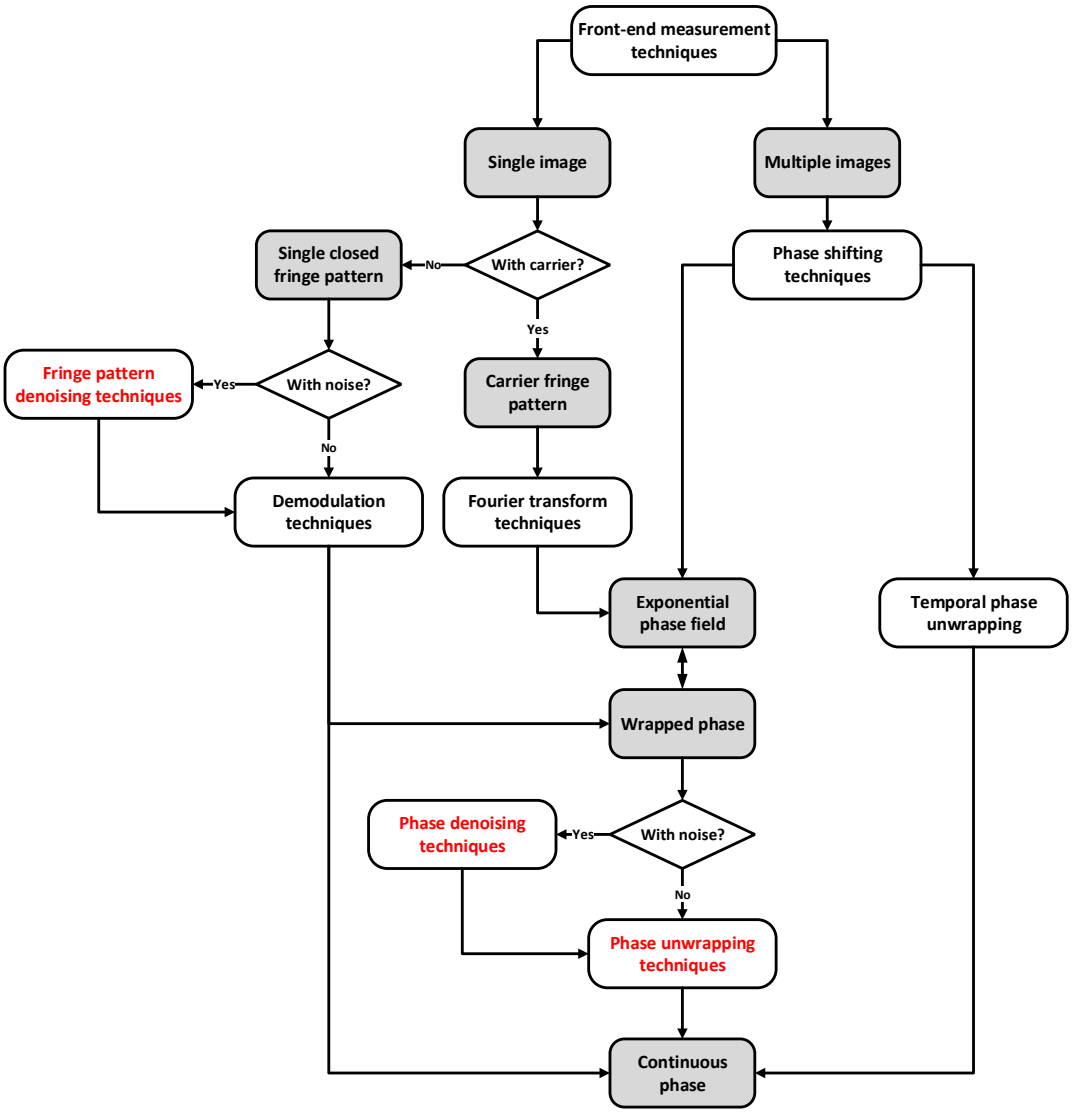


Figure 1.2: Fringe pattern analysis framework. Cells filled with gray color are data types, and others are fringe pattern analysis techniques. The techniques marked with red color will be covered in this dissertation.

to add more equations. By modifying the experiment setup, it is possible to add

some changes to the cosine term in Eq. 1.3:

$$\begin{aligned} I_0(x,y) &= a(x,y) + b(x,y)cos[\varphi(x,y)], \\ I_1(x,y) &= a(x,y) + b(x,y)cos[\varphi(x,y) + \omega_1(x,y)], \\ I_2(x,y) &= a(x,y) + b(x,y)cos[\varphi(x,y) + \omega_2(x,y)], \\ &... \\ I_n(x,y) &= a(x,y) + b(x,y)cos[\varphi(x,y) + \omega_n(x,y)], \end{aligned} \tag{1.10}$$

where $\omega_k(x,y), k = 1..n$ represents the phase shift at different time. Once $k \geq 3$, we will have enough equations, and obtaining a unique solution is possible. For example, the well-known four-step phase shifting algorithm [24] uses $\pi/2$ as the phase shift between two adjacent images. Then, Eq. 1.10 becomes

$$\begin{aligned} I_0(x,y) &= a(x,y) + b(x,y)cos[\varphi(x,y)], \\ I_1(x,y) &= a(x,y) + b(x,y)cos[\varphi(x,y) + \pi/2], \\ I_2(x,y) &= a(x,y) + b(x,y)cos[\varphi(x,y) + \pi], \\ I_3(x,y) &= a(x,y) + b(x,y)cos[\varphi(x,y) + 3\pi/2]. \end{aligned} \tag{1.11}$$

If we assume the background intensity term and the amplitude term are time-irrelevant, the phase can be expressed as

$$\varphi(x,y) = atan2[\frac{I_3(x,y) - I_1(x,y)}{I_0(x,y) - I_2(x,y)}]. \tag{1.12}$$

Phase shifting is able to provide a reliable input for temporal phase unwrapping [25,26] or generate an EPF for further analysis. Currently, many variants have been proposed to reduce systematic errors (such as the phase shift error and intensity distortion [19]) or fulfill different requirements of experiments. The details

of variants of phase shifting algorithms can be found in [24].

Fourier transform techniques

Fourier transform techniques [27,28] are another type of fundamental techniques, and it is effective to process carrier fringe patterns. Suppose we have a carrier fringe pattern (like Eq. 1.8). It can be rewritten as follows based on Euler's formula

$$f(x,y) = a(x,y) + c(x,y)exp[j(\omega_{cx}x + \omega_{cy}y)] + c^*(x,y)exp[-j(\omega_{cx}x + \omega_{cy}y)] \tag{1.13}$$

with

$$c(x,y) = \frac{1}{2}b(x,y)exp[j\varphi(x,y)] \tag{1.14}$$

where $*$ denotes the complex conjugate. Equation 1.15 is the FFT spectrum of Eq. 1.13, which gives

$$Ff(\xi_x,\xi_y) = Fa(\xi_x,\xi_y) + Fc(\xi_x - \omega_{cx},\xi_y - \omega_{cy}) + Fc^*(\xi_x + \omega_{cx},\xi_y + \omega_{cy}), \tag{1.15}$$

where the prefix F denotes the Fourier spectrum of corresponding terms. Usually, the added carrier signal should have a high frequency, and then the three spectrum terms $Fa(\xi_x,\xi_y)$, $Fc(\xi_x - \omega_{cx},\xi_y - \omega_{cy})$ and $Fc^*(\xi_x + \omega_{cx},\xi_y + \omega_{cy})$ are not overlapped and can be separated. If a band-pass filter whose center is at $(\omega_{cx},\omega_{xy})$ is applied to $Ff(\xi_x,\xi_y)$, only the term $Fc(\xi_x - \omega_{cx},\xi_y - \omega_{cy})$ will be kept. Then, with the help of the inverse Fourier transform, $c(x,y)$ could be obtained, which is an EPF. The corresponding wrapped phase of the obtained EPF can be expressed as

$$\varphi_w(x,y) = angle[c(x,y)] = atan2\{Imag[c(x,y)], Real[c(x,y)]\}. \tag{1.16}$$

The details of the Fourier transform techniques can be found in [24, 29].

Demodulation techniques

In some optical measurement techniques, only single closed fringe patterns can be acquired. Due to lacking carriers or other useful information, demodulating a single closed fringe pattern is very tough.

Usually, fringe pattern demodulation techniques are incorporated with background removal techniques [30, 31] and normalization techniques [32–34], which aim to remove the $a(x, y)$ term and the $b(x, y)$ term in Eq. 1.3, respectively. After these two steps, the fringe pattern becomes

$$f(x, y) = cos[\varphi(x, y)]. \tag{1.17}$$

The regularized phase tracker (RPT) technique is the first automatic demodulation method for single closed fringe patterns [35, 36]. It assumes that the phase distribution is locally smooth. The demodulated phase can be obtained by minimizing the following function

$$E_{total} = \sum_{x,y} E_r[\varphi(x, y), \omega_x(x, y), \omega_y(x, y)], \tag{1.18}$$

where

$$\begin{aligned} E_r[\varphi(x, y), \omega_x(x, y), \omega_y(x, y)] = & \sum_{(\xi,\eta)\in N} \{f(\xi, \eta) - cos[p(x, y, \xi, \eta)]\}^2 \\ & + \lambda \sum_{(\xi,\eta)\in N} [\varphi(\xi, \eta) - p(x, y, \xi, \eta)]^2 m(\xi, \eta), \end{aligned} \tag{1.19}$$

with

$$p(x, y, \xi, \eta) = \varphi(x, y) + \omega_x(x, y)(x - \xi) + \omega_y(x, y)(y - \eta).$$

In this equation, ω_x and ω_y are the local frequencies in x and y directions, respectively; N represents a small window around a point (x, y); ξ and η are the coordinates of the neighbor pixels in x and y directions, respectively; $m(x, y) = 1$, if point (x, y) has been estimated, otherwise $m(x, y) = 0$; λ is the regularization term. In Eq. 1.19, the first term describes the difference between the input fringe pattern and the model, while the second term controls the smoothness of the phase. In the variants of the RPT method [37–39], quadratic models are used, which provide better fitting of phase than a linear model used in Eq. 1.19.

Besides the RPT methods, some other demodulation methods are also developed, such as robust phase demodulation [40], tile based demodulation [41], windowed Fourier ridges (WFR2) [19,42], frequency-guided sequential demodulation (FSD) [43,44] and Fast FSD [45].

Phase unwrapping techniques

Once an EPF or a wrapped phase map is obtained, a phase unwrapping process is required. The phase unwrapping process is usually the last step of fringe pattern analysis, and it can be described as follows: given a wrapped phase $\varphi_w(x, y)$, we try to find a continuous phase $\varphi(x, y)$ which satisfies

$$\mathbb{W}[\varphi(x, y)] = \varphi_w(x, y), \tag{1.20}$$

where $\mathbb{W}()$ is a wrapping operator. From Eq. 1.20, it seems that the continuous phase can be obtained easily by adding or subtracting 2π. However, in practice, the phase unwrapping process has two major difficulties:

- **Noise**. Wrapped phase maps usually contain noises. The 2π jumps in the wrapped phase are hidden in noises, and the continuous phase could not be recovered by simply adding or subtracting 2π.

- **Discontinuity**. Discontinuity is another difficulty. Phase unwrapping process assumes $\varphi(x, y)$ in Eq. 1.20 is continuous, i.e. phase difference between two adjacent pixels should be smaller than π. However, this assumption will fail when the physical quantities to be measured is discontinuous.

Many phase unwrapping algorithms have been developed to overcome the difficulties, and they can be generally classified into two major types:

- **Path following algorithms**. These methods are more or less inspired by Itoh's linear integration method [46]. They consider the unwrapping process as a path integral, and the final phase can be described as

$$\varphi(x, y) = \int_C \mathbb{W}[\nabla \varphi_w(x, y)] dx dy + \varphi(x_0, y_0), \tag{1.21}$$

  where (x_0, y_0) is the initial point, (x, y) is the point to be unwrapped, C is the path connecting (x_0, y_0) and (x, y), and $\mathbb{W}[\nabla \varphi_w(x, y)]$ is the wrapped gradient of the given wrapped phase map. Example of the path following methods includes Goldstein's branch cut method [47], quality guided phase unwrapping methods [48–54].

- **Path independent algorithms**. These methods try to find the continuous phase by minimizing a cost function. For example, the least square method [55, 56] can be expressed as

$$E_{total} = \sum_{i,j} \{\mathbb{W}[\varphi(i, j)] - \varphi_w(i, j)\}^2 \tag{1.22}$$

  With the help of regularization and optimization techniques, the continuous phase can be obtained. Other examples are minimum L^p norm method [57] and graph cut method [58].

Phase unwrapping plays a very important role in the framework of fringe pattern analysis, and it thus attracts researchers to improve it, and make it more accurate and faster. In Sec. 2.3, details of phase unwrapping techniques are introduced and discussed.

Denoising techniques

Noise is an inevitable problem in fringe pattern analysis, and fringe patterns with noises must be denoised before further analysis. Current denoising techniques can be generally classified into two groups, spatial domain techniques [15, 59–67] and transform domain techniques [19, 42, 68–70]. The spatial domain techniques reduce noises by a smoothing process. Comparing general image denoising techniques, such as Gaussian filtering, these techniques consider the structure of fringe patterns, especially the fringe orientation, and they thus achieve better results. The mean and median filtering methods [15], oriented partial differential equations (OPDE) [59–62], coherence enhancing diffusion (CED) [63, 64] and spin filters [65–67] are examples of those spatial domain techniques. A comparison study of the spatial domain techniques can be found in Ref. [71].

In a noisy fringe pattern, useful signals are mixed with noises, and extracting them is quite difficult. When using transform domain techniques, the input fringe pattern is transformed to a transform domain. Due to its randomness, noises have small coefficients in the transform domain, while the useful signals usually provide larger coefficients. The noise can be reduced by a thresholding process. The transform domain techniques provide exciting denoised results and examples of transform domain techniques include Fourier transform [68], windowed Fourier transform (WFT) [19, 42], wavelet transform [69, 70].

1.3 Contributions and organization

1.3.1 Objective

In the framework of fringe pattern analysis, fringe pattern denoising and phase unwrapping are two important, but very challenging tasks. This dissertation thus focuses on these two tasks, and aims to make this two tasks more accurate and faster. Some new techniques are proposed and discussed in this dissertation.

1.3.2 Contributions

Following the objective above, the contributions of this dissertation are generally separated into two groups.

The first group of contributions is concentrated in windowed Fourier transform (WFT) based algorithms. The WFT based algorithms include two algorithms, windowed Fourier ridges (WFR2) and windowed Fourier filtering (WFF2). Since they were proposed, the WFR2/WFF2 have been widely used in fringe pattern analysis, especially for fringe pattern denoising [42,72]. The contributions are summarized as follows:

- **Multi-threaded WFR2/WFF2** [73]. The WFR2/WFF2 are CPU intensive and time consuming, so speeding them up is always preferred. The factors on accelerating the WFR2/WFF2 are discussed in this work, and a multi-threaded version of the WFR2/WFF2 is proposed.

- **Comparison between the WFF2 and block-matching and 3D filtering (BM3D) method** [74]. Fringe patterns are a special type of images, and we are curious that how the general image filtering methods perform on fringe patterns. In this work, the BM3D method [75], which is a state-of-the-art general im-

age filtering method, is compared with the WFF2, which is expert at fringe pattern denoising.

- **Integrating WFF2 and BM3D** [76]. The WFF2 achieves very good results in continuous areas, while the BM3D performs well in discontinuous areas. It is natural to combine the two algorithms to obtain better denoising results. In this work, a hybrid denoising scheme, WFF-BM3D, is proposed.

The second group of contributions focuses on quality guided phase unwrapping (QGPU), especially the WFT based QGPU. The QGPU is a popular phase unwrapping method, and a quality map is used to guided the unwrapping process [48]. The contributions are summarized as follows:

- **Comparison among quality maps** [53]. In QGPU, a quality map directly determines final unwrapped results. In this work, popular quality maps are compared in noisy condition and discontinuous condition separately.

- **Comparison among guiding strategies** [53]. In QGPU, the speed of the unwrapping process is determined by guiding strategies. In this work, guiding strategies are compared and discussed. A new data structure indexed interwoven linked list (I2L2) is proposed.

- **I2L2-H** [77]. In this work, the I2L2 is improved from three aspects: (1) how to find the highest non-empty level; (2) how to deal with badly distributed quality values; (3) how to implement each level of the I2L2 efficiently. The final data structure, which is named I2L2-H, achieves 6 times faster than the original I2L2.

- **Snake assisted QGPU** [78]. Discontinuity is a challenging problem in QGPU. In this work, a snake model, which can identify object boundaries, is used to

assist the WFT based phase unwrapping process. The result shows that the snake assisted QGPU offers better performance than the original WFT based QGPU when discontinuity occurs.

1.3.3 Organization

The dissertation has 8 chapters, and it is organized as follows:

In Chap. 1, the framework of fringe pattern analysis is briefly introduced. The objective, contributions and organization of this dissertation are also introduced in this chapter.

Due to the importance of the WFR2/WFF2, their principles and applications are first provided in Chap. 2. Literatures on parallelizing the WFR2/WFF2, popular phase unwrapping algorithms and the principles of the BM3D and snake models are also introduced and reviewed in Chap. 2.

In Chap. 3, a multi-threaded WFT implementation is proposed, and the used techniques, such as Fourier transform libraries, compilers and parallel techniques are introduced and compared. In Chap. 4, a comparison between the WFF2 and BM3D is first given, followed by WFF-BM3D, a newly proposed hybrid denoising scheme which combines the advantages of the WFF2 and BM3D.

Quality maps and guiding strategies for QGPU are presented in Chap. 5. In Chap. 6, I2L2-H is proposed, and the comparison between I2L2-H and the original I2L2 is also presented in this chapter. In Chap. 7, the snake assisted QGPU for discontinuous phase field is described.

Finally, the conclusion and future work are provided in Chap. 8.

Chapter 2

Literature review

2.1 Windowed Fourier transform

2.1.1 Principle

Windowed Fourier transform (WFT) is an important tool for fringe pattern analysis. As it is widely used in later chapters, its principle is first introduced in this section. The WFT spectrum [19,42] of a fringe pattern $f(x,y)$ can be described as

$$Sf(u,v;\xi_x,\xi_y) = f(u,v) \otimes g_{\xi_x,\xi_y}(u,v), \tag{2.1}$$

where $\otimes$ is the symbol of convolution; (x,y) and (u,v) are both spatial coordinates; ξ_x and ξ_y are frequency coordinates corresponding to x and y directions, respectively; $Sf(u,v;\xi_x,\xi_y)$ is the WFT spectrum; and $g_{\xi_x,\xi_y}(x,y)$ is the WFT kernel. The WFT kernel can be expressed as

$$g_{\xi_x,\xi_y}(x,y) = g(x,y)\exp(j\xi_x x + j\xi_y y), \tag{2.2}$$

with

$$g(x,y) = (\pi\sigma_x\sigma_y)^{-\frac{1}{2}} \exp(-\frac{x^2}{2{\sigma_x}^2} - \frac{y^2}{2{\sigma_y}^2}), \tag{2.3}$$

where j is the imaginary unit; σ_x and σ_y are the standard deviations of Gaussian function in x and y directions, respectively. The standard deviations control the spatial extension of $g(x,y)$, i.e. the window size. Because fringe patterns are two-dimensional (2D) signals, 2D transforms (Eq. 2.1) are applied. The transform could be extended to other dimensions as well [19].

In the 2-D WFR (WFR2), for a pixel [denoted as (u,v)] in a fringe pattern, a small patch around this pixel is compared with different WFT kernels $g_{\xi_x,\xi_y}(x,y)$. The WFT kernels are generated by using different frequency pairs. A frequency pair (ξ_x, ξ_y) is taken from $[\xi_{xl},\xi_{xh}] \times [\xi_{yl},\xi_{yh}]$ with sampling intervals ξ_{xi} and ξ_{yi} accordingly. After the comparisons, each kernel will have a matching score. The kernel which has the highest similarity offers the highest matching score. This highest matching score is called the ridge value of the pixel. The denoised phase can be estimated based on the ridge values. The WFR2 algorithm is described as

$$[\hat{\omega}_x(u,v), \hat{\omega}_y(u,v)] = \arg\max_{\xi_x,\xi_y} |Sf(u,v;\xi_x,\xi_y)|, \tag{2.4}$$

$$[\hat{\varphi}_w(u,v)] = \angle Sf(u,v;\hat{\xi}_x,\hat{\xi}_y) - \frac{1}{2}\arctan[{\sigma_x}^2\hat{c}_{xx}(u,v) - \frac{1}{2}\arctan[{\sigma_y}^2\hat{c}_{yy}(u,v)], \tag{2.5}$$

where $\hat{}$ is used to represent the estimated value for an given parameter; ω_x and ω_y are the phase derivatives; $\hat{c}_{xx}$ and $\hat{c}_{xx}$ are the second order phase derivatives; and $\hat{\varphi}_w$ is the estimated wrapped phase.

The noise signal is usually random, its windowed Fourier spectrum coefficients

are therefore very small. If we discard those small coefficients, noises can be reduced as well. In the 2D WFF (WFF2) algorithm, the filtered fringe pattern $\bar{f}(x,y)$ is obtained through an inverse WFT (IWFT), which is described as follows

$$\bar{f}(x,y) = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\overline{Sf}(x,y;\xi_x,\xi_y)\otimes g_{\xi_x,\xi_y}(x,y)\mathrm{d}\xi_x\,\mathrm{d}\xi_y, \tag{2.6}$$

with

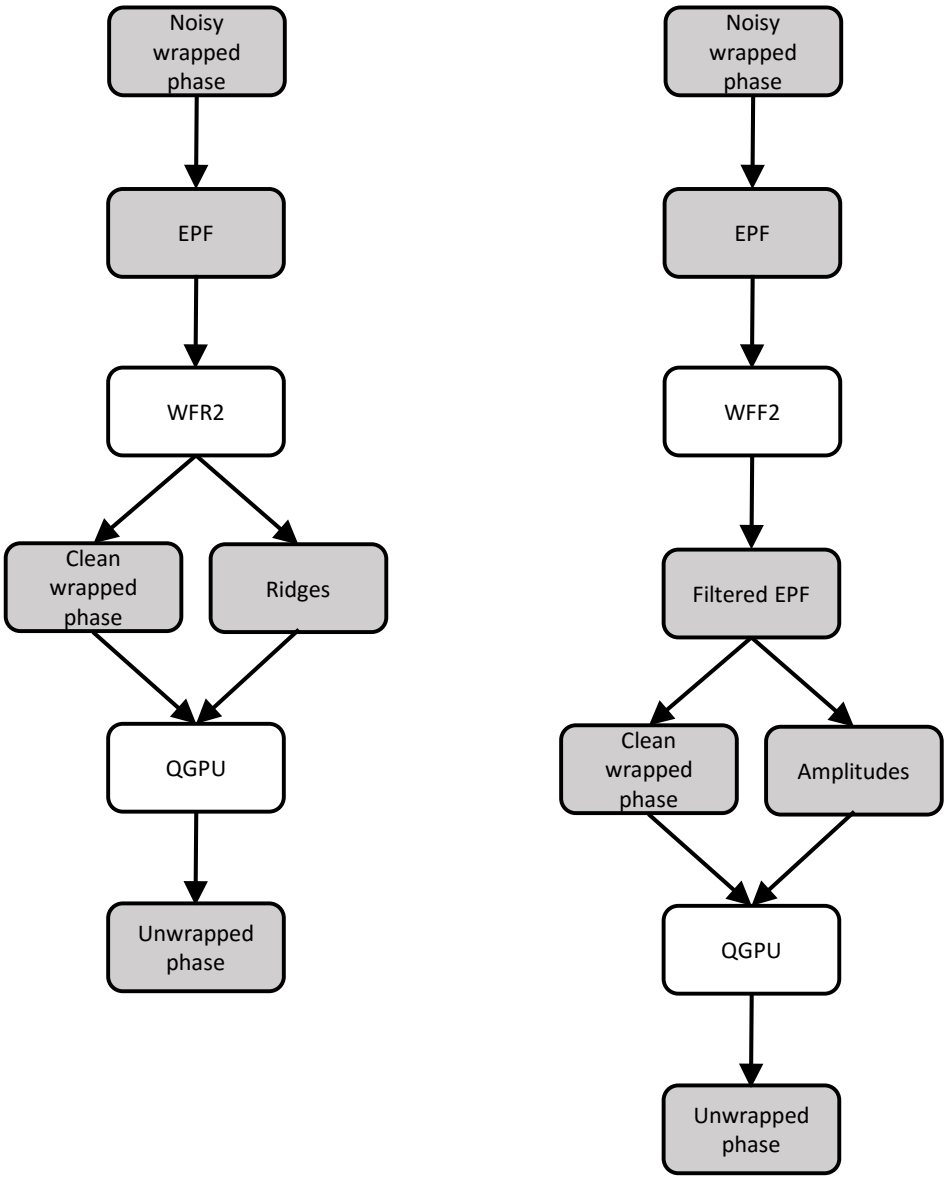
$$\overline{Sf}(x,y;\xi_x,\xi_y) = \begin{cases} Sf(x,y;\xi_x,\xi_y) & |Sf(x,y;\xi_x,\xi_y)| \geq threshold, \\ 0 & |Sf(x,y;\xi_x,\xi_y)| < threshold, \end{cases} \tag{2.7}$$

where $\overline{Sf}(x,y;\xi_x,\xi_y)$ is the filtered spectrum. The main body of the WFF2 is to compute $\overline{Sf}(x,y;\xi_x,\xi_y)$ for each combination of ξ_x and ξ_y taken from $[\xi_{xl},\xi_{xh}]\times[\xi_{yl},\xi_{yh}]$. The thresholded spectrum can be used for the reconstruction in Eq. 2.6.

2.1.2 Applications

One of the applications of the WFR2/WFF2 is denoising fringe patterns [42]. In the WFR2, it searches the best match (the ridge) of the input EPF for each small window, and phase values are estimated from the ridges. The WFF2 is a thresholding process, and phase values are computed with the remaining WFT coefficients after thresholding. Both of the WFR2/WFF2 can effectively reduce noises in fringe patterns.

Local properties of the WFR2/WFF2 results are also useful for other analyses. For example, the WFR2/WFF2 can be integrated with QGPU [52, 79]. The WFR2/WFF2 provide denoised wrapped phase to QGPU. Meanwhile, their local properties (the ridges from the WFR2 and the amplitudes of the WFF2 filtered EPF) represent the goodness of points, and they can thus be used as quality maps.



(a) (b)

Figure 2.1: The WFT based QGPU. (a) The WFR2 based QGPU; (b) the WFF2 based QGPU. Cells filled with gray color are data types. Others are the WFR2/WFF2 and QGPU techniques.

The steps of the WFR2/WFF2 based QGPU are shown in Figs. 2.1(a) and 2.1(b), respectively.

In addition, the WFR2/WFF2 can be also applied to demodulate fringe patterns [19] and extract phase derivatives [80–82].

2.1.3 Accelerating the WFT

From the expressions of the WFR2/WFF2, it is easy to find that they need to compute Fourier transform repeatedly. They are thus very time consuming. In Ref. [83], MATLAB Parallel Computing toolbox [84] was used to speed up the WFT process. In this work, the *parfor* command was selected to parallel loops in the WFT process. The parallelized version achieved 6 times faster than the original version with the assistance of a dual-die quad-core CPU which has 8 cores in total.

In Refs. [85] and [86], general-purpose computing on graphics processing units (GPGPU) was introduced to accelerate the WFT algorithms. Traditional programs are running on CPUs which equip a few cores specially designed for sequential data processing. Despite the cores of GPUs are much simpler, one GPU chip may contain thousands of cores. This makes GPU a very powerful tool for parallel computing. In Refs. [85] and [86], the authors replaced the original FT functions with computer unified device architecture (CUDA) cuFFT [87] and GPU enabled MATLAB FFT function. With the help of GPUs, the WFR2/WFF2 performance increased tremendously.

2.2 Block-matching and 3D filtering

Image denoising is a fundamental operation in digital image processing, and it has been studied for decades. In recent years, many powerful techniques have been proposed, such as K-SVD [88], non-local means (NLM) method [89], block-matching and 3D filtering (BM3D) [75], the learned simultaneous sparse coding (LSSC) method [90], and neural networks method [91].

In the K-SVD method, the sparsity and redundant concepts was used. The algorithm has three major steps: (1) compute the sparse representations for all

patches in the image with an initial dictionary; (2) update the dictionary to obtain better sparse representations; (3) reconstruct the denoised image with updated dictionary. The second step will be performed iteratively until the process is converged. The final obtained dictionary has a better fit of data, and noises are thus reduced. In Ref. [89], the NLM method first used similar patches for denoising. In NLM, similar patches are searched and grouped at first. Then the center pixel of each patch is denoised by averaging all center pixels of other similar patches. In Ref. [90], the authors combined the sparsity and non-local concepts with the assistance of simultaneous sparse coding [92]. One feature of this method is that similar patches can share the same dictionary elements in the decomposition process. In Ref. [91], the authors used a multi layer perceptron model to map noisy image patches to clean image patches. This neural networks approach achieves better results than the K-SVD method, and it runs faster than the K-SVD method with the assistance of GPU.

The BM3D method is a non-local denoising method through grouping and collaborative filtering [75]. The BM3D uses the non-local scheme as well. It searches and groups similar patches as the NLM does. However, the BM3D denoises pixels in a transform domain, while the NLM denoises pixels in a spatial domain. The BM3D is proven very effective on denoising [93]. It is better than, or at least equivalent to other leading alternatives.

Generally, the BM3D method can be divided into two major stages, basic estimate stage and final estimate stage. Each stage is further divided into three steps, grouping, filtering and aggregation. In the grouping step, the algorithm searches the input image for similar patches of each reference patch. The obtained patches are grouped and stacked into 3D arrays based on their corresponding reference patches. Each reference patch has one 3D array. A 3D transform based filtering is

applied to each 3D patch arrays. This process is similar as a 2D filtering process on images. A 2D transform is first performed on each level of the 3D array. The 2D transform is followed by a 1D transform along the third dimension of the 3D array. After that, the obtained 3D transform spectrums is filtered by a shrinkage process. Filtered patches are obtained through an inverse 3D transform. Finally, aggregation is performed for estimating pixels from different reference patches.

The only differences between the basic estimate and the final estimate are the input and the shrinkage process. The input of the basic estimate is the original noisy image, but the inputs of the final estimate are the original noisy image and the output of the basic estimate. In the shrinkage step, hard-thresholding and Wiener filtering are applied in the basic estimate and the final estimate, respectively.

2.3 Phase unwrapping

Researchers have developed dozens of phase unwrapping algorithms to solve the noise and discontinuity problems in these decades. In this section, some popular algorithms are introduced and reviewed.

2.3.1 Path following algorithms

The fundamental idea of the path following algorithms originates from Itoh's method [46], which is introduced in Sec. 1.2.3 (Eq. 1.21). In this method, the phase unwrapping process is considered as a path integral process. If the incoming wrapped phase map contains no errors, the integral process is path independent. However, incoming phase maps always contain noises or other problems. The result thus relies on the chosen integral path. Otherwise, if an incorrect path

is chosen, the integral path will enter problematic areas and the subsequent points will be incorrect completely.

Itoh's method inspires a large family of phase unwrapping methods. Branch cut methods [47, 94–98] and quality guided methods [49–52, 79, 99–105] are two typical examples. These two methods focus on isolating problematic areas from error-free areas. Then the points in error-free areas will not be effected by the points in problematic areas.

Branch cut algorithms

As introduced above, the phase unwrapping process is considered as a path integral process and it will be path-dependent when there are some inconsistencies. In Ref. [47], the authors pointed out that these inconsistencies are caused by residues. A residue is located in a loop of four pixels and the sum of wrapped phase differences of these pixels are not zeros.

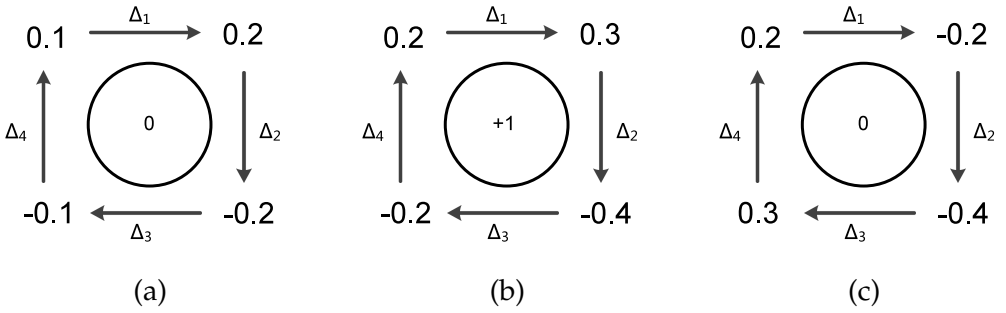


Figure 2.2: Examples of Residues. (a)Non-residue, (b) positive-residue, (c) negative-residue.

Figure 2.2 illustrates examples of detecting residues. The numbers represent the cycles of 2π, and the wrapped phase values can be obtained by multiply 2π. We use Fig. 2.2(a) to introduce residues. Firstly, the phase differences are com-

puted using the equations below (use the left top pixel as the initial point):

$$\begin{aligned}
\Delta_1 &= \mathbb{W}\{\varphi_w(m+1,n) - \varphi_w(m,n)\},\\
\Delta_2 &= \mathbb{W}\{\varphi_w(m+1,n+1) - \varphi_w(m+1,n)\},\\
\Delta_3 &= \mathbb{W}\{\varphi_w(m,n+1) - \varphi_w(m+1,n+1)\},\\
\Delta_4 &= \mathbb{W}\{\varphi_w(m,n) - \varphi_w(m,n+1)\}.
\end{aligned} \tag{2.8}$$

where Δ_1, Δ_2, Δ_3, and Δ_4 are wrapped phase differences, $\mathbb{W}(\ )$ is the wrapping operation, m and n are the row index and column index of the initial point, $\varphi_w(x,y)$ is the wrapped phase value.

We substitute the values in Fig. 2.2(a) into equations, and we can get:

$$\begin{aligned}
\Delta_1 &= 0.1,\\
\Delta_2 &= -0.4,\\
\Delta_3 &= 0.1,\\
\Delta_4 &= 0.2.
\end{aligned} \tag{2.9}$$

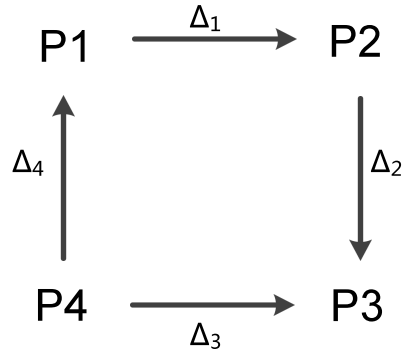


Figure 2.3: Example of four points loop.

The sum of these four phase differences is 0, so these four pixels do not form a residue. If we substitute the values in Fig. 2.2(b) and Fig. 2.2(c), we can find the sums are 1 and -1, and they form residues. Generally, there are two types of residues, positive or negative, based on the sum is $+1$ or -1. It is not a coincidence that the sum is $+1$ or -1. Let's consider the four points loop (see Fig. 2.3). If we compute the sum of the phase differences clockwise around the pixels, we can get:

$$\varphi_w(P_1)' = \varphi_w(P_1) + \sum_{i=1}^{4} \Delta_i, \tag{2.10}$$

where $\varphi_w(P_1)$ is the phase value of P_1, $\psi(P_1)'$ is the result after adding the sum of the phase differences and Δ_i is the difference between neighbor pixels. Point P_1 is the starting point and the ending point as well, so $\psi(P_1)$ and $\psi(P_1)'$ are equal. In order to fulfill this requirement, $\sum_{i=1}^{4} \Delta_i$ must be 0. Sometimes, the sum of these phase differences is not zero, and it brings a inconsistency. The phase difference between two adjacent points are less than π and the range of wrapped phase ($-\pi$, π], so the sum of phase differences can only be -2π or 2π, or -1 or $+1$ cycle. Figure 2.3 shows four pixels, and there are two paths from P_1 to P_4. One is P_1, P_2 and P_4, the other is P_1, P_3 and P_4. If these four pixels form a residue, choosing different paths will give different results.

As stated in Ref. [47], if positive and negative residues have the same number in any path integral, the residue theorem can guarantee that the surrounding integrals are all equal to zero, i.e. the integral is path-independent. This conclusion is the base of branch cut algorithms. The branch cut methods attempt to connect a pair of positive and negative residues with a cut. While performing phase unwrapping, the processing path will not across the cuts, i.e. the path will form a circle around the cuts. Then the integral will be path independent.

Branch cut method looks very simple, but it is arduous to implement. There

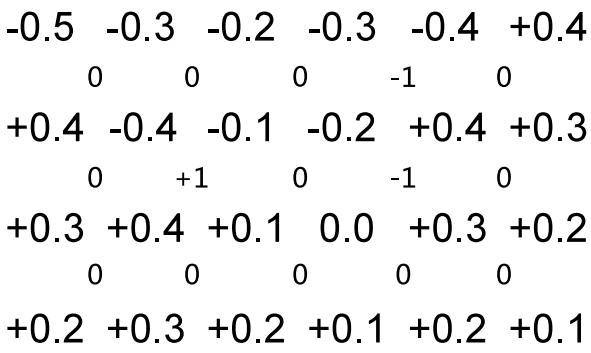


Figure 2.4: A branch cut example.

are a huge number of residues in one phase map. The key problem is how to find the pairs of residues. This can be considered as an optimization problem which involves the selection of cuts. The original branch cut method [47] computed the sum of cut lengths and chose the minimal one. Cusack et.al [94] and Buckland et.al [95] adopted nearest neighbor strategy and minimum-cost-matching method separately. In Ref. [96], they adopted reverse simulated annealing method and clustering method to improve the arrangement of cuts. Moreover, in Ref. [97], the authors added direction field of a phase map as a new energy term for the reverse simulated annealing computation. This new term pointed out the potential pairing direction of positive and negative residues. In Ref. [98], the authors introduced a new term called residue vector which was generated from residues. The residue vector pointed out toward the balancing residue of opposite polarity.

Quality guided phase unwrapping algorithms

The quality-guided phase unwrapping (QGPU) technique has been studied in the past decades [48–52, 79, 99–109]. It is an efficient, fast and automatic spatial phase unwrapping method. Figure 2.5 is an example for explaining the concept of QGPU. A real eyeglass case measured by fringe projection profilometry is used in this example. The wrapped phase of the case [shown in Fig. 2.5(a)] is unwrapped

into Fig. 2.5(c). By following the quality map in Fig. 2.5(b), the unwrapping path is recorded and shown in Fig. 2.5(c). The quality map is specially designed based on the texture of the case to demonstrate the process of QGPU intuitively. For the unwrapping path, colors of the pixels represents the unwrapping sequence. The pixels with light colors are unwrapped prior to those with dark colors.

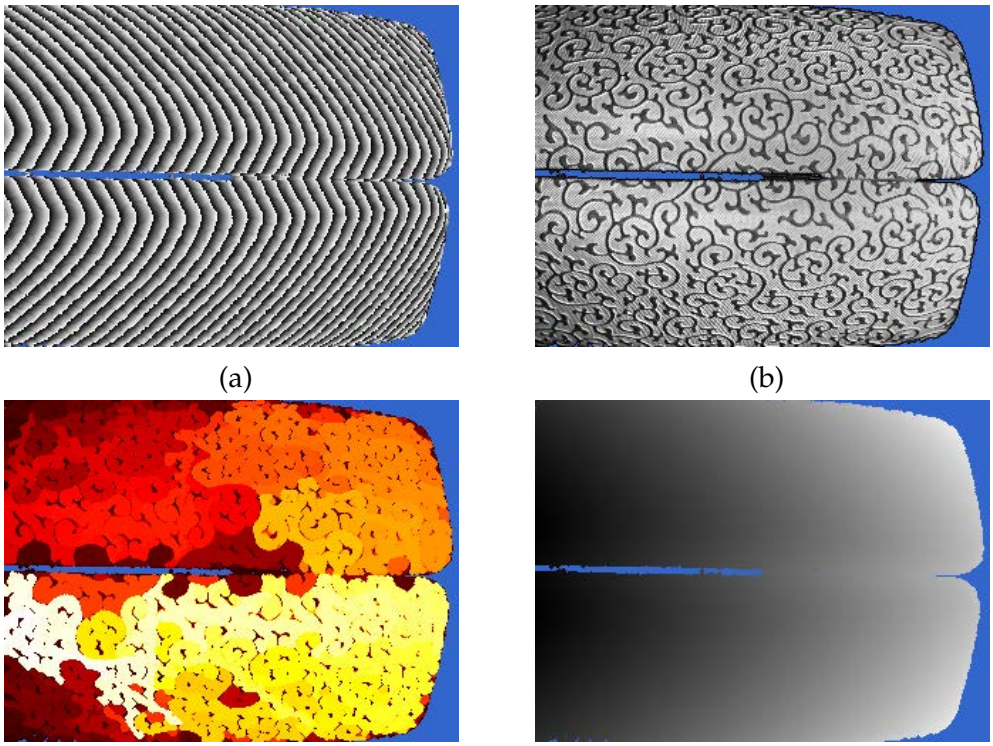


Figure 2.5: Example of quality guided phase unwrapping. (a) Wrapped phase map, (b) quality map, (c) unwrapping path map (pixels in light yellow are unwrapped first and those in dark red are unwrapped last), and (d) unwrapped phase map. Reprinted from "Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies," Appl. Opt. 50, 6214-6224 (2011) with permission from Optical Society of America.

In QGPU, quality maps and guiding strategies are two major aspects. The quality map defines how to measure the goodness of pixels which determines the accuracy of the algorithm. The guiding strategy controls how to follow the quality map in the QGPU process which majorly affects the speed of the algorithm.

Many quality maps have been proposed, including the correlation coefficient map in InSAR [109], modulation map or reliability map from optical metrology [50, 100], pseudo correlation coefficient map [48], phase derivative variance map [108], maximum phase gradient map [48], first phase difference map [50], second phase difference map [49,106,107], amplitude map or ridge map from transform methods [50, 52, 102]. For guiding strategy, the most common approach is the classical quality guiding strategy [50], which always select the pixel with the highest quality for unwrapping. Two-section guiding strategy and stack-chain guiding strategy are two examples of guiding strategies as well [99].

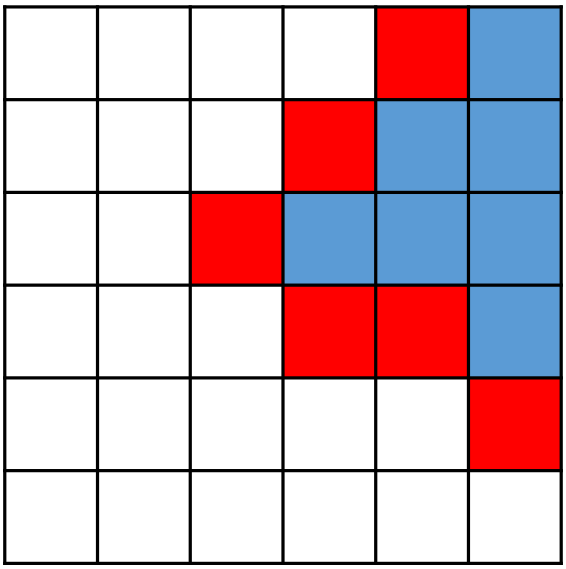


Figure 2.6: Demonstration of spatial phase unwrapping process. Blue pixels are unwrapped, and red pixels are in the adjoin list.

All red pixels in Fig. 2.6 are the candidates for unwrapping, and they can be unwrapped based on their unwrapped (blue) neighbors. Each red pixel are placed in a queue called adjoin list. The position in the adjoin list is determined by the quality value of the pixel. Each time, the pixel with the highest quality is selected for unwrapping. Pixels in noisy or discontinuous areas are usually assigned low quality values and they will moved to the back of the queue in the QGPU process.

The process can be described as follows.

Step 1: Look for the pixel with the highest quality in the quality map and insert it into the adjoin list.

Step 2: Look for the pixel with the highest quality in the adjoin list. The found pixel is then unwrapped and removed from the adjoin list. Its unprocessed neighbors are inserted into the adjoin list successively.

Step 3: Repeat step 2 until the adjoin list is empty.

Chapter 5 offers a detailed comparison of the different quality maps and guiding strategies, which is helpful for selecting a proper quality map and guiding strategy.

2.3.2 Path independent methods

In this section, a different family of algorithms is introduced, which imposes constraints in a mathematical manner, rather than following an explicit path as path following methods. These methods attempt to find the unwrapped phase values whose local gradients is close to the wrapped phase. The term "close to" is described by an optimization function.

Least squares methods

The first type is the least squares method. The least squares method has two forms, uniform weighted (sometimes called unweighted) and non-uniform weighted (sometimes called weighted), which is represented below:

$$minimize \sum_{i,j} w^x_{i,j}(\Delta\varphi^x_{i,j} - \Delta\varphi_{w}{}^x_{i,j})^2 + w^y_{i,j}(\Delta\varphi^y_{i,j} - \Delta\varphi_{w}{}^y_{i,j})^2, \tag{2.11}$$

where $\Delta\varphi^x_{i,j}$ and $\Delta\varphi_{w^x_{i,j}}$ are the phase gradients in x direction of unwrapped phase and wrapped phase, $\Delta\varphi^y_{i,j}$ and $\Delta\varphi_{w^y_{i,j}}$ are their counter part in y direction, $w^x_{i,j}$ and $w^y_{i,j}$ are user-defined weights.

If the weights are unweighted, the least squares problem can be solved by solving a Poisson's equation [110]:

$$\nabla^2\hat{\Phi} = \hat{\nabla}^2\Psi, \tag{2.12}$$

where the $\hat{\nabla}^2$ is the difference of the wrapped gradients, Ψ is the wrapped phase, and Φ is the estimation of unwrapped phase. Since the wrapped phase is already known and the right side can be obtained also, the estimation of unwrapped phase could be computed.

If the weights are non-uniform (weighted), the above Poisson's equation is not applicable. Reference [110] proposed a method which computed weighted least squares by computing unweighted least squares solutions iteratively.

Based on the work presented in Ref. [110], many researchers proposed their variations of least squares phase unwrapping methods. These methods include the multigrid method [108], the multi-resolution method [111] and the Green's functions method [112].

L^p norm methods

In Ref. [113], the authors improved original least squares method and proposed a more generalized solution. The new solution is called L^p norm method and represented as below:

$$minimize \sum_{i,j} w^x_{i,j}|\Delta\varphi^x_{i,j} - \Delta\varphi_{w^x_{i,j}}|^p + w^y_{i,j}|\Delta\varphi^y_{i,j} - \Delta\varphi_{w^y_{i,j}}|^p, \tag{2.13}$$

Just like the least squares method, this function could reach the minimal if the gradients of wrapped phase and unwrapped phase are close. The parameter p determines how differences between $\Delta\varphi$ and $\Delta\varphi_w$ are penalized. If $p = 2$, it reduces to the least squares method as introduced above.

Comparing to the original least squares method, the L^p Norm method is more flexible. We can change p freely to obtain different penalties of the difference between unwrapped phase and wrapped phase. Moreover, the L^p norm method can generate weights directly rather than using explicit weights like the least squares method. Also, extra weights or quality maps can be used to improve the results.

Graph cut methods

In Ref. [58], the authors introduced a new phase unwrapping approach, the phase unwrapping via max flow (PUMA), based on graph cut algorithm [114]. This approach is an extension of L^p norm methods. It rewrites Eq. 2.13 as:

$$E(k|\varphi_w) = \sum_{i\in[1,M],j\in[1,N]} |\Delta\varphi_{ij}^h|^p \bar{v}_{ij} + |\Delta\varphi_{ij}^v|^p \bar{h}_{ij} \tag{2.14}$$

where $p \geq 0$, φ_w is the wrapped phase map, $(\cdot)^h$ and $(\cdot)^v$ represent the horizonal and vertical phase differences, h_{ij} and v_{ij} are binary discontinuities for horizonal and vertical directions, $\bar{h}_{ij} = 1 - h_{ij}$ and $\bar{v}_{ij} = 1 - v_{ij}$, M and N are the numbers of rows and columns of the wrapped phase map, k is called "wrap count" and the unwrapped phase φ can be obtained by $\varphi = 2\pi k + \varphi_w$. The authors aimed to find a k that can minimize the energy function Eq. 2.14. Based on [115], they proved that if the minimum of E is obtained, there will be no binary map δk which makes $E(k + \delta k|\psi) < E(k|\psi)$, so the minimal of Eq. 2.14 could be reached by a series of binary operations. Meanwhile, according to Ref. [116], the optimization process could be described by a graph, and obtained by computing its max-flow/min-cut.

Based on the outcome of graph cut, the authors proposed some improved methods, phase estimation using adaptive regularization based on local smoothing (PEARLS) method [117] and combinatorial absolute phase estimation (CAPE) method [118]. The PEARLS adopted a new filtering technique which was based on local polynomial approximation [119] to reduce noise. Comparing with the original PUMA algorithm, the new filtering scheme got a better result in noisy conditions. The CAPE used a coarse to fine scheme to perform denoising process. According to the authors' results, the CAPE performed much better when discontinuity exist, but worse than the PEARLS if there were no discontinuities and Signal-Noise ratio(SNR) was low.

Bayesian/Regularization methods

Bayesian/Regularization methods consider phase unwrapping as an ill-posed problem, and regularization techniques are used to solve it. Nico et. al. [120] summarized the Bayesian based phase unwrapping methods as follows:

$$\mathcal{P}(f|g) = \mathcal{L}(g|f) + \lambda\mathcal{R}(f) \tag{2.15}$$

where g is the wrapped phase, f is unwrapped phase, $\mathcal{L}(g|f)$ represents how the reconstruction data is consist with the observed data, R is the regularization term, λ is the weight parameter. The unwrapped phase can be obtained with a *maximum a posteriori* (MAP) estimate:

$$\hat{f} = \arg\max_{f} \mathcal{P}(f|g) \tag{2.16}$$

Marroquin et. al. [56] introduced a method extended the classical least squares solution. Their method could solve some noise cases, but incapable to deal with

high noise and high phase gradient cases. Servin et. al. [121] proposed a regularized phase-tracking (RPT) system. This new system assumed pixels in a small area could be modeled as a plane and attempted to fit a least squares tangent plane in both wrapped and unwrapped phase map. The RPT system was more robust under high noise condition, but the setting of neighbor window size was very tricky. In Ref. [122], the authors improved the regularization method with a half-quadratic version. Compare with previous methods, their method obtained better performance on both noisy and discontinuous cases. In Ref. [123], the authors proposed an algorithm based on stochastic relaxation. This method considered the phase unwrapping problem as an optimization problem with constraints and used simulated annealing to obtain the solution. It performed better than least squares method under noisy condition, but the computation time was increased explicitly.

2.4 Snake models

Snake (or active contour) models are designed to find object boundaries in images. They can be represented as a curve, marching in an image to minimize the following energy function [124]

$$E = \int_0^1 \frac{1}{2}[\alpha|\mathbf{x}'(s)|^2 + \beta|\mathbf{x}''(s)|^2] + E_{ext}[\mathbf{x}(s)]ds, \tag{2.17}$$

where $\mathbf{x}(s) = [x(s), y(s)]$, $s \in [0, 1]$; $\mathbf{x}'(s)$ and $\mathbf{x}''(s)$ are first-order and second-order derivatives of $\mathbf{x}(s)$, respectively; α and β are their weights; and E_{ext} is the external energy term. The first two terms represent the internal forces. The first-order derivative controls the tension of the curve, which makes the curve like a rubber band. The second derivative controls the rigidity of the curve, which prevents the

curve from bending. As for the external force, one possible form is as follows [124]

$$E_{ext} = -|\nabla I(x,y)|^2, \tag{2.18}$$

where $I(x,y)$ is the given image and ∇ is the gradient. To minimize E, the following equation needs to be satisfied

$$\alpha \mathbf{x}''(s) - \beta \mathbf{x}''''(s) - \nabla E_{ext}[\mathbf{x}(s)] = 0. \tag{2.19}$$

This equation can be further represented as a force balance equation

$$F_{int} + F_{ext} = 0, \tag{2.20}$$

where the internal force term $F_{int} = \alpha \mathbf{x}''(s) - \beta \mathbf{x}''''(s)$ and the external force term $F_{ext} = \nabla E_{ext}[\mathbf{x}(s)]$. The internal force prevents stretching and bending, while the external force pulls the curve to the discontinuous areas. This equation can be solved by a semi-implicit iterative method [124, 125].

The original snake model performs well, but it highly relies on the initial position of the snake, and performs poorly in concave regions. To solve these problems, a gradient vector flow (GVF) snake model is proposed [126]. A GVF field $\mathbf{v}(x,y) = [u(x,y), v(x,y)]$ is used as a new external force term. The GVF field is computed by minimizing the following energy function

$$E_{GVF} = \int\int \mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla e|^2 |\mathbf{v} - \nabla e|^2 dxdy, \tag{2.21}$$

where e is an edge map obtained from the image; μ is a regularization term; ∇ is

a gradient operator; and u and v can be obtained by solving following equations

$$\mu\nabla^2 u - (u - e_x)(e_x^2 + e_y^2) = 0, \tag{2.22}$$

$$\mu\nabla^2 v - (v - e_y)(e_x^2 + e_y^2) = 0, \tag{2.23}$$

with ∇^2 as the Laplace operator.

Snake models are widely used in image segmentation and object recognition. For example, in medical imaging [such as magnetic resonance imaging (MRI), computed tomography (CT), and ultrasound], extracting regions of interest (ROIs) from images are necessary for further analysis. However, the images are usually low contrast and noisy. It is thus very challenging to segment ROIs. In Refs. [127–130], snake models were applied to MRI and CT images, and achieved very good performance.

The snakes have been previously applied to fringe pattern analysis as well. In Ref. [131], snakes are used to detect fringe lines of 2π jumps for phase unwrapping, where the underlying phase is still continuous. In Ref. [132], snakes are used to find object boundaries without phase unwrapping. In Ref. [133], snakes are used to find π jumps to solve the sign ambiguity problem in phase retrieval from a single fringe pattern, rather than to solve the order ambiguity problem in phase unwrapping.

Chapter 3

Multi-core implementation of the windowed Fourier transform algorithms

The WFT based algorithm are powerful tools for fringe pattern analysis, and a fast implementation of the WFT based algorithms is always desired in real applications. In this chapter, how to efficiently implement the WFT based algorithms is investigated[1]. First, Fourier transform, the core component of the WFT based algorithms, is first studied and improved. Second, the main body of the WFT algorithms are parallelized, because the operations in the main body are independent. With these two improvements, the new implementation performs 10 times faster than the original program, and subsequent analyses, such as denoising and phase unwrapping, are also benefited.

[1]This chapter is reproduced from "Multi-core implementation of the windowed Fourier transform algorithms for fringe pattern analysis," Appl. Opt. 54, 587-594 (2015) with permission from Optical Society of America.

3.1 Introduction

In previous chapters, the principles of the WFT based algorithms have already been introduced. From the principles, it is easy to find that the WFR2/WFF2 have an exhaustive searching process. This searching process is usually time-consuming. For example, the current WFR2/WFF2 MATLAB programs [134] costs about 3253.94s and 703.7s to process a 1024×1024 fringe pattern with default parameters. The high time cost limits the usage of the WFT based algorithm, and a fast implementation is expected. Previously, some attempts have been made to improve the performance of the WFT based algorithms, such as MATLAB parallel toolbox for a multi-core CPU [83] and a graphics processing unit (GPU) [86], and CUDA C for a GPU [85]. In general, MATLAB program and GPU program usually have different focuses. The MATLAB program is easy to use and takes short time to develop, and thus suitable for research areas. However, its poor performance makes it inappropriate for industrial applications. On the contrast, the GPU program has high performance and great potential. However, it cannot be used in the computers without GPU chips, and it takes much longer time to develop and tune. A multi-core CPU based implementation fulfills both speed and convenience requirements, and it is thus explored in this chapter.

The WFT algorithms have a common characteristic that they perform Fourier transforms repeatedly in their main bodies. This characteristic inspires us to improve the performance from two aspects: (1) directly improve the speed of a single Fourier transform; (2) use parallel techniques to utilize the power of multi-core CPUs by assigning different Fourier transforms to different CPU cores. The novelties of this chapter are summarized as follows:

(i) The implementation of Fourier transform has been studied for decades, and a well engineered implementation can achieve a very good performance. Two

popular libraries, FFTW [135] and Intel's Math Kernel library (MKL) [136], are selected and compared. The influences of compilers and size of input fringe patterns to Fourier transform are also discussed.

(ii) To parallelize Fourier transforms in the WFT algorithms, an efficient parallel technique is necessary. In this chapter, four popular CPU based C++ parallel techniques, including OpenMP [137], Microsoft's Parallel Patterns Library (PPL) [138], Intel's Cilk Plus [139], and Intel's Threading Building Blocks Library (TBB) [140], are investigated.

After the two investigations above, the faster Fourier transform library and the fastest parallel technique will be chosen and integrated into our new WFT implementation. The final implementation achieves about 10 times improvement than the original MATLAB program on average. When testing on the 1024×1024 example above, the new WFR2/WFF2 implementations spend 297.2s and 40.04s, respectively, which are $11\times$ and $17\times$ faster than the original programs.

3.2 Analysis of the WFR2/WFF2

The pseudo-codes of the WFR2 and WFF2 algorithms are shown in Algorithm 1 and Algorithm 2, respectively. For each frequency pair (ξ_x and ξ_y), the WFR2 needs to perform one convolution, and the WFF2 requires two convolutions. In practice, a convolution is replaced by forward and inverse Fourier transforms which are equivalent and more efficient. The forward and inverse transforms are denoted as $\mathcal{F}$ and $\mathcal{F}^{-1}$, respectively [19],

$$f(x,y) \otimes g(x,y) = \mathcal{F}^{-1}[\mathcal{F}(f)\mathcal{F}(g)], \tag{3.1}$$

In the WFR2/WFF2, all frequency pairs of (ξ_x and ξ_y) from $[\xi_{xl}, \xi_{xh}] \times [\xi_{yl}, \xi_{yh}]$

are computed, which means we need to compute a huge number of forward and inverse Fourier transforms. Repeated Fourier transforms make the WFR2/WFF2 very protracted. Using a faster Fourier transform library is needed to optimize the performance. Detailed discussions for optimizing the Fourier transform performance is presented in Sec. 3.4.

Algorithm 1 Pseudo-codes of the WFR2. The input fringe pattern is denoted as f. Reproduced from "Multi-core implementation of the windowed Fourier transform algorithms for fringe pattern analysis," Appl. Opt. 54, 587-594 (2015) with permission from Optical Society of America.

generate a window g {Eq. 2.3}
for $\xi_{yt} = \xi_{yl} : \xi_{yi} : \xi_{yh}$ **do**
  for $\xi_{xt} = \xi_{xl} : \xi_{xi} : \xi_{xh}$ **do**
    compute $g_{\xi_x,\xi_y}(x,y)$ {Eq. 2.2}
    Sf=conv2(f, $g_{\xi_x,\xi_y}(x,y)$) {Eq. 2.1}
    update the ridges from Sf {Eq. 2.4}
    update the wrapped phase {Eq. 2.5}
  end for
end for

Algorithm 2 Pseudo-codes of the WFF2. The input fringe pattern is denoted as f. Reproduced from "Multi-core implementation of the windowed Fourier transform algorithms for fringe pattern analysis," Appl. Opt. 54, 587-594 (2015) with permission from Optical Society of America.

generate a window g {Eq. 2.3}
for $\xi_{yt} = \xi_{yl} : \xi_{yi} : \xi_{yh}$ **do**
  for $\xi_{xt} = \xi_{xl} : \xi_{xi} : \xi_{xh}$ **do**
    compute $g_{\xi_x,\xi_y}(x,y)$ {Eq. 2.2}
    Sf=conv2(f, $g_{\xi_x,\xi_y}(x,y)$) {Eq. 2.1}
    filter Sf with *threshold* {Eq. 2.7}
    Rf = conv2(f, $g_{\xi_x,\xi_y}(x,y)$) {Eq. 2.6}
  end for
end for

From the pseudo-codes in Algorithm 1 and Algorithm 2, we can observe that the operations of different frequency pairs are independent. It indicates that parallel computing is applicable in the WFR2/WFF2. Different iterations can be distributed to different compute units, and computed simultaneously. How to parallelize the WFR2/WFF2 is presented in Sec. 3.4.

3.3 Accelerating the Fourier transform

As shown in Algorithm 1 and Algorithm 2, the computation of Fourier transform is the major source of time consumption. In this section, we focus on three aspects of implementing a single Fourier transform.

3.3.1 Library selection

Fourier transform is usually computed by fast FT (FFT) algorithms, such as the Cooley-Turkey algorithm and the prime factor algorithm [141]. Without utilizing those FFT algorithms, Fourier transform could not be widely applied. The FFTW and Intel's MKL are two well engineered and widely used FFT libraries. Their outstanding performances have been examined in benchmarks [142, 143] and real applications. In this paper, they are selected for further comparison.

The FFTW is a popular FFT library and specially designed for computing FFT. The library contains several FFT algorithms. Before a FFT computation, the library examines the environment and decides the fastest way to compute FFT. Comparing with using a fixed algorithm, the strategy of FFTW is more adaptive and efficient [135]. Intel's MKL is a widely used numerical library which includes common categories of math functions, such as linear algebra, statistics and vector math. The MKL also offers functions to compute FFT. As Intel is a CPU manufacturer, it is possible to provide fine optimizations based on the features of CPUs to the MKL. These optimizations cover different generations of Intel or other compatible CPUs, and most users can be benefited without any changes of their programs [136]. The FFTW and the MKL are compared to find the faster FFT implementation.

3.3.2 Compiler selection

Before executing in CPU, programs must be compiled by a compiler. Compilers determine the quality of final programs. Therefore, after studying the selection of FFT libraries, the selection of compilers are also studied. As we focus on the Microsoft's Windows platform, two popular compilers on this platform, Microsoft C++ compiler (MSVC) and Intel C++ compiler (ICC), are selected and compared.

The MSVC is provided by the Microsoft's integrated development environment (IDE), Visual Studio, and most of Windows developers use it for their programs. The ICC is specially designed for high performance programs. It can generate optimized binary codes for Intel's or other compatible CPUs.

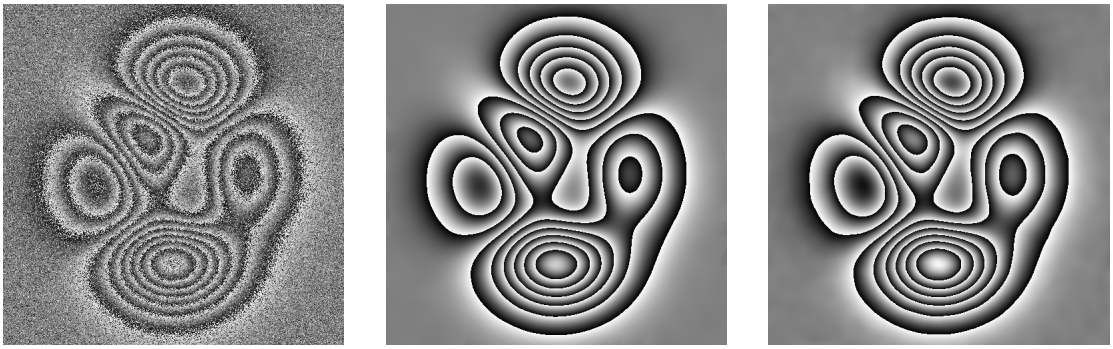
3.3.3 Size of the input fringe pattern of the FFT

When applying FFT on a fringe pattern, its size influences the speed of the FFT. To reduce computational complexity, the FFT algorithms ususally decompose the original transform to smaller transforms. This decomposition process is performed recursively. When the decomposed transforms reach certain pre-defined transform, the process stops. For instance, if the size of the input fringe pattern can be expressed as the form of $2^a3^b5^c7^d11^e13^f$, the FFTW will be more efficient [144], and the MKL has a similar preference [145]. Due to the preferences of the FFTW and the MKL, the size of the input fringe pattern should be the multiple of the pre-defined values. However, most of input fringe patterns cannot satisfy this requirement. A zero-padding is needed to expand the input fringe patterns to preferred sizes. In addition, a fringe pattern may be expanded to multiple preferred sizes. To make the padding process clearer and more convenient, we test all valid combinations and record the fastest FFT one into a look-up table. Given a fringe pattern, its preferred size can be easily obtained with the assistance of the look-up table.

3.3.4 Experiments

Three factors of implementing a single Fourier transform is introduced above. To have a better understanding, different combinations of those three factors are examined. Three phase maps with the sizes of 256×256, 512×512, and 1024×1024

are generated by MATLAB's built-in *peaks* function. In practise, exponential phase fields (EPF) is the most common input of the WFT based algorithms [19], so the test wrapped phase maps are all converted to EPFs to simulate real applications. In addition, normally distributed random noise is added to these EPFs. Because all EPFs have similar looking, only the wrapped phase of the simulated EPF with the size of 512×512 is shown in Fig. 3.1(a) for demonstration. The WFR2/WFF2 outputs of the wrapped phase map are shown in Figs. 3.1(b) and 3.1(c).



(a) (b) (c)

Figure 3.1: Test wrapped phase map for FFT comparison. (a) The wrapped phase map; (b) and (c) are the results of WFR2 and WFF2. Reproduced from "Multi-core implementation of the windowed Fourier transform algorithms for fringe pattern analysis," Appl. Opt. 54, 587-594 (2015) with permission from Optical Society of America.

From Algorithm 1, it is obvious that the fringe pattern $f(x, y)$ of size $m \times n$ is convolved with a kernel $g_{\xi_x,\xi_y}(x, y)$ of size $k_m \times k_n$ in the WFR2. Before the convolution starts, an initial padding process is applied to both the fringe pattern and the kernel, which expands them to a size of $(m + k_m - 1) \times (n + k_n - 1)$ [146]. After that, an extra padding which expands the size to a preferred size is applied subsequently. The second padding process aims to increase the speed of the FFT. In the WFF2, the same padding process is also applied. In our experiments, the kernel size is selected as 61×61, thus the image sizes after initial padding are 316×316, 572×572, 1084×1084, for three EPFs, respectively. After the second padding, the sizes become $320 \times 320 = (2^6 \times 5) \times (2^6 \times 5)$, $576 \times 576 = (2^6 \times 3^2) \times$

$(2^6 \times 3^2)$, and $1120 \times 1120 = (2^5 \times 5 \times 7) \times (2^5 \times 5 \times 7)$, respectively.

The computer for testing is a Dell Precision T3600 workstation. This workstation equips a six-core Intel Xeon E5-1650 CPU with a frequency of 3.2GHz and 16 GB memory. The operating system is Windows Server 2012 R2. The versions of the MKL and FFTW are 11.2 and 3.3.4, respectively. The compilers are the MSVC which is shipped with Visual Studio 2013 and ICC 15.0. For each fringe pattern, forward and inverse FFTs are computed for 1000 times, and the average is used as the final benckmark.

Table 3.1: FFT Comparison, time (ms) per pair of forward and inverse FFTs. Reproduced from "Multi-core implementation of the windowed Fourier transform algorithms for fringe pattern analysis," Appl. Opt. 54, 587-594 (2015) with permission from Optical Society of America.

FFT Library	Compiler	Preferred Size	Original image size		
			256×256	512×512	1024×1024
FFTW	MSVC	No	13.41	14.55	176.44
		Yes	2.74	11.08	61.77
	ICC	No	9.75	10.48	153.35
		Yes	2.94	10.33	56.73
MKL	MSVC	No	12.2	13.63	174.36
		Yes	2.53	10.32	58.42
	ICC	No	9.17	10.38	153.31
		Yes	**2.51**	**10.09**	**50.21**

In Table 3.1, the comparison results of different combinations of the three factors above are demonstrated. From the results, the MKL runs faster than the FFTW in the library comparison, and the difference is about 18%. In the comparison of compilers, the ICC is better, while the difference is only 5%. For the comparison on the effect of the preferred sizes, the program with the second padding runs 2$\times$ faster than the ones with the initial padding sizes. Among these three factors, using preferred sizes offers the most significant improvement. Based on the results, in the following study, the MKL, the ICC, and the preferred sizes are used as the default configuration.

3.4 Parallelizing the WFR2 and the WFF2

In the WFR2/WFF2, a large number of Fourier transforms are performed repeatedly. The performance of a single Fourier transform has been improved in the previous section. In this section, we aim to improve the main body of the algorithms. From Algorithms 1 and 2, the main body of the WFR2/WFF2 is nested loops. As described in Sec. 3.2, the computations of different frequency pairs are independent. Therefore, parallelizing the main body of the WFR2/WFF2 is applicable. In this section, a parallel implementation on multi-core CPUs which are ubiquitous in modern computers is introduced.

3.4.1 Parallel techniques

When using early parallel techniques, such as Pthreads [147], programmers have to create, maintain, and synchronize threads explicitly. These low-level operations are vulnerable to errors [148]. In modern parallel techniques, those low-level operations are hidden by simpler high-level operations, and programmers are thus liberated. Those new parallel techniques can be simply classified into two types. One type of techniques, such as OpenMP [135] and Intel's Cilk Plus [139], provides parallelization support by adding compiler directives to existing programming languages. The other type of techniques, such as Intel's Thread Building Blocks (TBB) and Microsoft's Parallel Patterns Library (PPL), provides parallelization support by offering libraries to existing programming languages. These techniques support different parallelization scheme. In this section, due to the characteristic of the WFR2/WFF2, only the loop parallelization scheme of the four techniques are considered.

OpenMP

OpenMP [135] is an Application Program Interface (API) which supports multi-threaded and shared memory parallelism. It defines a set of compiler directives which make compilers capable to parallelize codes. Also, OpenMP contains a few of functions and environment variables to assist the parallelization. Currently, most modern compliers and CPU architectures support the usage of OpenMP. In OpenMP, regions to be parallelized are defined by a *parallel* directive. The loops in the parallel region are declared by a *for* directive, and different iterations of a loop can be automatically distributed to threads.

Cilk Plus

Intel's Cilk Plus [139] is an extension for the C and C++ programming language to support parallelism. It adds three keywords to create parallel loops or spawn functions. In Cilk Plus, parallelization can be simply achieved by replacing the original keyword *for* in C++ with a new keyword *cilk_for*. The parallelization is implicit and automatic.

Thread Building Blocks (TBB)

Unlike OpenMP and Cilk Plus, Intel's TBB [140] is a C++ template library for parallelism. It encapsulates low-level operations into tasks. Programmers can use those tasks to parallelize codes via defining a class or using a lambda expression [149]. The TBB is supported by many compilers and Intel compatible processors.

In the TBB, parallelizing a loop needs two steps. First, we need to define a class for parallelization. The original loop body will be put in the overloaded *()* operator of the defined class. The iteration range of the loop will be imported as an argument of the *()* operator. Second, the original *for* loop is replaced by a TBB

built-in template function, *parallel_for*. This function takes the pre-defined class and iteration range as arguments. When calling the *parallel_for* function, the TBB will identify the loop body and iteration range, and automatically dispatch tasks to different threads.

Parallel Patterns Library (PPL)

Microsoft's PPL [138] is also a C++ template library for parallelism. It runs on the Windows platform, and has similar semantics as the TBB. A *parallel_for* function is provided for parallelism, and it is compatible with the function with the same name in the TBB [140].

MATLAB/Math Kernel Library

In current Fourier transform libraries, such as MATLAB and the MKL, parallelism is also provided in their built-in functions. When the functions are called, the computation of Fourier transform will be implicitly parallelized. The original MATLAB programs of the WFR2/WFF2 use this parallel method.

3.4.2 Loop parallelization

The main body of the WFR2/WFF2 is nested loops (as shown in Algorithms 1 and 2), thus both the outer loop (represents the iterations of frequencies in y direction) and the inner loop (the iterations of frequencies in x direction) can be parallelized. Parallelizing the outer loop is better, because it has less overhead cost caused by thread maintenance. Once the frequency bound (ξ_{yl} and ξ_{yh}) and sampling interval (ξ_{yi}) are determined, the number of threads and the amount of tasks for each thread only depends on the number of CPU cores. If the CPU has n cores, n threads are required, and the iterations of the outer loop will be equally divided

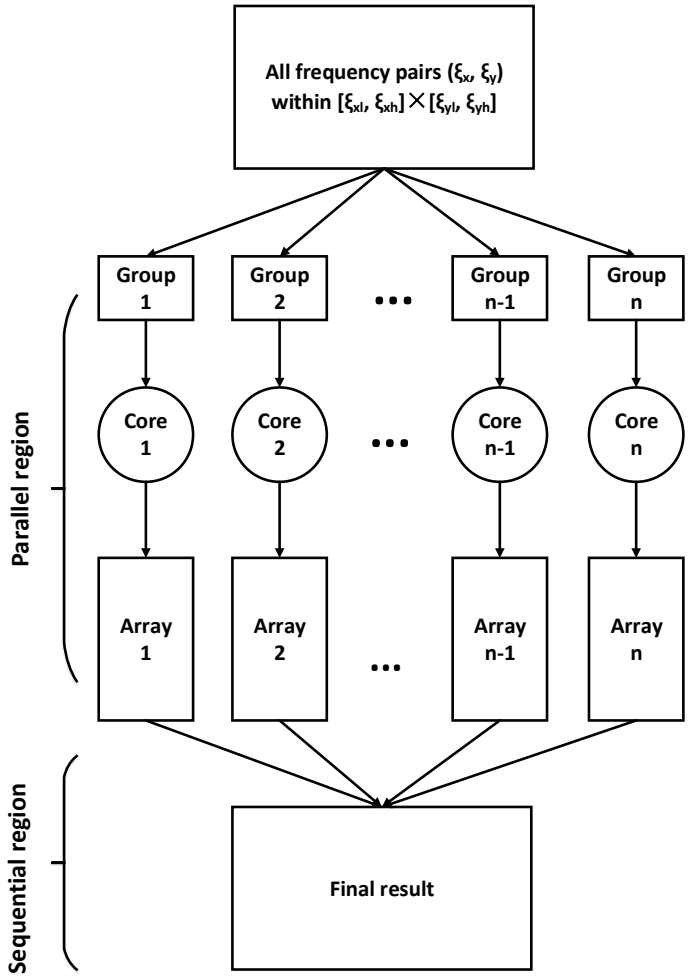


Figure 3.2: Parallelization of the WFR2/WFF2 loops. Reproduced from "Multi-core implementation of the windowed Fourier transform algorithms for fringe pattern analysis," Appl. Opt. 54, 587-594 (2015) with permission from Optical Society of America.

into n groups. One thread processes one task group. Meanwhile, a dedicated array for intermediate result is also prepared for each thread. Therefore, each thread has its own tasks and data arrays, and no interference exists among different threads. When the algorithm starts, the operating system schedules all working threads automatically until the outer loop is completed. Finally, after the outer loop is completed, intermediate results from threads are merged to obtain the final result. The entire process is demonstrated in Fig. 3.2.

3.4.3 Experiments

The WFR2/WFF2 algorithms are implemented by the above four techniques. The comparison result provides a clear impression of the performance differences.

Configurations

The same configurations used in Sec. 3.3 are adopted again. OpenMP, Cilk Plus, and the TBB are from Intel Parallel Studio XE 2015. The PPL comes from Microsoft Visual Studio 2013. In the experiments, we also use the three wrapped phase maps used above. To reduce the effect of background programs, each testing program will be executed for 5 times, and the average is computed as the final result. During the entire process of the programs, six CPU cores are all fully utilized. The WFR2/WFF2 parameters are set as follows: $\sigma_x = \sigma_y = 10$; $\xi_{xl} = \xi_{yl} = -2$; $\xi_{xh} = \xi_{yh} = 2$; $\xi_{xi} = \xi_{yi} = 0.025$ (the WFR2); $\xi_{xi} = \xi_{yi} = 0.1$ (the WFF2); $threshold = 6$ (the WFF2). From Figs. 3.1(b) and 3.1(c), noises are successfully reduced, so only speed comparison is covered in this section.

To have a better understanding about the improvements from parallel techniques, two MATLAB programs, a sequential version and a build-in parallel version of the WFR2/WFF2 [134], are are chosen to compare with the parallel programs. In addition, the performance of a sequential C++ program and a C++ program with build-in MKL parallel Fourier transform is also examined.

Parallel Performance

Tables 3.2 and 3.3 illustrate the time costs for processing the 1024×1024 fringe pattern. The costs for remaining fringe patterns are not shown in the tables, but will be discussed below. From the results, we can observe follow findings

- All the C++ implementations are faster than their MATLAB counterparts.

Table 3.2: The WFR2 time costs of the 1024 × 1024 example. MATLAB (S) and C++ (S) are the sequential versions of the MATLAB and C++ programs, respectively. Reproduced from "Multi-core implementation of the windowed Fourier transform algorithms for fringe pattern analysis," Appl. Opt. 54, 587-594 (2015) with permission from Optical Society of America.

Item	Technique	Time (s)	Speedup Comparing with A	Speedup Comparing with B	Speedup Comparing with C
A	MATLAB (S)	11916.59	-	-	-
B	MATLAB	3253.94	3.66×	-	-
C	C++ (S)	1549.89	7.69×	2.10×	-
D	MKL	665.41	17.91×	4.89×	2.33×
E	OpenMP	**297.2**	**40.10**×	**10.95**×	**5.21**×
F	Cilk Plus	318.35	37.70×	10.29×	4.90×
G	TBB	313.05	37.43×	10.22×	4.87×
H	PPL	316.13	38.07×	10.39×	4.95×

Table 3.3: The WFF2 time costs of the 1024 × 1024 example. Reproduced from "Multi-core implementation of the windowed Fourier transform algorithms for fringe pattern analysis," Appl. Opt. 54, 587-594 (2015) with permission from Optical Society of America.

Item	Technique	Time (s)	Speedup Comparing with A	Speedup Comparing with B	Speedup Comparing with C
A	MATLAB (S)	1136.08	-	-	-
B	MATLAB	703.7	1.61×	-	-
C	C++ (S)	208.6	5.45×	3.37×	-
D	MKL	78.76	14.42×	8.93×	2.65×
E	OpenMP	**40.04**	**28.37**×	**17.57**×	**5.22**×
F	Cilk Plus	42.64	26.91×	16.67×	4.94×
G	TBB	42.75	26.64×	16.50×	4.89×
H	PPL	42.21	26.57×	16.50×	4.87×

Even the sequential C++ implementation still outperforms the MATLAB parallel program. This situation is not unexpected. Because MATLAB is an interpreted programming language and always requires an interpreter to execute the programs. C++ is a compiled programming language. Before executing in CPU, the programs are compiled to machine codes which can be executed by a computer directly.

- All parallel implementations (MATLAB and C++) outperform their sequen-

tial counterparts. It proves that parallelization is an effective way to improve the performance of programs.

- All four parallel techniques (OpenMP, Cilk Plus, the TBB, and the PPl) need to modify existing codes to utilize parallelization (see Sec. 3.4.3). Using built-in parallelized functions in the MKL, but the MKL built-in parallelization is slower than other parallel techniques.

- Among four parallel techniques (OpenMP, Cilk Plus, the TBB and the PPL), OpenMP is the fastest one, and the differences are about 5-10%.

- Comparing with the existing WFR2/WFF2 programs, the OpenMP programs achieve about speed-ups of 11$\times$ and more than 17$\times$, respectively.

- For the other two sizes fringe patterns (256×256 and 512×512), we can observe similar trends. The time costs of OpenMP are 18.7s (256×256, the WFR2), 2.51s (256×256, the WFF2), 77s (512×512, the WFR2) and 10.63s (512×512, the WFF2). The time costs of their MATLAB counterparts are 153.37s (256×256, the WFR2), 16.28s (256×256, the WFF2), 644.04s (512×512, the WFR2) and 136.40s (512×512, the WFF2). The corresponding speed-up are 8.20$\times$, 6.50$\times$, 8.36$\times$ and 12.83$\times$, respectively.

- In our experiments, the used frequency bounds can cover most of fringe patterns [19]. However, a smaller frequency is sufficient for many fringe patterns. If we set $\xi_{xl} = \xi_{yl} = -1$ and $\xi_{xh} = \xi_{yh} = 1$, the execution time can be further reduced by 4 times.

Table 3.4: Code churn results. Reproduced from "Multi-core implementation of the windowed Fourier transform algorithms for fringe pattern analysis," Appl. Opt. 54, 587-594 (2015) with permission from Optical Society of America.

	OpenMP		Cilk Plus		TBB		PPL	
	WFR2	WFF2	WFR2	WFF2	WFR2	WFF2	WFR2	WFF2
LA	31	13	27	9	40	18	26	8
LD	0	0	0	0	2	2	2	2
LM	51	41	51	41	53	47	53	47
Total	82	54	78	50	85	67	81	57

Code Modification

Once we would like to parallelize existing programs, code modification on original programs is inevitable. Evaluating the complexity of the modification is therefore necessary. In this section, the Code Churn method [150] is adopted to estimate the complexity. The Code Churn method measures three factors: the amount of added lines (LA), deleted lines (LD), and modified lines (LM) of the program. The total churn value, which is the sum of LA, LD and LM, represents the estimated total complexity.

The parallel programs are compared with the original sequential program, we use the Code Churn method to measure changes of each parallel techniques. The sequential WFR2 program has 150 lines, and the sequential WFF2 program has 179 lines. Table 3.4 summarizes the estimated complexities of parallel techniques. Memory allocation and parallel loop control are two major sources of code changes. In our parallel programs, each thread are provided a dedicated array for storing intermediate results, and this operation causes extra code changes. These four parallel techniques use the same memory allocation codes. For the parallel loop control, OpenMP and Cilk Plus require fewer code changes, because they only need to add a few of compiler directives, while the TBB and PPL need to define classes to encapsulate the codes to be parallelized. The differences are reflected by their churn values.

3.5 Summary

In this chapter, how to accelerate the existing windowed Fourier transform based algorithms is discussed. Fourier transform is the core component of the WFR2/WFF2. We thus first investigate factors of implementing a Fourier transform. Three factors, Fourier transform libraries, compilers and sizes of the input fringe pattern, are studied. From our results, the combination of using the MKL, the ICC, and the preferred sizes offers the best performance.

Another improvement comes from parallelizing the WFR2/WFF2 algorithms. We use four parallel techniques, OpenMP, Cilk Plus, the TBB and the PPL, to parallelize the main body of the WFR2/WFF2 algorithms. From the comparison results, OpenMP is the fastest technique. For a 1024×1024 fringe pattern, the new implementation increases the speed of the WFR2/WFF2 by 11 and 17 times, respectively.

Chapter 4

Comparison and integration of windowed Fourier filtering and BM3D

Image denoising has been studied for decades. Fringe patterns are images as well, we thus would like to find how the general image denoising methods perform on fringe patterns. In this chapter, block-matching and 3D filtering (BM3D) method [75] is used, and compared with the windowed Fourier filtering (WFF2) method. From the results, the WFF2 provides better results in continuous areas than in discontinuous areas, such as the boundary of two fringe pattern pieces. This situation indicates that fringe pattern information are lost in discontinuous areas during the denoising process. Meanwhile, the BM3D performs quite well in the discontinuous areas. A hybrid denoising scheme by combining the WFF2 and the BM3D is proposed. From the results, with the new scheme, noises in the residuals could be removed and lost fringe pattern information could be restored [1].

[1]This chapter is reproduced from "A comparison study of denoising techniques in fringe pattern analysis", Proc. SPIE 9302 and "WFF-BM3D: a hybrid denoising scheme for fringe patterns ", Proc. SPIE 9524 with permission from the International Society for Optics and Photonics (SPIE).

4.1 Introduction

In practise, fringe patterns from front-end measurement techniques usually contain noises. Those noises must be reduced before subsequent studies. To denoise the noisy fringe patterns, a few methods are proposed. One common type of techniques is spatial domain methods, which includes the mean and median filtering methods [15], the second-order oriented partial-differential equations (OPDEs) method [60] and the coherence enhancing diffusion (CED) method [64]. Transform domain methods are another important type of denoising methods. Fourier transform [68], windowed Fourier transform (WFT) [42], and wavelet transform methods [69] are typical examples of transform domain methods. Windowed Fourier filtering (WFF2) method is a WFT based algorithm, and it can suppress noises effectively.

The WFF2 is a local filter, and it denoises fringe patterns in a four-dimensional (4D) transform domain. When the input fringe pattern is transformed to the transform domain, the WFT coefficients of the noises are very small, and the noises could be removed by discarding those small coefficients. Meanwhile, fringe patterns are also images. Using general image denoising algorithms to process fringe patterns is also applicable. Image denoising is a fundamental task in image processing area. It has been studied for a long time, and many denoising techniques are proposed. We are curious to see the difference between the existing fringe pattern denoising algorithms and the general image denoising methods. The block-matching 3D filtering (BM3D) is the state of art image denoising algorithm [75]. The BM3D method uses a non-local strategy for denoising. In the BM3D method, similar patches are first searched in the given noisy image, and stacked into a three-dimensional (3D) array. Then the 3D array is transformed to a 3D transform domain, and the filtering process is performed in the 3D domain. Due to

its good performance, we select it as the representative of general image denoising method to compare with the WFF2, which is the representative of the fringe pattern denoising method.

4.2 Comparison between WFF2 and BM3D

4.2.1 Experiments

In this section, we use two types of fringe patterns, intensity fringe patterns (IFP) and exponential phase filed (EPF), to test the performance of the WFF2 and the BM3D methods. For each type, three fringe patterns are used, including straight, circular and peaks. Those fringe patterns are all simulated by MATLAB, and the size of the fringe patterns is 512×512. For IFPs, the fringe amplitude is set to be $b(x, y) = 127.5$, while additive Gaussian noises are added to the fringe patterns with $\sigma = 20, 40, 60, 80, 100, 120$ and 140. Thus approximately, σ / b ranges from 0 to 1. For EPFs, their real and imaginary parts are individually simulated as two IFPs. Figure 4.1 illustrates IFPs and the wrapped phase map of their corresponding EPFs. The source codes for the WFF2 and BM3D are downloaded from Refs. [134] and [151]. In the tests, both the WFF2 and the BM3D use their default parameters. Using the default parameters for the WFF2 is a convenient yet effective way to use the WFF2, while manually tuned parameters can provide better results [19]. Using the default parameters of the BM3D is because the default parameters has nearly reach optimal results [152]. All the programs are tested on a HP Z620 workstation with a 3.2 GHz CPU, and each case will be run 5 times, and the averages will be used.

The WFF2 can process both IFPs and EPFs, while the BM3D can only process IFPs. When using the BM3D to denoise an EPF, the real and imaginary parts of the

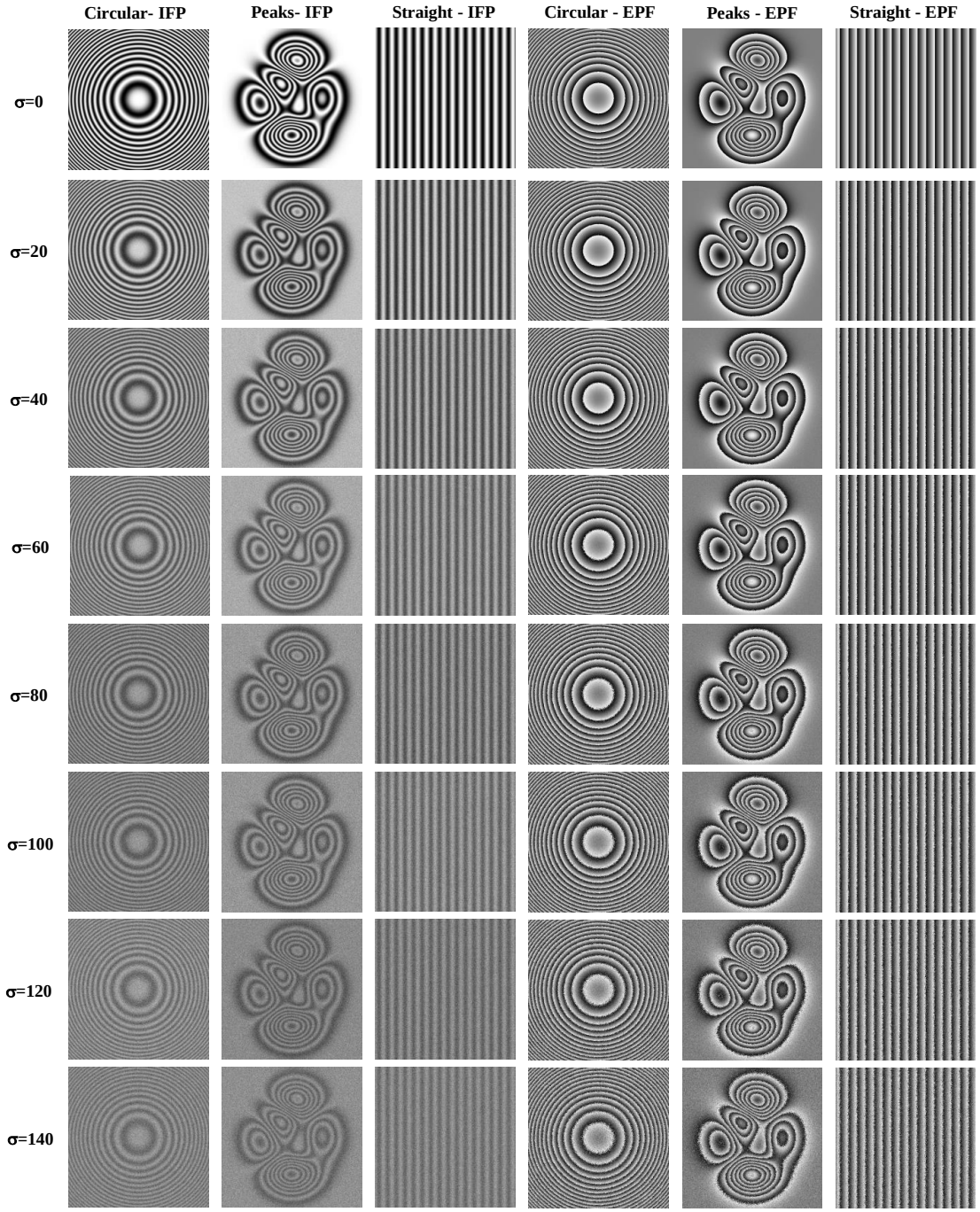


Figure 4.1: The IFPs and EPFs of the straight, circular and peaks examples. Reproduced from "A comparison study of denoising techniques in fringe pattern analysis", Proc. SPIE 9302, International Conference on Experimental Mechanics 2014 with permission from the International Society for Optics and Photonics (SPIE).

EPF are processed separately, and the final denoised EPF is obtained by combining the denoised real and imaginary parts.

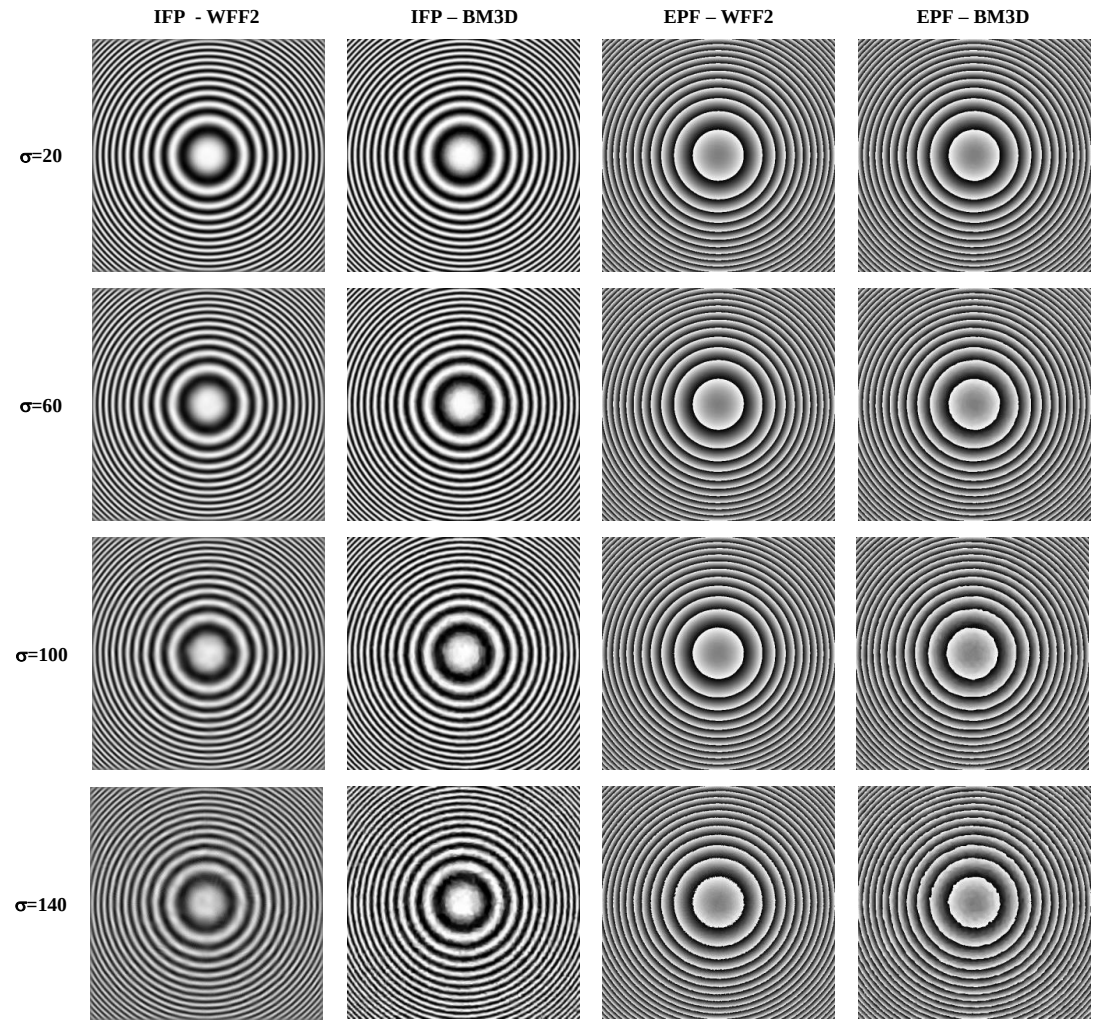


Figure 4.2: Selected denoised IFPs and EPFs of the Circular example. Reproduced from "A comparison study of denoising techniques in fringe pattern analysis", Proc. SPIE 9302, International Conference on Experimental Mechanics 2014 with permission from the International Society for Optics and Photonics (SPIE).

4.2.2 Results

The results of three fringe patterns are similar, and only the results of the circular fringe pattern are illustrated in Fig. 4.2. The mean absolute errors (MAEs) are computed to compare the results quantitatively. In Figs. 4.3 and 4.4, the MAEs between the filtered and ideal intensities of IFPs and the MAEs between the filtered and ideal phase values of EPFs are illustrated. From the results, the WFF2 and the BM3D are all capable to remove noises in fringe patterns, and the WFF2 performs better than the BM3D. In IFP examples, the MAEs of the BM3D under different noise levels are about 30% higher than the WFF2. In the EPF examples, the MAEs of the BM3D are more than 3 times higher than the WFF2. It is noteworthy that the BM3D may perform better if the noise level continues increasing, but the WFF2 can outperform it again by simply increasing the threshold. In our experiments, when using the BM3D to denoise an EPF, the real and imaginary parts of the EPF are filtered individually. In Ref. [153], the BM3D is directly applied to EPFs, and its performance on EPFs might be better. For the speed, the BM3D needs about 4 seconds for one IPF and 9 seconds for one EPF on average. The WFF2 needs about 23 seconds for both one IPF and one EPF.

4.2.3 Discussions

Apparently, the WFF2 and the BM3D represent two types of denoising strategies. The WFF2 selects the local processing strategy. It is very reasonable to use local processing strategy to process fringe patterns. Because the fringe patterns are provided by front-end measurement techniques, and pixels in a small patch are somehow correlated. Because of this characteristic, the phase of a fringe pattern in a small patch is usually modeled as a polynomial. In addition, the fact that fringe patterns are in the form of cosine or exponential functions (as shown in Eqs. 1.4

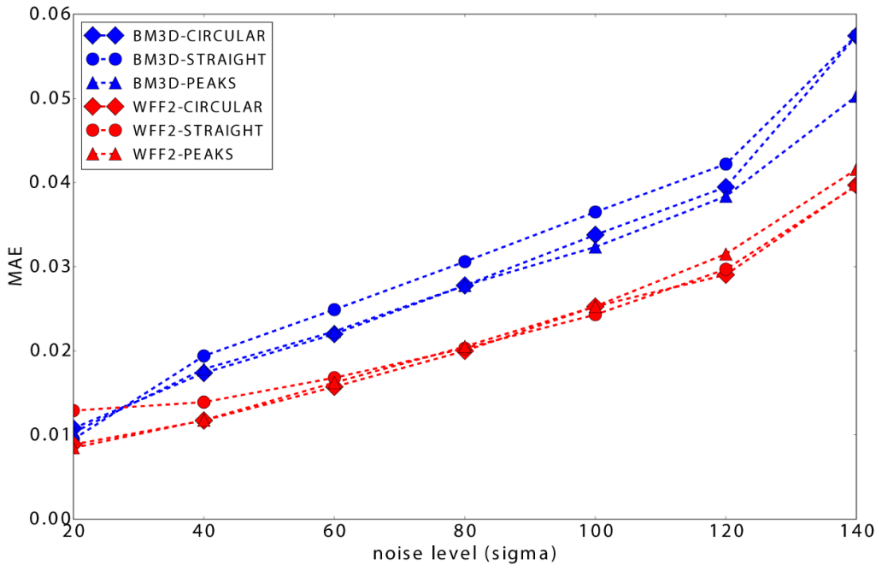


Figure 4.3: The comparison of the mean absolute errors of the WFF2 and the BM3D in the IFPs. Reproduced from "A comparison study of denoising techniques in fringe pattern analysis", Proc. SPIE 9302, International Conference on Experimental Mechanics 2014 with permission from the International Society for Optics and Photonics (SPIE).

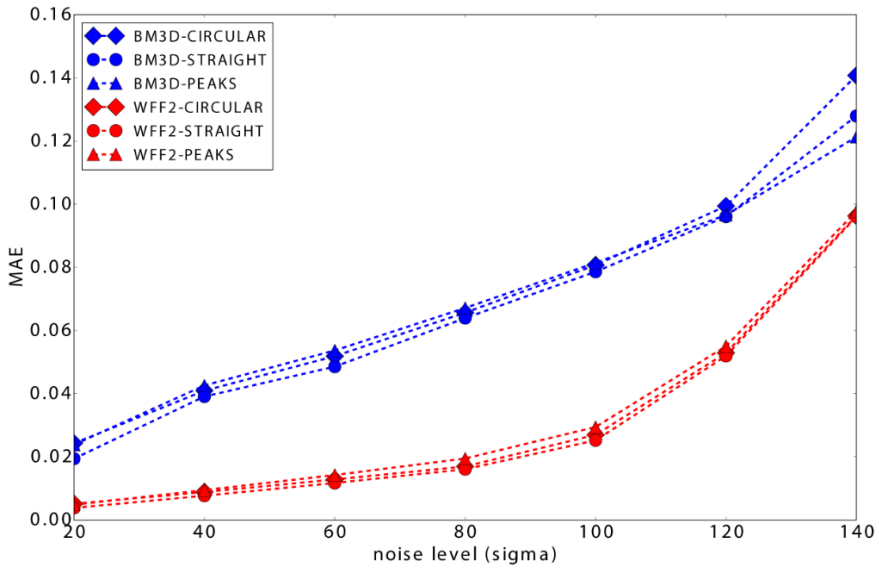


Figure 4.4: The comparison of the mean absolute errors of the WFF2 and the BM3D in the EPFs. Reproduced from "A comparison study of denoising techniques in fringe pattern analysis", Proc. SPIE 9302, International Conference on Experimental Mechanics 2014 with permission from the International Society for Optics and Photonics (SPIE).

and 1.6) is used as a prior knowledge in the fringe pattern analysis algorithms. Consequently, a local (windowed) Fourier transform can describe fringe patterns

with a sparse representation, and thus denoising is successful. The BM3D is a non-local method, and builds correlation through a block-matching process. It can then provide a sparse representation by grouping similar patches in fringe patterns. This strategy makes the BM3D capable to utilize more data for collaborative filtering than the WFF2. Since the BM3D has no assumptions on the input images, it can be applied to any type of images theoretically. However, this is also a minus for denoising fringe patterns due to the fringe pattern structure information has been ignored. Since both of them achieve high correlation and sparsity, the good performance they achieved is not unexpected.

4.3 Integration of WFF2 and BM3D

4.3.1 The discontinuity problem of the WFF2

The WFF2 algorithm uses a linear model to fit phase values in a small patch of a fringe pattern. For most cases, this assumption is tenable, but it is problematic in discontinuous areas, such as the boundary of two pieces of fringe patterns or image boundaries. Sudden phase changes in those discontinuous areas make the linear model used by the WFF2 less effective.

As introduced in Chap. 1, a fringe pattern with noise can be modeled as

$$f(x,y) = a(x,y) + b(x,y)cos[\varphi(x,y)] + n(x,y). \tag{4.1}$$

In this section, if we focus on the IFPs, Eq. 4.1 can be rewritten as follows:

$$f = a + \frac{b}{2}exp(j\varphi) + \frac{b}{2}exp(-j\varphi) + n, \tag{4.2}$$

and the filtered fringe pattern can be further rewritten as [19]

$$\bar{f} = \bar{a} + \frac{\bar{b}}{2} exp(j\varphi) + \frac{\bar{b}}{2} exp(-j\varphi) + \bar{n}, \tag{4.3}$$

where $\bar{b} = [1 - exp(\frac{-R^2}{4\pi})]b$, $R = \sqrt{8\pi ln[\frac{b}{threshold}(\frac{4\pi{\sigma_x}^2}{1+{\sigma_x}^4{c_{xx}}^2}\frac{4\pi{\sigma_y}^2}{1+{\sigma_y}^4{c_{yy}}^2})^{\frac{1}{4}}]}$, $c_{xx} = \frac{\partial^2\varphi}{\partial x^2}$, and $c_{yy} = \frac{\partial^2\varphi}{\partial y^2}$. The detailed derivation is available in Ref. [19]. According to Eq. 4.3, the phase is retained, and the amplitude of the filtered fringe pattern is reduced by $[1 - exp(\frac{-R^2}{4\pi})]$. This indicates that the filtered fringe pattern discards some useful information. Moreover, regions with high local curvatures, such as the discontinuous areas, loses more information than the regions with low curvatures. The discarded information exists in the residuals, and the WFF2 residual signal f_r can be defined as follows,

$$f_r = (f_0 - \bar{f}_0) + (n - \bar{n}). \tag{4.4}$$

To improve the quality of filtered fringe patterns, two potential solutions are applicable. One solution is to improve the WFF2 algorithm itself to reduce the lost information. If the value of $f - \bar{f}$ in Eq. 4.4 can be reduced, $f \approx \bar{f}$ and most information is retained. It is possible with modeling phase values by a higher order polynomial, but the WFF2 algorithm will become too complicated. The other solution is to recover the lost information from the residual signal f_r by removing $n - \bar{n}$. The recovered information is denote as $\bar{f}_r$. The final denoised fringe pattern is obtained by combining the original WFF2 result and the recovered information, and it is denoted as f_H.

$$f_H = \bar{f}_0 + \bar{f}_r. \tag{4.5}$$

The residual f_r contains fringe pattern information and noises. If the noise term can be removed, fringe pattern information can be extracted easily. This inspires us to use image denoising techniques to process f_r. The BM3D method is the state of the art, we thus select it to collaborate with the WFF2 method. The details of this hybrid method is presented in the next section.

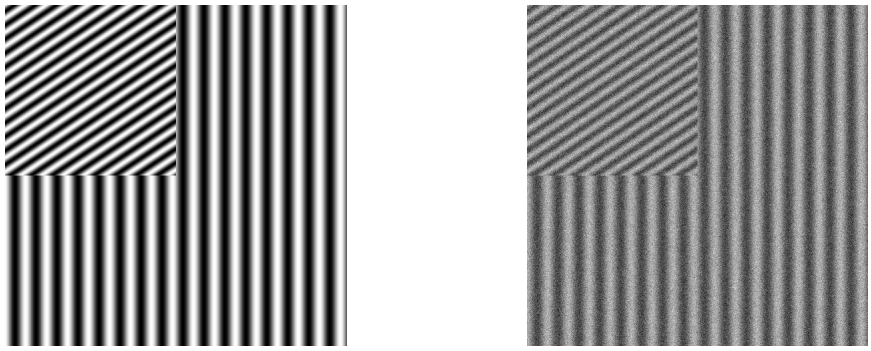
4.3.2 WFF-BM3D: a hybrid denoising scheme

The BM3D is expert in image denoising, and its performance on denoising fringe patterns has been examined already [74]. In most areas, the WFF2 can outperform the BM3D, however, once in discontinuous areas, the BM3D can achieve better results than the WFF2 with the assistance of its non-local strategy. If we can combine these two algorithms, we can obtain good results in both continuous and discontinuous areas. A hybrid scheme is proposed to achieve this goal. The WFF2 (a local method) is used first to remove noises in fringe patterns, and, subsequently, the BM3D (a non-local method) is applied to recover lost information from the WFF2 residuals. The final denoised result is obtained by adding the WFF2 result and the filtered residuals. This hybrid method is named WFF-BM3D. It is noteworthy that another straightforward method uses similar strategy. This method first defines continuous and discontinuous areas in fringe patterns, and then directly combines the continuous areas of the WFF2 result and the discontinuous areas of the BM3D result. We name this method as the Mosaic method, and compare it with the proposed WFF-BM3D method in the following section.

4.3.3 Experiments

To examine the effectiveness of the WFF-BM3D method, we simulate a clean image by MATLAB [as shown in Fig. 4.5(a)]. The size of the image is 512×512, and

the image consists of two pieces of fringe patterns with different frequencies and directions. Then, we add additive Gaussian noise to the clean image, and obtain the corresponding noisy image [as shown in Fig. 4.5(b)]. The standard deviation of the noise is 50. Due to their good performance and convenience, the default parameters of the WFF2 and BM3D are used [19, 152]. The configurations are the same as the experiments in Sec. 4.2.1.



(a) (b)

Figure 4.5: Test fringe pattern for WFF-BM3D. (a) Simulated clean fringe pattern; (b) corresponding noisy fringe pattern. Reproduced from "WFF-BM3D: a hybrid denoising scheme for fringe patterns ", Proc. SPIE 9524 with permission from the International Society for Optics and Photonics (SPIE).

The WFF2, the BM3D and the Mosaic method are first compared, and the results are shown in Figs. 4.6(a) - 4.6(c), respectively. The error maps of three methods are shown in Figs. 4.6(d) - 4.6(f). In these experiments, the discontinuous areas are defined as areas within 30 pixels of the boundary; the remaining areas are continuous. Based on this definition, the mean absolute errors (MAEs) for different methods in discontinuous and continuous areas are computed individually, and the results are shown in Table 4.1. Among the WFF2, the BM3D and the Mosaic methods, the Mosaic method is the best.

The WFF-BM3D is implemented as follows. The residual map of the WFF2 [shown in Fig. 4.7(a)] is first computed. It represents the information discarded by the WFF2. From Fig. 4.7(a), we can observe that, besides noises, some fringe

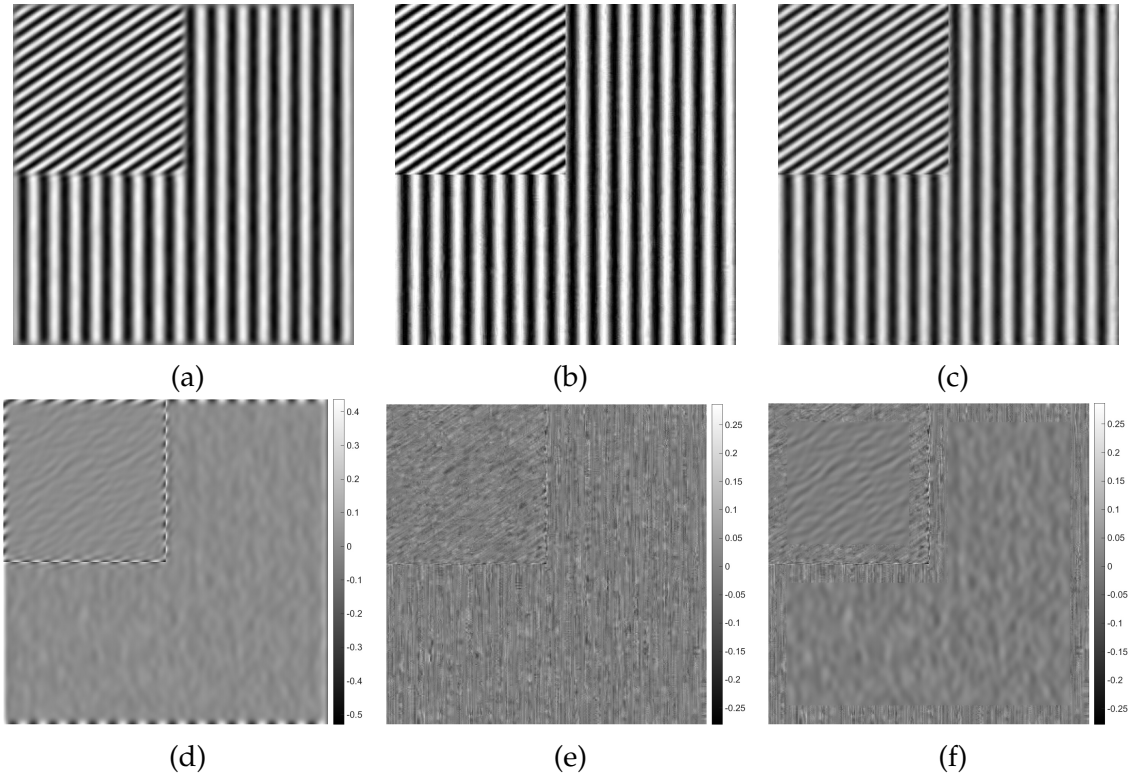


Figure 4.6: The results of (a) the WFF2, (b) the BM3D, and (c) the Mosaic; the error maps of (d) the WFF2, (e) the BM3D, and (f) the Mosaic. Reproduced from "WFF-BM3D: a hybrid denoising scheme for fringe patterns ", Proc. SPIE 9524 with permission from the International Society for Optics and Photonics (SPIE).

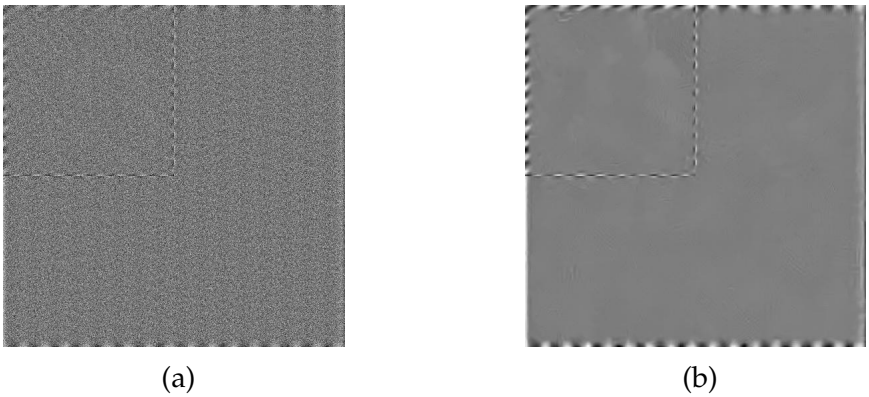
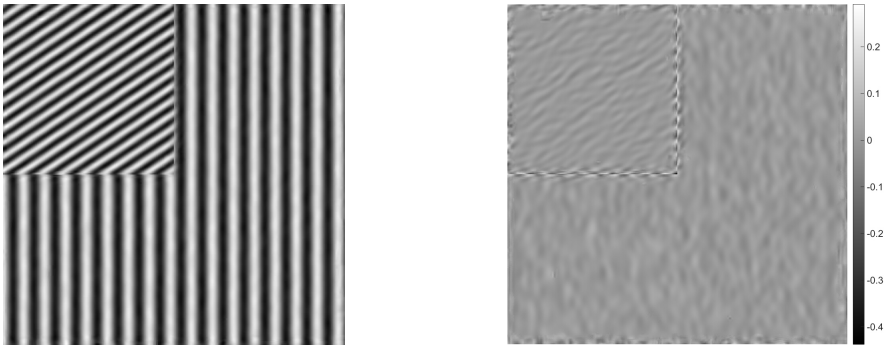


Figure 4.7: (a)The residual map of the WFF2; (b) the residual map after filtered by the BM3D. Reproduced from "WFF-BM3D: a hybrid denoising scheme for fringe patterns ", Proc. SPIE 9524 with permission from the International Society for Optics and Photonics (SPIE).



(a) (b)

Figure 4.8: (a) The result of the WFF-BM3D (selective update); (b) the error map of the WFF-BM3D (selective update). Reproduced from "WFF-BM3D: a hybrid denoising scheme for fringe patterns ", Proc. SPIE 9524 with permission from the International Society for Optics and Photonics (SPIE).

Table 4.1: The MAEs of different algorithms in both discontinuous and continuous areas. WFF-BM3D (FU): WFF-BM3D (full update); WFF-BM3D (DU): WFF-BM3D (discontinuous area update); WFF-BM3D (SU): WFF-BM3D (selective update). Reproduced from "WFF-BM3D: a hybrid denoising scheme for fringe patterns ", Proc. SPIE 9524 with permission from the International Society for Optics and Photonics (SPIE).

	WFF2	BM3D	Mosaic	WFF-BM3D (FU)	WFF-BM3D (DU)	WFF-BM3D (SU)
Continuous Areas	0.0131	0.0218	0.0131	0.0139	0.0131	0.0131
Discontinuous Areas	0.0413	0.0228	0.0228	0.0223	0.0223	0.0217

pattern information exist in the discontinuous areas, and these information should be recovered. Then, the BM3D method is applied to the residual map. From the filtered residual map [shown in Fig. 4.7(b)], the lost information is clearly presented. The next step is combining the result of the WFF2, and the recovered information. This step can be achieved in three ways: (1) full update: add the filtered residual map to the WFF2 result directly; (2) discontinuous area update: only update the pixels in the predefined discontinuous areas; (3) selective update: only update pixels with high local standard deviation. As the results of these three update methods are visually similar, only the outputs of the third method

are illustrated in Figs. 4.8(a) and 4.8(b). It is obvious that the boundary areas are improved after adding back the filtered residual map. Table 4.1 provides the quantitative differences among the three update methods. The WFF-BM3D with selective update has the lowest average MAE (the standard deviation of the MAEs is 0.00013 in the continuous areas and 0.00021 in discontinuous areas). This result is not unexpected, because the Mosaic method has already shown the advantages of combining the WFF2 and BM3D results. Meanwhile, the WFF2 has already provided a good fit in continuous areas. Updating those continuous areas is redundant. The third update method is the most adaptive, and thus has the best result.

4.3.4 Discussions

Based on analyses and results shown above, we can observe that:

- In the continuous areas, the WFF2 provides better results, while in the discontinuous areas, the BM3D is more reliable. The proposed hybrid scheme is originated from this fact.

- The WFF2 follows the local strategy, while the BM3D selects the non-local strategy. The non-local strategy helps the BM3D perform better in the discontinuous areas. The collaboration of local and non-local algorithms performs better than either single algorithm.

- Among different collaboration combinations, the WFF-BM3D (selective update) is the most adaptive and performs best.

- It can be seen from Eq. 4.3 that amplitude is reduced from b to $\bar{b}$. Further exploration is required to study the effect.

4.4 Summary

The WFF2 and the BM3D are two different denoising techniques. The WFF2 is specially designed for fringe pattern analysis with a local processing strategy, while the BM3D is a general image denoising method with a non-local strategy. In this chapter, we first compare their denoising performance on fringe patterns. The results indicate that both of them work satisfactorily, while the WFF2 outperforms the BM3D in the examples presented in this chapter.

The WFF2 is less accurate in discontinuous areas of fringe patterns, and the BM3D performs well in the discontinuous areas. In this chapter, a hybrid method combing the advantages of the WFF2 and BM3D is introduced. The results show that this hybrid method is more reliable in discontinuous areas.

Chapter 5

Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies

In Chap. 2, we introduced the QGPU method for phase unwrapping. This method adopts a quality map to guide the phase unwrapping progress. The quality map describes the goodness of pixels, and the good pixels are processed prior to the bad ones. Generally, the QGPU method contains two parts: the quality map and the guiding strategy. In this chapter, popular quality maps and guiding strategies are introduced and reviewed. A new data structure for implementing the adjoin list is also introduced[1].

[1]This chapter is a teamwork with Dr. Huang Lei, and the contents of this chapter is reproduced from "Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies," Appl. Opt. 50, 6214-6224 (2011) with permission from Optical Society of America.

5.1 Comparison of quality maps

5.1.1 Resources for quality calculation

Two common approaches are usually used to compute quality maps from front-end measurement data.

Case A: The quality map is directly computed from the raw measurement data. For instance, through a phase shifting process, a complex-valued fringe pattern f_c can be obtained as Ref. [42]

$$f_c = M \cdot exp(j \cdot \varphi_w); \tag{5.1}$$

where j is the imaginary unit, φ_w is the wrapped phase map, and M is the intensity modulation. From f_c, both φ_w and M can be delineated.

Case B: The quality map is computed from the wrapped phase or the normalized complex-valued fringe pattern f_n as:

$$f_n = 1 \cdot exp(j \cdot \varphi_w) = \frac{f_c}{|f_c|}, \tag{5.2}$$

5.1.2 Classification of quality maps

According to the sources, quality maps are classified as follows:

(i) Quality maps only from case A, including the correlation coefficient map in InSAR [48]; the modulation map Q_{MOD} [50], and the reliability map Q_{REL} [50,100] from least squares fitting in phase shifting method;

(ii) Quality maps only from case B, including the pseudo correlation coefficient map Q_{PCC} [48], the phase derivative variance map Q_{PDV} [48], the first phase

difference map Q_{FPD} [50], the second phase difference map Q_{SPD} [49] and their combinations [101, 154];

(iii) Quality maps from both case A and case B, which are based on transform methods, such as the Fourier transform method [28, 155, 156], the windowed Fourier filtering method [42,52,157,158], the windowed Fourier ridges method [42, 52, 157, 158], and the wavelet transform method [102, 159]. The quality maps include the amplitude map from Fourier transform method Q_{FT} [50], amplitude map from windowed Fourier filtering method Q_{WFF} [52], ridge map from windowed Fourier ridges method Q_{WFR} [52], and ridge map from wavelet transform method Q_{WT} [52].

The modulation map M from the first type can be expressed as Ref. [113]

$$Q_{MOD} = \frac{2}{N}\sqrt{[\sum_{n=0}^{N-1} I_n \cdot \sin(\frac{2n\pi}{N})]^2 + [\sum_{n=0}^{N-1} I_n \cdot \cos(\frac{2n\pi}{N})]^2}, \tag{5.3}$$

where N is the total phase stepping number with its running number denoted as n. I_n is the intensity of the nth phase-shifted fringe pattern. Combined with the least squares fitting error, the reliability map R is defined as [50, 113]

$$Q_{REL} = \frac{Q_{MOD}}{1 + c \cdot E/Q_{MOD}}, \tag{5.4}$$

where c is a coefficient to adjust the effect weight of least squares fitting error in R, and the definition of least squares fitting error map E is

$$E = \sqrt{\sum_{n=1}^{N} (I_n - f_n)^2/N}, \tag{5.5}$$

where f_n is the nth estimated fringe intensity from least squares fitting with phase shifting method.

The PCC map from the second type can be calculated through the following equation [52]

$$Q_{PCC} = \frac{\sqrt{(\sum_{x,y} \cos \varphi_w x, y)^2 + (\sum_{x,y} \sin \varphi_w x, y)^2}}{k^2}, \tag{5.6}$$

where the sums with index of (x, y) are only evaluated within a k by k neighboring square around the current pixel. The PDV map is defined by the equation [52]

$$Q_{PDV} = \frac{k^2}{\sqrt{\sum_{x,y}(\Delta^x_{x,y} - \frac{\sum_{x,y} \Delta^x_{x,y}}{k^2})^2 + \sum_{x,y}(\Delta^y_{x,y} - \frac{\sum_{x,y} \Delta^y_{x,y}}{k^2})^2}}, \tag{5.7}$$

where the terms $\Delta^x_{x,y}$ and $\Delta^y_{x,y}$ are the partial derivatives of the phase, which can be directly calculated from the wrapped phase map φ_w as these two equations

$$\Delta^x_{x,y} = \gamma(\Delta^w_{x+1,y} - \Delta^w_{x,y})\Delta^y_{x,y} = \gamma(\Delta^w_{x,y+1} - \Delta^w_{x,y}), \tag{5.8}$$

where the function $\gamma(\ )$ is an ordinary unwrapping operation to directly remove any 2π jumps in phase derivative calculation.

The definition of FPD map can be expressed with equation 5.9

$$Q_{FPD} = \frac{1}{|\Delta^x_{x-1,y}| + |\Delta^x_{x,y}| + |\Delta^y_{x,y-1}| + |\Delta^y_{x,y}|}, \tag{5.9}$$

The SPD map can be determined by 5.10

$$Q_{SPD} = \frac{1}{\sqrt{H^2 + V^2 + U^2 + W^2}} \tag{5.10}$$

where those variables are defined as

$$
\begin{aligned}
H_{x,y} &= \gamma(\varphi_{w_{x+1,y}} - \varphi_{w_{x,y}}) - \gamma(\varphi_{w_{x,y}} - \varphi_{w_{x-1,y}}),\\
V_{x,y} &= \gamma(\varphi_{w_{x,y+1}} - \varphi_{w_{x,y}}) - \gamma(\varphi_{w_{x,y}} - \varphi_{w_{x,y-1}}),\\
U_{x,y} &= \gamma(\varphi_{w_{x+1,y+1}} - \varphi_{w_{x,y}}) - \gamma(\varphi_{w_{x,y}} - \varphi_{w_{x-1,y-1}}),\\
W_{x,y} &= \gamma(\varphi_{w_{x+1,y-1}} - \varphi_{w_{x,y}}) - \gamma(\varphi_{w_{x,y}} - \varphi_{w_{x-1,y+1}}),
\end{aligned}
\tag{5.11}
$$

Transform methods compute quality maps in transform domains. It should be noted that the transform methods are used for phase unwrapping only. Although they are capable to demodulate fringe patterns, the demodulation process is completed by phase shifting method. Transform methods are applicable to either complex-valued fringe pattern f_c in case A, or phase data only, i.e. normalized complex-valued fringe pattern f_n in case B. In addition, besides the quality map, transform methods can provide filtered wrapped phase map for the unwrapping process. The WFR2/WFF2 are selected as examples of transform methods, and introduced in detail.

The WFR2 and WFF2 are all based on Window Fourier Transform (WFT). The WFT and inverse WFT are a pair of transforms, and they are represented below:

$$
Sf(u,v,\xi,\eta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y) g^{*}_{u,v,\xi_x,\xi_y}(x,y)dxdy, \tag{5.12}
$$

$$
f(x,y) = \frac{1}{4\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} Sf(u,v,\xi_x,\xi_y) \times g^{*}_{u,v,\xi_x,\xi_y}(x,y)\mathrm{d}u\mathrm{d}v\mathrm{d}\xi_x\mathrm{d}\xi_y, \tag{5.13}
$$

where $f(x,y)$ is a 2D image, $Sf(u,v,\xi,\eta)$ is the corresponding spectrum, * represents the complex conjugate operation, and $g_{u,v,\xi_x,\xi_y}(x,y)$ is the WFT basis which

is represented as

$$g_{u,v,\xi_x,\xi_y}(x,y) = g(x-u, y-v)\exp(j\xi_x x + j\xi_y y), \tag{5.14}$$

The WFT has a similar scheme as Fourier Transform, so it can be used to denoise a fringe pattern by filtering the corresponding WFT spectrum. This process is called the WFF2. Given a WFT spectrum, a predefined threshold is applied. Then noise is reduced by discarding small amplitude coefficients. After that, a filtered fringe pattern is obtained by applying IWFT on the filtered spectrum. Equation 5.15 describes the whole WFF2 process:

$$\overline{f}(x,y) = \frac{1}{4\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{\xi_{yl}}^{\xi_{yh}}\int_{\xi_{xl}}^{\xi_{xh}} \overline{Sf}(u,v,\xi_x,\xi_y) \times g^*_{u,v,\xi_x,\xi_y}(x,y)\mathrm{d}u\mathrm{d}v\mathrm{d}\xi_x\mathrm{d}\xi_y, \tag{5.15}$$

with

$$\overline{\mathrm{Sf}}(\mathrm{u},\mathrm{v},\text{„},\mathrm{J}) = \begin{cases} Sf(u,v,\xi,\eta), & |f(u,v,\xi_x,\eta)| \geq threshold \\ 0, & |f(u,v,\xi,\eta)| < threshold \end{cases} \tag{5.16}$$

The WFR2 is another application of the WFT, and it attempts to describe the similarity between the fringe pattern and its WFT elements. The element with the highest similarity is called ridge, as mentioned in Ref. [160]. The computed ridge can be used as the quality map for phase unwrapping. The whole WFR2 process is represented below:

$$[\omega_x(u,v), \omega_y(u,v)] = arg\underset{\xi_x,\xi_y}{m}ax|Sf(u,v,\xi_x,\xi_y)| \tag{5.17}$$

where $\omega_x(u,v)$ and $\omega_y(u,v)$ are local frequencies:

$$\omega_x(u,v) = \frac{\partial\varphi(x,y)}{\partial x}, \tag{5.18}$$

$$\omega_y(u,v) = \frac{\partial \varphi(x,y)}{\partial y}, \tag{5.19}$$

$\omega_x(u,v)$ and $\omega_y(u,v)$ obtain the values of ξ_x and ξ_y when $Sf(u,v,\xi_x,\xi_y)$ reaches the maximal. Then the phase values and ridges can be computed as:

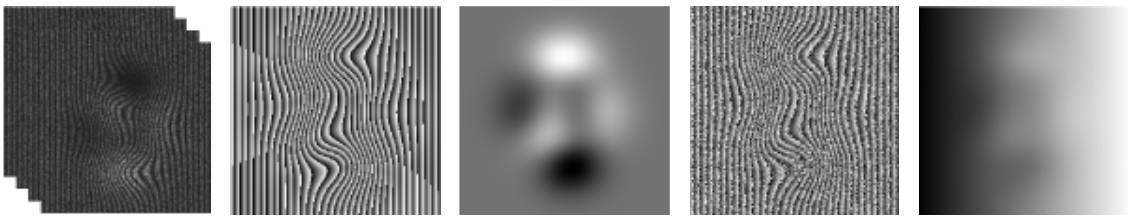
$$r(u,v) = |Sf[u,v,\omega_x(u,v),\omega_y(u,v)]|, \tag{5.20}$$

$$\varphi(u,v) = angle\{Sf[u,v,\omega_x(u,v),\omega_y(u,v)]\} + \omega_x(u,v)u + \omega_y(u,v)v, \tag{5.21}$$

In summary, quality maps from case A include Q_{MOD}, Q_{REL}, Q_{FT}, Q_{WFF}, Q_{WFR}, and Q_{WT}, while quality maps from case B include Q_{PCC}, Q_{PDV}, Q_{PFG}, Q_{SPD}, Q_{FT}, Q_{WFF}, Q_{WFR}, and Q_{WT}. These two cases will be compared separately, where noise and discontinuities are two major concerns.

5.1.3 Comparison of different quality maps for noisy phase maps

Generation of noisy phase maps



(a) (b) (c) (d) (e)

Figure 5.1: Simulated of noisy phase maps. (a) Four-step phase-shifted fringe patterns with speckle and random noise, (b) true wrapped phase, (c) true modulation, (d) retrieved noisy wrapped phase, (e) retrieved unwrapped phase (with phase noise). Reproduced from "Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies," Appl. Opt. 50, 6214-6224 (2011) with permission from Optical Society of America.

The test fringe pattern is simulated by the four-step phase-shifting technique with the size of 256×256. Speckle noise and normally distributed random noise are also added to the fringe pattern [shown in Fig. 5.1(a)]. The true wrapped phase map shown in Fig. 5.1(b) is simulated as

$$\varphi(x,y) = \frac{2\pi}{8}(x-128) + 3 \times peaks(x,y) \tag{5.22}$$

where $\varphi(x,y)$ denote the true phase map and the function, $peaks(\ )$ is a MATLAB® built-in function. The true modulation map M in Fig. 5.1(c) is simulated as

$$M(x,y) = \frac{255}{2} \times normalize[peaks(x,y)] \tag{5.23}$$

where the function $normalize(\ )$ linearly scales the input to the range of $[0,1]$. With the least squares phase-shifting algorithm [161], the wrapped phase is obtained [shown in Fig. 5.1(d)]. In Fig. 5.1(e), an unwrapped phase map is obtained by a congruence operation on the simulated true phase. In the low modulation areas, the signal-to-noise ratio is nearly zero, i.e. both the wrapped and unwrapped phase values in these regions are not reliable. However, this is specially designed to examine the quality maps.

Comparison of quality maps in case A

Six quality maps, Q_{MOD}, Q_{REL}, Q_{FT}, Q_{WFF}, Q_{WFR}, are calculated from raw fringe patterns and their performance are examined. The results of these quality maps are shown in Fig. 5.2. The results indicate that when noise presents, transform-based quality maps, Q_{FT}, Q_{WFF}, Q_{WFR}, and Q_{WT}, are more reliable than Q_{MOD} and Q_{REL}.

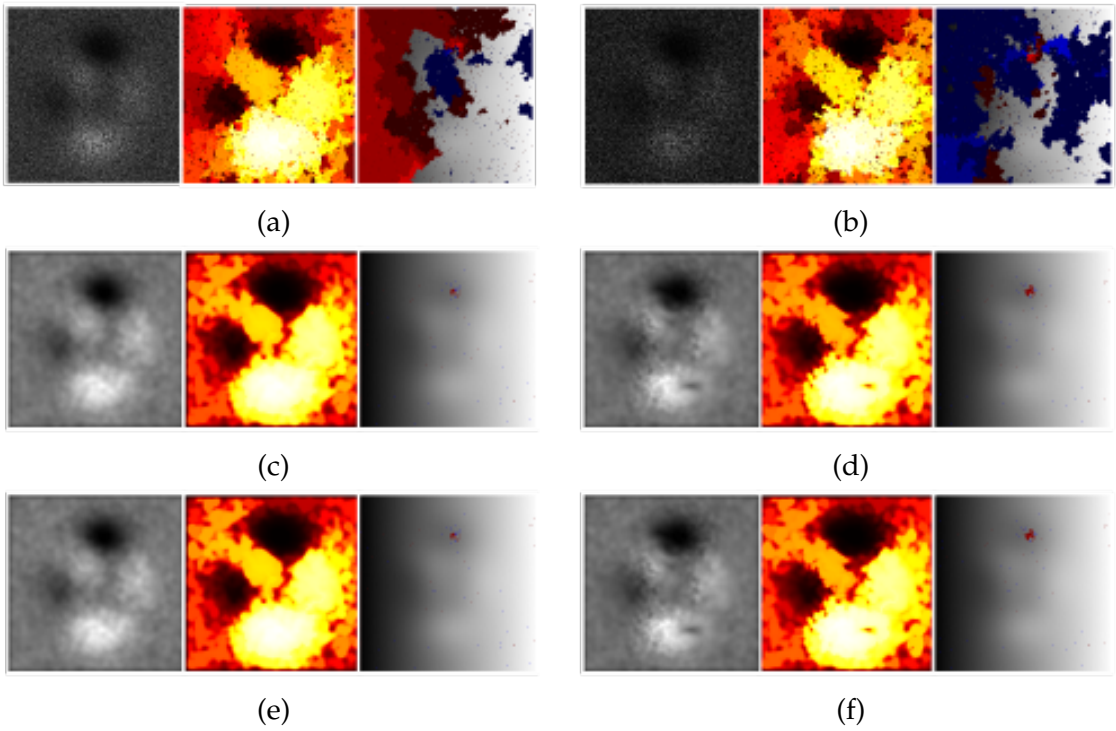


Figure 5.2: Comparison of phase unwrapping from noisy phase maps in case A. The quality maps used are (a) Q_{MOD}, (b) Q_{REL}, (c) Q_{FT}, (d) Q_{WFF}, (e) Q_{WFR} and (f) Q_{WT}, respectively. From left to right, each group shows quality maps, path maps, and unwrapping error maps. In unwrapping error maps, correctly unwrapped pixels are shown in gray scale and incorrectly unwrapped pixels are shown in red or blue. Red and blue indicate that the unwrapped phase value is larger and smaller than its true value, respectively. Darker shades indicate larger errors. Reproduced from "Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies," Appl. Opt. 50, 6214-6224 (2011) with permission from Optical Society of America.

Comparison of quality maps in case B

Eight quality maps, Q_{PCC}, Q_{PDV}, Q_{FPD}, Q_{SPD}, Q_{FT}, Q_{WFF}, Q_{WFR}, and Q_{WT}, are computed from the wrapped phase map and then their performance are compared. The results are shown in Fig. 5.3, and from the results, it is obvious to see that transform-based quality maps perform better than others.

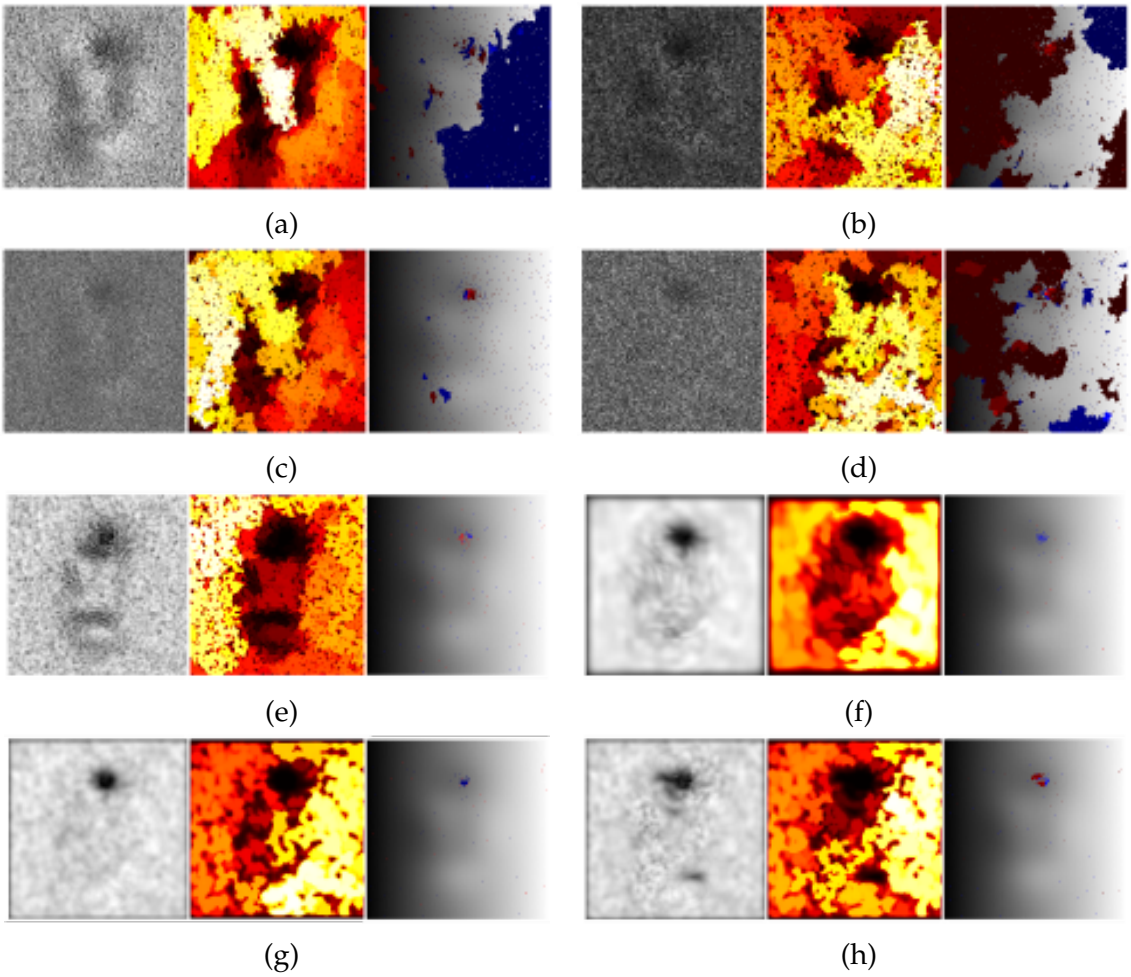
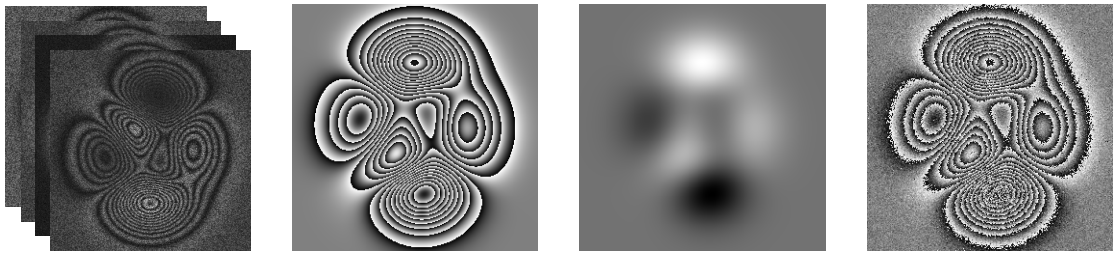


Figure 5.3: Comparison of phase unwrapping from noisy phase maps in case B. The quality maps used are (a) Q_{PCC}, (b) Q_{PDV}, (c) Q_{FPD}, (d) Q_{SPD}, (e) Q_{FT}, (f) Q_{WFF}, (g) Q_{WFR}, and (h) Q_{WT}, respectively. From left to right, each group shows quality maps, path maps, and unwrapping error maps. The meanings of colors in the unwrapping error maps are the same as in Fig. 5.2. Reproduced from "Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies," Appl. Opt. 50, 6214-6224 (2011) with permission from Optical Society of America.

Further comparison of transform-based quality maps

To further compare the performance among transform-based methods, we simulate an example which has a wider spectrum and no carrier fringes. The fringe patterns, true phase, true modulation, and retrieved wrapped phase are shown in Figs. 5.4(a) to 5.4(d), respectively.



(a) (b) (c) (d)

Figure 5.4: Phase unwrapping simulation with wide spectrum and no carrier fringes. (a) Phase-shifted fringe patterns, (b) wrapped phase without noise, (c) modulation distribution, and (d) retrieved wrapped phase with phase noise. Reproduced from "Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies," Appl. Opt. 50, 6214-6224 (2011) with permission from Optical Society of America.

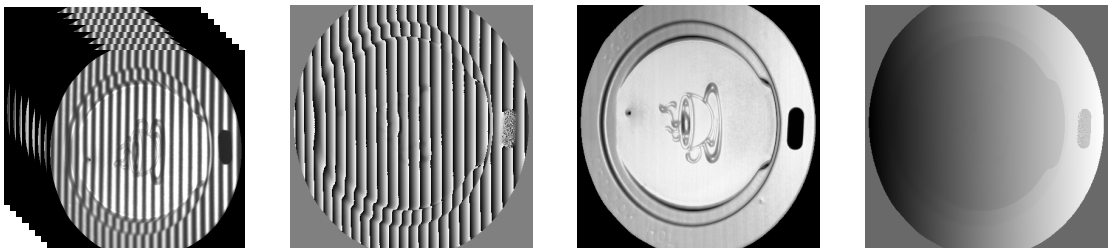
The unwrapping results of transform methods in case A are shown in Fig. 5.4. In Fig. 5.4(a), we can find that the Fourier transform method could not reduce noise in high frequency regions due to its global characteristic. Figures 5.4(b) and 5.4(c) illustrate the results of the WFR2/WFF2. From the results, noises are effectively reduced. The results indicate that local processing is suitable for denoising. Figure 5.4(d) shows the unwrapping result of the wavelet method. The wavelet method does not perform well in low frequency regions. From Fig. 5.4(d), errors are mainly located in the low frequency regions. From Case B, similar results are also observed.

To summarize, when noise presents, transform-based methods show better performance because they can provide both denoised wrapped phase maps and reasonable quality maps.

5.1.4 Comparison of different quality maps for discontinuous phase maps

Generation of discontinuous phase maps

Another problem of phase unwrapping process is discontinuity that the true phase distribution is not continuous. The test wrapped phase map is generated by fringe pattern projection (FPP) technique [17, 18] with eight frames of phase-shifted fringe patterns [shown in Fig. 5.5(a)]. Figures 5.5(b) and 5.5(c) are the retrieved wrapped phase map and modulation map. The unwrapped phase map obtained by temporal phase unwrapping technique is shown in Fig. 5.5(d) as the ground truth.



(a) (b) (c) (d)

Figure 5.5: Influence of fringe discontinuity on phase unwrapping. (a) eight frames of phase-shifted fringe pattern using FPP, (b) retrieved wrapped phase, (c) retrieved modulation map, (d) unwrapped phase by using temporal phase unwrapping technique. Reproduced from "Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies," Appl. Opt. 50, 6214-6224 (2011) with permission from Optical Society of America.

Comparison of quality maps in case A

In case A, six quality maps are compared. The results are shown in Fig. 5.6. It can be seen that four quality maps, Q_{MOD}, Q_{REL}, Q_{FT}, and Q_{WFF}, lead correct results in the discontinuity test. Q_{WFR} and Q_{WT} fail in this test, because the unwrapping path passes through the discontinuous area.

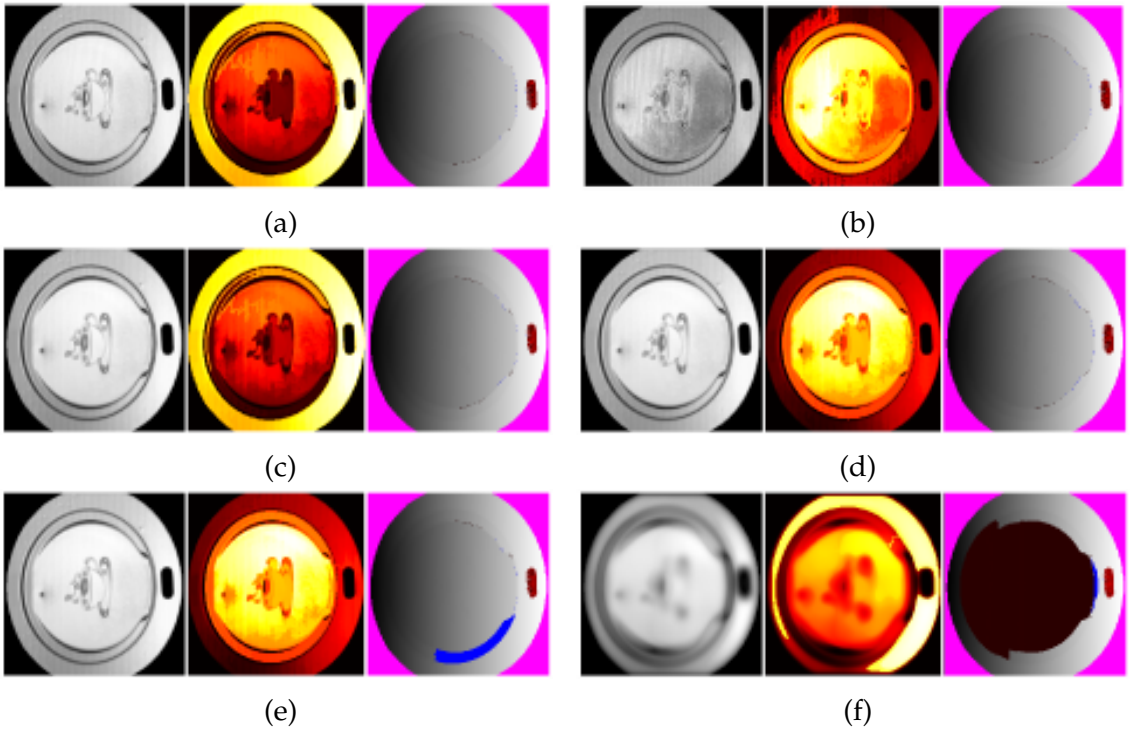


Figure 5.6: Comparison of phase unwrapping under discontinuity condition in case A. The quality map used are (a) Q_{MOD}, (b) Q_{REL}, (c) Q_{FT}, (d) Q_{WFF}, (e) Q_{WFR}, and (f) Q_{WT}, respectively. From left to right, each group shows quality maps, path maps, and unwrapping error maps. Reproduced from "Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies," Appl. Opt. 50, 6214-6224 (2011) with permission from Optical Society of America.

Comparison of quality maps in case B

In case B, eight quality maps are used. The results are shown in Fig. 5.7. From the results, six quality maps provide correct results, including Q_{PDV}, Q_{FPD}, Q_{SPD}, Q_{FT}, and Q_{WFF}, and Q_{WFR}. Q_{PCC} and Q_{WT} are failed in this test, because the unwrapping path passes through the discontinuous area.

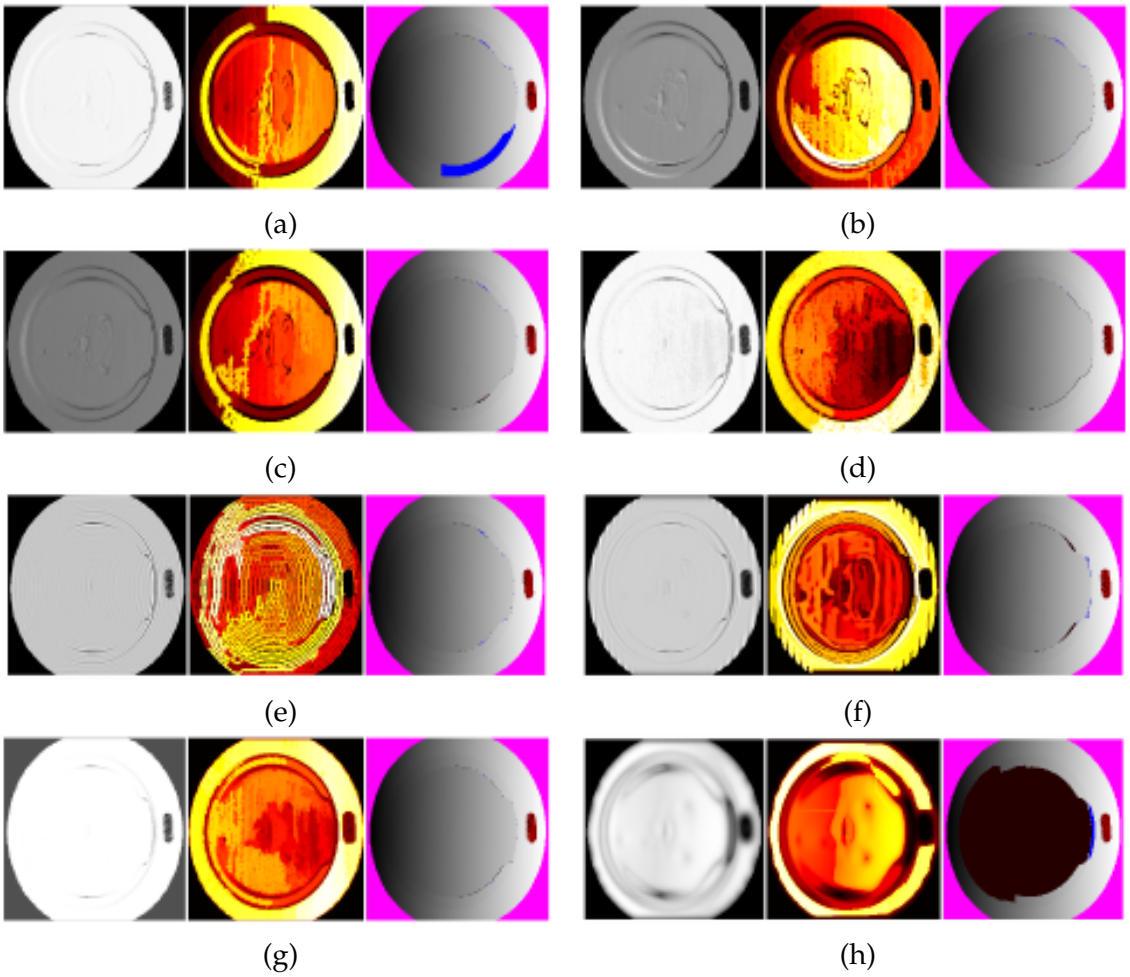


Figure 5.7: Comparison of phase unwrapping under discontinuity condition in case B. The quality map used are (a) Q_{PCC}, (b) Q_{PDV}, (c) Q_{FPD}, (d) Q_{SPD}, (e) Q_{FT}, (f) Q_{WFF}, (g) Q_{WFR}, and (h) Q_{WT}, respectively. From left to right, each group shows quality maps, path maps, and unwrapping error maps. Reproduced from "Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies," Appl. Opt. 50, 6214-6224 (2011) with permission from Optical Society of America.

Failure of all quality maps

Till now, no quality map could guarantee a successful unwrapping result. Figure 5.8 is an example that when all quality maps fail. QGPU under discontinuity condition is thus still a challenging problem and unresolved with QGPU alone. In Chap. 7, a snake assisted QGPU is presented to solve the discontinuity problem.

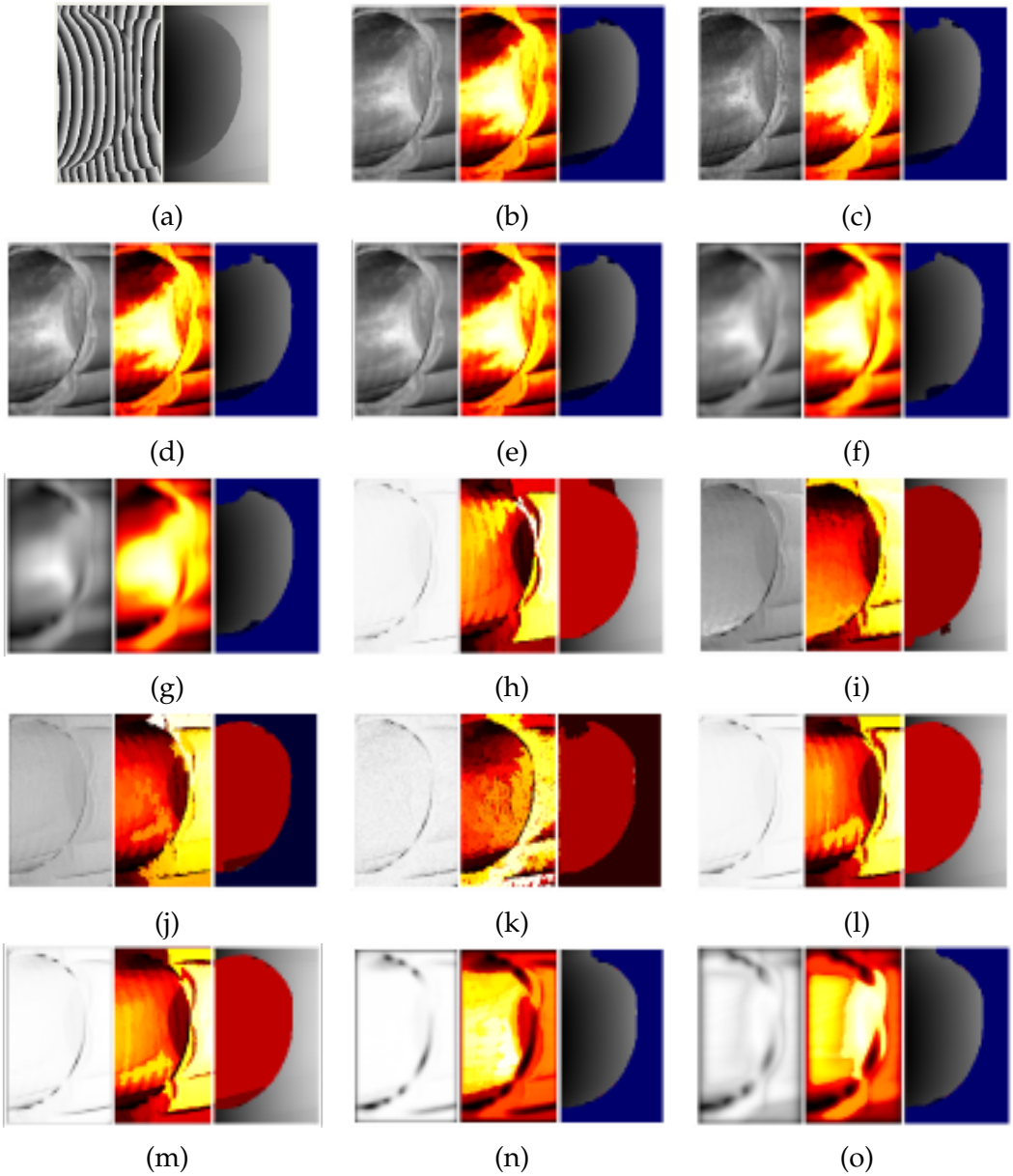


Figure 5.8: Failure of QGPU. (a) Wrapped phase and unwrapped phase with temporal phase unwrapping, and unwrapping result under case A (b)-(g) and case B (h)-(o). The quality map used are (b) Q_{MOD}, (c) Q_{REL}, (d)Q_{FT}, (e) Q_{WFF}, (f) Q_{WFR}, (g) Q_{WT}, (h) Q_{PCC}, (i) Q_{PDV}, (j) Q_{FPD}, (k) Q_{SPD}, (l) Q_{FT}, (m) Q_{WFF}, (n) Q_{WFR}, and (o) Q_{WT}, respectively. From left to right, each group shows quality maps, path maps, and unwrapping error maps. Reproduced from "Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies," Appl. Opt. 50, 6214-6224 (2011) with permission from Optical Society of America.

5.2 Comparison of guiding strategies

5.2.1 Introduction to guiding strategies

Classical quality guiding strategy

The simplest and classical guiding strategy was described in Sec. 2.3.1. Pixels to be unwrapped are stored in the adjoin list. In the adjoin list, those pixels can be either unorder or ordered based on their quality values. If the pixels have no order, maintaining the adjoin list requires three operations, inserting a new pixel, looking for the pixel with the highest quality value for unwrapping and removing the unwrapped pixel. If the pixels are ordered, we still need three operations to maintain the adjoin list, locating the position for inserting a new pixel, inserting the new pixel and removing the pixel with the highest quality value. Comparing these two approaches, with a ordered list, we only need to visit a part of the adjoin list to locate the inserting position, while we need to traverse the entire list to find the pixel with the highest quality value in the unordered list. Thus, the ordered list is selected as the default approach.

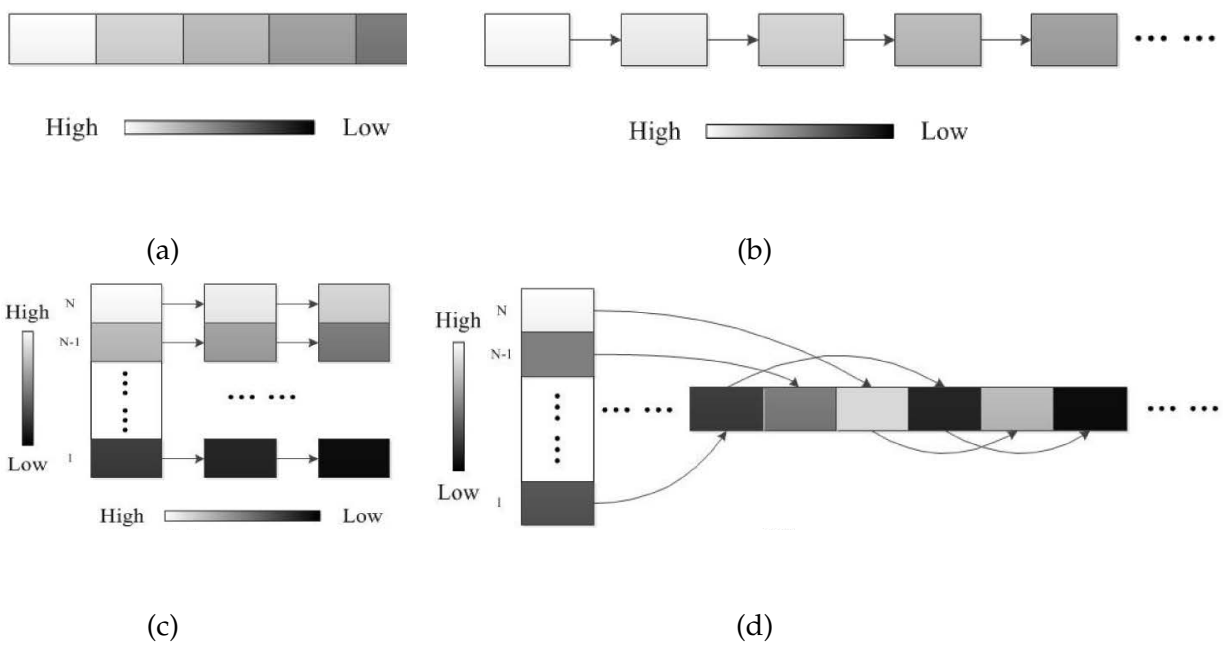


Figure 5.9: Data structures for sorting in quality guiding. (a) Array, (b) linked list, (c) indexed linked list, and (d) indexed interwoven linked list. Reproduced from "Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies," Appl. Opt. 50, 6214-6224 (2011) with permission from Optical Society of America.

Several data structures can be used to implement the adjoin list. Figure 5.9(a) shows the data structure of an array [162]. An array is usually stored in a continuous memory area. It is efficient at random access, but inefficient at inserting or removing pixels due to memory re-allocation. Figure 5.9(b) shows the data structure of a linked list (LL) [162]. A linked list consists of nodes, and a node of a linked list contains the value of the node and a pointer to the following node. Different from an array, an LL is efficient at inserting or removing pixels, which can be achieved by modifying pointers only. However, traversing a LL is usually inefficient, because a LL does not support random access and it is very long.

One possible solution to improve the efficiency of traversing a LL is to cut the LL into several short LLs based on the quality values. We name the short LLs as "mini-LLs". To connecting all mini-LLs, an array of pointers is used, and each pointer in the array points to a mini-LL. This data structure is called indexed

linked list (ILL) [shown in Fig. 5.9(c)]. In this data structure, quality values decrease from upper rows to lower rows. In each mini-LL, quality values decrease from left to right. The ILL is efficient at inserting or removing pixels, and the efficiency of traversal is also improved due to each mini-LL becomes much shorter than the original LL.

When inserting or removing pixels from a mini-LL, dynamical memory allocation and deallocation are required. In practise, to reduce the time consumption, memory should be pre-allocated. However, as the length of each mini-LL is undetermined before the computation, the amount of pre-allocated memory is very tricky to choose. Too little pre-allocated memory causes frequent allocation and deallocation operations, while too much pre-allocated memory is just a waste. Fortunately, mini-LLs can be interwoven in a pre-allocated array. The pre-allocated array is one-dimensional (1-D), and its index can be converted to a 2D coordinates which represents the location of the pixel in the wrapped phase map and quality map. The pre-allocated array and the header array in the ILL form a new data structure called Interwoven Indexed Linked List (I2L2) and I2L2 is shown in Fig. 5.9(d). Based on our experiences, the square root of pixel number of the input wrapped phase map is suitable as the the number of mini-LL in I2L2. To the best of our knowledge, the data structures ILL and I2L2 are proposed by the authors, with the hint from the data structure for stack-chain guiding strategy which is introduced later.

In addition, an algorithm called TRIM [48] is also discussed and implemented. The TRIM algorithm splits the original list into two sections when it is longer than a predefined threshold. The adjoin list is one section and the other one is named as the postponed list. The pixels in the postponed list can be seen as candidates of the adjoin list. When the adjoin list becomes empty, high quality pixels in the

postponed list are inserted into the adjoin list.

The results of the comparison among these data structures are summarized in Table 5.1. From the worst to the best, data structures are ranked as an array, an LL, an ILL and an I2L2, which is the same as our previous analysis. The TRIM algorithm is better than an array and an LL, but slower than the ILL and the I2L2.

Table 5.1: Speed comparison for different data structures in classical quality guiding (C++). Reproduced from "Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies," Appl. Opt. 50, 6214-6224 (2011) with permission from Optical Society of America.

	Data Structures	Array	LL	ILL	I2L2	TRIM
Time (s)	Box (960×1280) in Fig. 2.5	33.441	14.437	0.813	0.595	2.749
	Noisy Peaks (256×256) in Fig. 5.1	1.762	0.831	0.051	0.037	0.07
	Coffee Cup Cover (400×400) in Fig. 5.5	2.687	1.214	0.069	0.045	0.128

Two-section guiding strategy

In the classical quality guiding strategy, only the pixel with the highest quality value in the adjoin list can be unwrapped. However, pixels with high quality values can provide correct results, even they are not the highest. Based on this idea, a two-section guiding strategy is proposed. In the two-section strategy, pixels are divided into two section. Pixels with high quality values are inserted into the high-quality section directly without any comparison, while the remaining pixels are inserted into the low-quality section following the classical guiding strategy. The high-quality and low-quality sections could be implemented by an LL and I2L2, respectively. Since only pixels in the low-quality section are ordered, the two-section guiding strategy is expected to be faster than the classical strategy. However, setting the threshold for separating high and low quality values is very

trickly. The results of different thresholds are shown in Table 5.2. If we only accept zero-error rate, the improvement is about 0.1s. If we can tolerate higher error rate, such as 1%, the speed can be 2 times faster.

Table 5.2: Threshold level in two-section guiding. Sample: Box (960×1280) in Fig. 2.5. Language: C++. Reproduced from "Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies," Appl. Opt. 50, 6214-6224 (2011) with permission from Optical Society of America.

Threshold (Percentage of the highest quality)	Time (s)	Error Rate (%)
10%	0.202	0.0994
20%	0.281	0.1085
30%	0.34	0.4278
40%	0.399	0.0139
50%	0.445	0.0008
60%	0.487	0
70%	0.527	0
80%	0.566	0
90%	0.593	0
100%	0.595	0

Stack-chain guiding strategy

Since the data structure shown in Fig. 5.9(c) contains many mini-LLs and pixels in each mini-LL have similar quality values, it is possible to ignore the order of pixels in the same mini-LL for further speed-up. This idea was proposed in [99,163] and also formed through personal communication [164]. Following this strategy, each LL acts as a stack, which uses the "first in, last out" strategy for pixels. Similar to Fig. 5.9(d), all stacks could be interwoven, and this strategy is named stack-chain guiding strategy [164]. The number of mini-LL is set as the square root of the pixel number of the input wrapped phase map empirically.

5.2.2 Comparison and discussion

Three wrapped phase maps are used to examine the performance of above guiding strategies, including glass case in Fig. 2.5, noisy peaks in Fig. 5.1, and coffee cup lid in Fig. 5.5. Four methods, TRIM, I2L2, two-section, and stack-chain, are implemented by MATLAB and C++. MATLAB is a programming languange which is widely used by researchers for designing algorithms, while C++ is preferred by industry for its high performance. These two programming languages are selected due to their popularity. Three thresholds, 25%, 50% and 75% of the largest quality value are used in the two-section strategy to separate high and low quality values. Tables 5.3-5.5 summarize the results.

Table 5.3: Comparison results of the glass case example (960×1280) in Fig. 2.5. Reproduced from "Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies," Appl. Opt. 50, 6214-6224 (2011) with permission from Optical Society of America.

		Time (s)		Error Rate (%)
		MATLAB	C++	
CQ	TRIM	65.226	2.749	0
	I2L2	19.406	0.595	0
Two-section	75%	18.323	0.548	0
	50%	16.462	0.445	0.0008
	25%	11.914	0.305	0.1152
Stack-chain		1.007	0.162	0.003

From these results, it can be summarized that,

1. I2L2 sorts all pixels in it, and therefore provides the most reasonable and acceptable result. Its efficient structure reduces time cost effectively. The TRIM algorithms can provide the same unwraping results as I2L2, but it is much slower.

2. Among all guiding strategies introduced above, the stack-chain strategy gives the fastest speed. However, it is not error-free. The algorithm is applicable for

Table 5.4: Comparison results of noisy peaks (256×256) in Fig. 5.1. Reproduced from "Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies," Appl. Opt. 50, 6214-6224 (2011) with permission from Optical Society of America.

		Time (s)		Error Rate (%)
		MATLAB	C++	
CQ	TRIM	2.335	0.07	0
	I2L2	0.705	0.037	0
Two-section	75%	0.692	0.037	0
	50%	0.66	0.035	0
	25%	0.54	0.03	0.4486
Stack-chain		0.056	0.01	0.0656

Table 5.5: Comparison results of coffee cup lid (400×400) in Fig. 5.5. Reproduced from "Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies," Appl. Opt. 50, 6214-6224 (2011) with permission from Optical Society of America.

		Time (s)		Error Rate (%)
		MATLAB	C++	
CQ	TRIM	4.783	0.128	0
	I2L2	1.936	0.045	0
Two-section	75%	1.821	0.042	0
	50%	1.492	0.031	0.0006
	25%	1.431	0.029	22.5375
Stack-chain		0.102	0.025	0.2338

high speed applications.

3. Comparing with other two strategies, the two-section strategy is less attractive regarding the error rate and speed. However, it is suitable for processing a batch of wrapped phase maps with similar properties. In this case, those wrapped phase maps can share one threshold, and the requirements for error rate and speed can be reconciled.

5.3 Summary

Quality map of the QGPU method is first discussed in this chapter. Without a good quality map, successful QGPU is impossible. When noise presents, transform-based methods are more reliable than spatial methods, but they usually need more time. Meanwhile, selecting proper parameters for transform-based methods is necessary, but the manual operation makes the QGPU difficult to be automatic. The WFF2 and WFR2 algorithms can explicitly reduce noise and identify some discontinuity boundaries. However, when discontinuities present, QGPU alone is not always successful, and other assistances are required.

Besides the quality map, guiding strategy is another important part of the QGPU. In this chapter, three guiding strategies, classical, two-section and stack-chain, are introduced and compared. The classical strategy is the most reliable, and it can be accelerated by well-engineered data structures, such as the I2L2. The two-section performs faster than the classical, but its success highly depends on setting the threshold properly. The stack-chain strategy gives the fastest speed, but it is not error-free.

Chapter 6

Quality-guided phase unwrapping implementation: an improved I2L2

Quality-guided phase unwrapping (QGPU) is a common phase unwrapping technique. In the QGPU, the adjoin list determines the speed of the unwrapping process. Indexed interwoven linked list (I2L2) is a data structure for implementing the adjoin list. In this chapter, the I2L2 is improved from three aspects. With these three improvements, I2L2 is speeded up by over 6 times than the original implementation. For some phase maps, the new implementation is almost in real time [1].

6.1 Introduction

The quality guided phase unwrapping (QGPU) technique is an effective, efficient and automatic spatial phase unwrapping method [48,50]. Quality map is a major part of the QGPU. In a quality map, the goodness of each pixels are measured

[1] The contents of this chapter is reproduced from "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," Appl. Opt. 53, 3492-3500 with permission from Optical Society of America.

and recorded, and pixels with high quality values are unwrapped before the ones with low quality values. Usually, noisy pixels or pixels in the discontinuous areas usually have bad quality, and they are assigned low quality values. Several quality maps have been proposed to achieve more accurate measurement on the goodness of pixels [53].

During the unwrapping process, pixels to be unwrapped are stored in a list called adjoin list. Each time, the pixel with the highest quality value in the adjoin list is selected for unwrapping. Once a pixel is unwrapped, its unprocessed neighbors will be inserted into the adjoin list. The QGPU process is described as follows.

Step 1: Choose the pixel with the highest quality value in the quality map as the starting pixel, and add it into the adjoin list;

Step 2: Unwrap the pixel with the highest quality value in the adjoin list. Then, remove this pixel from the adjoin list, and insert its unprocessed neighbors into the adjoin list;

Step 3: Repeat Step 2 till the adjoin list is empty.

In general, an adjoin list has three basic operations, inserting a new pixel, removing the unwrapped pixels, and searching the pixel with the highest quality value. If a data structure can support these three operations, it can be used to implement the adjoin list. In Chap. 5, a data structure named indexed interwoven linked list (I2L2) is introduced, which satisfies these requirements very well. From the result, it has achieved high speed in the unwrapping process.

Among the three basic operations, inserting a new pixel is the most time-consuming operation, because the pixels are ordered in the adjoin list. With a data structure shown in Fig. 6.1(a), pixels in the adjoin list are distributed in different short linked lists. It explicitly improves the speed of inserting a new pixel.

However, due to a large amount of inserting/deleting operations, frequency allocating/deallocating memory is inevitable. The indexed interwoven linked list (I2L2) is proposed for solving this problem. In the I2L2, a one-dimensional (1D) array is pre-allocated, and the indices of the 1D array have one-to-one relationship with the indices of the two-dimensional (2D) wrapped phase map and quality map [shown in Fig. 6.1(b)]. From the comparison results, the I2L2 is faster than the indexed linked list by 35% [53]. As the index linked list and I2L2 use an equivalent strategy, the former is used for explanation in the following sections, while the latter is used for implementation.

In this chapter, three improvements on I2L2 are introduced. While searching the highest non-empty level, the original I2L2 has redundant operations. The first improvement maintains a record of the highest non-empty level and therefore reduces the searching time. The second improvement focuses on quality value distribution. When quality values are highly concentrated, the I2L2 becomes less efficient. An adaptive mapping method is proposed to adjust the distribution of quality values, and keep the I2L2 in an efficient state. Finally, heap is introduced to replace the original short linked list, and a new data structure I2L2-H is proposed. Heap is a popular data structure, and it is very efficient at inserting element in order. The I2L2-H combines the advantage of the I2L2 and heap, and outperforms other structures in the experiments. It is considered as a good implementation for the adjoin list.

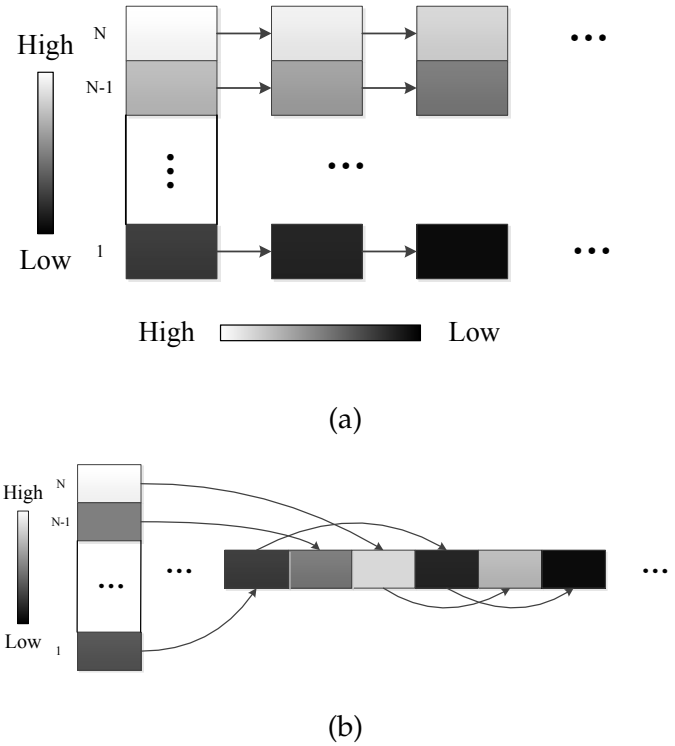


Figure 6.1: (a) Indexed linked list; (b) indexed interwoven linked list. Reproduced from "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," Appl. Opt. 53, 3492-3500 with permission from Optical Society of America.

6.2 Improved I2L2 with resumed searching

6.2.1 Recording the highest non-empty level

In the QGPU, only the pixel with the highest quality value can be used for unwrapping. It is the head of the highest non-empty level in the I2L2. When an level of I2L2 is empty, searching the highest non-empty level is necessary, and this operation will be performed repeatedly. In the original implementation of the I2L2, the searching operation always starts from the beginning of the index array till a non-empty level is found [illustrated in Fig. 6.2(a)]. Frequent searching operations generates redundant time costs. To reduce the time cost, a record of the highest non-empty level is maintained. The I2L2 begins the searching operation from the

recorded index rather than the beginning of the index array [shown in Fig. 6.2(b)]. For each new pixel inserted into the adjoin list, its level number is checked. If the level number of the new pixel is higher than the recorded number, the record will be updated. To distinguish these two approaches, the original one is named refreshed searching and the new one is named resumed searching. The resumed searching has fewer comparisons, and is expected to be faster.

In the original refreshed searching, the level number L is selected by the user. The recommended value of L is $\sqrt{MN)}$, where M and N are the row and column numbers of the input wrapped phase map [53]. Because the resume searching needs fewer comparisons, its level number is increased to achieve higher overall searching performance. In our experiments, the level number is set as $L = k \cdot \sqrt{MN}$, where k is a positive integer (typically 16).

6.2.2 Comparison between the refreshed searching and the resumed searching

Three wrapped phase maps with different sizes are chosen to evaluate the performance of the resumed searching (Fig. 6.3), including peaks (512×512), coffee cup lid (960×1280), and toy duck (1000×1000). For the wrapped phase maps, the peaks is simulated by MATLAB built-in *peaks* function, and normally distributed random noise is added. The other two wrapped phase maps are obtained from fringe pattern projection technique. For the quality maps, the coffee cup lid uses its modulation map as its quality map, while the quality maps of the other two wrapped phase maps are generated by the windowed Fourier ridges algorithm [52]. All the quality maps are illustrated in Fig. 6.4. It should be noted that all quality maps are generated in advance, and the time costs for generating quality maps are excluded from the results.

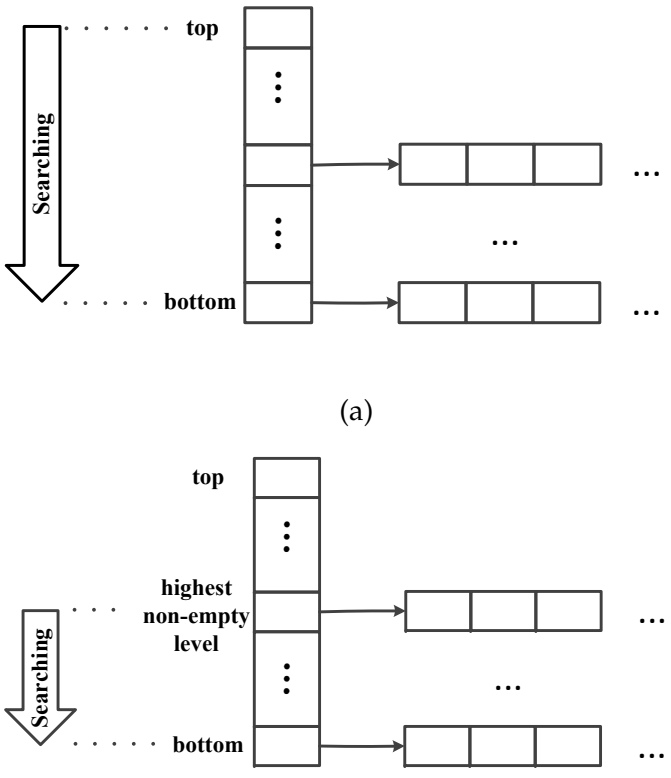
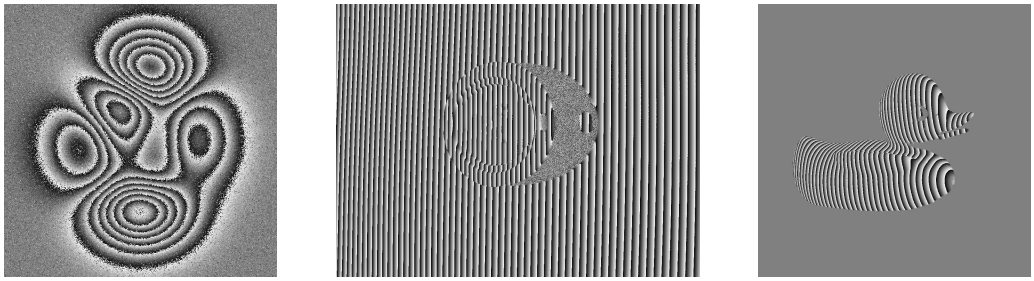


Figure 6.2: Searching the highest non-empty level. (a) Refreshed searching, (b) resumed searching. Reproduced from "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," Appl. Opt. 53, 3492-3500 with permission from Optical Society of America.

Both the refreshed searching and the resumed searching are implemented by C++. The test programs run on a DELL Precision T3600 workstation. The workstation equips a 3.20 GHz CPU and 16 GBytes memory. All tests are repeated 10 times to reduced the effect of background programs, and the average is used. Due to two searching strategies use the same guiding strategy, their unwrapping results are the same (shown in Fig. 6.5). Therefore, the unwrapping results are not covered, only the time costs are discussed. Table 6.1 summarizes the comparison between the refreshed searching and the resumed searching.

From the results, it is obvious that the resumed search costs less time than the



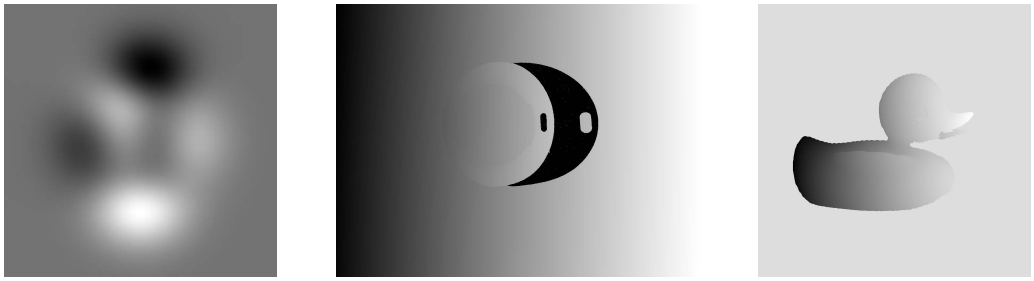
(a) (b) (c)

Figure 6.3: Wrapped phase maps. (a) Peaks (512×512), (b) coffee cup lid (960×1280), (c) toy duck (1000×1000). Reproduced from "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," Appl. Opt. 53, 3492-3500 with permission from Optical Society of America.

(a) (b) (c)

Figure 6.4: Quality maps. (a) Peaks, (b) coffee cup lid, (c) toy duck. Reproduced from "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," Appl. Opt. 53, 3492-3500 with permission from Optical Society of America.

original refreshed searching. In the coffee cup lid example, the resumed searching outperforms the refreshed searching by over 300%. In the other two examples, the differences are not so significant. The reason is the majority pixels in the peaks and the toy duck examples are in good quality, and they are concentrated in the levels close to the top level. In this case, redundancy caused by the refreshed searching is not high, and the improvement by the refreshed searching is as significant as the coffee cup lid example. Due to the good performance, the resumed searching is used by default.

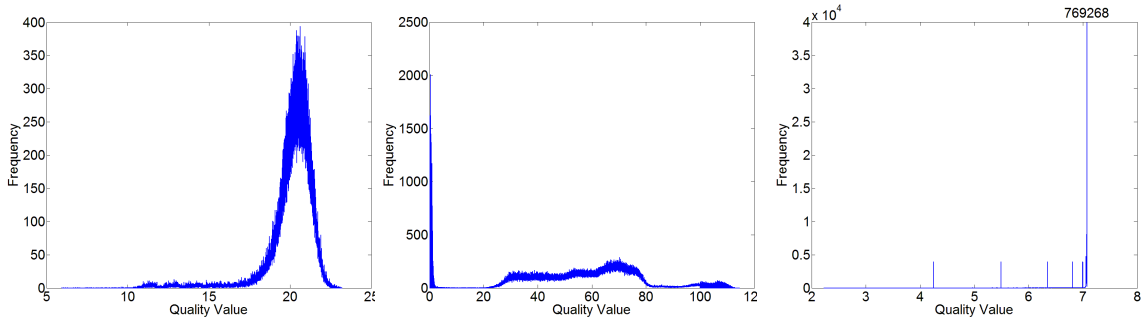


(a) (b) (c)

Figure 6.5: Unwrapped phase maps. (a) Peaks, (b) coffee cup lid, (c) toy duck. Reproduced from "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," Appl. Opt. 53, 3492-3500 with permission from Optical Society of America.

Table 6.1: Comparison between the refreshed searching and resumed searching. Reproduced from "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," Appl. Opt. 53, 3492-3500 with permission from Optical Society of America.

	Searching	Level number	Peaks	Coffee cup lid	Toy duck
Time(ms)	Refreshed	$(MN)^{1/2}$	36.28	436.27	63.29
	Resumed	$(MN)^{1/2}$	26.92	140.93	58.39
	Resumed	$16 \times (MN)^{1/2}$	23.04	138.19	51.37



(a) (b) (c)

Figure 6.6: Quality distribution curves. (a) Peaks, (b) coffee cup lid, (c) toy duck. Reproduced from "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," Appl. Opt. 53, 3492-3500 with permission from Optical Society of America.

6.3 Improved I2L2 with adaptive levels

6.3.1 Assigning levels adaptively to quality value distribution

The I2L2 is designed to cut a long linked list into a group of short linked lists. As linked lists is shortened, searching in the linked lists becomes more efficient. To implement this intention, the entire range of quality values is divided into L sub-ranges uniformly, where L represents the level number of the I2L2. Each short linked list represents a fix range of quality values, and the quality value of each pixel could be linearly mapped to a short linked list.

1	30	19	3	13	5	8	6	6	2
1	28	11	5	7	5	8	4	15	8
7	5	3	6	1	2	7	7	4	1
33	5	13	2	8	4	1	6	8	4
38	23	5	31	6	35	3	8	25	2
38	2	4	3	4	6	3	8	35	8
5	23	11	3	8	14	6	3	7	6
5	9	7	4	2	3	2	6	7	25
7	20	15	5	4	18	6	2	5	4
40	19	8	1	8	1	1	1	2	1

Figure 6.7: A quality Map Example (10×10). Reproduced from "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," Appl. Opt. 53, 3492-3500 with permission from Optical Society of America.

Quality distributions are usually various in different quality maps. In some cases, quality values may congregate into a small range. Consequently, a large number of pixels will locate in a small group of short linked lists, and these short linked lists will become very long and less efficient. The peaks [shown in Fig. 6.6(a)] and toy duck [shown in Fig. 6.6(c)] serve as such examples. In addition, this problem also exists in the coffee cup lid example [shown in Fig. 6.6(b)]. Actually, this problem is very common in wrapped phase maps.

One possible solution is to sort the whole quality map and distribute pixels into short linked list equally. This method can solve the problem, but it costs time on

sorting the quality map and assigning pixels to short linked lists. Another solution is to modify the structure of the I2L2, making it more adaptive. For example, if the number of pixels of one short linked list exceeds a pre-defined threshold, the short linked list will be divided into two parts, and the part with lower quality values will be moved to the head of the next short linked list [165]. This adaptive I2L2 structure outperforms the I2L2 in badly distributed cases. However, frequent moving operations costs too much time, and if most pixels are in low quality ranges, there may not be enough short linked lists to move points.

Rather than modifying the structure of the I2L2, we focus on the mapping between the quality map and the I2L2 levels. In this section, we use a nonlinear and adaptive mapping to replace the original linear mapping. The basic idea is similar to histogram equalization [166]. A histogram is first established. The bin number is much larger than the level number of the I2L2 (usually it is a multiple of the level number). Then we combine the bins in the histogram to new levels to establish a more dispersed quality value distribution. The steps of establishing the new adaptive mapping is listed below:

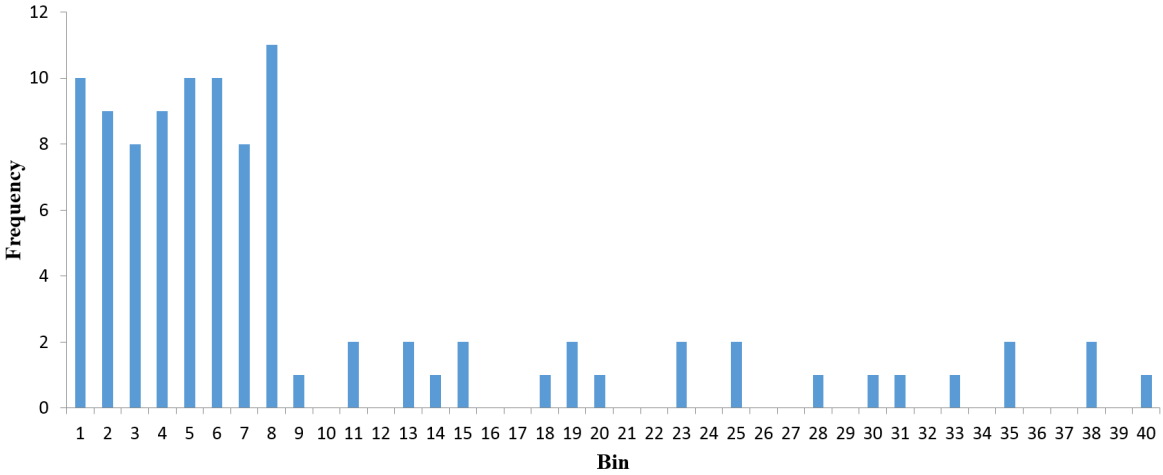
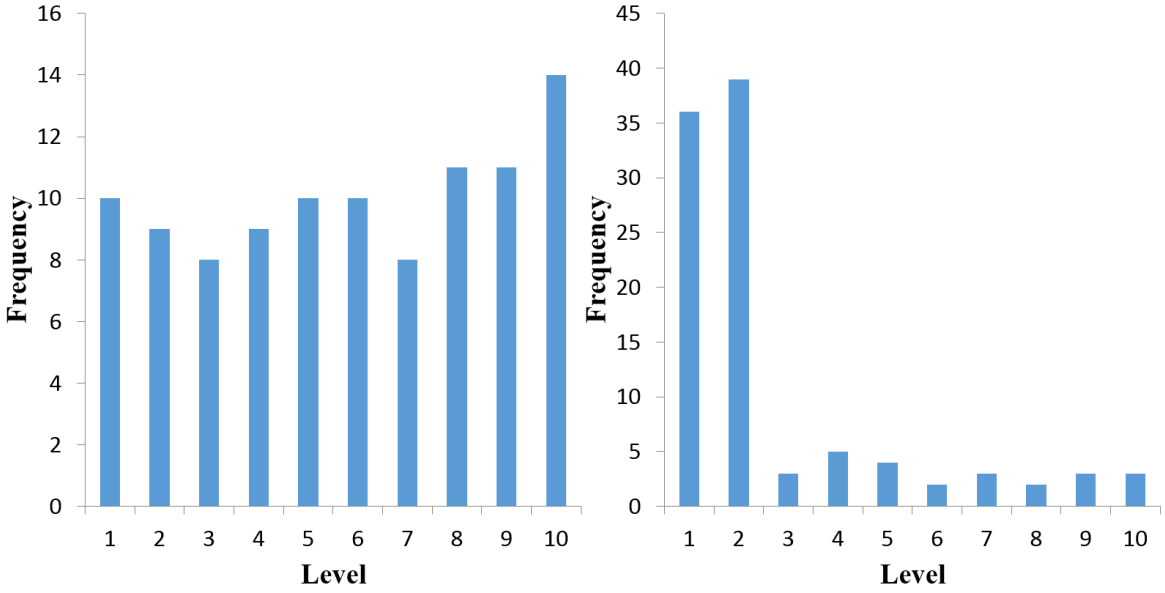


Figure 6.8: Histogram with 40 bins.

Step 1: Establish a histogram with $n \times L$ bins (n is an integer), and also assign



(a) (b)

Figure 6.9: (a) Histogram of quality distribution with adaptive mapping; (b) histogram of quality distribution with fixed mapping. Reproduced from "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," Appl. Opt. 53, 3492-3500 with permission from Optical Society of America.

the bin indices to all pixels in the quality map. Higher n gives a narrower bin width and a better discrimination of quality values, especially in dense areas. In our experiments, $n = 8$.

Step 2: Merging neighboring bins from left to right to create L new levels. Each bin should not have more than $M \times N/L$ pixels, except the last bin or the bins which already have more than $M \times N/L$ pixels before merging.

Step 3: Assign the new level indices to all pixels based on the relationship between bins and levels in Step 2.

An example is provided to explain the steps of creating the new mapping. Figure 6.7 illustrates a 10×10 quality map. The quality values in this quality map are within the range of $[1, 40]$. We set $L = 10$ and $n = 4$, i.e., the I2L2 uses 10 levels, and the histogram has 40 bins. Figure 6.8 shows the computed histogram.

Following Step 2, those 40 bins are merged in to 10 levels. The new quality distribution is shown in Fig. 6.9(a), while the original distribution is shown in Fig. 6.9(b). It is clear that the new adaptive mapping provides a better distribution than the original fixed mapping.

Table 6.2: Comparison of the timing cost for assigning level numbers to pixels between fixed mapping and adaptive mapping. Reproduced from "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," Appl. Opt. 53, 3492-3500 with permission from Optical Society of America.

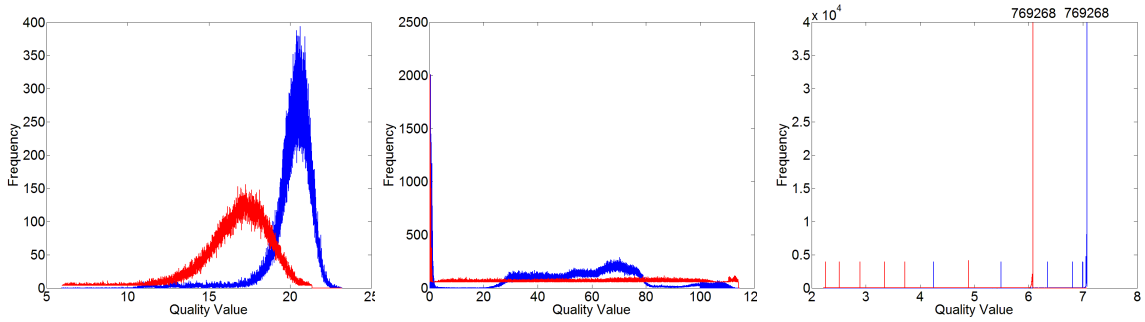
		Peaks	Coffee cup lid	Toy duck
Time(ms)	Fixed	2.79	13.44	10.96
	Adaptive	3.9	21.78	16.22

Table 6.3: Comparison of speed among refreshed searching and resumed searching with both fixed and adaptive mappings. Reproduced from "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," Appl. Opt. 53, 3492-3500 with permission from Optical Society of America.

	Searching	Mapping	Level number	Peaks	Coffee cup lid	Toy duck
Time(ms)	Refreshed	Fixed	$(MN)^{1/2}$	36.28	436.27	63.29
	Refreshed	Adaptive	$(MN)^{1/2}$	32.26	417.36	61.57
	Resumed	Fixed	$16 \times (MN)^{1/2}$	23.04	138.19	51.37
	Resumed	Adaptive	$16 \times (MN)^{1/2}$	16.89	85.34	42.71

6.3.2 Comparison between fixed and adaptive mapping

To have a better understanding of the new adaptive mapping, the three wrapped phase maps and configurations used in Sec. 6.2.2 are adopted again. The time costs for establishing the fixed and adaptive mappings are first compared. Table 6.2 summarizes the results, and it can be seen that the difference establishing the fixed mapping and the adaptive mapping is not significant, even the adaptive mapping is more complex. The comparisons of quality value distributions are shown in Fig. 6.10. After the adaptive mapping, quality values in the peaks and coffee cup lid examples are better distributed. Only a small number of bins cannot



(a) (b) (c)

Figure 6.10: Quality distribution curves comparison between the fixed and adaptive mapping. The blue curve represents the fixed mapping, and the red curve is the adaptive mapping. (a) Peaks, (b) coffee cup lid, (c) toy duck. Reproduced from "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," Appl. Opt. 53, 3492-3500 with permission from Optical Society of America.

be redistributed. Due to its extreme quality distribution, the toy duck example is not significantly redistributed, only a small number of pixels are optimized.

We next compare the unwrapping time costs of these two different mapping methods. Table 6.3 summarizes the results. Obviously, the adaptive mapping outperforms the fixed mapping as expected. In the coffee cup lid example (the best case), with the resumed searching and adaptive mapping, the time cost is reduced from 436.27ms to 85.34ms, which is over 5 times faster. Even in the toy duck example (the worst case), the new I2L2 implementation is still 48% faster. However, it should be noted that the adaptive mapping could not convert a extremely concentrated quality distribution to a well distributed distribution. To fully solve the distribution problem, further study is necessary.

6.4 I2L2-H: a new I2L2 variant

In the I2L2, short linked lists are ordered, therefore, a new pixel must be compared all pixels larger than it till its location is found. Heap is efficient at insertion, we

thus combine it with the I2L2 to achieve higher performance.

6.4.1 The proposed hybrid I2L2 with heaps

Structure and basic operations of a heap

Heap [162] is a tree-based data structure. Each node of a heap has one pointer to its parent (i.e., a node in its upper level), and pointers to its children (i.e. nodes in its lower level). All parent-child pairs in a heap must follow the same value order. For example, if it is a binary max heap (shown in Fig. 6.11), values of the parent nodes are always larger than or equal to the values of their corresponding children. The root gives the largest value in the heap.

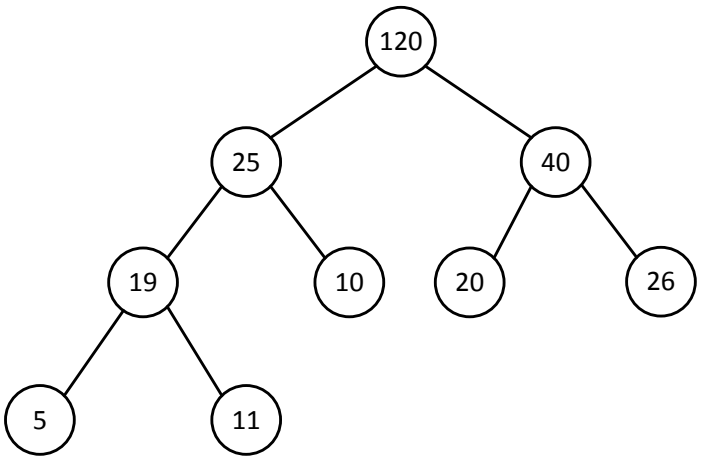


Figure 6.11: A binary max heap data structure. Reproduced from "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," Appl. Opt. 53, 3492-3500 with permission from Optical Society of America.

Inserting new points and removing the root point are two basic operations of a heap. The new point is first added to the tail of the heap, which is the leftmost available position in the bottom level of the heap. The new added node will be compared with its parent node. If the parent node is smaller than the new added node, these two nodes are swapped, and the new node will continue comparing with its new parent node. Figure 6.12 illustrates an example of inserting a new

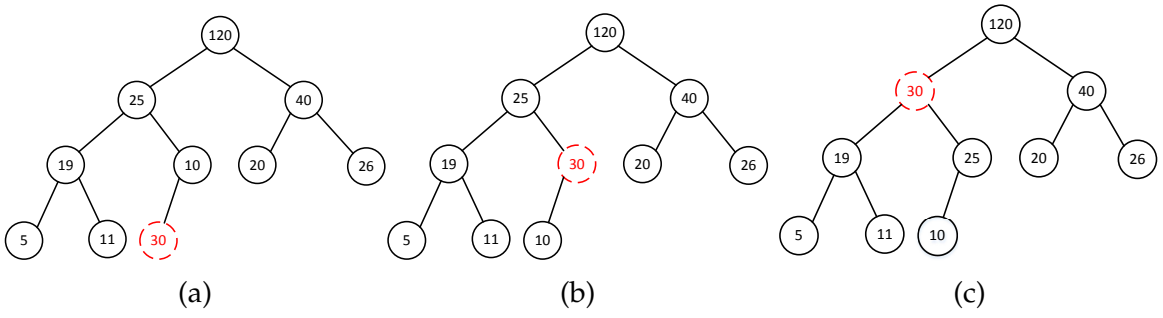


Figure 6.12: The operations of inserting a new element. (a) The new element is inserted to the end of the heap. (b)-(c) Subsequent steps of moving the new element to the proper position. Reproduced from "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," Appl. Opt. 53, 3492-3500 with permission from Optical Society of America.

point. A new point with a value of 30 is added to the end of the heap [represented by the red dotted circle in Fig. 6.12(a)]. Then it is compared with its parent node whose value is 10. As the new added node has a larger value, it is exchanged with its parent node [Fig. 6.12(b)]. The new added node continues the comparison with its new parent whose value is 25. As the new added node is still larger, these two nodes swapped again. At last, the new added node is compared with the root node whose value is 120. The root node is larger, and the new added node stops the comparison [Fig. 6.12(c)]. During the insertion process, the new added node only compares with partial nodes in the heap, and thus heap has faster insertion speed than a linked list.

For a max heap, the root node has the largest value in the heap. After removing the root node, the heap needs a "heapify" operation to restore the order in the heap. Usually, the last node in the heap is selected to fill the empty root position. Then the new root node is compared with its children. The operation will stop immediately if the root is larger than its children. Otherwise, the new root is swapped with its largest child. The comparison continues till the new root node reaches it proper position. Figure 6.13 presents how the heapify operation

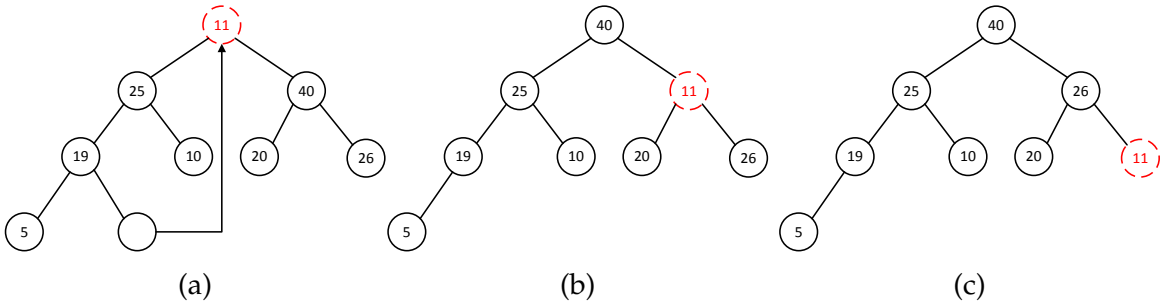


Figure 6.13: The operations of restoring heap order after deleting the root element. (a) Replace the root with the last element. (b)-(c) Subsequent steps of restoring the order of heap. Reproduced from "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," Appl. Opt. 53, 3492-3500 with permission from Optical Society of America.

performs. After the original root node is removed, the last node is selected to be the new root node [Fig. 6.13(a)], and compared with its new children. As its value is smaller, the new root is exchanged with its larger child node whose value is 40 [Fig. 6.13(b)]. The heapify operation terminates after comparing the new root node and the node with a value of 26 [Fig. 6.13(c)].

Structure of I2L2-H

Due to a heap is more efficient than a linked list at inserting an element, the shorted linked lists in the I2L2 are replaced by heaps. The new structure is illustrated in Fig. 6.14 and named as I2L2-H. Like the original I2L2, the nodes in I2L2-H represent the pixels in the wrapped phase map, and the values of the nodes are the quality values of the pixels. For each time, the root of the highest non-empty level is chosen for unwrapping.

When we try to insert a pixel to the I2L2-H, the proper level index of the pixel is first determined. Then the pixel will be added to the end of the corresponding heap, and relocated to its proper position. In the unwrapping step, the root node of the highest non-empty level is first retrieved and unwrapped. After the root node is removed, the heap performs the heapify operation to restore the order. The

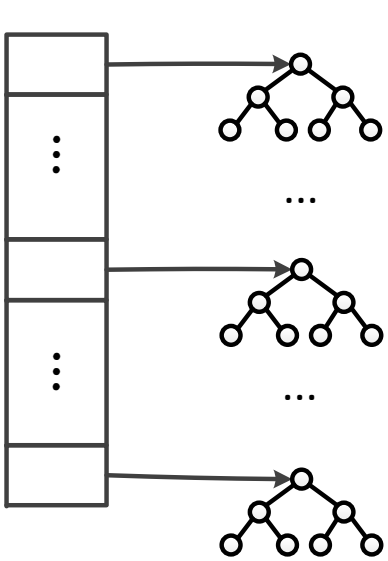


Figure 6.14: The structure of I2L2-H. Reproduced from "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," Appl. Opt. 53, 3492-3500 with permission from Optical Society of America.

time complexities of inserting a new pixel and removing the root pixel for I2L2-H are all $O(\log_2 n)$, while the complexities for I2L2 are $O(n)$ and $O(1)$, respectively. The I2L2-H has lower time consumptions.

A single heap also satisfies the requirements of the adjoin list mentioned in Sec. 6.1, so it can be used in the QGPU implementation [167, 168]. A QGPU implementation using a heap as the adjoin list is compared with the I2L2 and I2L2-H.

6.4.2 Comparison among I2L2-H, I2L2 and a single heap

The wrapped phase maps used in previous sections are adopted again for comparison. In our experiments, we select our own heap implementation. This implementation is specially optimized for QGPU operations and runs about 30% faster than the heap provided by C++ Standard Template Library. In the peaks and coffee cup lid examples, the I2L2-H outperforms other two data structures as ex-

pected. In the toy duck example, due to its quality distribution, the I2L2 achieves a faster speed than the I2L2-H. A single heap is the slowest in all cases.

Table 6.4: Comparison among I2L2, I2L2-H, and heap. Reproduced from "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," Appl. Opt. 53, 3492-3500 with permission from Optical Society of America.

		Peaks	Coffee cup lid	Toy Duck
Time(ms)	I2L2	16.89	85.34	42.71
	I2L2-H	12.84	66.19	54.27
	Heap	20.75	115.2	58.61

To summarize, these three data structures have their own strengths and weaknesses. From the concept viewpoint, the I2L2 replaces the original 1D search process with a 2D search and thus reduces time cost. Heap is a well known data structure, and familiar to programmers. The I2L2-H combines the advantages of the I2L2 and heap. From the implementation viewpoint, heap is the simplest, and has the richest resources. From the performance viewpoint, the I2L2-H is the best, because it is more efficient at inserting than the I2L2, and restoring orders than a single heap.

6.4.3 Comparison between the original and improved I2L2

In this section, a comparison between the proposed I2L2-H and the original I2L2 proposed in [53] is presented. Table 6.5 shows the comparison results. From the results, with the assistance of the three improvements, the I2L2-H is faster than the I2L2 by over 6 times in the coffee cup lid example, and about 3 times in the peaks example. In the toy duck example, the I2L2-H outperforms the I2L2 by 17%.

Meanwhile, from the results of the coffee cup lid and toy duck examples (which have more than one million pixels), the time cost is only 70ms, i.e. about 15 frames per second (fps), which is near the requirements of real-time (24 fps). If the I2L2-H is applied to smaller wrapped phase map, it can exceed the real time requirement.

For instance, it achieves 75 fps in the peaks example.

Table 6.5: Comparison between the original and improved I2L2. Reproduced from "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," Appl. Opt. 53, 3492-3500 with permission from Optical Society of America.

		Peaks	Coffee cup lid	Toy Duck
Time(ms)	I2L2	36.28	436.27	63.29
	I2L2-H	12.84	66.19	54.27

6.5 Summary

In this chapter, three improvements of I2L2, the resumed searching, the adaptive mapping, and I2L2-H are introduced. The resumed searching focuses on the array of header pointers. It records the highest non-empty level during the unwrapping process, and reduces redundant comparisons. The adaptive mapping focuses on the quality value distribution in a quality map. With the adaptive mapping, the quality distribution is optimized, and the effect of bad distributions can be lessened. The I2L2-H focuses on the data structure for implementing levels of the I2L2. It replaces the original short linked lists with heap which is more efficient. Based on these three improvements, the I2L2-H outperforms the original I2L2 by 6 times in the best case, and it is thus considered as a good implementation of the adjoin list in the QGPU.

Chapter 7

A snake assisted quality-guided phase unwrapping for discontinuous phase fields

When phase discontinuity occurs, it prevents us from obtaining correct unwrapping results. Piecewise phase unwrapping is an effective solution of discontinuity. In this chapter, a snake model is introduced to achieve the piecewise phase unwrapping.[1]

7.1 Introduction

Quality-guided phase unwrapping (QGPU) is an effective, efficient, and automatic phase unwrapping method [48,53]. A quality map is used to represent the goodness of pixels and guide the phase unwrapping process from good pixels to bad ones. With the assistance of the filter-based quality maps, the QGPU has been

[1]The contents of this chapter is reproduced from "Snake-assisted quality-guided phase unwrapping for discontinuous phase fields," Appl. Opt. 54, 7462-7470 with permission from Optical Society of America.

shown to be very robust to noise [53]. However, because the QGPU assumes that the true phase map is continuous, its performance is affected by the discontinuities in the true phase map [122]. These discontinuities appear when the physical quantity being measured is discontinuous. A specimen with multiple components provides such an example.

Assume that a phase field has two separated, but both individually continuous, pieces. If these two continuous pieces are unwrapped sequentially, one after the other, the QGPU is expected to be successful. However, if the QGPU partially unwraps one piece, enters the other piece, and returns to the first piece, the unwrapping process usually fails. A piecewise unwrapping process is thus preferred when discontinuity occurs. To achieve the piecewise unwrapping, it is natural to segment the phase field into two pieces for individual unwrapping.

Interestingly, in the previous work [48,53], many successful piecewise unwrapping examples have been observed even without the segmentation. This is because the quality map used has low values at pixels in discontinuous areas, which automatically forms a barrier between two pieces of phase fields [48,53]. Obtaining such a quality map is possible because phase values within each piece are smooth while those between two pieces are discontinuous. The discontinuity level can be measured by, for example, windowed Fourier transform (WFT) [19]. The WFT is selected in this chapter as a background technique to build the proposed one on top of the WFT, because it provides a reliable quality measure as well as a clean denoised phase map [52].

Unfortunately, the success of the WFT based QGPU is not guaranteed. When the discontinuity is not obvious, the quality values may fail to automatically build a clear barrier between two pieces. To deal with such failed cases, in this chapter, we propose to explicitly segment the different pieces for piecewise unwrapping.

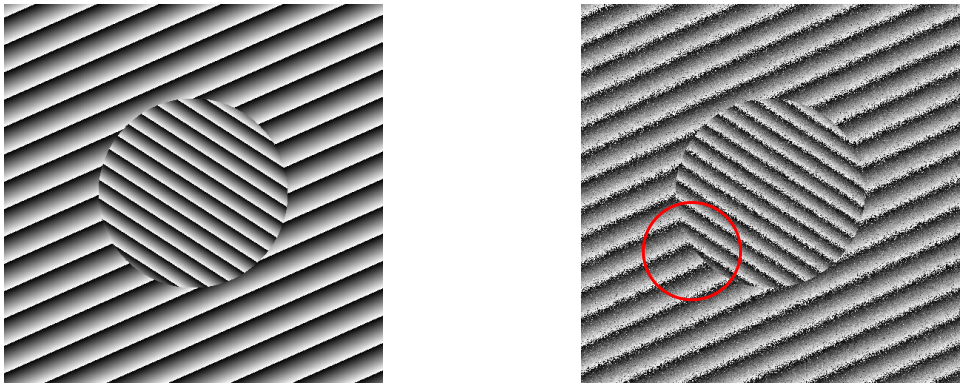
Because a discontinuity boundary is arbitrarily shaped, the snake models [124] are selected to fulfill the segmentation. The snakes are deformable curves and estimate the object boundaries via an energy minimizing process. They are therefore suitable for building boundaries of objects in the quality map. In particular, a gradient vector flow (GVF) snake model is used due to its good performance [126]. In this chapter, the proposed snake assisted QGPU (sQGPU) will be shown to be effective in solving challenging phase unwrapping problems with discontinuities.

7.2 The WFT based QGPU

The WFT based QGPU have been proposed [52]. When the WFR2 is used, the denoised phase is unwrapped with the ridge value $r(x, y)$ as the quality. When the WFF2 is used, the filtered phase $angle[\bar{f}(x, y)]$ is unwrapped with the filtered amplitude $|\bar{f}(x, y)|$ as the quality. Both of them work very effectively for smooth phase. Even when discontinuities exist, the WFT based QGPU works surprisingly well [52]. In Sec. 7.2.1, an example is given for demonstration and to gain insights. In Sec. 7.2.2, another example is provided to discuss the problem of this technique.

7.2.1 Demonstration of the WFT based QGPU

A wrapped phase map (512×512) with discontinuity is simulated [Fig. 7.1(a)]. Its corresponding noisy wrapped phase map is subsequently generated [Fig. 7.1(b)]. The WFR2/WFF2 algorithms are applied to the noisy wrapped phase map. The WFR2 algorithm has two outputs, a denoised wrapped phase map [Fig. 7.2(a)] and a ridge map [Fig. 7.2(b)]. The unwrapping process is performed on the denoised wrapped phase map following the guidance of the ridge map. In the ridge map, the pixels in the boundary area have lower values than other parts. These pixels



(a) (b)

Figure 7.1: A wrapped phase map with noise and discontinuity. (a) Clean wrapped phase, (b) wrapped phase with noise. The size of the wrapped phase is 512×512. Reproduced from "Snake-assisted quality-guided phase unwrapping for discontinuous phase fields," Appl. Opt. 54, 7462-7470 with permission from Optical Society of America.

automatically form an implicit barrier, and the two pieces of wrapped phase map are unwrapped separately. The path map and unwrapping result are shown in Figs. 7.2(c) and 7.2(d), respectively. The WFF2 based QGPU is similar to the WFR2 based QGPU, as shown in Fig. 7.3. It is noticed that a small portion of the unwrapping result is not correct, as highlighted in a red circle. Throughout this chapter, only the default parameters of the WFR2/WFF2 are used in order to simulate general situations. However, carefully tuned parameters can achieve better results. For example, by using a larger window size or a larger threshold value, the wrong portion can be easily rectified. For the details of the default and adjusted parameters, please refer to [19].

The WFR2/WFF2 are both local processing methods [19]. They work in a small window that slides in the wrapped phase map. The quality value of a pixel depends on the surrounding pixels in the small window. Smoother phase of the surrounding pixels gives the central pixel a higher quality value. However, although the boundary pixels are supposed to be discontinuous, they can be smooth in a small window. Consequently, the WFR2/WFF2 obtain a high quality and fail to

detect the discontinuity. A red circle is drawn in Fig. 7.1(b) to show such a case, which results in the partial failure shown in Fig. 7.3(d). Generally speaking, the WFR2/WFF2 based QGPU is quite successful even for discontinuous phase. Although it is considered as a global feature, a boundary still has "local presence and influence" [19], which are picked up by the WFR2/WFF2 for effective unwrapping.

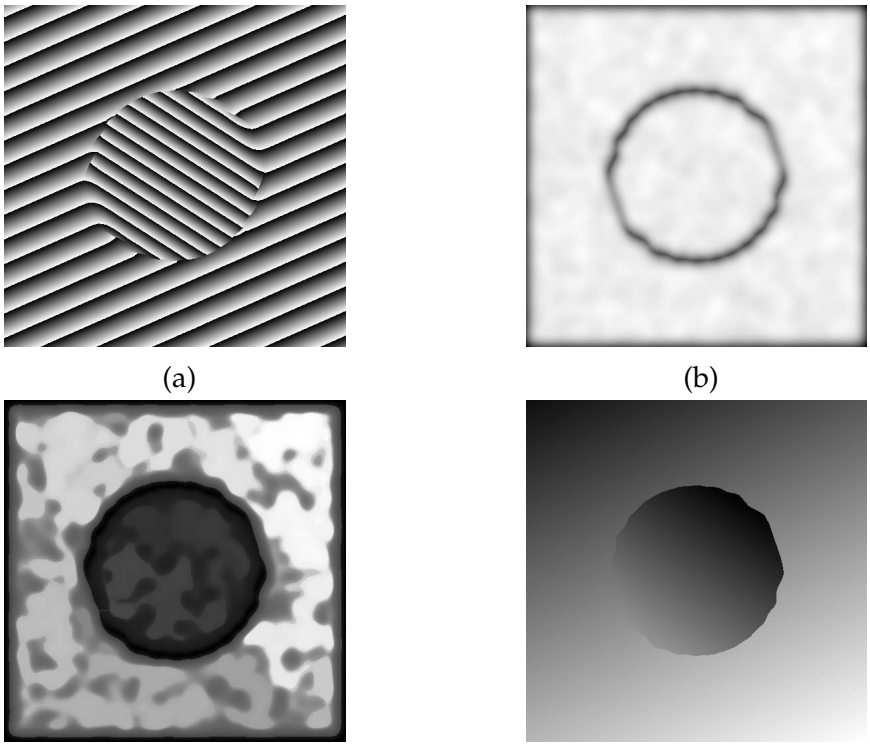


Figure 7.2: The WFR2 and discontinuity in the circle example. (a) Denoised wrapped phase, (b) quality map, (c) path map (pixels in light are unwrapped first and those in dark are unwrapped last. All the path maps below follow this color scheme), and (d) unwrapped phase. Reproduced from "Snake-assisted quality-guided phase unwrapping for discontinuous phase fields," Appl. Opt. 54, 7462-7470 with permission from Optical Society of America.

7.2.2 A remaining problem

Usually, boundaries are locally visible and easy to be captured by the WFR2/WFF2. However, when their local presence is insignificant, the boundaries are no longer

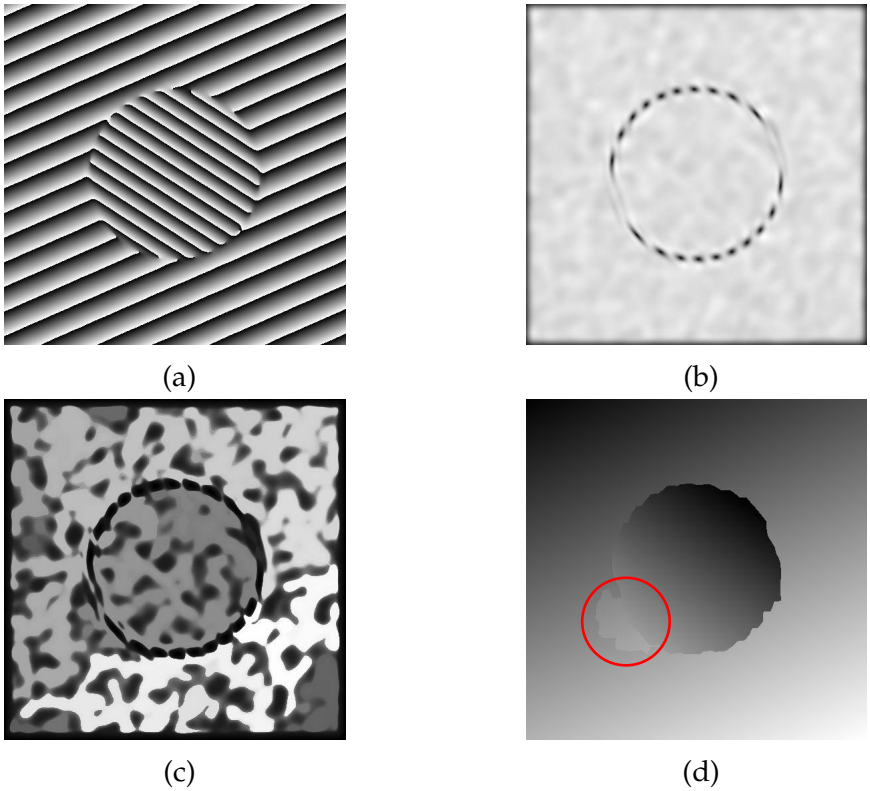
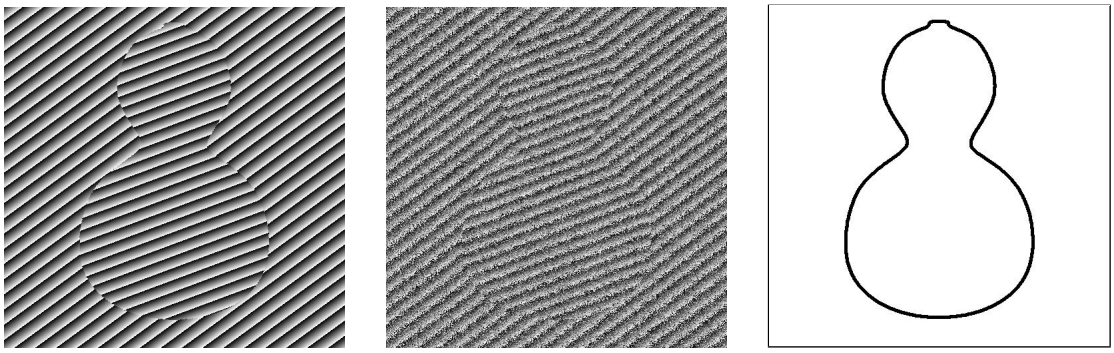


Figure 7.3: The WFF2 and discontinuity in the circle example. (a) Denoised wrapped phase map, (b) quality map, (c) path map, and (d) unwrapped phase. Reproduced from "Snake-assisted quality-guided phase unwrapping for discontinuous phase fields," Appl. Opt. 54, 7462-7470 with permission from Optical Society of America.

detectable, and the obtained quality maps are not reliable as well. The unwrapping results may be correct, but this success is not guaranteed. Figure 7.4 is a failed example that even adjusting parameters cannot improve the results. Due to the heavy noise, the boundary of the object is almost invisible in the wrapped phase map [Fig. 7.4(b)], and the WFR2/WFF2 can only detect a few segments of the boundary [Figs. 7.5(b) and 7.6(b)]. The openings of the boundary make the unwrapping path trespass the boundaries easily [Figs. 7.5(c) and 7.6(c)], and lead to incorrect results [Figs. 7.5(d) and 7.6(d)]. We aim at solving this problem in this chapter.



(a) (b) (c)

Figure 7.4: Another wrapped phase map with noise and discontinuity. (a) Clean wrapped phase, (b) wrapped phase with noise, (c) the outline of the object. The size of the wrapped phase is 512×512. Reproduced from "Snake-assisted quality-guided phase unwrapping for discontinuous phase fields," Appl. Opt. 54, 7462-7470 with permission from Optical Society of America.

7.3 Snake assisted QGPU

The problem exposed in Sec. 7.2.2 is encountered when the discontinuous phase looks smooth locally. This inspires us to use a global method to assist the local WFT based algorithms. Snake models are global and they can provide a continuous outline of objects to be detected. Therefore, they are selected to improve the WFT based QGPU. The principle of snake model, especially the GVF model, is introduced in Sec. 2.4. In this section, how to use the GVF snake model to assist the WFT based QGPU is introduced and discussed.

7.3.1 Getting the boundary via a GVF snake

A GVF snake is used to get the boundary of discontinuous phase for the subsequent piecewise QGPU. Getting the boundary involves two main steps: snake initialization and snake evolvement. For snake initialization, an automatic method is most desirable and considered first. In a quality map obtained from the WFR2/WFF2, a large number of boundary pixels have very low quality values. Thresholding is

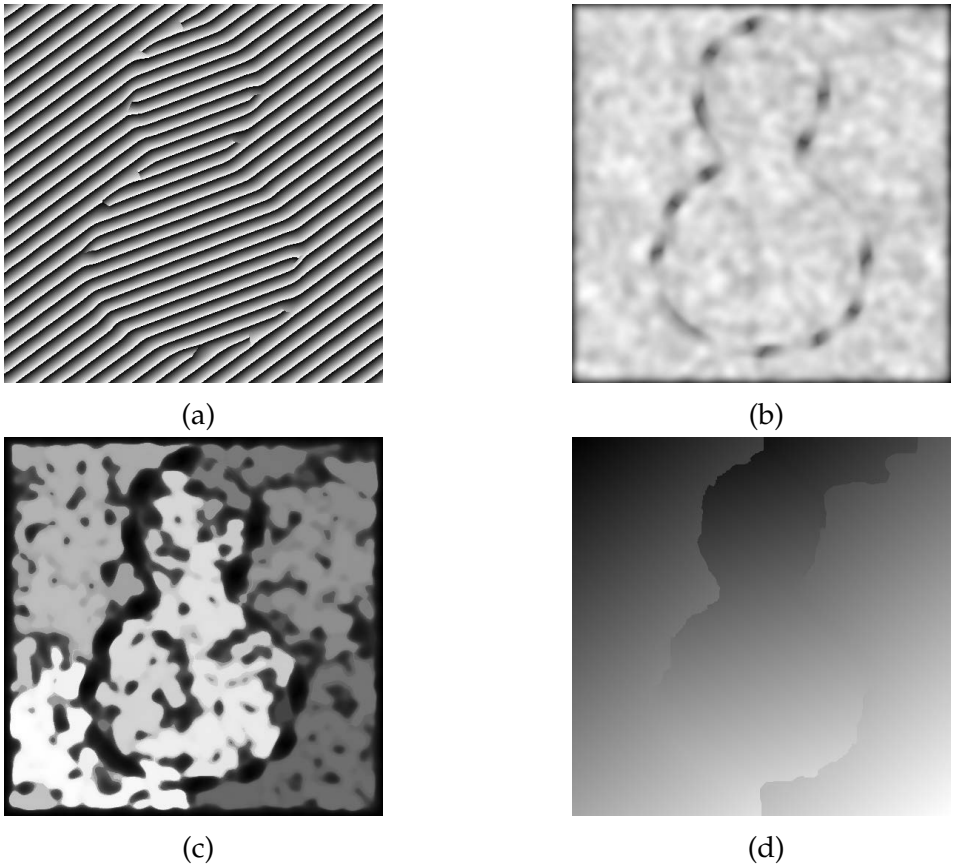


Figure 7.5: The WFR2 and discontinuity in the calabash shaped example. (a) Denoised wrapped phase map, (b) quality map, (c) path map, and (d) unwrapped phase. Reproduced from "Snake-assisted quality-guided phase unwrapping for discontinuous phase fields," Appl. Opt. 54, 7462-7470 with permission from Optical Society of America.

thus applied to the quality map to pick these boundary pixels, and subsequently a binary quality map is generated. The boundary pixels are marked as black, and other pixels are white in the binary image. From the binary quality map, a convex hull can be automatically and uniquely computed as the initial position of the snake.

As an example, Figs. 7.5(b) and 7.6(b) are thresholded into binary quality maps first. A proper threshold is easily and dynamically set. We start from a large threshold value in order to keep the majority of the boundary pixels below the threshold and thus recognized. However, some undesired non-boundary pixels

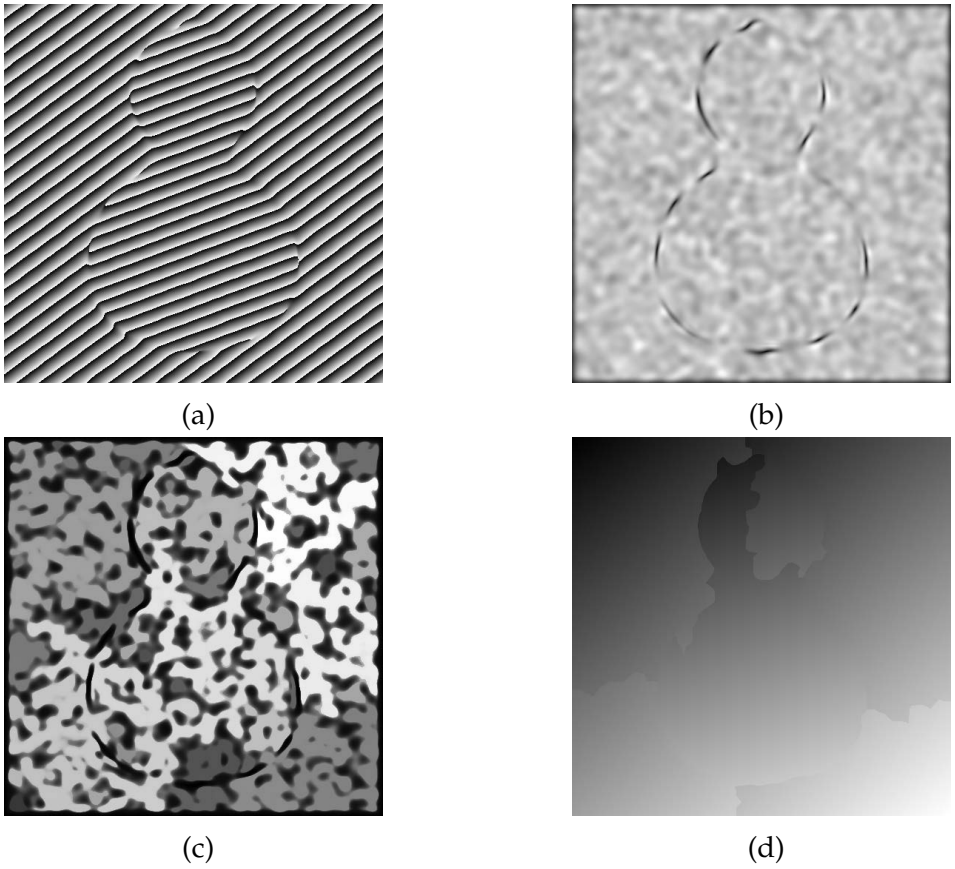


Figure 7.6: The WFF2 and discontinuity in the calabash shaped example. (a) Denoised wrapped phase map, (b) quality map, (c) path map, and (d) unwrapped phase. Reproduced from "Snake-assisted quality-guided phase unwrapping for discontinuous phase fields," Appl. Opt. 54, 7462-7470 with permission from Optical Society of America.

may also be below this threshold. We then gradually decrease the threshold until all these non-boundary pixels are removed. The final thresholded results are shown in Figs. 7.7(a) and 7.7(d), respectively. Next, in each binary quality map, a convex hull of the black points is computed [shown as the red lines in Figs. 7.7(a) and 7.7(d)]. The automatic initialization is simple to use, and performs well for simple objects, such as the circle example shown in Sec. 7.2. However, as seen from Figs. Figures. 7.7(a) and 7.7(d), the initial boundaries (convex hulls) could be away from the real boundary position and stuck in the subsequent optimization for snake evolution.

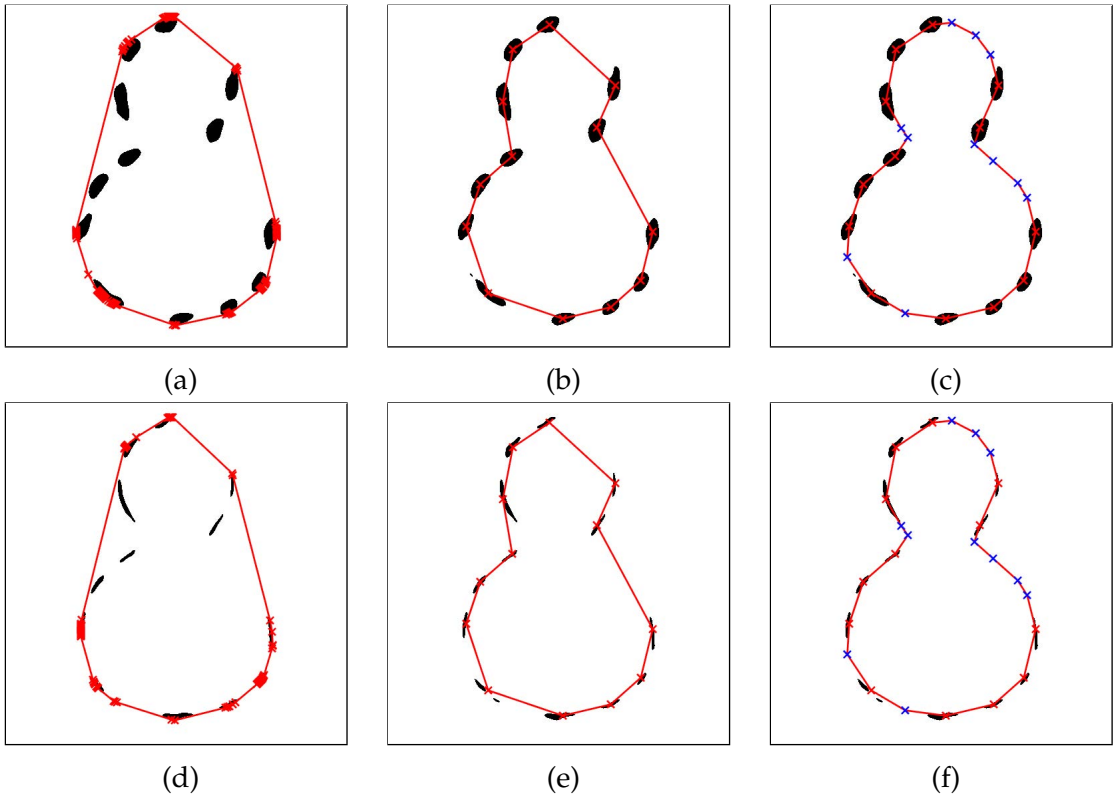


Figure 7.7: The WFR2/WFF2 binary images with different initialization methods. The WFR2: (a) automatic, (b) semi-automatic, and (c) manual; the WFF2: (d) automatic, (e) semi-automatic, and (f) manual. The red lines show the initial positions of the snakes. Reproduced from "Snake-assisted quality-guided phase unwrapping for discontinuous phase fields," Appl. Opt. 54, 7462-7470 with permission from Optical Society of America.

A semi-automatic method can help to improve the initialization by integrating user interaction. The user only needs to click a few key points in the black regions of a binary quality map, and the initial snake (a polygon) is generated by connecting the key points. In Figs. 7.7(b) and 7.7(e) , 13 key points are clicked for the 13 black regions, and the polygon connecting these 13 key points is used as the initial snake. Comparing with the automatic method, the semi-automatic method maximizes the utility of the black regions and initializes a better snake.

In both the automatic and the semi-automatic methods, the snakes are initialized solely based on the recognized boundary pixels. This leaves uncertainty for the snake to evolve between these black pixels in some challenging cases. To alle-

viate this problem, extra key points can be inserted manually between the black pixels. To insert meaningful extra key points, the original wrapped phase map and the original quality map can be used as a reference. Shown in Figs. 7.7(c) and 7.7(f) as an example, besides the 13 key points from the semi-automatic method, 11 extra key points [shown as blue crosses in Figs. 7.7(c) and 7.7(f)] on the boundary are clicked for insertion based on our observation of the WFR2/WFF2 quality maps [Figs. 7.5(b) and 7.6(b)]. The initial snake is then generated by connecting these 24 points, which is seen to be the most satisfactory.

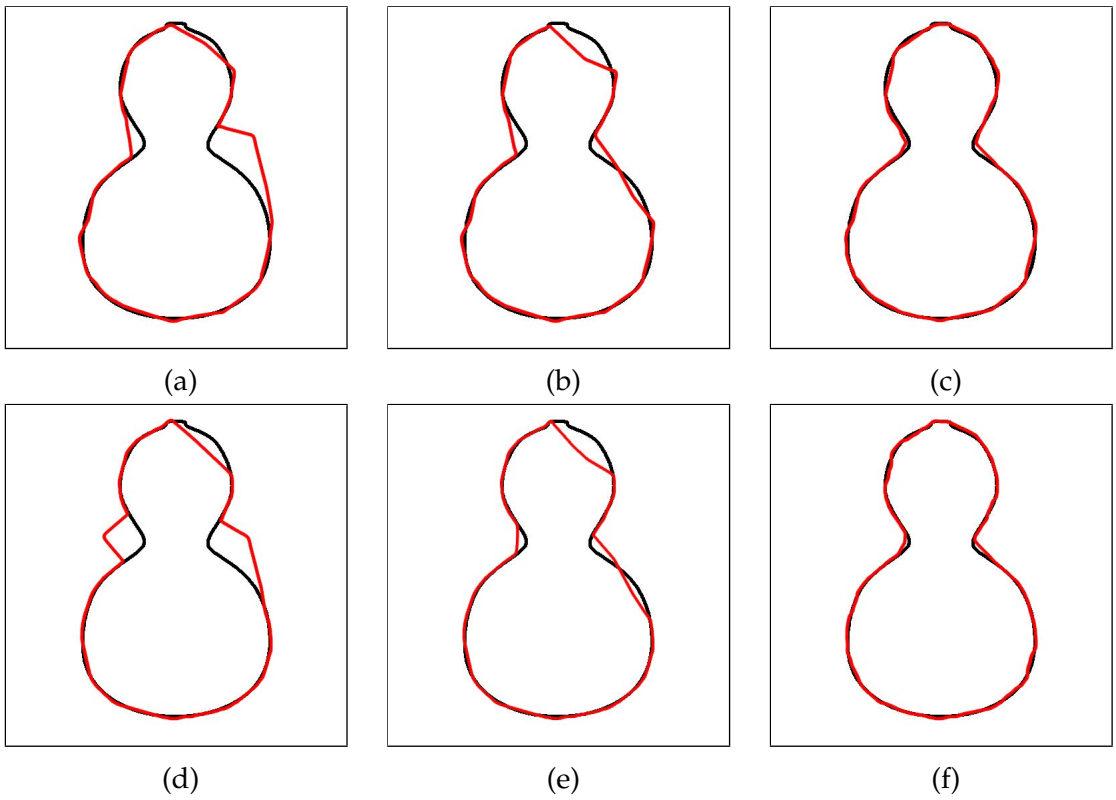


Figure 7.8: Obtained boundaries via three initialization methods. The WFR2: (a) automatic, (b) semi-automatic, and (c) manual; the WFF2: (d) automatic, (e) semi-automatic, and (f) manual. The black curves are the boundary of the object, and the red curves are the final positions of snakes. Reproduced from "Snake-assisted quality-guided phase unwrapping for discontinuous phase fields," Appl. Opt. 54, 7462-7470 with permission from Optical Society of America.

To evolve a snake, a GVF field is computed using the quality map as the edge

map. Usually the original quality map contains a large number of low quality pixels besides the ones in the discontinuous areas. These pixels outside the discontinuous areas will distort the GVF field. Thus the binary quality map is used to generate the GVF field. Once the GVF field is generated, the snake can then be evolved. The final snakes are shown in Fig. 7.8, corresponding to different initial snakes given in Fig. 7.7. Our algorithm is implemented by MATLAB and executed on a HP Z620 Workstation equipped with a 2.5 GHz quad-core CPU and 16 GB memory. In this example, 200 iterations are executed for the GVF generation and 150 iterations for the marching of the snakes. The costs are about 8s and 19s, respectively.

Table 7.1: Numbers of error points of obtained boundaries via three initialization methods.

	Automatic	Semi-automatic	Manual
WFR2	8409	6034	2801
WFF2	8354	4766	1751

Table 7.1 shows the numbers of error points of the final boundaries after snake marching with the three initialization methods. The manual method provides the best boundaries among the three initialization methods. The length of the true boundary is about 1140 pixels, and the average errors of the manual method are about 2.5 pixels (the WFR2) and 1.5 pixels (the WFF2), respectively.

7.3.2 Piecewise unwrapping using the detected boundaries

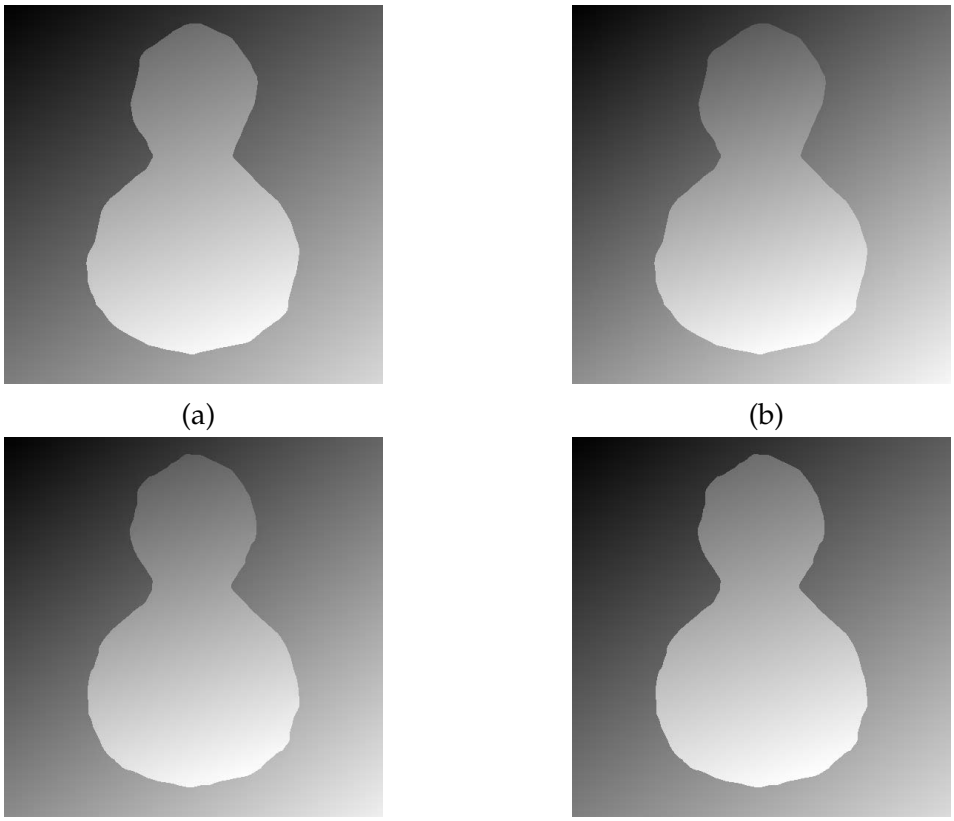
The boundaries detected above can be used to segment the whole wrapped phase map into individually continuous pieces. Each piece is separately unwrapped by the QGPU, for which the original quality map and the desnoised phase from the WFR2/WFF2 are used.

The WFR2/WFF2 assume that the phase being processed is continuous, but

this assumption is violated for the phase in the discontinuous areas. As a result, in these areas, the phase values estimated by the WFR2/WFF2 are slightly distorted from the true phase values. This distortion will be inherited by the unwrapped phase. Fortunately, it has been proven that, if there are two pieces of discontinuous phase and one of these two pieces contains only zero-phase, then the other piece of phase will not be distorted by the WFR2/WFF2 [169]. Therefore, we propose to apply the WFR2/WFF2 again to each segmented piece of phase to obtain a new quality map and new denoised phase. For each segmented piece for the WFR2/WFF2 processing, the phase values of the pixels inside the piece are taken from the original wrapped phase map, while others are filled with zeros. To easily differentiate these two types of processing, the previous WFR2/WFF2 processing is named as WFR2/WFF2-Full, while the processing proposed in this section is named as WFR2/WFF2-Piecewise. The subsequent piecewise unwrapping can be realized by the following two equivalent ways: (1) unwrap the segmented pieces one by one and then combine the results; (2) combine the denoised wrapped phase maps, and set the quality of the boundary pixels to the lowest value and then unwrap the combined wrapped phase.

To show the performance, the piecewise unwrapping results using the quality map and denoised phase from the WFR2/WFF2-Full are shown in Figs. 7.9(a) and 7.9(c), respectively. For comparison, the results using those from the WFR2/WFF2-Piecewise are shown in Figs. 7.9(b) and 7.9(d), respectively. All of them are able to unwrap the phase successfully and all the results are visually similar. Furthermore, each result is compared with the true phase and the error is rewrapped and shown in Fig. 7.10, from which it is easily observable that the WFR2/WFF2-Piecewise provide better results. Quantitatively, the mean absolute errors of the boundary pixels of both the WFR2-Full and WFF2-Full are about 0.25 rad, and they

are reduced to 0.11 rad (the WFR2-Piecewise) and 0.09 rad (the WFF2-Piecewise). The WFR2/WFF2-Piecewise are more accurate and thus recommended. It is worthy to see that the piecewise unwrapping is applicable to any QGPU algorithms, such as [77] and [104]. In this chapter, a high speed unwrapping algorithm reported in [77] is used.



(a) (b) (c) (d)

Figure 7.9: Unwrapping results of Full/Piecewise methods. (a) WFR2-Full, (b) WFR2-Piecewise, (c) WFF2-Full, and (d) WFF2-Piecewise. Reproduced from "Snake-assisted quality-guided phase unwrapping for discontinuous phase fields," Appl. Opt. 54, 7462-7470 with permission from Optical Society of America.

7.4 Experiments

In this section, an application of the sQGPU to a real example is demonstrated. A wrapped phase shown in Fig. 7.11(a) is obtained from a phase-shifting fringe

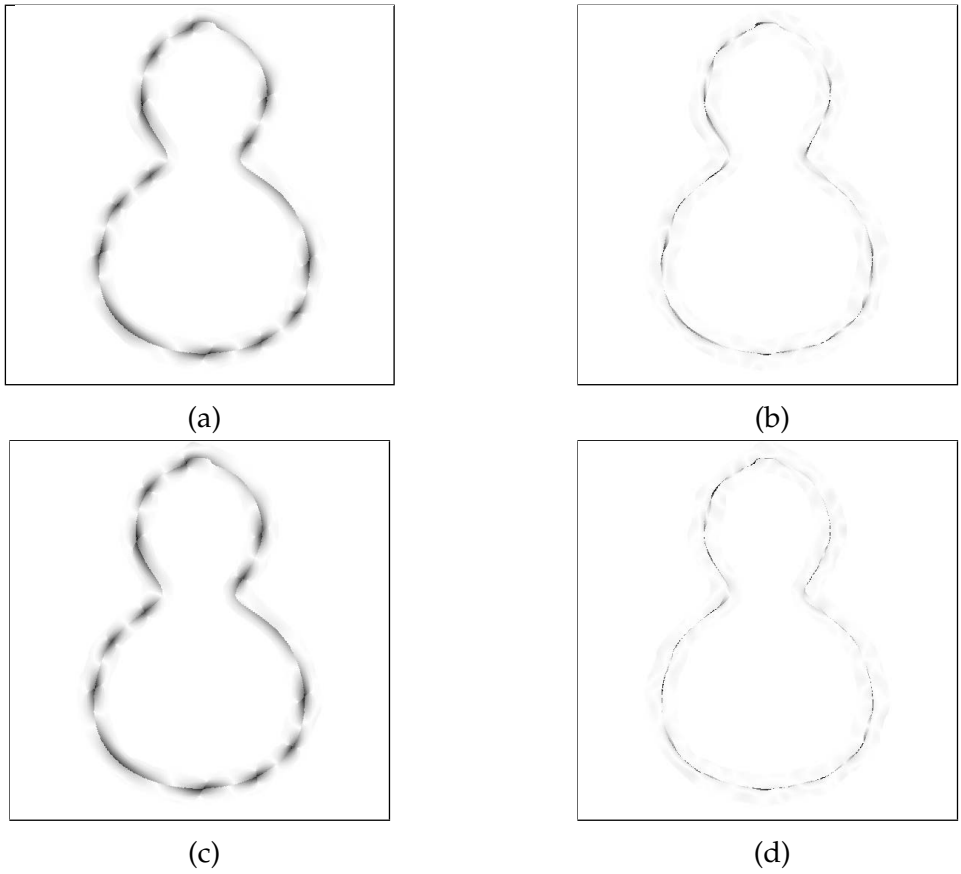
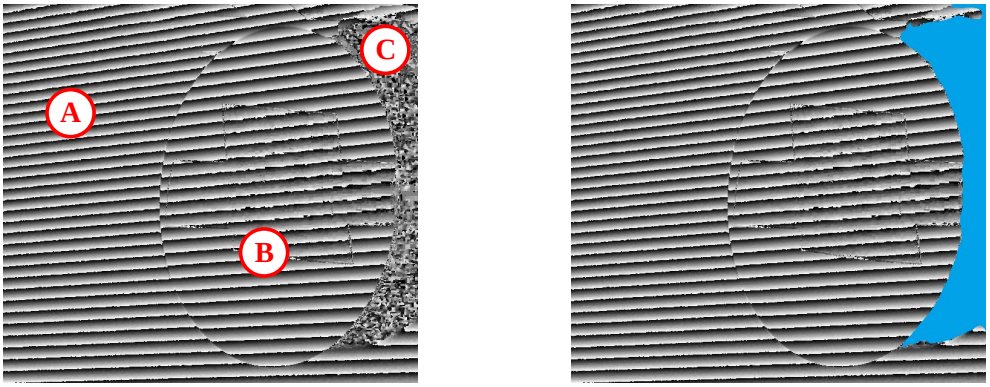


Figure 7.10: Rewrapping error maps of Full/Piecewise methods. (a) WFR2-Full; (b) WFR2-Piecewise; (c) WFF2-Full; (d) WFF2-Piecewise. Darker regions have higher errors. Reproduced from "Snake-assisted quality-guided phase unwrapping for discontinuous phase fields," Appl. Opt. 54, 7462-7470 with permission from Optical Society of America.

projection profilometry system [170], measuring an elliptic object (Zone B) on a background (Zone A). The noisy Zone C is due to the shadow of the object, and it is marked off using the WFR2 quality map in advance [Fig. 7.11(b)].

The denoised wrapped phase and the quality map (the ridge map) by the WFR2 are shown in Figs. 7.12(a) and 7.12(b), respectively. When they are directly used for the QGPU, the resulting path map and the unwrapped phase are shown in Figs. 7.12(c) and 7.12(d), respectively. From the path map, it is seen that the unwrapping process starts from the lower part of Zone B, proceeds to the left part of Zone A, and terminates at the upper part of Zone B again (i.e., B$\rightarrow$A$\rightarrow$B). The



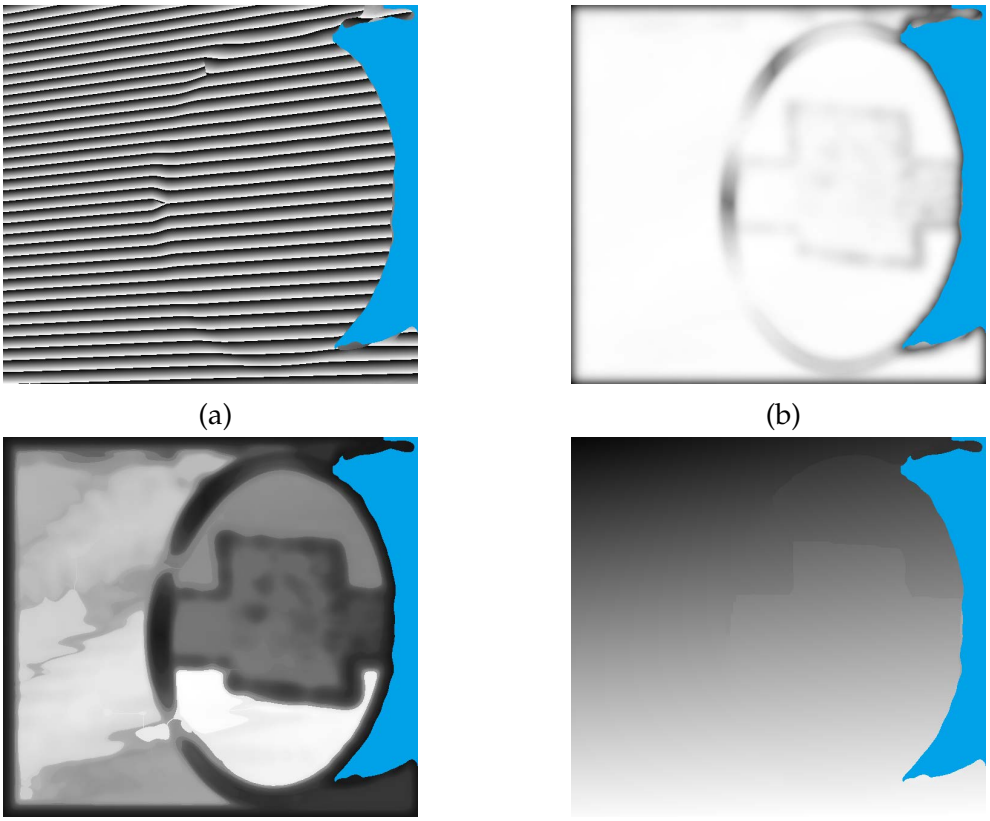
(a) (b)

Figure 7.11: (a) The wrapped phase of the real example, (b) the masked wrapped phase. The size of the wrapped phase is 686×626. Reproduced from "Snake-assisted quality-guided phase unwrapping for discontinuous phase fields," Appl. Opt. 54, 7462-7470 with permission from Optical Society of America.

quality map is not able to build up a boundary barrier for piecewise unwrapping and the unwrapped phase is not correct. Next, the WFF2 is used for denoising [Fig. 7.13(a)] and quality map generation [Fig. 7.13(b)]. The path map and the unwrapped phase are shown in Figs. 7.13(c) and 7.13(d), respectively. The path starts from Zone A, enters the lower part of Zone B, then returns to Zone A again, and ends at the upper part of Zone B again (i.e., A$\rightarrow$B$\rightarrow$A$\rightarrow$B). The unwrapping is also unsuccessful. A common characteristic of the two failed examples is that at least one zone has been visited more than once, making the unwrapping values in that zone inconsistent.

The sQGPU is then attempted. The WFR2/WFF2 quality maps in Figs. 7.12(b) and 7.13(b) are thresholded to generate binary images, as shown in Figs. 7.14(a) and 7.14(d), respectively. The snakes are initialized manually [Figs. 7.14(a) and 7.14(d)]. After the GVF snake marching, the final snakes are used as the phase boundaries, which are overlaid on the original wrapped phase map and shown in Figs. 7.14(b) and 7.14(e), respectively. Then the wrapped phase map [Fig. 7.11(a)] is unwrapped piecewise based on the boundaries. Computing the GVF fields

takes 200 iterations, and obtaining the final boundary uses 150 iterations, whose time costs are about 12s and 22s, respectively. The successful unwrapping results are given in Figs. 7.14(c) and 7.14(f). From Figs. 7.14(b) and 7.14(e), we can find the final boundaries are close to the real boundary in the wrapped phase map. The WFF2 provides a smoother boundary than the WFR2 in this example.



(a) (b)

(c) (d)

Figure 7.12: The WFR2 results of the real example: (a) denoised wrapped phase, (b) quality map, (c) path map, and (d) unwrapped phase. Reproduced from "Snake-assisted quality-guided phase unwrapping for discontinuous phase fields," Appl. Opt. 54, 7462-7470 with permission from Optical Society of America.

7.5 Discussions

First, in the experimental example, after careful selection of the parameter tuning, we find that with a very small window size, using the WFR2/WFF2 directly is able

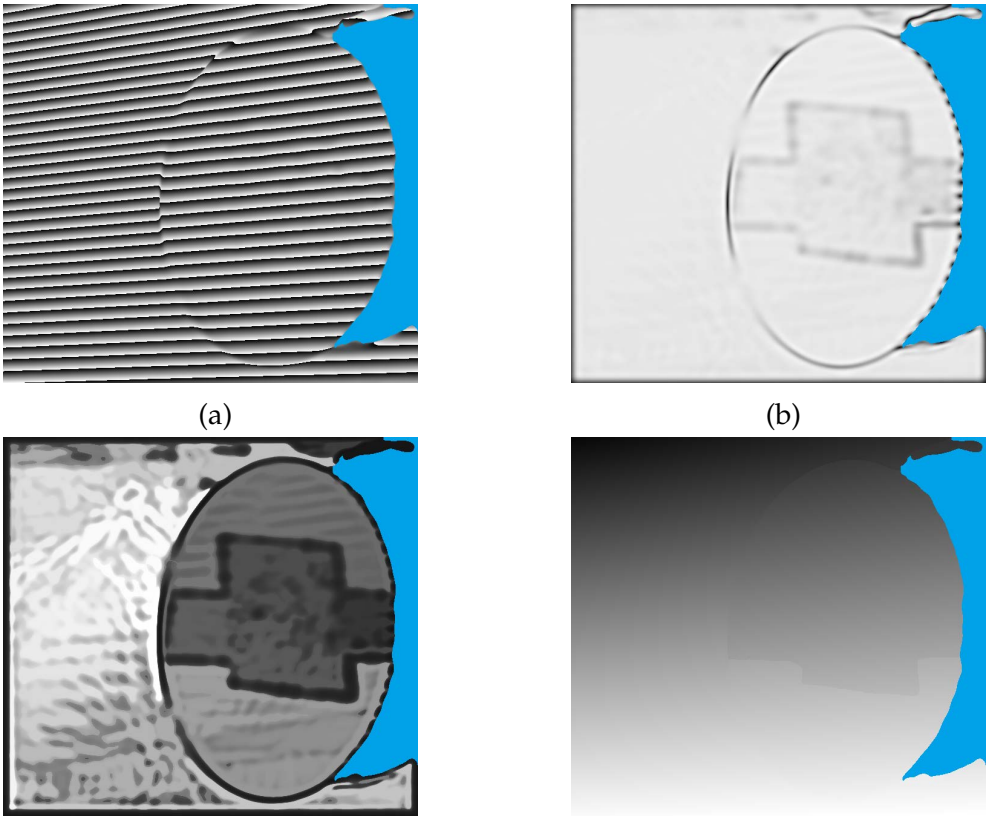


Figure 7.13: The WFF2 results of the real example: (a) denoised wrapped phase, (b) quality map, (c) path map, and (d) unwrapped phase. Reproduced from "Snake-assisted quality-guided phase unwrapping for discontinuous phase fields," Appl. Opt. 54, 7462-7470 with permission from Optical Society of America.

to unwrap the phase successfully, but adding in a little noise will fail it. On the contrary, the sQGPU performs very robust to the noise. This makes the sQGPU is good backup technique for unwrapping challenging discontinuous phase maps.

Second, the user interaction in the sQGPU makes it less automatic. This is another reason that we view it as a necessary backup technique.

Third, as can be seen throughout the chapter, the WFR2 and WFF2 perform similarly, while it could happen that one is slightly better than the other [52]. Thus both WFR2 and WFF2 are encouraged for attempting.

Fourth, in this chapter, the quality maps from the WFR2/WFF2 are solely used

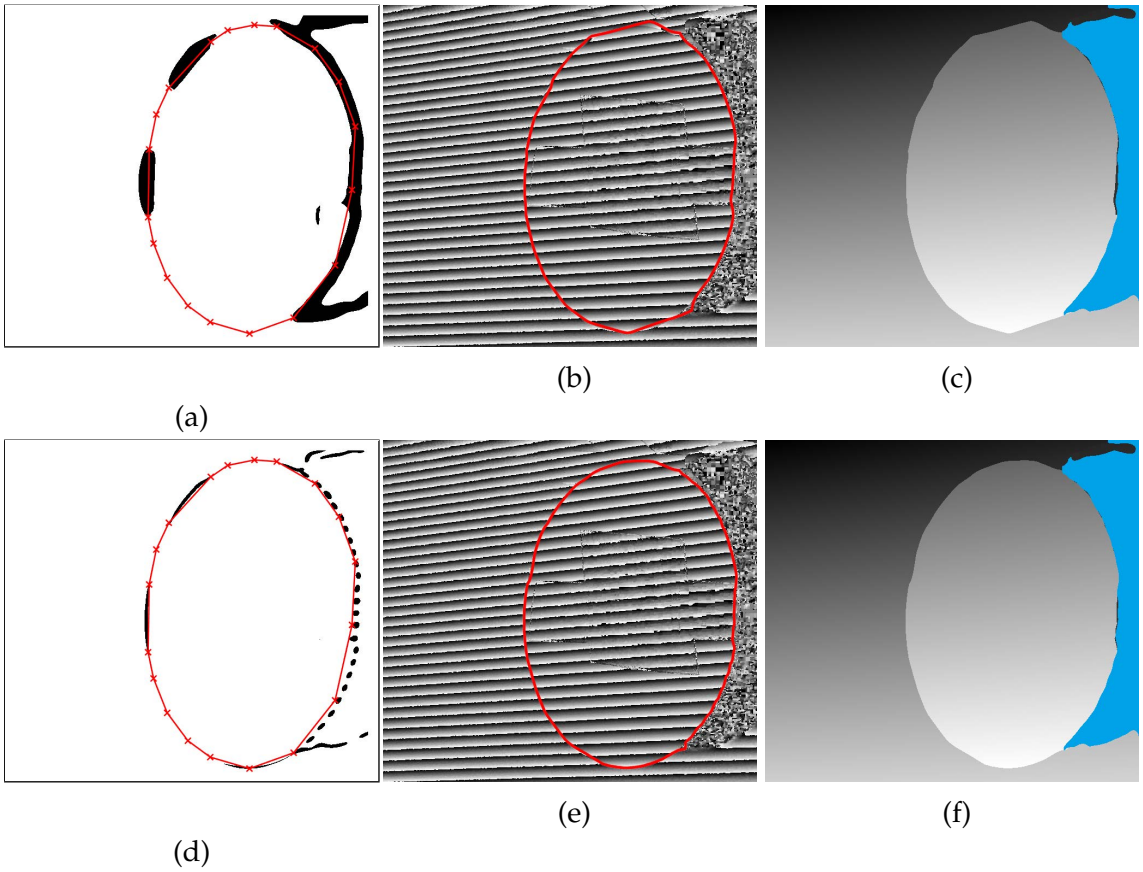


Figure 7.14: The sQGPU results. The WFR2: (a) the initial snake, (b) the final boundary, (c) the unwrapped phase; the WFF2: (d) the initial snake, (e) the final boundary, (f) the unwrapped phase. Reproduced from "Snake-assisted quality-guided phase unwrapping for discontinuous phase fields," Appl. Opt. 54, 7462-7470 with permission from Optical Society of America.

because of their established good performances, especially in a noisy condition [53]. However, this does not prevent the readers from using other quality maps suitable for the problem being solved, such as fringe modulation.

Fifth, the proposed technique is designed for a wrapped phase map containing multiple well-defined pieces, but may not be a placebo for a wrapped phase with complicated discontinuities.

Finally, as both the WFR2/WFF2 and QGPU are automatic and towards real-time [77,85], reducing user interaction in snakes and making the algorithm faster are of interest in the future work.

7.6 Summary

In this chapter, the snake assisted quality guided phase unwrapping (sQGPU) is proposed to solve the phase discontinuity problem in phase unwrapping. The sQGPU works on the WFR2/WFF2 quality maps, and affords a closed boundary to implement piecewise phase unwrapping by using the GVF snake model. For different samples, the sQGPU provides three initialization methods to make a balance between accuracy and automaticity. From the results, the sQGPU is able to unwrap some wrapped phase maps that the WFT based QGPU fails. Though it requires extra steps to compute the GVF fields and evolve the snake, the sQGPU is a reliable solution when other quality measures could not work properly.

Chapter 8

Conclusions and future work

8.1 Summary

Fringe pattern based measurement techniques provide an effective way to measure objects. Due to their good performance, fringe pattern analysis techniques are extensively used in metrology applications. However, some difficulties, such as noises and discontinuities, may prevent people from obtaining correct results. In this dissertation, we focus on two steps of the fringe pattern analysis framework, denoising and phase unwrapping, because these two steps are very challenging, but important and closely connected.

Windowed Fourier transform (WFT) based algorithms are powerful tools for fringe pattern analysis. Two WFT based algorithms, the windowed Fourier ridges (WFR2) and the windowed Fourier filtering (WFF2) have been widely adopted for denoising, demodulation, and extracting local properties. However, the WFR2/WFF2 are highly CPU-intensive and time consuming. This situation limits the usage of the WFR2/WFF2 in real-time applications. In Chap. 3, we discuss how to implement the WFR2/WFF2 efficiently. Fourier transform (FT) is the core part of the WFR2/WFF2. A fast implementation of FT can explicitly improve the perfor-

mance of the algorithms. Different selections of FT libraries, compilers and the size of the FT input are examined and discussed. With the combination of the MKL, the ICC, and the preferred sizes, the WFR2/WFF2 can achieve the highest speed. Parallelization of the WFR2/WFF2 algorithms offers another improvement. Four parallel techniques, OpenMP, Cilk Plus, the TBB and the PPL are used to parallelize the main body of the WFR2/WFF2, and OpenMP gives the fastest speed. For a 1024×1024 fringe pattern, the OpenMP technique improves the performance of the WFR2/WFF2 by 11 and 17 times, respectively.

In Chap. 4, the comparison and integration of the WFF2 and the block-matching and 3D filtering (BM3D) are introduced. The WFF2 and the BM3D methods are two types of denoising techniques. The WFF2 is specially designed for processing fringe patterns with a local strategy. The BM3D is a general image denoising method which follows a non-local strategy. In this chapter, their denoising performance on fringe patterns are first compared. The results reveal that both of them are effective to remove noises, while the WFF2 is better. Meanwhile, the WFF2 becomes less reliable in discontinuous areas of a fringe pattern, but the BM3D works well in those areas. A hybrid method, WFF-BM3D, which combines the advantages of the WFF2 and BM3D is thus proposed. The WFF-BM3D outperforms the WFF2 method in discontinuous areas as expected.

Phase unwrapping is the final step of the fringe pattern analysis framework. It accepts denoised wrapped phase maps as the input, and outputs unwrapped phase maps which can be directly used for further analyses. Quality guided phase unwrapping method (QGPU) is a reliable and efficient phase unwrapping method. How to select quality maps and guiding strategies determines the final output of the QGPU. Therefore, we try to seek reliable and high-speed combination of quality maps and guiding strategies. In Chap. 5, different quality maps are compared.

For noisy phase maps, transform-based methods perform better, but need more time. When discontinuities present, successful unwrapping results are not guaranteed with existing quality maps. For guiding strategies, three strategies, classical, two-section and stack-chain, are discussed. The classical quality guiding strategy is the most reliable, and it can be can be accelerated by well-designed data structures, such as the proposed interwoven index linked list (I2L2). The two-section method runs faster, but it needs a threshold which makes it less automatic. The stack-chain method is fastest but is not error-free.

In Chap. 6, we further improve the performance of the I2L2. Three improvements, the resumed searching, the adaptive mapping, and I2L2-H are made. The resumed searching records and updates the highest non-empty level during the unwrapping process. This operation reduces the time cost for finding the highest non-empty level, and redundant comparisons are removed. The adaptive mapping focuses on the problem of concentrated distribution. It maps the quality values to the levels of the I2L2 in a nonlinear and adaptive way, rather than the original linear mapping. After the mapping, the quality value distribution is optimized, and the unwrapping process can benefit from the optimized distribution. Heap is a data structure which is efficient on insertion. The new data structure, I2L2-H, replaces short linked lists with heaps. As the result reveals, the I2L2-H performs better than the I2L2 and the heap. In the best case, the I2L2-H achieves more than 6 times faster than the original I2L2.

Finally, in Chap. 7, the snake assisted quality guided phase unwrapping (sQGPU) is proposed to solve the phase discontinuity problem in phase unwrapping. The GVF snake model is used to identify object boundaries in the WFR2/WFF2 quality maps. With the assistance of the detected boundaries, piecewise phase unwrapping is achievable which directly avoid the discontinuity problem. In the

sQGPU, three initialization methods of the GVF snake are available, and people can make a choice based on preference for accuracy or automaticity. From the results, the sQGPU is able to unwrap some wrapped phase maps that the WFT based QGPU fails. When other quality measures could not work properly, sQGPU can be used as the backup solution.

8.2 Future work

8.2.1 Future work on the WFR2/WFF2

Current WFR2/WFF2 algorithms select an exhaustive search strategy to find suitable frequencies. This strategy guarantees all possible combinations of frequencies can be examined during the WFT process. Otherwise, the output may be inaccurate. However, this strategy requires a large number of FT computations. Implementations with parallelization techniques, such as GPU and multi-core, can explicitly improve the performance, but they do not reduce the number of FT computations. Recently, in Ref. [171], the authors applied 3-point Gaussian fitting method to interpolate ridges in the WFR2 algorithm, and obtained positive results. Further studies on the distribution of frequencies could be performed. Once a better estimation of ridges is obtained, the WFR2 algorithm can be directly accelerated without any accuracy lost, and the WFF2 algorithm can be benefited as well.

8.2.2 Future work on parallelizing QGPU

From Chaps. 5 and 6, it is obvious that I2L2 and I2L2-H are all sequential data structures. New pixels must be inserted sequentially based on their quality values. This restrains the data structures from exploiting the power of modern multi-core

CPUs. To fully utilize multi-core CPUs, parallel data structures could be helpful. Possible candidates include parallel linked list [172], parallel heap [173, 174]. However, due to the characteristic of the QGPU, only three pixels can be inserted concurrently at most. The efficiency of the parallel data structures might be decreased. Seeking a suitable parallel data structure for QGPU is still a challenging task.

8.2.3 Future work on quality metric for fringe pattern denoising algorithm

To evaluate a denoising method, a proper quality metric is always necessary. Different quality metrics describe the difference between the clean and filtered fringe patterns in different ways. Seeking a suitable combination of quality metrics is helpful to have a better understanding of the denoising algorithms.

References

[1] K. Harding, *Handbook of Optical Dimensional Metrology*. CRC Press, 2013.

[2] K. E. Peiponen, R. Myllylä, and A. V. Priezzhev, *Optical measurement techniques: innovations for industry and the life sciences*, vol. 136. Springer, 2009.

[3] P. Hariharan, *Basics of interferometry*. Academic Press, second ed., 2010.

[4] T. J. Wright, Z. Lu, and C. Wicks, "Source model for the mw 6.7, 23 October 2002, Nenana Mountain Earthquake (Alaska) from InSAR," *Geophy. Res. Lett.*, vol. 30, no. 18, 2003.

[5] T. J. Wright, B. E. Parsons, and Z. Lu, "Toward mapping surface deformation in three dimensions using InSAR," *Geophy. Res. Lett.*, vol. 31, no. 1, 2004.

[6] A. Hooper, H. Zebker, P. Segall, and B. Kampes, "A new method for measuring deformation on volcanoes and other natural terrains using insar persistent scatterers," *Geophy. Res. Lett.*, vol. 31, no. 23, 2004.

[7] Y. Y. Hung, "Shearography: a new optical method for strain measurement and nondestructive testing," *Opt. Eng.*, vol. 21, no. 3, pp. 213391–213391, 1982.

[8] Y. Y. Hung, "Shearography for non-destructive evaluation of composite structures," *Opt. Lasers Eng.*, vol. 24, no. 2–3, pp. 161–182, 1996.

[9] Y. Y. Hung, Y. S. Chen, S. P. Ng, L. Liu, Y. H. Huang, B. L. Luk, R. W. L. Ip, C. M. L. Wu, and P. S. Chung, "Review and comparison of shearography and active thermography for nondestructive evaluation," *Mater. Sci. and Eng.: R: Reports*, vol. 64, no. 5, pp. 73–112, 2009.

[10] B. Han and Y. Guo, "Thermal deformation analysis of various electronic packaging products by moiré and microscopic moiré interferometry," *J. Electron. Packaging*, vol. 117, no. 3, pp. 185–191, 1995.

[11] D. Post, B. Han, and P. Ifju, *High sensitivity moiré: experimental analysis for mechanics and materials*. Springer Science & Business Media, 2012.

[12] F. Charrière, J. Kühn, T. Colomb, F. Montfort, E. Cuche, Y. Emery, K. Weible, P. Marquet, and C. Depeursinge, "Characterization of microlenses by digital holographic microscopy," *Appl. Opt.*, vol. 45, pp. 829–835, Feb 2006.

[13] C. Mann, L. Yu, C.-M. Lo, and M. Kim, "High-resolution quantitative phase-contrast microscopy by digital holography," *Opt. Express*, vol. 13, pp. 8693–8698, Oct 2005.

[14] P. Marquet, B. Rappaz, P. J. Magistretti, E. Cuche, Y. Emery, T. Colomb, and C. Depeursinge, "Digital holographic microscopy: a noninvasive contrast imaging technique allowing quantitative visualization of living cells with subwavelength axial accuracy," *Opt. Lett.*, vol. 30, pp. 468–470, Mar 2005.

[15] D. W. Robinson and G. T. Reid, *Interferogram analysis, digital fringe pattern measurement techniques*. CRC Press, 1993.

[16] D. Malacara, *Optical shop testing*, vol. 59. John Wiley & Sons, 2007.

[17] S. S. Gorthi and P. Rastogi, "Fringe projection techniques: whither we are?," *Opt. Lasers Eng.*, vol. 48, no. IMAC-REVIEW-2009-001, pp. 133–140, 2010.

[18] S. Zhang, "Recent progresses on real-time 3D shape measurement using digital fringe projection techniques," *Opt. Lasers Eng.*, vol. 48, no. 2, pp. 149–158, 2010.

[19] Q. Kemao, *Windowed Fringe Pattern Analysis*. SPIE Press, 2013.

[20] J. H. Bruning, D. R. Herriott, J. E. Gallagher, D. P. Rosenfeld, A. D. White, and D. J. Brangaccio, "Digital wavefront measuring interferometer for testing optical surfaces and lenses," *Appl. Opt.*, vol. 13, pp. 2693–2703, Nov 1974.

[21] I. Yamaguchi and T. Zhang, "Phase-shifting digital holography," *Opt. Lett.*, vol. 22, pp. 1268–1270, Aug 1997.

[22] M. Gupta and S. Nayar, "Micro phase shifting," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012*, pp. 813–820, June 2012.

[23] I. Chivers and J. Sleightholme, *Introduction to Programming with Fortran. With Coverage of Fortran 90, 95, 2003, 2008 and 77*. Springer London, 2012.

[24] E. P. Goodwin and J. C. Wyant, *Field Guide to Interferometric Optical Testing*. SPIE Press, 2006.

[25] J. M. Huntley and H. Saldner, "Temporal phase-unwrapping algorithm for automated interferogram analysis," *Appl. Opt.*, vol. 32, pp. 3047–3052, Jun 1993.

[26] H. O. Saldner and J. M. Huntley, "Temporal phase unwrapping: application to surface profiling of discontinuous objects," *Appl. Opt.*, vol. 36, pp. 2770–2775, May 1997.

[27] M. Takeda, H. Ina, and S. Kobayashi, "Fourier-transform method of fringe-pattern analysis for computer-based topography and interferometry," *J. Opt. Soc. Am.*, vol. 72, pp. 156–160, Jan 1982.

[28] M. Takeda and K. Mutoh, "Fourier transform profilometry for the automatic measurement of 3-D object shapes," *Appl. Opt.*, vol. 22, pp. 3977–3982, Dec 1983.

[29] T. Kreis, *Handbook of holographic interferometry: optical and digital methods*. John Wiley & Sons, 2006.

[30] S. Li, X. Su, W. Chen, and L. Xiang, "Eliminating the zero spectrum in fourier transform profilometry using empirical mode decomposition," *J. Opt. Soc. Am. A*, vol. 26, pp. 1195–1201, May 2009.

[31] X. Zhou, A. G. Podoleanu, Z. Yang, T. Yang, and H. Zhao, "Morphological operation-based bi-dimensional empirical mode decomposition for automatic background removal of fringe patterns," *Opt. Express*, vol. 20, pp. 24247–24262, Oct 2012.

[32] J. A. Quiroga, J. A. Gómez-Pedrero, and Á. García-Botella, "Algorithm for fringe pattern normalization," *Opt. Commun.*, vol. 197, no. 1-3, pp. 43 – 51, 2001.

[33] J. A. Quiroga and M. Servin, "Isotropic n-dimensional fringe pattern normalization," *Opt. Commun.*, vol. 224, no. 4–6, pp. 221 – 227, 2003.

[34] J. A. Guerrero, J. L. Marroquin, M. Rivera, and J. A. Quiroga, "Adaptive monogenic filtering and normalization of espi fringe patterns," *Opt. Lett.*, vol. 30, pp. 3018–3020, Nov 2005.

[35] M. Servin, J. L. Marroquin, and F. J. Cuevas, "Demodulation of a single interferogram by use of a two-dimensional regularized phase-tracking technique," *Appl. Opt.*, vol. 36, pp. 4540–4548, Jul 1997.

[36] M. Servín, J. A. Quiroga, and M. Padilla, *Fringe pattern analysis for optical metrology: theory, algorithms, and applications*. John Wiley & Sons, 2014.

[37] C. Tian, Y. Yang, D. Liu, Y. Luo, and Y. Zhuo, "Demodulation of a single complex fringe interferogram with a path-independent regularized phase-tracking technique," *Appl. Opt.*, vol. 49, pp. 170–179, Jan 2010.

[38] L. Kai and Q. Kemao, "A generalized regularized phase tracker for demodulation of a single fringe pattern," *Opt. Express*, vol. 20, pp. 12579–12592, May 2012.

[39] L. Kai and Q. Kemao, "Improved generalized regularized phase tracker for demodulation of a single fringe pattern," *Opt. Express*, vol. 21, pp. 24385–24397, Oct 2013.

[40] M. Rivera, "Robust phase demodulation of interferograms with open or closed fringes," *J. Opt. Soc. Am. A*, vol. 22, pp. 1170–1175, Jun 2005.

[41] O. S. D.-C. no, M. Rivera, and R. Legarda-Saenz, "Fast phase recovery from a single closed-fringe pattern," *J. Opt. Soc. Am. A*, vol. 25, pp. 1361–1370, Jun 2008.

[42] Q. Kemao, "Two-dimensional windowed fourier transform for fringe pattern analysis: Principles, applications and implementations," *Opt. Lasers Eng.*, vol. 45, no. 2, pp. 304 – 317, 2007.

[43] Q. Kemao and S. H. Soon, "Sequential demodulation of a single fringe pattern guided by local frequencies," *Opt. Lett.*, vol. 32, pp. 127–129, Jan 2007.

[44] H. Wang and Q. Kemao, "Frequency guided methods for demodulation of a single fringe pattern," *Opt. Express*, vol. 17, pp. 15118–15127, Aug 2009.

[45] L. Kai and Q. Kemao, "Fast frequency-guided sequential demodulation of a single fringe pattern," *Opt. Lett.*, vol. 35, pp. 3718–3720, Nov 2010.

[46] K. Itoh, "Analysis of the phase unwrapping algorithm," *Appl. Opt.*, vol. 21, pp. 2470–2470, Jul 1982.

[47] R. M. Goldstein, H. A. Zebker, and C. L. Werner, "Satellite radar interferometry: Two-dimensional phase unwrapping," *Radio Sci.*, vol. 23, no. 4, pp. 713–720, 1988.

[48] D. C. Ghiglia and M. D. Pritt, *Two-dimensional phase unwrapping: theory, algorithms, and software*. Wiley New York, 1998.

[49] M. A. Herráez, D. R. Burton, M. J. Lalor, and M. A. Gdeisat, "Fast two-dimensional phase-unwrapping algorithm based on sorting by reliability following a noncontinuous path," *Appl. Opt.*, vol. 41, pp. 7437–7444, Dec 2002.

[50] X. Su and W. Chen, "Reliability-guided phase unwrapping algorithm: a review," *Opt. Lasers Eng.*, vol. 42, no. 3, pp. 245–261, 2004.

[51] S. Zhang, X. Li, and S.-T. Yau, "Multilevel quality-guided phase unwrapping algorithm for real-time three-dimensional shape reconstruction," *Appl. Opt.*, vol. 46, pp. 50–57, Jan 2007.

[52] Q. Kemao, W. Gao, and H. Wang, "Windowed fourier-filtered and quality-guided phase-unwrapping algorithm," *Appl. Opt.*, vol. 47, pp. 5420–5428, Oct 2008.

[53] M. Zhao, L. Huang, Q. Zhang, X. Su, A. Asundi, and Q. Kemao, "Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies," *Appl. Opt.*, vol. 50, no. 33, pp. 6214–6224, 2011.

[54] M. Zhao and Q. Kemao, "Quality-guided phase unwrapping implementation: an improved indexedinterwoven linked list," *Appl. Opt.*, vol. 53, pp. 3492–3500, Jun 2014.

[55] D. C. Ghiglia and L. A. Romero, "Robust two-dimensional weighted and unweighted phase unwrapping that uses fast transforms and iterative methods," *J. Opt. Soc. Am. A*, vol. 11, pp. 107–117, Jan 1994.

[56] J. L. Marroquin and M. Rivera, "Quadratic regularization functionals for phase unwrapping," *J. Opt. Soc. Am. A*, vol. 12, pp. 2393–2400, Nov 1995.

[57] D. C. Ghiglia and L. A. Romero, "Minimum Lp-norm two-dimensional phase unwrapping," *J. Opt. Soc. Am. A*, vol. 13, pp. 1999–2013, Oct 1996.

[58] J. Bioucas-Dias and G. Valadao, "Phase unwrapping via graph cuts," *IEEE T. Image Process.*, vol. 16, pp. 698–709, March 2007.

[59] C. Tang, F. Zhang, H. Yan, and Z. Chen, "Denoising in electronic speckle pattern interferometry fringes by the filtering method based on partial differential equations," *Opt. Commun.*, vol. 260, no. 1, pp. 91 – 96, 2006.

[60] C. Tang, L. Han, H. Ren, D. Zhou, Y. Chang, X. Wang, and X. Cui, "Second-order oriented partial-differential equations for denoising in electronic-speckle-pattern interferometry fringes," *Opt. Lett.*, vol. 33, pp. 2179–2181, Oct 2008.

[61] F. Zhang, W. Liu, J. Wang, Y. Zhu, and L. Xia, "Anisotropic partial differential equation noise-reduction algorithm based on fringe feature for ESPI," *Opt. Commun.*, vol. 282, no. 12, pp. 2318 – 2326, 2009.

[62] C. Tang, L. Wang, and H. Yan, "Overview of anisotropic filtering methods based on partial differential equations for electronic speckle pattern interferometry," *Appl. Opt.*, vol. 51, pp. 4916–4926, Jul 2012.

[63] J. Weickert, "Coherence-enhancing diffusion filtering," *Int. J. Comput. Vision*, vol. 31, no. 2-3, 1999.

[64] H. Wang, Q. Kemao, W. Gao, F. Lin, and H. S. Seah, "Fringe pattern denoising using coherence-enhancing diffusion," *Opt. Lett.*, vol. 34, pp. 1141–1143, Apr 2009.

[65] Q. Yu, "Spin filtering processes and automatic extraction of fringe centerlines in digitial interferometric patterns," *Appl. Opt.*, vol. 27, pp. 3782–3784, Sep 1988.

[66] Q. Yu, X. Liu, and K. Andresen, "New spin filters for interferometric fringe patterns and grating patterns," *Appl. Opt.*, vol. 33, pp. 3705–3711, Jun 1994.

[67] Q. Yu, X. Sun, X. Liu, and Z. Qiu, "Spin filtering with curve windows for interferometric fringe patterns," *Appl. Opt.*, vol. 41, pp. 2650–2654, May 2002.

[68] A. Shulev, I. Russev, and V. C. Sainov, "New automatic fft filtration method for phase maps and its application in speckle interferometry," in *Proc. SPIE*, vol. 4933, pp. 323–327, 2003.

[69] G. H. Kaufmann and G. E. Galizzi, "Speckle noise reduction in tv holography fringes using wavelet thresholding," *Opt. Eng.*, vol. 35, no. 1, pp. 9–14, 1996.

[70] A. Federico and G. H. Kaufmann, "Comparative study of wavelet thresholding methods for denoising electronic speckle pattern interferometry fringes," *Opt. Eng.*, vol. 40, no. 11, pp. 2598–2604, 2001.

[71] H. Wang and Q. Kemao, "Comparative analysis on some spatial-domain filters for fringe pattern denoising," *Appl. Opt.*, vol. 50, pp. 1687–1696, Apr 2011.

[72] Q. Kemao, "Applications of windowed fourier fringe analysis in optical measurement: a review," *Opt. Lasers Eng.*, vol. 66, pp. 67–73, 2015.

[73] M. Zhao and Q. Kemao, "Multicore implementation of the windowed fourier transform algorithms for fringe pattern analysis," *Appl. Opt.*, vol. 54, pp. 587–594, Jan 2015.

[74] M. Zhao and Q. Kemao, "A comparison study of denoising techniques in fringe pattern analysis," in *Proc. SPIE*, vol. 9302, pp. 930208–930208–6, 2015.

[75] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE T. Image Process.*, vol. 16, pp. 2080–2095, Aug 2007.

[76] M. Zhao and K. Qian, "WFF-BM3D: a hybrid denoising scheme for fringe patterns," in *Proc. SPIE*, vol. 9524, pp. 952423–952423–6, 2015.

[77] M. Zhao and Q. Kemao, "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," *Appl. Opt.*, vol. 53, no. 16, pp. 3492–3500, 2014.

[78] M. Zhao, H. Wang, and Q. Kemao, "Snake-assisted quality-guided phase unwrapping for discontinuous phase fields," *Appl. Opt.*, vol. 54, pp. 7462–7470, Aug 2015.

[79] Q. Kemao, W. Gao, and H. Wang, "Windowed fourier filtered and quality guided phase unwrapping algorithm: on locally high-order polynomial phase," *Appl. Opt.*, vol. 49, pp. 1075–1079, Mar 2010.

[80] Y. Fu, R. M. Groves, G. Pedrini, and W. Osten, "Kinematic and deformation parameter measurement by spatiotemporal analysis of an interferogram sequence," *Appl. Opt.*, vol. 46, pp. 8645–8655, Dec 2007.

[81] H. Shi, Y. Fu, C. Quan, C. J. Tay, and X. He, "Vibration measurement of a micro-structure by digital holographic microscopy," *Meas. Sci. Technol.*, vol. 20, no. 6, p. 065301, 2009.

[82] Y. Fu, M. Guo, and P. B. Phua, "Multipoint laser doppler vibrometry with single detector: principles, implementations, and signal analyses," *Appl. Opt.*, vol. 50, pp. 1280–1288, Apr 2011.

[83] W. Gao, Q. Kemao, H. Wang, F. Lin, and H. S. Seah, "Parallel computing for fringe pattern processing: A multicore cpu approach in MATLAB® environment," *Opt. Lasers Eng.*, vol. 47, no. 11, pp. 1286–1292, 2009.

[84] I. The MathWorks, "MATLAB parallel computing toolbox." http://www.mathworks.com/products/parallel-computing/.

[85] W. Gao, N. T. T. Huyen, H. S. Loi, and Q. Kemao, "Real-time 2d parallel windowed fourier transform for fringe pattern analysis using Graphics Processing Unit," *Opt. Express*, vol. 17, no. 25, pp. 23147–23152, 2009.

[86] W. Gao and K. Qian, "A simple method to accelerate fringe pattern analysis algorithms based on graphics processing unit and MATLAB," in *Proc. ISEM-ACEM-SEM-7th-ISEM'12-Taipei*, 2012.

[87] N. Corporation, "The NVIDIA CUDA fast fourier transform library." https://developer.nvidia.com/cufft/.

[88] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE T. Image Process.*, vol. 15, pp. 3736–3745, Dec 2006.

[89] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR),2005*, vol. 2, pp. 60–65 vol. 2, June 2005.

[90] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *IEEE 12th International Conference on Computer Vision, 2009*, pp. 2272–2279, Sept 2009.

[91] H. Burger, C. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with BM3D?," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012*, pp. 2392–2399, June 2012.

[92] J. A. Tropp, A. C. Gilbert, and M. J. Strauss, "Algorithms for simultaneous sparse approximation. part i: Greedy pursuit," *Signal Process.*, vol. 86, no. 3, pp. 572–588, 2006.

[93] P. Chatterjee and P. Milanfar, "Is denoising dead?," *IEEE T. Image Process.*, vol. 19, pp. 895–911, April 2010.

[94] R. Cusack, J. Huntley, and H. Goldrein, "Improved noise-immune phase-unwrapping algorithm," *Appl. Opt.*, vol. 34, pp. 781–789, Feb 1995.

[95] J. R. Buckland, J. Huntley, and S. Turner, "Unwrapping noisy phase maps by use of a minimum-cost-matching algorithm," *Appl. Opt.*, vol. 34, pp. 5100–5108, Aug 1995.

[96] B. Gutmann and H. Weber, "Phase unwrapping with the branch-cut method: Clustering of discontinuity sources and reverse simulated annealing," *Appl. Opt.*, vol. 38, pp. 5577–5593, Sep 1999.

[97] B. Gutmann and H. Weber, "Phase unwrapping with the branch-cut method: Role of phase-field direction," *Appl. Opt.*, vol. 39, pp. 4802–4816, Sep 2000.

[98] S. Karout, M. Gdeisat, D. Burton, and M. Lalor, "Residue vector, an approach to branch-cut placement in phase unwrapping: theoretical study," *Appl. Opt.*, vol. 46, no. 21, pp. 4712–4727, 2007.

[99] A. Asundi and W. Zhou, "Fast phase-unwrapping algorithm based on a Gray-Scale mask and Flood Fill," *Appl. Opt.*, vol. 37, pp. 5416–5420, Aug 1998.

[100] W. Li and X. Su, "Phase unwrapping algorithm based on phase fitting reliability in structured light projection," *Opt. Eng.*, vol. 41, p. 1365, 2002.

[101] Y. Lu, X. Wang, and G. He, "Phase unwrapping based on branch cut placing and reliability ordering," *Opt. Eng.*, vol. 44, p. 055601, 2005.

[102] S. Li, W. Chen, and X. Su, "Reliability-guided phase unwrapping in wavelet-transform profilometry," *Appl. Opt.*, vol. 47, no. 18, pp. 3369–3377, 2008.

[103] M. Ramji and K. Ramesh, "Adaptive quality guided phase unwrapping algorithm for whole-field digital photoelastic parameter estimation of complex models," *Strain*, vol. 46, no. 2, pp. 184–194, 2010.

[104] H. Zhong, J. Tang, S. Zhang, and M. Chen, "An improved quality-guided phase-unwrapping algorithm based on priority queue," *IEEE Geosci. Remote S.*, vol. 8, no. 2, pp. 364–368, 2011.

[105] H. Wang, J. Weaver, I. Perreard, M. Doyley, and K. Paulsen, "A three-dimensional quality-guided phase unwrapping method for MR elastography," *Phys. Med. Biol.*, vol. 56, p. 3935, 2011.

[106] D. Bone, "Fourier fringe analysis: the two-dimensional phase unwrapping problem," *Appl. Opt.*, vol. 30, no. 25, pp. 3627–3632, 1991.

[107] J. Quiroga, A. Gonzalez-Cano, and E.Bernabeu, "Phase-unwrapping algorithm based on an adaptive criterion," *Appl. Opt.*, vol. 34, no. 14, pp. 2560–2563, 1995.

[108] M. Pritt, "Phase unwrapping by means of multigrid techniques for interferometric SAR," *IEEE T. Geosci. Remote*, vol. 34, no. 3, pp. 728–738, 1996.

[109] C. Jakowatz, *Spotlight-mode synthetic aperture radar: a signal processing approach*. Springer, 1996.

[110] D. Ghiglia and L. Romero, "Robust two-dimensional weighted and unweighted phase unwrapping that uses fast transforms and iterative methods," *J. Opt. Soc. Am. A*, vol. 11, no. 1, pp. 107–117, 1994.

[111] G. Davidson and R. Bamler, "Multiresolution phase unwrapping for SAR interferometry," *IEEE T. Geosci. Remote*, vol. 37, no. 1, pp. 163–174, 1999.

[112] I. Lyuboshenko and H. Maître, "Phase unwrapping for interferometric synthetic aperture radar by use of Helmholtz equation eigenfunctions and the first Green's identity," *J. Opt. Soc. Am. A*, vol. 16, no. 2, pp. 378–395, 1999.

[113] D. Ghiglia and L. Romero, "Minimum L^p norm two-dimensional phase unwrapping," *J. Opt. Soc. Am. A*, vol. 13, no. 10, pp. 1999–2013, 1996.

[114] Y. Boykov and O. Veksler, "Graph cuts in vision and graphics: Theories and applications," in *Handbook of Mathematical Models in Computer Vision* (N. Paragios, Y. Chen, and O. Faugeras, eds.), pp. 79–96, Springer US, 2006.

[115] J. Bioucas-Dias and J. Leitao, "The zπm algorithm: a method for interferometric image reconstruction in sar/sas," *IEEE T. Image Process.*, vol. 11, no. 4, pp. 408–422, 2002.

[116] K. Lange, *Optimization*. Springer Verlag, 2004.

[117] J. Bioucas-Dias, V. Katkovnik, J. Astola, and K. Egiazarian, "Absolute phase estimation: adaptive local denoising and global unwrapping," *Appl. Opt.*, vol. 47, no. 29, pp. 5358–5369, 2008.

[118] G. Valadão and J. Bioucas-Dias, "CAPE: combinatorial absolute phase estimation," *J. Opt. Soc. Am. A*, vol. 26, no. 9, pp. 2093–2106, 2009.

[119] V. Katkovnik, K. Egiazarian, and J. Astola, *Local Approximation Techniques in Signal and Image Processing*. SPIE Publications, 2006.

[120] G. Nico, G. Palubinskas, and M. Datcu, "Bayesian approaches to phase unwrapping: theoretical study," *IEEE T. Signal Proces.*, vol. 48, no. 9, pp. 2545–2556, 2000.

[121] M. Servin, J. L. Marroquin, D. Malacara, and F. J. Cuevas, "Phase unwrapping with a regularized phase-tracking system," *Appl. Opt.*, vol. 37, pp. 1917–1923, Apr 1998.

[122] M. Rivera and J. Marroquin, "Half-quadratic cost functions for phase unwrapping," *Opt. Lett.*, vol. 29, no. 5, pp. 504–506, 2004.

[123] S. Stramaglia, G. Nico, G. Pasquariello, and L. Guerriero, "Phase-unwrapping method based on stochastic relaxation," in *Proc. SPIE*, vol. 3217, pp. 4–12, 1997.

[124] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vis.*, vol. 1, no. 4, pp. 321–331, 1988.

[125] J. P. Ivins and J. Porrill, "Everything you always wanted to know about snakes (but were afraid to ask)," tech. rep., AIVRU University of Sheffield, July 1993, revised June 1995 and March 2000.

[126] C. Xu and J. L. Prince, "Snakes, shapes, and gradient vector flow," *IEEE T. Image Process.*, vol. 7, no. 3, pp. 359–369, 1998.

[127] J. Yezzi, A., S. Kichenassamy, A. Kumar, P. Olver, and A. Tannenbaum, "A geometric snake model for segmentation of medical imagery," *IEEE T. Med. Imaging*, vol. 16, pp. 199–209, April 1997.

[128] D. J. Kang, "A fast and stable snake algorithm for medical images," *Pattern Recogn. Lett.*, vol. 20, no. 5, pp. 507 – 512, 1999.

[129] F. Liu, B. Zhao, P. K. Kijewski, L. Wang, and L. H. Schwartz, "Liver segmentation for ct images using gvf snake," *Med. Phys.*, vol. 32, no. 12, pp. 3699–3706, 2005.

[130] H. Ng, K. Foong, S. Ong, P. Goh, and W. Nowinski, "Medical image segmentation using feature-based gvf snake," in *Engineering in Medicine and Biology Society, EMBS 2007. 29th Annual International Conference of the IEEE*, pp. 800–803, Aug 2007.

[131] W. Feng, W. Gang, V. Prinet, and R. Kun, "Phase unwrapping based on active contour model," in *Proc. IEEE Int. Geosci.Remote Se. 2003*, vol. 7, pp. 4386–4388, IEEE, 2003.

[132] A. B. Suksmono, A. Handayani, and A. Hirose, "Snake in phase domain: A method for boundary detection of objects in phase images," in *Proceedings of International Joint Conference on Neural Networks*, pp. 481–485, IEEE, 2006.

[133] A. Federico and G. H. Kaufmann, "Phase retrieval in digital speckle pattern interferometry by application of two-dimensional active contours called snakes," *Appl. Opt.*, vol. 45, no. 9, pp. 1909–1916, 2006.

[134] Q. Kemao, "Windowed fourier transform for fringe pattern analysis." http://www.mathworks.com/matlabcentral/fileexchange/24852.

[135] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005.

[136] Intel, "Reference manual for intel math kernel library 11.2." https://software.intel.com/en-us/mkl_11.2_ref.

[137] L. Dagum and R. Enon, "OpenMP: an industry standard api for shared-memory programming," *Computational Science & Engineering, IEEE*, vol. 5, no. 1, pp. 46–55, 1998.

[138] C. Campbell and A. Miller, *Parallel Programming with Microsoft Visual C++: Design Patterns for Decomposition and Coordination on Multicore Architectures*. Microsoft Press, 2011.

[139] Intel, "Intel cilk plus documentation." https://software.intel.com/en-us/node/522579.

[140] Intel, "Intel threading building blocks documentation." https://software.intel.com/en-us/intel-tbb.

[141] P. Duhamel and M. Vetterli, "Fast fourier transforms: a tutorial review and a state of the art," *Signal Process.*, vol. 19, no. 4, pp. 259–299, 1990.

[142] Intel, "Intel's mkl benchmark results." https://software.intel.com/en-us/intel-mkl#pid-12780-836.

[143] FFTW, "Fftw benchmark results." http://www.fftw.org/speed.

[144] FFTW, "Fftw documentation version 3.3.4." http://www.fftw.org/#documentation.

[145] D. Baksheev, "Fft length and layout advisor." https://software.intel.com/en-us/articles/fft-length-and-layout-advisor.

[146] A. V. Oppenheim and R. W. Schafer, *Discrete-time Signal Processing*. Prentice Hall, 2 ed., 1999.

[147] D. R. Butenhof, *Programming with POSIX Threads*. Addison-Wesley Professional, 1997.

[148] P. Kegel, M. Schellmann, and S. Gorlatch, "Using openmp vs. threading building blocks for medical imaging on multi-cores," in *Euro-Par 2009 Parallel Processing*, pp. 654–665, Springer, 2009.

[149] S. B. Lippman, J. Lajoie, and B. E. Moo, *C++ Primer*. Addison-Wesley Professional, 5 ed., 2012.

[150] J. C. Munson and S. G. Elbaum, "Code churn: A measure for estimating the impact of code change," in *International Conference on Software Maintenance, 1998.*, pp. 24–31, IEEE, 1998.

[151] K. Dabov, A. Danieyan, and A. Foi, "BM3D demo software for image/video restoration and enhancement." http://www.cs.tut.fi/~foi/GCF-BM3D/index.html#ref_software.

[152] M. Lebrun, "An analysis and implementation of the BM3D image denoising method," *Image Processing On Line*, vol. 2, pp. 175–213, 2012.

[153] V. Katkovnik, K. Egiazarian, and J. Bioucas-Dias, "Phase imaging via sparse coding in the complex domain based on high-order svd and nonlocal BM3D techniques," in *IEEE International Conference on Image Processing (ICIP), 2014*, pp. 4587–4591, Oct 2014.

[154] Y. Lu, X.Wang, X. Zhong, G. He, Y. Liu, and D. Zheng, "A new quality map for quality-guided phase unwrapping," *Chin. Opt. Lett.*, vol. 2, pp. 698–700, Dec 2004.

[155] D. Bone, H. Bachor, and R. Sandeman, "Fringe-pattern analysis using a 2-d Fourier transform," *Appl. Opt.*, vol. 25, no. 10, pp. 1653–1660, 1986.

[156] X. Su and W. Chen, "Fourier transform profilometry: a review," *Opt. Lasers Eng.*, vol. 35, no. 5, pp. 263–284, 2001.

[157] Q. Kemao, "Windowed fourier transform for fringe pattern analysis," *Appl. Opt.*, vol. 43, pp. 2695–2702, May 2004.

[158] Q. Kemao, H. Wang, and W. Gao, "Windowed Fourier transform for fringe pattern analysis: theoretical analyses," *Appl. Opt.*, vol. 47, pp. 5408–5419, Oct 2008.

[159] Z. Wang and H. Ma, "Advanced continuous wavelet transform algorithm for digital interferogram analysis and processing," *Opt. Eng.*, vol. 45, p. 045601, 2006.

[160] S. Mallat, *A Wavelet Tour of Signal Processing*. Academic Press Inc, 2nd edition ed., 1999.

[161] V. Srinivasan, H. Liu, and M. Halioua, "Automated phase-measuring profilometry of 3-D diffuse objects," *Appl. Opt.*, vol. 23, pp. 3105–3108, Sep 1984.

[162] R. Sedgewick, *Algorithms in C: Parts 1-4, Fundamentals, Data Structures, Sorting, and Searching*. Addison-Wesley Longman Publishing Co., Inc, 1997.

[163] Y. Li and X. Su, "Fast algorithm for reliability-guided phase unwrapping," *Opto-Electronic Engineering*, 2005.

[164] Y. Li and Z. Hou. Personal communication, 2011.

[165] M. Zhao, L. Huang, and Q. Kemao, "Dynamic adjoin list for quality guided phase unwrapping," in *CIOMP-OSA Summer Session: Lasers and Their Applications*, p. Th19, Optical Society of America, 2011.

[166] R. C. Gonzalez and R. E. Woods, *Digital Image Processing, 3rd Edition*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.

[167] F. Li, Y. Zhan, D. Hu, and C. Ding, "A fast method for InSAR phase unwrapping based on quality guide," *Journal of Radars*, vol. 1, no. 2, pp. 196–202, 2012.

[168] J. Wesner. Personal communication, 2012. In an email to the authors, dated June 14, 2012, Dr. J. Wesner asked whether we had tested using a heap for QGPU.".

[169] Q. Kemao, "A simple phase unwrapping approach based on filtering by windowed fourier transform: the phase near edges," *Opt. Laser Technol.*, vol. 39, no. 7, pp. 1364–1369, 2007.

[170] H. Wang, Q. Kemao, and H. S. Seah, "Valid point detection in fringe projection profilometry," *Opt. Express*, vol. 23, no. 6, pp. 7535–7549, 2015.

[171] S. Dehaeck and P. Colinet, "Improving speed and precision of local frequency analysis using gaussian ridge interpolation for wavelet and windowed fourier ridge algorithms," *Opt. Lasers Eng.*, vol. 77, pp. 54 – 63, 2016.

[172] P. Tang, P. C. Yew, and C. Zhu, "A parallel linked list for shared-memory multiprocessors," in *Computer Software and Applications Conference, 1989. COMPSAC 89., Proceedings of the 13th Annual International*, pp. 130–135, Sep 1989.

[173] N. Deo and S. Prasad, "Parallel heap: An optimal parallel priority queue," *The Journal of Supercomputing*, vol. 6, no. 1, pp. 87–98, 1992.

[174] X. He, D. Agarwal, and S. Prasad, "Design and implementation of a parallel priority queue on many-core architectures," in *19th International Conference on High Performance Computing (HiPC), 2012*, pp. 1–10, Dec 2012.

Publications

1. Ming Zhao, Haixia Wang, and Qian Kemao, "Snake-assisted quality-guided phase unwrapping for discontinuous phase fields," Applied Optics 54, 7462-7470 (2015);

2. Ming Zhao, Qian Kemao, "WFF-BM3D: a hybrid denoising scheme for fringe patterns ", Proc. SPIE 9524, International Conference on Optical and Photonic Engineering (icOPEN 2015), (July 17, 2015);

3. Ming Zhao, Qian Kemao; "A comparison study of denoising techniques in fringe pattern analysis", Proc. SPIE 9302, International Conference on Experimental Mechanics 2014, (March 4, 2015);

4. Ming Zhao, Qian Kemao, "Multicore implementation of the windowed Fourier transform algorithms for fringe pattern analysis," Applied Optics 54, 587-594 (2015);

5. Ming Zhao, Qian Kemao, "Quality-guided phase unwrapping implementation: an improved indexed interwoven linked list," Applied Optics 53, 3492-3500 (2014);

6. Haixia Wang, Qian Kemao, Ronghua Liang, Huayin Wang, Ming Zhao and Xiaofei He, "Oriented boundary padding for iterative and oriented fringe pattern denoising techniques", Signal Processing 102, 112-121 (2015);

7. Ming Zhao, Qian Kemao, "Window size selection of windowed fourier ridges for phase discontinuity in quality-guided phase unwrapping", Proc. SPIE 8769, International Conference on Optics in Precision Engineering and Nanotechnology (icOPEN2013), (June 22, 2013);

8. Ming Zhao, Lei Huang, and Qian Kemao, "Dynamic Adjoin List for Quality Guided Phase Unwrapping," in International Summer Session: Lasers and Their Applications, (Optical Society of America, 2011), paper Th19.

9. Ming Zhao, Lei Huang, Qican Zhang, Xianyu Su, Anand Asundi, and Qian Kemao, "Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies," Applied Optics 50, 6214-6224 (2011).