# Two-Dimensional Phase Unwrapping

Salah Karout

A thesis submitted in partial fulfilment of the
requirements of Liverpool John Moores
University for the degree of Doctor of Philosophy

General Engineering Research Institute (GERI),
Liverpool John Moores University.

April 2007

## Abstract

Many applications that rely on phase data, such as: synthetic aperture radar (SAR), magnetic resonance imaging (MRI) and interferometry involve solving the two-dimensional phase unwrapping problem. The phase unwrapping problem has been tackled by a number of researchers who have attempted to solve it in many ways. This thesis examines the phase unwrapping problem from two perspectives. Firstly it develops two new techniques based upon the principles of Genetic Algorithms. Secondly it examines the reasons for failure of most of the common existing algorithms and proposes a new approach to ensuring the robustness of the phase unwrapping process. This new method can be used in conjunction of a number of algorithms including, but not limited to, the two Genetic Algorithm methods developed here.

Some research effort has been devoted to solving the phase unwrapping problem using artificial intelligence methods. Recent developments in artificial intelligence have led to the creation of the Hybrid Genetic Algorithm approach which has not previously been applied to the phase unwrapping problem. Two hybrid genetic algorithm methods for solving the two dimensional phase unwrapping problem are proposed and developed in this thesis. The performance of these two algorithms is subsequently compared with several existing methods of phase unwrapping.

The most robust existing phase unwrapping techniques use exhaustive computations and approximations, but these approaches contribute little towards understanding the cause of failure in the phase unwrapping process. This work undertakes a thorough investigation to the phase unwrapping problem especially with regard to the problem of residues. This investigation has identified a new feature in the wrapped phase data, which has been named the *residue-vector*. This residue-vector is generated by the presence of a residue, it has an orientation that points out towards the balancing residue of opposite polarity and it can be used to guide the manner in which branch-cuts are placed in phase unwrapping. Also, the residue-vector can be used for the determination of the weighting values used in different existing phase unwrapping methods such as minimum cost flow and least squares. In this work, the theoretical foundations of the residue-vector method are presented and a residue-vector extraction method is developed and implemented. This technique is then demonstrated both as an unwrapping tool and as an objective method for determining a quality map, using only the data in the wrapped phase map itself. Finally a general comparison is made between the residue-vector map and other existing quality map generation methods.

## Acknowledgement

It has been an honour and a privilege to be associated with ***Professor Michael Lalor*** and ***Professor David Burton*** for carrying out the research work towards my PhD degree. I have greatly benefited from their deep insight and expertise into the subject. With their professional guidance, invaluable advice, patience, constant support and encouragement, throughout the different stages of the project I was able to publish my work presented in this thesis in two international journal papers and to be recognized in the field of research by many researchers. Therefore I would like to take this opportunity to express my gratitude and sincere thanks to them.

Also, I acknowledge valuable time spent by ***Dr. Francis Lilley*** in helping me correct and edit my papers I have published. I would like to express my thanks and appreciation especially for his quick response for help.

I would like to thank my supervisor, ***Dr. Munther Gdeisat***, for his help, time, encouragement, advice and motivation that he provided me through the period of my PhD degree. Besides, learning his engineering techniques and understanding the project. Again, I would like to express my deep thanks and gratitude.

On a more personal note, I must thank my parent for their endless support throughout my studies. Without their constant assurance and assistance, completion of this project would have not been possible. Their enthusiasm, drive, strength of character, tenacity and determination have inspired me to carry out this research work. And as a sign of my love, gratitude and affection I dedicate this work to them.

I want to thank my wife for her support and motivation in the completion of my PhD degree.

Salah Karout

## Table of Contents

# List of Abbreviations and Symbols

| | |
|---|---|
| MRI | Magnetic Resonance Imaging |
| SAR | Synthetic Aperture Radar |
| HGA | Hybrid Genetic Algorithm |
| TSP | Travelling Salesman Problem |
| FFT | Fast Fourier Transform |
| DCT | Discrete Cosine Transform |
| PCG | Preconditioned Conjugate Gradient |
| BWT | Biorthogonal Wavelet Transform |
| MCF | Minimum Cost Flow |
| PU | Phase Unwrapping |
| AI | Artificial Intelligence |
| SA | Simulated Annealing |
| GA | Genetic Algorithm |
| NN | Neural Network |
| RSA | Reversed Simulated Annealing |
| MCM | Minimum Cost-Matching |
| BCP | Branch-Cut Problem |
| *ARACB* | All Residues and All Corresponding Boundaries Coding |
| *ARMCB* | All Residues and Minimum Corresponding Boundaries Coding |
| *RI* | Random Initialization |
| *NNRI* | Nearest Neighbour and Random Initialization |
| *NNR2OPTI* | Nearest Neighbour and Random 2-opt Initialization |
| SCX | Smallest Edge Crossover Operator |
| *LK* | Lin-Kernighan Algorithm |
| IFSAR | Interferometric Synthetic Aperture Radar |

| | |
|---|---|
| $P$ | Phase unwrapping integration path, |
| $\Phi(p_i)$ | The value of the wrapped phase at pixel $p_i$ in phase map, |
| $p_i$ | Pixel in phase map such that: $p_i \in \{P\}$, |
| $Z$ | The set of integer numbers, |
| $W[.]$ | Wrapping operator, where $-\pi \leq W[.] \leq +\pi$. |

| | |
|---|---|
| $\hat{\nabla}\Phi(p_i)$ | Wrapped phase gradient at pixel $p_i$ in phase map, |
| $p_o$ | Index of the starting pixel in phase unwrapping integration path, |
| $p_e$ | Index of the end pixel in phase unwrapping integration path, |
| $\Psi(p_i)$ | The value of the unwrapped phase at pixel $p_i$ in phase map, |
| $k(p_i)$ | An integer number at pixel $p_i$ in phase map, where $k(p_i) \in Z$, |
| $n$ | An integer number such that $n \in \{-1,0,1\}$, |
| $\min E$ | The total number of minimum errors (discontinuities) in the unwrapped phase map, |
| $\Phi_{i,j}$ | The value of the wrapped phase, |
| $\Psi_{i,j}$ | The estimated unwrapped phase, |
| $\nabla\Phi^x_{i,j}$ | Horizontal wrapped phase gradient in x direction, |
| $\nabla\Phi^y_{i,j}$ | Vertical wrapped phase gradients in y direction , |
| $k^x_{i,j}$ | Integer number of horizontal discontinuities, |
| $k^y_{i,j}$ | Integer number of vertical discontinuities, |
| $c^x_{i,j}$ | Horizontal weightings or cost functions, |
| $c^y_{i,j}$ | Vertical weightings or cost functions, |
| *Int(.)* | Rounds to the nearest integer, |
| $i$ | Horizontal index of a pixel location in the image, |
| $j$ | Vertical index of a pixel location in the image, |
| $N$ | Horizontal dimension of a phase map, |
| $M$ | Vertical dimension of a phase map, |
| $\varepsilon^2$ | Minimum discontinuity error in $L^2$-norm sense, |
| $Q$ | Matrix that perform the discrete Laplacian operation on the vector $\Psi$ as shown on the left hand side of Eq. (2.11), |
| $\Psi$ | Column vector containing the unwrapped phase values which is the solution, |
| $\rho$ | Column vector containing the discrete Laplacian operation on the wrapped phase differences as shownin Eq. (2.11), |
| $\min(a,b)$ | Minimum of $a$ and $b$, |
| $w^x_{i,j}$ | Horizontal weightings. |

| | |
|---|---|
| $w_{i,j}^{y}$ | Vertical weightings, |
| $W$ | Matrix of weight values of every pixel in the phase map, |
| $A^T$ | Transpose of a vector or matrix $A$, |
| $U(i,j)$ | Horizontal data dependent weights, |
| $V(i,j)$ | Vertical data dependent weights, |
| $p$ | Degree of the norm, |
| $\varepsilon$ | Degree at which the solution gradients match the measured gradients and $p \neq 2$, |
| $q_{m,n}$ | Quality value at the pixel *(m, n)*, |
| $k$ | Box size has odd number dimension, |
| $B_{m,n}$ | Measure of poor quality of pixel *(m, n)*, |
| $\overline{\hat{\nabla}\Phi_{i,j}^{x}}$ | Mean value of $\hat{\nabla}\Phi_{i,j}^{x}$ in $k \times k$ window, |
| $\overline{\hat{\nabla}\Phi_{i,j}^{x}}$ | Mean value of $\hat{\nabla}\Phi_{i,j}^{y}$ in $k \times k$ window, |
| *h(i, j)* | Horizontal second wrapped phase differences, |
| *v(i, j)* | Vertical second wrapped phase differences, |
| $w(i,j)$ | Generated weight at a pixel, |
| t | Distance from the centre pixel, |
| $I$ | Quality map in the $k \times k$ window, |
| $\bar{I}$ | Average value of $I$, |
| $\sigma_I$ | Standard deviation of $I$, |
| $A_0$ | Normalised constant, |
| $K$ | Damping factor, |
| NP-hard | Problem complexity class, |
| NP-complete | Problem complexity class, |
| $T_{start}$ | Initial starting temperature for simulated annealing algorithm, |
| $s_{ini}$ | Initial solution, |
| $s_i$ | Solution at iteration $i$, |
| *f(.)* | Solution evaluation function, |
| $E_i$ | Quality of the solution at iteration $i$, |
| $\Delta E_k$ | Difference in the quality of two successive solutions at iteration k of the same temperature level, |

| | |
|---|---|
| $r$ | A random number, where $r \in [0,1]$. |
| $\alpha$ | Temperature reduction function, |
| $T_{stop}$ | Simulated annealing termination Temperature. |
| Pop | Genetic algorithm population size, |
| $P_x$ | Probability of performing crossover, |
| $P_m$ | Probability of performing mutation, |
| **C** | Set of cities where $C = \{c_1, c_2, c_3 \ldots c_{Nc}\}$, |
| $N_c$ | Travel salesman problem number of cities, |
| $d(c_i, c_{i+1})$ | Distance between $c_i$ and $c_{i+1}$, |
| Total_Dist | Total distance covered by travelling salesman problem, |
| G | Graph **G** = (V, E, W), |
| V | Set of graph vertices, |
| E | Set of graph Edges, |
| W | Set of edge weights, |
| R | Set of residues in a phase map, where $R = \{r_1, r_2, r_3 \ldots r_N\}$, |
| B | Set of branch-cuts in a phase map, where $B = \{b_1, b_2, b_3 \ldots b_{N-1}\}$, |
| $L(r_i, r_j)$ | Cut length between to residues of opposite polarity $r_i$ and $r_j$, |
| $x_i^+$ | Horizontal index of positive gene *i*, |
| $x_i^-$ | Horizontal index of negative gene *i*, |
| $y_i^+$ | Vertical index of positive gene *i*, |
| $y_i^-$ | Vertical index of negative gene *i*, |
| $f_n$ | Fitness value of solution n, |
| $f_k$ | Fitness value of solution k, |
| e | Fitness equivalency Error, |
| *L* | Variable number varying between {2, *N*}, |
| $N_{ng}$ | Total number of negative genes, |
| $z_i$ | Surface data pixel to be fitted at pixel *i*, |
| $f(x_i, y_i)$ | Polynomial function at coordinate $x_i$ & $y_i$ in the x and y direction respectively, |
| $a_n$ | Polynomial coefficient at pixel *i*, |
| *A* | Matrix representing the polynomial coefficients. |
| *X* | Matrix representing the left side of Eq. (5.4). |
| *Z* | Matrix representing the right side of Eq. (5.4), |

| | |
|---|---|
| rand | A random number, |
| $\rho[.]$ | A rounding function, |
| *dx* | Derivative in the horizontal direction, |
| *dy* | Derivative in the vertical direction. |
| $\hat{\nabla}\Phi(b,p)$ | Gradient estimate in a 2×2 closed loop at pixel 'p' of branch-cut 'b' in the phase map, |
| $N_{bc}$ | Number of optimal branch-cuts in a wrapped phase map, |
| $N_{bp}$ | Number of optimal branch-cut pixels in a branch-cut 'b'. |
| $N_{rf}$ | Total number of residue-vector pixels overlapped by branch-cut in the phase map, |
| $N_{br}$ | Total number of branch-cut pixels in the phase map, |
| *min* | Minimum, |
| *max* | Maximum. |

# List of Figures

**Chapter 4**

# Chapter 1

# Introduction

# Chapter 1

## 1. Introduction

Many digital image processing techniques can be used to extract phase distributions from images generated by applications such as: optical and microwave interferometry, magnetic resonance imaging (MRI), synthetic aperture radar (SAR), synthetic aperture sonar, adaptive optics, seismic processing and aperture synthesis radio astronomy, *etc* [Huntley (2001)]. In many of the mentioned applications, the extracted phase relates to physical quantities such as surface topography in interferometry, wavefront distortion in adaptive optics, the degree of magnetic field inhomogeneity in the water/fat separation problem of magnetic resonance imaging or the relationship between the object phase and its bi-spectrum phase in astronomical imaging [Huntley (2001)]. Such techniques that rely on calculating the phase distribution suffer from one disadvantage; they employ the arctangent function in order to extract the phase. However, the arctangent operator produces results wrapped onto the range $-\pi$ to $+\pi$. Thus, in order to retrieve the contiguous form of the phase map, an unwrapping step has to be added to the phase retrieval process [Cusack (1995), Buckland (1995)].

Phase unwrapping is a technique used on wrapped phase images to remove the $2\pi$ discontinuities embedded within the phase map. It detects a $2\pi$ phase jump and adds or subtracts an integer offset of $2\pi$ to successive pixels following that phase jump based on a threshold mechanism, thus, retrieving the contiguous form of the phase map. A complete review of phase unwrapping is presented by Ghiglia *et al.* [Ghiglia and Pritt (1998)]. Phase unwrapping is not a straightforward step because of the possible presence of different kinds of noise and geometric layout of fringe lines. In the case of analysing interferometric fringes such noise sources (residues) are:

- Low signal-to-noise ratio of the fringes caused by electronic noise, speckle noise, or a low fringe modulation;
- Violation of Shannon's theorem;
- Object discontinuities [Gutmann (1999)].

On the other hand, geometric layout problems are:

- Intersection of fringe lines due to the use of some phase extraction algorithms causes the phase to be lost and in some algorithms this is classified as a potential residue (residue is an inconsistency in the wrapped phase data preventing straightforward unwrapping).
- Object shadow causes the loss of fringe lines, thus, loss of phase [Gutmann (1999)].

As a result, many phase unwrapping algorithms have been developed in an attempt to solve these problems. However, the variety of forms, shapes and densities of noise that might be found in real wrapped phase maps makes the problem of phase unwrapping complex and difficult to solve, even given the significant amount of research effort expended to date and the large number of existing phase unwrapping algorithms.

So, phase unwrapping algorithms were developed to overcome the pixels affected by noise. These algorithms calculate a gradient estimate by evaluating the difference between two consecutive pixels, then if the absolute value of the gradient estimate is greater than $\pi$, then an offset of + or $-2\pi$ is add to correct the phase [Fornaro *et al.* (1997)]. Because of the presence of corrupted regions within the image, this step cannot be performed unless an integration step is performed. Integration can be of two forms either local or global and it can be a combination of both as in the hybrid model [Fornaro *et al.* (1997)].

Local integration involves integrating the phase gradient of pixels in an image over a path starting from a certain point and going over all the pixels, in essence, unwrapping the image. Path independent unwrapping is obtained in the absence of residues that can arise from either noise or object discontinuities. The unwrapped result is independent of the unwrapping path; hence, the complete phase map is consistent. However, in the presence of corrupted pixels (residues), taking just any path is not possible anymore. Consequently, unwrapping becomes path dependent, where it has to manoeuvre between pixels choosing the best path to follow where the pixels are not corrupted by error. To overcome path dependence, many ways have been suggested and implemented. One of the first algorithms developed to overcome path dependence using local integration is Goldstein's branch cut algorithm [Goldstein (1988)]. This algorithm

finds corrupted pixels called residues characterized by having two kinds of polarities, positive and negative. Then, by growing the search area, it locates close opposite polarity residues and connects them by a branch cut, thus joining each pair with all the pixels that separate them. These branch cuts, then, will be avoided while unwrapping until unwrapping of non-corrupted pixels has been accomplished. Therefore, by identifying corrupted regions and excluding them from integration, unwrapping can follow any path independently achieving an unwrapped image. This algorithm is one of the fastest still. It gives accurate results where it can unwrap but it is limited to areas of moderate residue density. Moreover, a wrong choice of a single branch cut will cause errors to propagate over the whole image [Fornaro *et al.* (1997)]. Other algorithms based on local integrations are quality-guided [Ghiglia and Pritt (1998)], mask cut [Ghiglia and Pritt (1998)], Flynn's minimum discontinuity [Flynn (1997)], and phase unwrapping by means of genetic algorithm [Collaro *et al.* (1998)].

Global integration uses a different way for unwrapping images while still using the estimated phase gradient. The procedure is to minimize the least squared distance (the squared difference) between the estimated phase gradient and the true gradient of the unknown unwrapped phase [Fornaro *et al.* (1997)]. In this way, a smooth solution is achieved by the resultant minimization. That can be done by integrating over all the possible paths within the image not like local integration, which integrates over one single path, thus, spreading the error over the whole image. Like the previous method, this method also encounters a large number of errors once a corrupted region is present in the image. So, weighting measures in Weighted Least-Squared and $L^P$-norm algorithms were introduced to exclude corrupted regions [Ghiglia and Romero (1996)]. However, the success of algorithms using such a method relies on choosing the weights, which puts a huge load on the performance of the algorithm. One advantage of this method over algorithms that use residue branch cut technique is unwrapping residue-rich regions. This method actually uses noisy areas in the image while unwrapping, which was difficult or impossible to unwrap using the local methods. Thus, global integration forces unwrapping in these regions providing estimates, which are more accurate than the local methods.

These two methods have their own advantages and disadvantages, therefore, a hybrid model was introduced to use their merits and exclude their demerits. In essence,

researchers had found that there is a connection between methods using local and global integration. In other words, the least-squares solution (global method) at a given point is the average of all the solutions obtained by simple path-following (local method) radial paths from the point to the boundary [Ghiglia and Pritt (1998)]. It was also found that in some cases, the global and the local approaches were finding the same solution by minimizing the same error measure even though using completely different methods [Ghiglia and Pritt (1998)]. Moreover, another link exists; when Goldstein's branch cut algorithm minimizes the branch cut lengths in the local method, it is actually equivalent to minimizing the number of discontinuities in the global method [Ghiglia and Pritt (1998)]. Therefore, one way to achieve the merits of both methods is by using a joint method. This is accomplished by using the excellent unwrapping way of the path-following method when unwrapping non-corrupted regions. Whilst in the case of the corrupted regions, a global method is used to minimize the error by spreading it over the whole wrapped image until it is unwrapped. Then, unwrapped data of the corrupted region is added to the unwrapped data of the non-corrupted region, in essence, achieving a minimal error unwrapped phase image.

In this work, a new local integration phase unwrapping method using artificial intelligence in the form of a hybrid genetic algorithm (HGA) is used to optimize the unwrapped phase by minimizing the total cut length in a wrapped phase map globally before unwrapping. Phase unwrapping in this proposed algorithm is presented in the form of the travelling salesman problem (TSP) with the exception of the 'matching phenomena' instead of the 'tour' concept. Therefore, most of the profound advances in solving the TSP will be used in favour of the branch-cut phase unwrapping problem. The newly developed genetic algorithm is then tested on simulated and real wrapped phase maps to verify its characteristics and the results are compared with three existent branch-cut phase unwrapping algorithms, which are: simulated annealing [Cusack *et al.* (1995)], reverse simulated annealing [Gutmann (1999)] and minimum-cost matching algorithms [Buckland *et al.* (1995)]. This proposed algorithm is designed to unwrap wrapped phase maps with contiguous object features.

Also, a global integration phase unwrapping algorithm is proposed that uses a hybrid genetic algorithm to estimate the parameter coefficients of an $n^{th}$-order polynomial used to create the unwrapped phase solution that minimizes the $L^p$-norm error between the

gradient of the solution and the gradient of the wrapped phase map. This method is similar in concept to least-square and $L^p$-norm phase unwrapping methods developed by Ghiglia *et al.* [Ghiglia and Pritt (1998)] except it does not rely directly on the wrapped phase data to construct the unwrapped solution. However, it uses a polynomial to construct the unwrapped surface solution. The wrapped and the unwrapped phase maps are not totally independent of each other as the difference between them is used to optimize the solution. The only relation between the wrapped and the unwrapped phase maps is the $L^p$-norm error minimization. The other advantage of the proposed algorithm is that it generates noise-free unwrapped phase maps within a bandwidth or spatial extent as governed by the order of the polynomial and achieves a global smoothness constraint. This proposed algorithm is designed to unwrap wrapped phase maps with contiguous object features.

Many of the phase unwrapping methods developed until now rely on exhaustive search and approximation. There is no knowledge up till now on why phase algorithms still fail, even the most robust of them. The evident disadvantage of up to date advances in phase unwrapping is that any phase unwrapping algorithm cannot be used to unwrap any kind of wrapped phase. They are specific to certain applications with a possibility of failing especially in the case of very complex, noisy or under-sampled wrapped images. Also, phase unwrapping can take huge amount of processing time to achieve somewhat acceptable results. However, demands on the phase unwrapping process are getting larger and larger by the fast progress of technology, such demands are: larger images, real time applications, unwrapping in the 3rd Dimension video data, need to unwrap very complex surfaces, medical applications that do not leave a margin for error (MRI and Cancer X-ray Therapy), need to more details is larger, precision in results, and the need to limit the human interaction in the unwrapping process to a minimum.

Synthetic Aperture Radar phase unwrapping research was mostly driven into using exhaustive optimization but the performance and the quality of these techniques rely on weights which can be extracted from the data or provided as an additional data item from another source. These weights are more general than specific to the phase unwrapping problem. The weight factor has opened the research widely and brought phase unwrapping to square one. In essence, there exists no standard for defining the

weight factor which would result into acceptable results [Gens (2003)]. MRI phase unwrapping research has approached the problem by relying on certain features in the data but used also another set of data to assist in phase unwrapping. They also faced problems and failure of the unwrapping algorithm in certain cases which it is still ambiguous why it did fail. In metrology, phase unwrapping has faced great challenges especially when the data contains discontinuous and contiguous features at the same time. There exists no algorithm that can solve this problem even though less complex featured data rely extensively on weights to produce acceptable results.

To this present time, researchers and engineers in the industrial and medical field complain of the failure or unacceptable performance of the existing phase unwrapping methods. Researchers have even attempted to use a collection of existing phase unwrapping algorithms to benefit from the capabilities of each method corresponding to a variety of applications [Gens (2003)]. This desperate move shows the reliance of researchers on exhaustive computations and approximations but reveals little about the cause of failure of the phase unwrapping process.

Most phase unwrapping techniques attempt to identify or approximate the position of the residue ghost discontinuity lines (or branch-cuts) in the wrapped phase map to achieve a successful unwrapping. They rely on information such as quality maps, loops that are not neutral (these are set of pixels forming a closed loop where the summation of their corrected phase gradients does not equal to zero), areas that generate large discontinuities after a pre-unwrapping step, etc. However, there exists no exact knowledge on how to identify these ghost discontinuity lines except that they are located in the vicinities of high gradients. Unfortunately, residues and their ghost discontinuity lines are not the only sources that create high gradients. This work presents a discovery of the residue-vector lines embedded in the wrapped phase gradient maps in the horizontal and vertical directions. It was also found out that ghost discontinuity lines could take a form different from the high gradient information. This work demonstrates the residue-vector and proposes a residue-vector map that can be used as a weighting factor to many existing phase unwrapping algorithms. The performance of the residue-vector map is compared with other existing quality maps by implementing it as a weighting factor to Flynn's minimum discontinuity algorithm.

In essence, a thorough investigation to the residue problem in phase unwrapping has led to what we named "residue-vector", which are gradient information features embedded in the wrapped phase maps that give all the necessary guidance on how to unwrap an image successfully no matter the amount of noise, under-sampling, and the complex features in the wrapped phase map. Early investigation has produced more accurate branch-cut by means of the residue-vector as it localises to the actual discontinuities present in the wrapped data unlike existing techniques. This investigation also provides insights to why phase unwrapping methods succeed or fail. It was found that failure of phase unwrapping algorithms was due to a special type of residue-vector which is named a "zero-vector". When this "zero-vector" exists in the data the unwrapping process gets disrupted and as a result non-robust existing phase unwrapping techniques fail and the robust ones make large number of approximations to an extent that the smoothness of the data is lost and large sections of the results are deleted, *i.e.*, they loose large sections of good data. This discovery has pulled the weighting factor from being general to problem-specific because it provides the rules and standards onto how to extract weights and approach the phase unwrapping problem. The phase unwrapping technique that uses the proposed residue-vector information is likely to unwrap any wrapped phase map containing either contiguous or discontinuous object features or noise or under-sampling.

## 1.1. Synopsis of the Thesis

In chapter 2, an overview of the phase unwrapping problem is presented. In chapter 3, a brief study of some of the existing artificial intelligence algorithms is introduced with in depth presentation of the genetic algorithm and its hybrid form together with a review of up-to-date phase unwrapping methods using artificial intelligence. Chapter 4, presents a new phase unwrapping algorithm aided by artificial intelligence. This algorithm, based on the branch-cut method of solving the phase unwrapping problem, is introduced as a travelling salesman problem. A hybrid genetic algorithm which is well known in terms of its capability of solving the travelling salesman problem was developed to solve the branch-cut phase unwrapping problem. A detailed explanation of the hybrid genetic algorithm is presented that uses local and global methods in solving the problem. A demonstration of this algorithm's performances and results is presented in this chapter accompanied by a thorough discussion. Chapter 5 introduces a new global phase unwrapping method that also uses a hybrid genetic algorithm. This

algorithm approaches the phase unwrapping as a global minimization problem by surface fitting. The algorithm is tested on real and simulated wrapped phases to demonstrate its performance. On the other hand, the phase unwrapping problem was studied deeply and thoroughly especially in relation to the problem of residues. A new theory is introduced and explained in chapter 6. It is tested and verified in this chapter with simulated and real wrapped phase maps to prove its validity and capabilities. Finally, the work accomplished is concluded in chapter 7 with comments regarding the significance of the work accomplished together with suggestions for future work.

## 1.2. Contributions

The contributions of this research are summarized as following:

1. It demonstrates the equivalences between the branch-cut phase unwrapping problem and the travelling salesman problem.
2. It has designed and developed a local integration phase unwrapping algorithm using a hybrid genetic algorithm.
3. It has designed and developed a global integration phase unwrapping algorithm using a hybrid genetic algorithm.
4. It has investigated the residue, branch-cut and quality map problems.
5. The causes of failure or unacceptable results of phase unwrapping algorithms have been investigated.
6. It has led to the discovery of the residue-vector information embedded in the wrapped phase map.
7. It has proposed and implemented a residue-vector extraction method.
8. The performance of residue-vector maps in comparison with other quality maps has been investigated.
9. It presents a brief theory of the residue-vector and the methods of branch-cutting using residue-vector information.

# Chapter 2

# Phase Unwrapping

# Chapter 2

## 2. Phase Unwrapping

### 2.1. Introduction

Phase unwrapping has been a research area for more than two decades. Hundreds of papers have been published aimed at solving the phase unwrapping problem. Many phase unwrapping algorithms have been suggested and implemented. The reason for such interest in phase unwrapping is due to many applications in applied optics that require an unwrapping process. Many phase unwrapping algorithms has been developed only for data from a particular application. There is no universal phase unwrapping algorithm that can solve wrapped phase data from any application. Moreover, phase unwrapping algorithms are generally a trade off between accuracy of solution and computational requirements. Even so, even the most robust phase unwrapping algorithm cannot guarantee in giving successful or acceptable unwrapped results without a good set of weights. Existing quality or weighting maps are not problem-specific to phase unwrapping. They are general and do not specifically aid phase unwrapping all the time. Unfortunately, there is no standard objective method of defining weights that guarantee good phase unwrapping.

In this section, the phase unwrapping process will be defined and explained in detail. Two phase unwrapping methods will be explained in detail; the path-following method and the minimum-norm method. In essence, this chapter will explain the residue problem specifying how to locate residues in the phase map. Residues are local inconsistencies that prevent straight-forward unwrapping. It will present the branch-cut technique used to localize the residue effect, thus preventing it from affecting the rest of the pixels in the phase map by not allowing unwrapping paths to cross the localized residue areas. Moreover, an overview of the existing path-following, minimum-norm phase unwrapping and hybrid methods are presented and discussed. The quality or weight maps are defined and methods of extraction are presented. This chapter will help in understanding the material presented in later chapters.

### 2.2. Definition

Phase unwrapping (PU) is a technique used on wrapped phase images to remove the $2\pi$ discontinuities embedded within the phase map. It detects a $2\pi$ phase jump and adds or

subtracts an integer offset of $2\pi$ to successive pixels following that phase jump based on a threshold mechanism. The threshold mechanism states that if the phase difference between two successive pixels in a path $\{P\}$ is greater than $+\pi$, then, subtract a $2\pi$ offset to all successive pixels in the path. The phase difference can be calculated using Eq. (2.1):

$$\Delta\Phi(p_i) = \Phi(p_i) - \Phi(p_{i-1}) \tag{2.1}$$

where $\Phi(p_i)$ is the wrapped phase at pixel $p_i$ in phase map. However, if the phase difference is less than $-\pi$, add a $2\pi$ offset to all successive pixels in the path. Then, by locating all discontinuities in the wrapped phase map, the phase at every pixel will change by an integer $k$ multiples of $2\pi$ depending on the pixel position in the unwrapping path. This can be summarized by the wrapping operator in Eq. (2.2):

$$W[\Psi(p_i)] = \Psi(p_i) + 2\pi k(p_i) \qquad k(p_i) \in Z \tag{2.2}$$

where $-\pi \le W[\Psi(p_i)] \le +\pi$, $\Psi(p_i)$ is the unwrapped phase at pixel $p_i$ in phase map and $Z$ is the set of integer numbers. The wrapping operator $W[.]$ could be modified to specify the corrected gradient phase difference $\hat{\nabla}\Phi(p_i)$ between two successive pixels in the unwrapping path as is stated in Eq. (2.3):

$$\hat{\nabla}\Phi(p_i) = W[\Phi(p_i) - \Phi(p_{i-1})] \tag{2.3}$$

The phase unwrapping process is an integration process that could be performed by local pixel-to-pixel integration or by global integration or could even be performed in a hybrid form which employs both local and global integrations. The local integration technique could be termed the "path-following method". Whilst, the global integration technique is usually referred to as the "minimum-norm method". These integration techniques will be explained in the following sections.

## 2.3. Path-Following Methods

A simple local phase unwrapping method uses independent path integration between the starting point ($p_o$) and the end point ($p_e$) to retrieve the true unwrapped phase in the absence of residues in the wrapped phase map. It is a pixel-to-pixel integration technique that relies on local wrapped phase values along a chosen path to construct the

correct true phase refered to as unwrapped phase. This can be summarised in the discrete form using Eq. (2.4) of an $N$ pixels phase map:

$$\Psi(p_e) = \Phi(p_o) + \sum_{i=1}^{N} \hat{\nabla}\Phi(p_i) \tag{2.4}$$

Thus, by using Eq. (2.4), Phase Unwrapping will be capable of retrieving the contiguous form of the phase map [Gutmann (1999)].

However, this is not always the case. Because of the presence of noise or corrupted areas in the wrapped phase map, the path of integration becomes dependent. If Eq. (2.4) is used by itself to retrieve the unwrapped phase map, it may result in the addition or subtraction of incorrect multiples of $2\pi$, which will then propagate throughout the rest of the phase map. Restrictions must be used on the unwrapping path in the corrupted areas, which result in the path being dependent. To avoid this situation, corrupted areas, or residues, must be identified, balanced and isolated using barriers (branch-cuts) from the rest of the good pixels in the phase map. Once residues are isolated, phase unwrapping will take an independent path avoiding these branch-cuts, thus, retrieving the true phase.

### 2.3.1. Residues

Residues are defined to be local inconsistencies, which mark the beginning and end of $2\pi$ discontinuities. These residues are identified when the value of '$n$' in Eq. (2.5) is 1 or $-1$ in a $2 \times 2$ closed path, otherwise $n = 0$, which indicates that no residue exists.

$$\sum_{i}^{M} \hat{\nabla}\Phi(p_i) = 2\pi n \tag{2.5}$$

However, residues have two forms of discontinuity. One is a positive polarity when $n$ in Eq. (2.5) is +1; the other form is a negative polarity when $n$ is $-1$.

In the case of a residue is present; the result of Eq. (2.5) is always a +1 or -1 because the $2 \times 2$ closed path cannot encircle more than one residue.

A summary of how to identify residues in the wrapped phase map is shown in Fig. 2.2.

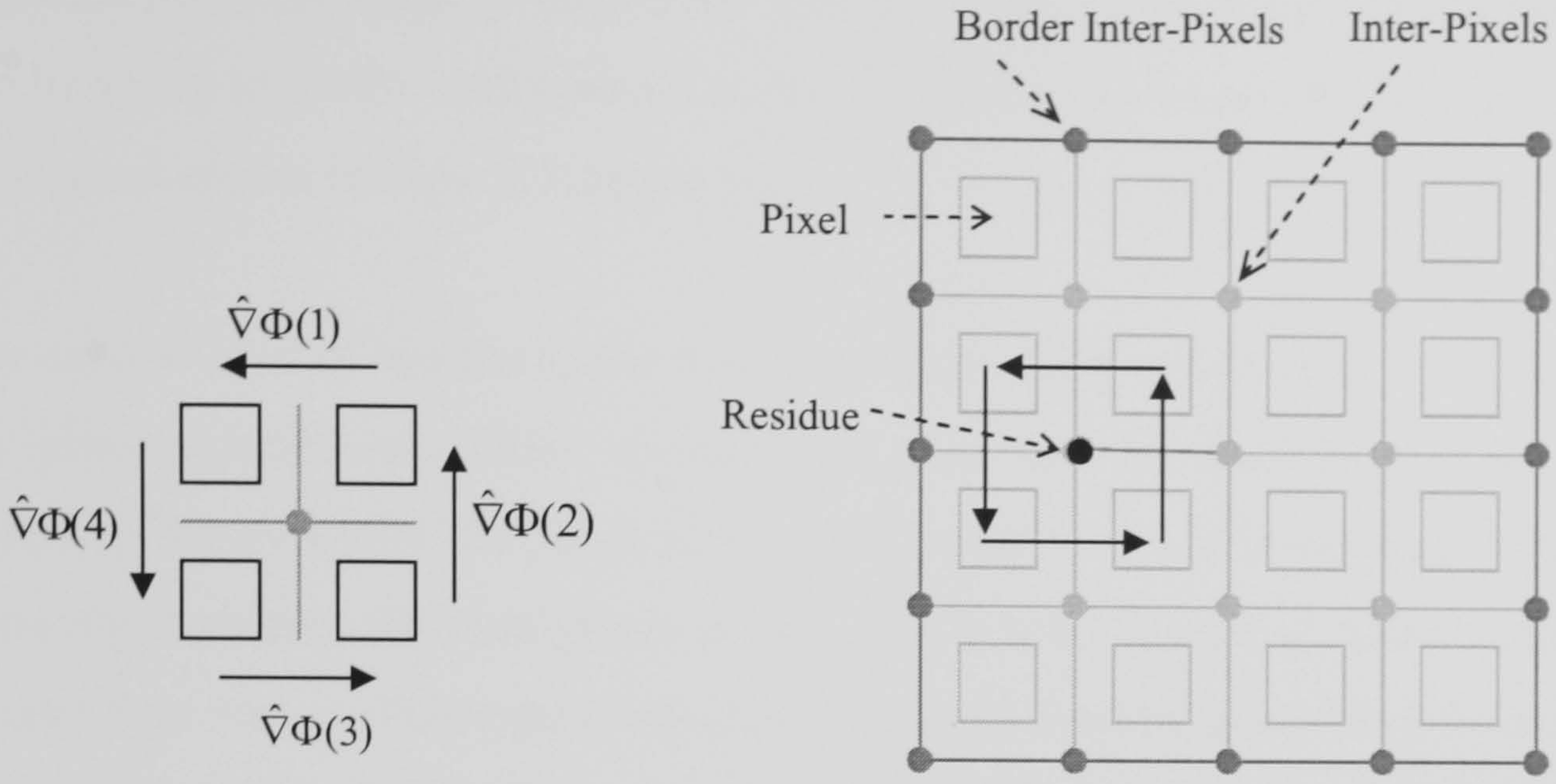



Fig. 2.1. (a) Visualizing residue calculation and (b) an inter-pixel network with a 2×2 closed loop and marked inter-pixel residue.

```
For (each pixel in the wrapped phase map)
        Sum the wrapped phase differences using Eq. (2.5) as shown in Fig. 2.1(a)
        & (b)
        If (the sum is 2π) then mark the inter-pixel as a positive residue
        If (the sum is -2π) then mark the inter-pixel as a negative residue
End
```

Fig. 2.2. A Pseudo-code for Residue identification.

### 2.3.2. Types of Residues

Many different kinds of residues may exist in a wrapped phase map caused by phase noise, spatial under-sampling of phase, object discontinuity, *etc*. Moreover, residues can be of two forms: dipole residues and monopole residues. Dipole residues are those that exist in pairs of two opposite polarity states or charges and monopoles that are single value residues for which no corresponding opposite-sign partner exists in a wrapped phase map [Gutmann (1999)].

One specific type of residue is the so-called phase noise generated dipole residue. These are caused by the random fluctuation of phase due to noise, which results in the wrapped phase gradient exceeding $\left|\hat{\nabla}\Phi\right| > \pi$. Each pair of dipole residues generated in

this case often lay close to each other (generally one pixel apart). This kind of residue can be easily identified and isolated in the phase map. An example of phase noise dipole residues is shown in Figs. 2.3(a) and (b).

The second kind of residue is the dipole residue which results from under-sampling of the phase distribution. These residues are generated by the violation of Shannon's sampling theory where the phase is not represented with sufficient spatial resolution to correctly represent the contiguous phase. This results in spatial under-sampling steps greater than $+\pi/-\pi$. This type of residue is characterized by generating dipoles that tend to be well separated when Shannon's law is broken, which makes them hard to identify as shown in the example shown in Fig. 2.3(c).

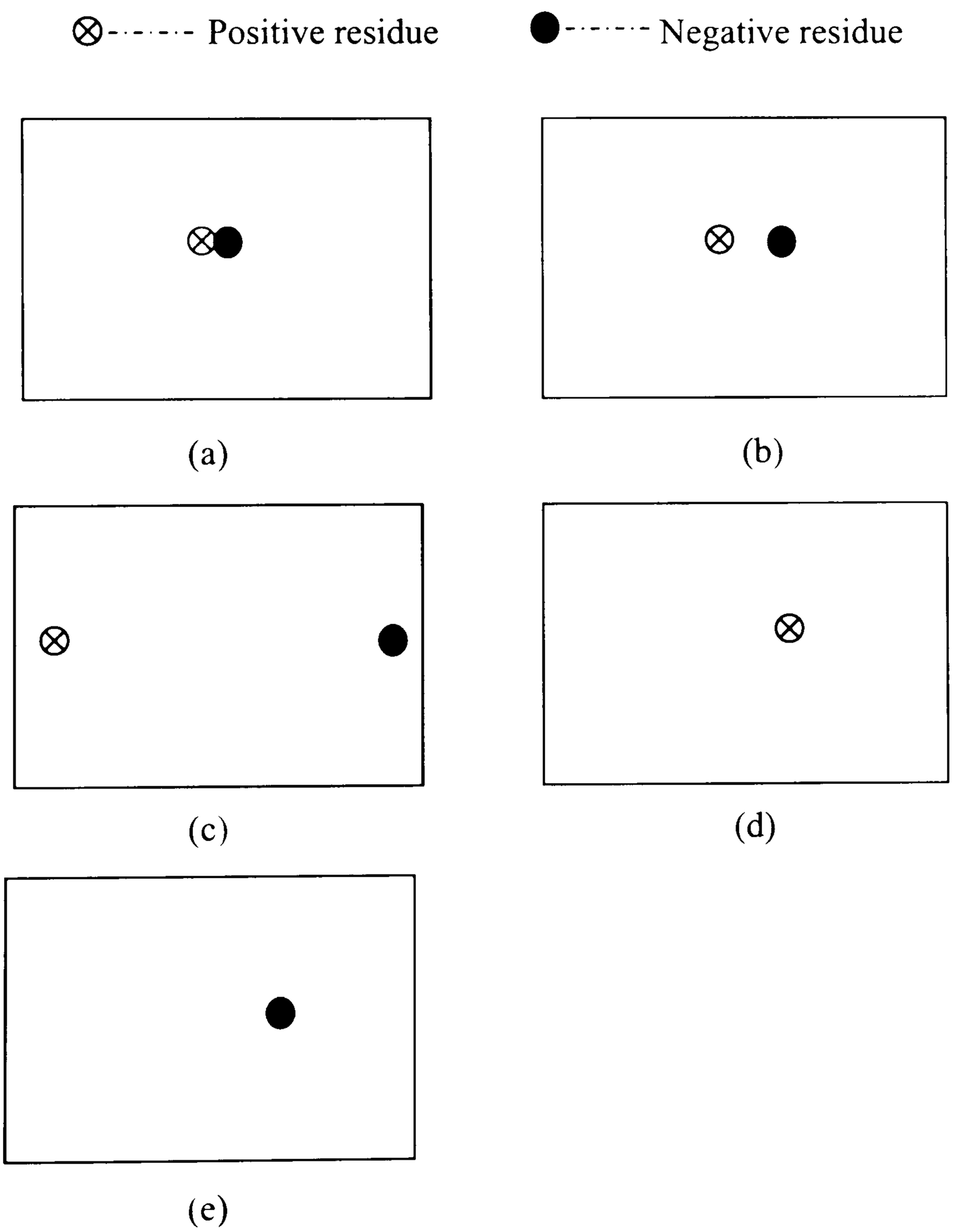



Fig. 2.3. (a) One pixel apart dipole residues generated by phase noise, (b) dipole residues generated by phase noise several pixels apart, (c) dipole residues generated by discontinuous objects and under-sampling have the tendency of lying far apart from each other and (d) & (e) are monopole residues.

Another kind of residue is dipole residues caused by object discontinuity. Sometimes wrapped phase maps contain objects that are discontinuous by nature such as holes, sharp edges, cracks or fluids of varying refractive index [Huntley (2001), Buckland *et al.* (1995)]. These discontinuous objects often generate dipole residues that lie on the discontinuity edges. The existence of these dipole residues depends on the discontinuity size of the object. If the object discontinuity exceeds $\pi$ when wrapped; then, this case will cause these residues. Object discontinuity dipole residues are characterized by generating dipoles that tend to be well separated depending on the nature of the discontinuity, which makes them hard to identify as shown in the example shown in Fig. 2.3(c).

The number of opposite polarity residues in the image is not always equal, due to the existence of monopoles. This may occur for two reasons, leading to two distinct types of monopoles; dipole-split monopoles and real monopoles. Dipole-split monopoles only occur close to the borders of the image, the simple fact of their border location causing their opposite polarity residue to lie outside the field of measurement. However, real monopoles may lie anywhere within the image, although they are usually found deeper inside the image than dipole-split monopoles, far away from the border regions of the image. Real monopoles generally occur in regions of high phase gradient or areas where the phase map contains true object discontinuities [Gutmann (1999)]. However; it is difficult to locate a monopole in a wrapped phase map with a large number of residues. It is assumed that any residue has a high probability of being a monopole if the boundary lies closer than twice the distance between the residue and its closest opposite polarity residue. Figs. 2.3(d) and (e) show two cases of monopole residues. The theory for approximating the number of monopoles in a wrapped phase map is presented by Gutmann [Gutmann (1999)].

### 2.3.3. Branch-Cut Concept for Phase Unwrapping

The branch-cut technique is a powerful method that has the potential of providing correct phase unwrapping without any solution approximations. This method relies on the fact that the summation of the gradient estimate of any closed path in the wrapped phase map must be equal to zero. This principle is defined in Eq. (2.6) for any closed loop path $\{P\}$ in the wrapped phase maps:

$$\sum_{i}^{M} \hat{\nabla}\Phi(p_i) = 0 \tag{2.6}$$

where $\Phi(p_i)$ is the wrapped phase value at pixel $p_i \in \{P\}$, $\hat{\nabla}\Phi(p_i)$ is the wrapped phase gradient and $M$ is the number of pixels in a path $\{P\}$. This principle is applicable to noise free wrapped phase maps. However, in the presence of noise in the wrapped phase map, this principle will be violated in areas, which mark the start and the end of a $2\pi$ discontinuity.

The branch-cut algorithm must ensure that the condition of Eq. (2.6) is not violated to achieve successful phase unwrapping. However, the condition of Eq. (2.6), in the case of perfect data without noise and in which the field variable described by the phase is everywhere contiguous, is satisfied at every pixel in the image. However in dealing with most real data we often find that Eq. (2.6) is not respected at all points. This can occur due to real discontinuities in the underlying field variable or because of noise in the phase. Whatever causes the violation of Eq. (2.6); it must be identified and localized.

Branch-cuts restrict the unwrapping path from passing through corrupted areas and achieve a balance between the phases of opposite residues to satisfy the condition of Eq. (2.6).

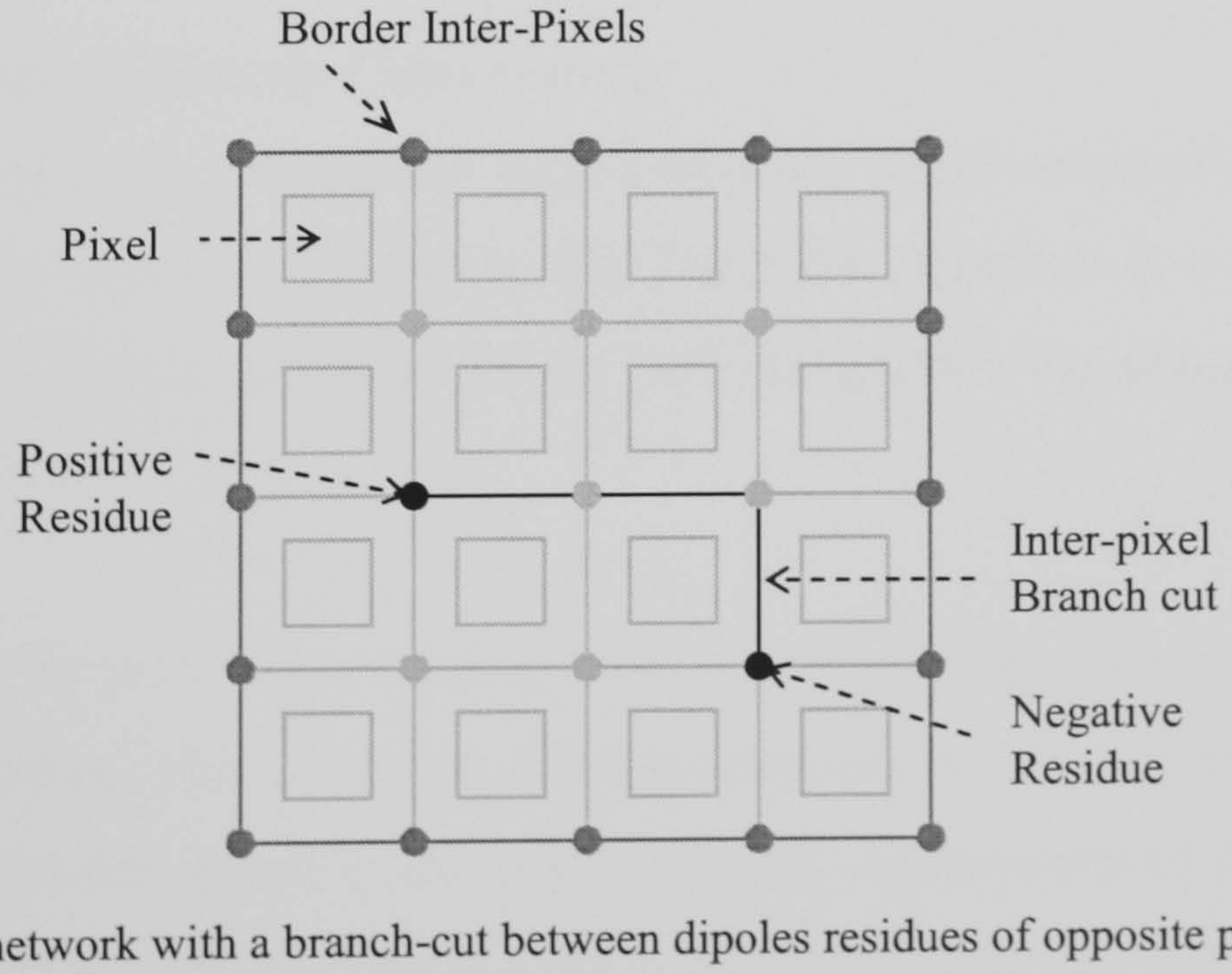



Fig. 2.4. An inter-pixel network with a branch-cut between dipoles residues of opposite polarity.

Once branch-cuts are placed between all residues in a phase map as shown in the example of Fig. 2.4, the unwrapping path can take any independent path in the phase map that respects the branch cut barriers and this will result in a correct phase

unwrapping which is consistent with the condition laid down in Eq. (2.6) as shown in Fig. 2.5(a).

However, the wrong phase unwrapping path is generated whenever the unwrapping path crosses the branch-cut barriers as shown in Fig. 2.5(b). This creates $2\pi$ discontinuities in the unwrapped phase map.

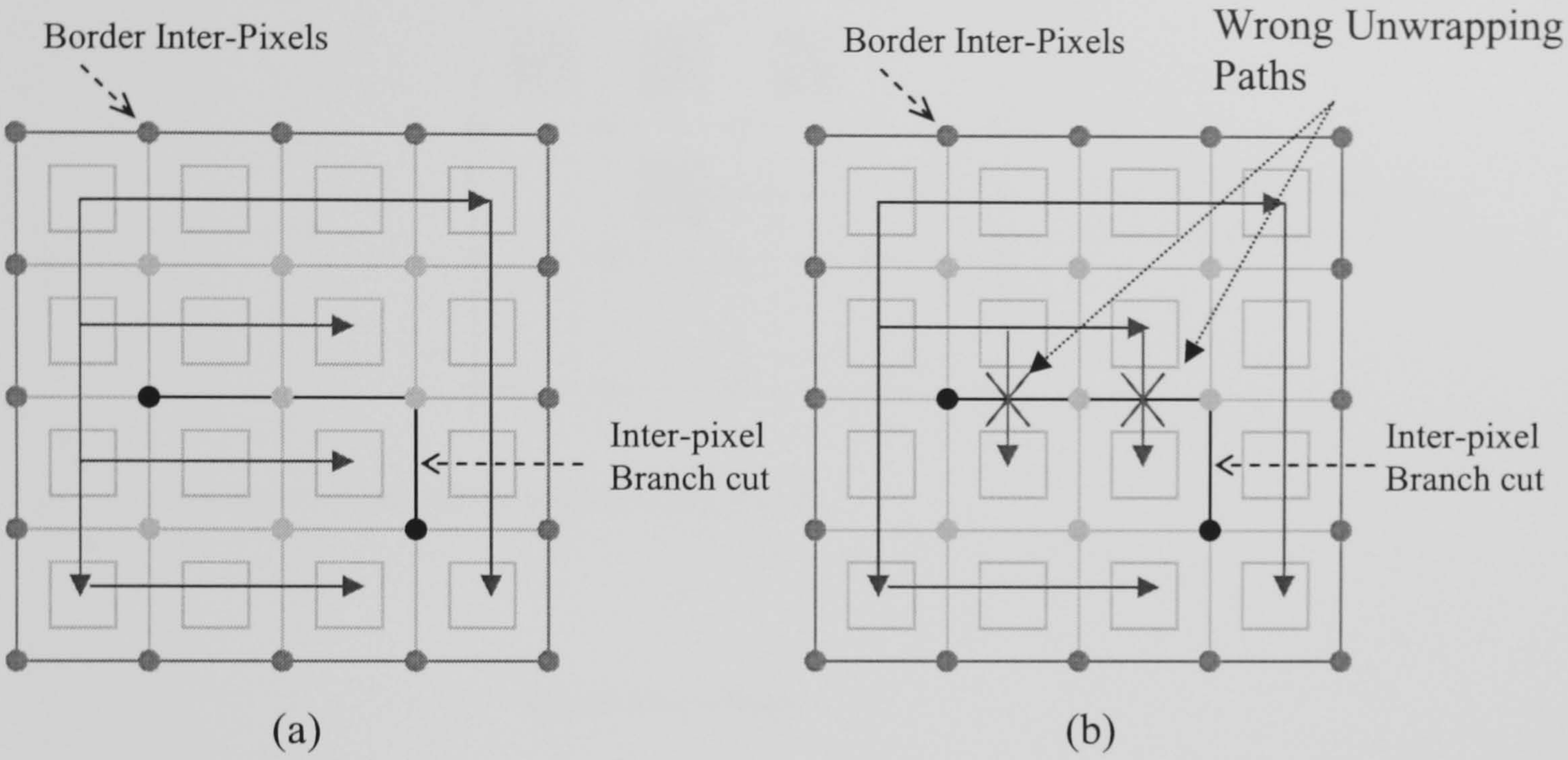



Fig. 2.5. (a) Correct phase unwrapping paths avoiding the branch-cuts (there is several combination of paths could be done on this figure but this is one of them) and (b) Wrong phase unwrapping paths cross over the branch-cuts.

### 2.3.4. Flood Fill Method in Phase Unwrapping

The path integration process required by path following phase unwrapping algorithms may performed by the flood-fill algorithm. The flood-fill algorithm is a general image processing algorithm that is modified to do path-integration for phase unwrapping [Ghiglia and Pritt (1998)].

This algorithm starts by selecting a starting pixel and its corresponding phase value is stored in a solution array. Then, the four neighbouring pixels are then unwrapped and their unwrapped values are stored in the solution array. An example of the unwrapped pixel and its four neighbouring pixels is shown in Fig. 2.6. These four unwrapped pixels are stored in a track list. Then, the algorithm proceeds by iteratively choosing a pixel from the track list and unwrapping the four neighbouring pixels, inserting their unwrapped values in the solution array and then inserting these pixels in the track list.

In this algorithm, the unwrapping path should not cross a branch-cut as shown in Fig. 2.5(b) and it must not unwrap unwrapped pixels [Ghiglia and Pritt (1998)].

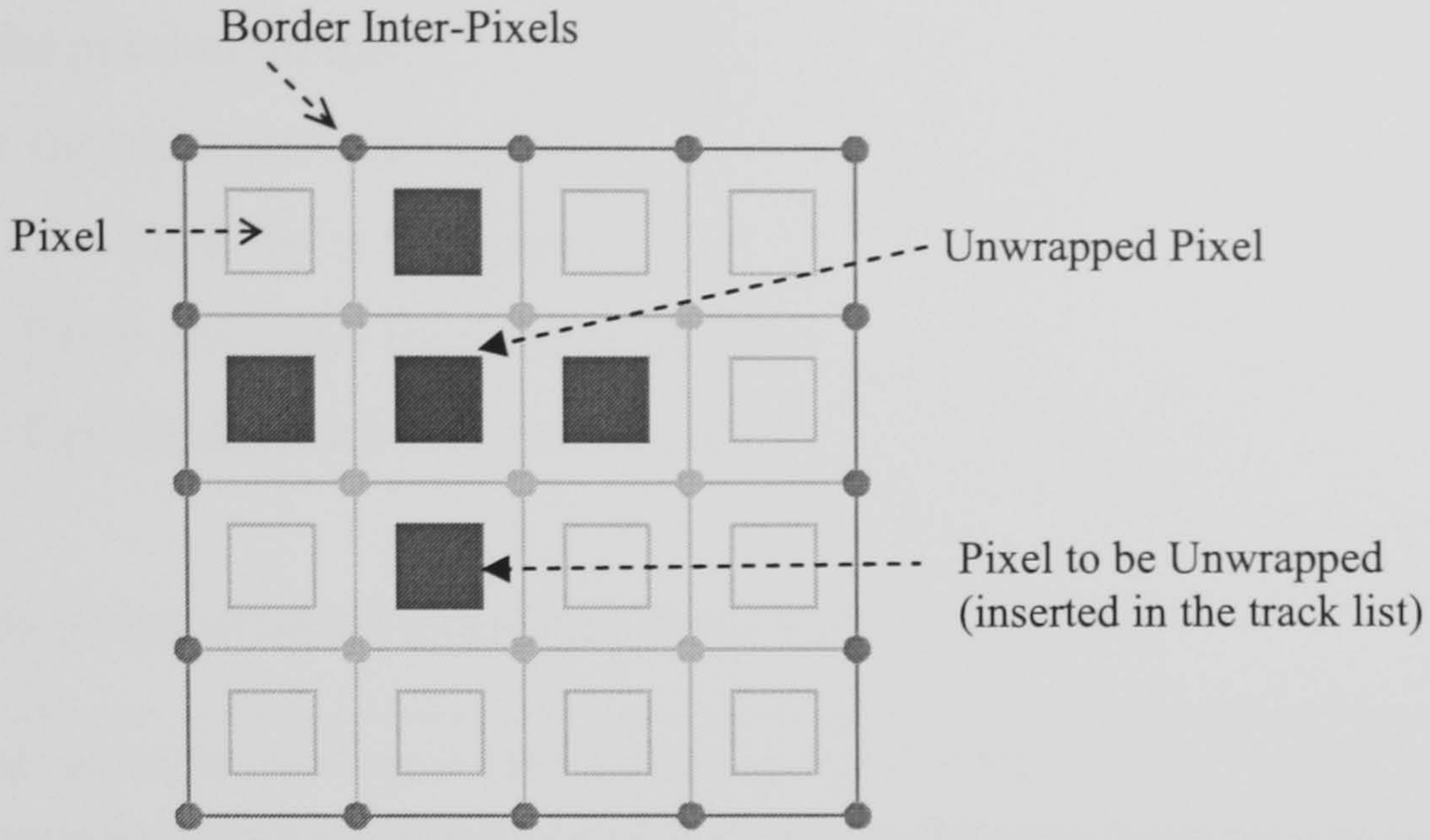



Fig. 2.6. Unwrapped pixel and its neighbouring pixels.

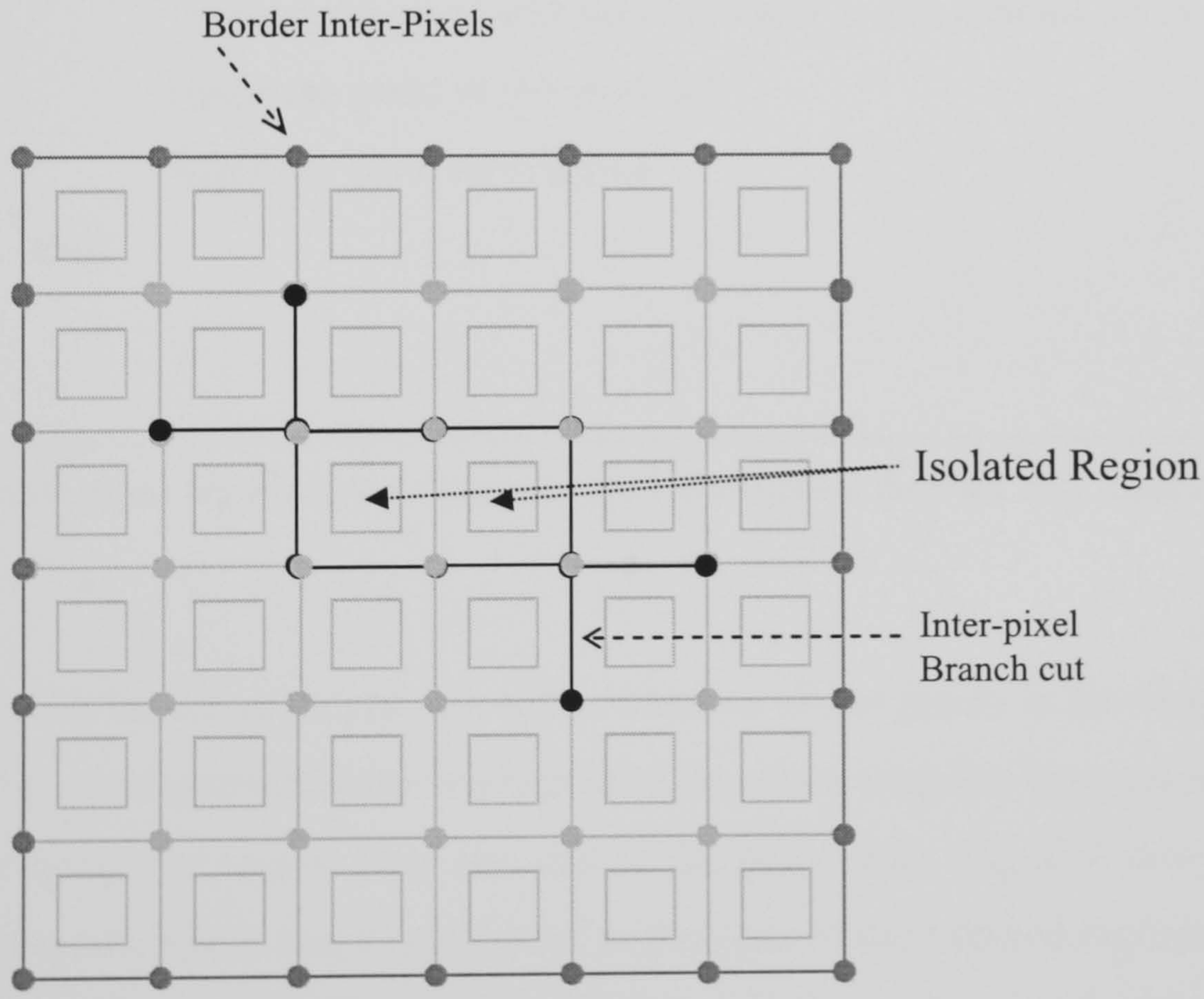



Fig. 2.7. Isolated regions created by encircling branch-cuts.

The Flood-fill algorithm is summarized in Figs. 2.8 and 2.9.

```
Repeat
        Select a starting pixel
        Store its phase value in the solution array
        Mark the pixel unwrapped
        Update the track list (see below)
        While ( the track list is not empty)
                Fetch any pixel from the track list
                Update the track list (see below)
        End
Until (all pixels in the wrapped phase map have been unwrapped)
```

Fig. 2.8. A Pseudo-code of the flood-fill algorithm [Ghiglia and Pritt (1998)].

```
For (each of the four neighboring pixels)
        If (the pixel is not on the track list and not unwrapped)
                If (the pixel is not crossing a branch-cut)
                        Unwrap the pixel and store its value in the solution array
                        Insert the pixel in the track list
                        Mark the pixel unwrapped
                End
        End
End
```

Fig. 2.9. A Pseudo-code of the algorithm that update the track list (Update the track list) [Ghiglia and Pritt (1998)].

In the case of the track list being empty, it is either because all the pixels in the wrapped phase map has been unwrapped or there is a region in the phase map has been encircled by branch-cuts isolating that region from the rest of the phase map. Fig. 2.7 shows an isolated region created by encircling branch-cuts. In the case of the isolated regions due to encircled branch-cuts, the flood-fill is repeated by choosing a starting pixel within the chosen region and repeating the previously mentioned process [Ghiglia and Pritt (1998)].

### 2.3.5. Branch-Cut Methods

Several methods have been developed to implement the branch-cut placement in a phase map such as: tree, dipole, mask quality guided and minimum cost flow (or cost guided minimum discontinuity) branch-cut placement methods.

The tree branch-cut placement method is one of the earliest branch-cut methods introduced by Goldstein *et al.* in 1988 who was first to propose the branch-cut technique[Goldstein *et al.* (1988)]. This method creates trees that connect a number of nearest neighbour residues where the net charge of every tree should be zero. Also, if there exists a tree which not neutral and is closer to the border than any neutralizing residue, then this tree is neutralized by connecting it to the nearest border pixel. This method is very fast but it tends to isolate areas with dense residues because branch-cuts in such areas often close on themselves. The weakness of this method is the lack of a weighting factor. A modification of this method is the minimum spanning tree method, which introduced a weighting factor to the tree method and also used a new minimum Sterner tree. This method was introduced by Chen and Zebker [Chen and Zebker (2000)]. It uses Dijkstra's shortest path algorithm for searching for the nearest charge to the tree. In this method, the cuts are associated with the phase differences, which guarantees that the tree does not close on itself [Chen and Zebker (2000)].

The dipole branch-cut method was first introduced by Huntley [Huntley (1989)]. This method uses the nearest neighbour heuristic search to find the nearest opposite polarity residue for every residue in the phase map. It connects the nearest possible pair of opposite polarity residues in a single branch-cut. It similarly does the same procedure to the rest of the residues until there exist no residue pairs to branch-cut. In the case of a residue having the border closer than any balancing residue; the residue is branch-cut with the nearest border pixel. One of the disadvantages of this branch-cut method is that it often ends up with very long branch-cuts. The method was improved by using more sophisticated search strategies such as: improved nearest neighbour, simulated annealing, minimum-cost matching, stable marriages and reverse simulated annealing that try to find the corresponding dipoles with the minimum total connection length [Cusack *et al.* (1995), Buckland *et al.* (1995), Gutmann (1999)]. An advantage of these dipole methods over the tree method is that they are less likely to create branch-cuts that isolate noisy regions in the phase map. The major disadvantage of the tree and the dipole branch-cut

methods is that they use straight line branch-cuts which leads to unrealistic discontinuities distorting the unwrapped phase map even though they attempt to balance the overall charge of resides in the unwrapped phase map.

A different approach to the branch-cut method was introduced by Flynn [Flynn (1996)]. This method uses the quality map to guide the placement of every branch-cut segment. It starts from a residue and follows the pixels with lowest quality until it finds another residue. It stops when it neutralizes the branch-cut growing tree. It repeats the same procedure until all the residues in the phase map are balanced by branch-cuts. Then, a post-processing procedure is used to thin the branch-cuts created in order to unwrap larger areas in the wrapped phase map. This method, like the tree map, tends to create isolated regions in the phase map. Also, the quality of the unwrapped phase map relies totally on the quality map used. In essence, different quality maps lead to different results and unfortunately no method has been found to identify the "best" quality map automatically in all cases [Ghiglia and Pritt (1998)]. A major disadvantage of this method is that it is not intelligent in the way the branch-cuts are placed this can lead to a somewhat unpredictable and random arrangement of branch cuts which is contrary to an intuitive view of where the branch cuts should lie [Ghiglia and Pritt (1998)].

The latest most efficient method of branch-cut placement is the minimum cost flow (MCF) or the cost guided minimum discontinuity method. This method was first introduced by Costantini [Costantini (1998)]. It uses a network of flows which defines the placement of every segment of the branch-cut guided by a cost factor and a global minimization strategy defined by Eqs. (2.7) and (2.8). This method minimizes the global sum of integer multiples of +/- $2\pi$ added to the original gradient estimate at every pixel before integration. Costs define where the branch-cuts are likely (low cost), or unlikely (high cost) to be placed [Costantini (1998)]. In essence, finding the best possible branch-cut placement aids the minimization criteria to achieve a global minimum value. Costs in this method are weighting factors, which if they are constant then the minimum cost flow minimizes the total length of branch-cuts. However, costs in this method are usually defined by either user defined weights or quality maps on the local interferogram such as: coherence, correlation coefficient, pseudo-correlation, phase gradient variance, maximum gradient, residue density, flatness or smoothness of the unwrapped phase.

This method is solved by network flow algorithms (a technique from graph theory and network programming) and has received wide interest in recent years. So far, this method is applied using a general package software. Thus, it suffers from a disadvantage in requiring huge computational and memory resources and also, the non-existence of the perfect or optimum distribution weighting factors which these algorithms need [Costantini (1998)].

$$\min E = \sum_{i}^{M}\sum_{j}^{N} c_{i,j}^{x}\left|k_{i,j}^{x}\right| + \sum_{i}^{M}\sum_{j}^{N} c_{i,j}^{y}\left|k_{i,j}^{y}\right| \tag{2.7}$$

Subject to

$$k_{i,j+1}^{y} - k_{i,j}^{y} - k_{i+1,j}^{x} + k_{i,j}^{x} = \frac{-1}{2\pi}\left[\Phi_{i,j+1}^{y} - \Phi_{i,j}^{y} - \Phi_{i+1,j}^{x} + \Phi_{i+1,j}^{x}\right] \tag{2.8}$$

Where min $E$ is the total number of minimum errors (discontinuities) in the unwrapped phase map,

$\Phi_{i,j}$ is the given wrapped phase,

$\Psi_{i,j}$ is the estimated unwrapped phase,

$\nabla\Phi_{i,j}^{x} = W[\Phi_{i+1,j} - \Phi_{i,j}]$ and $\nabla\Phi_{i,j}^{y} = W[\Phi_{i,j+1} - \Phi_{i,j}]$ are the wrapped phase gradients in x and y directions respectively where $W[\Phi_{i,j}] = \Phi_{i,j} + 2\pi k$ for $k \in Z$ and $-\pi \le W[\Phi_{i,j}] \le +\pi$,

$k_{i,j}^{x} = Int\left(\frac{\Psi_{i,j} - \Psi_{i-1,j} - \hat{\nabla}\Phi_{i,j}^{x}}{2\pi}\right)$ is the integer number of horizontal discontinuities,

$k_{i,j}^{y} = Int\left(\frac{\Psi_{i,j} - \Psi_{i,j-1} - \hat{\nabla}\Phi_{i,j}^{y}}{2\pi}\right)$ is the integer number of vertical discontinuities,

$c_{i,j}^{x}$ and $c_{i,j}^{y}$ are weighting or cost functions,

*Int(.)* rounds to the nearest integer,

$i$ and $j$ are indices of the pixel location in the image respectively,

$N$ and $M$ are the size of the wrapped phase map in x and y directions, respectively [Costantini (1998)].

A similar method to minimum cost flow is minimum discontinuity developed by Flynn [Flynn (1997)]. This method is a special case of the minimum cost flow algorithm. It relies on finding the unwrapped solution with a minimum number of discontinuities by tracing paths of discontinuities in the wrapped phase map then detecting paths that form loops where Eq. (2.7) is not satisfied. It then minimizes these discontinuities by adding a multiple of $2\pi$ to the phase values enclosed by the loops. This algorithm creates a mask that will act as a branch-cut mask which then requires a local path-integration method like the flood-fill method to unwrap the phase map. This method works with or without weighting factors or quality map. However, in the case of wrapped phase maps containing objects of discontinuous nature, this algorithm requires a quality map as it is the case for all algorithms for this kind of object. The lack of a good quality map or weighting factor means that this algorithm does not guarantee successful results all the time even though it is one of the most robust phase unwrapping methods. Another disadvantage of this algorithm is that it requires high memory and computational requirements.

Another phase unwrapping algorithm that does not use straight branch-cuts is pole-guided-cutline phase unwrapping [Chavez (2002)]. This algorithm tries to identify fringe-lines generated by the existence of two opposite polarity residues. It uses the wrapped phase map to extract these fringe-lines by generating a score map that is a form of differential phase gradient map that only highlights phase gradients caused by residues. The algorithm performs a region growing process; which starts from one residue and stops when a balancing residue is reached. The region growing relies on following all possible fringe-lines from one residue to another [Chavez (2002)]. Then, a post processing mask-thinning algorithm is used to thin the cuts [Ghiglia and Pritt (1998)].

A general summary of any branch-cut phase unwrapping algorithm techniques is presented in the following steps in Fig. 2.10:

- (Optional) Calculate the summation of the gradient estimate of a 2 × 2 closed loop at every pixel in the wrapped phase map using Eq. (2.5),
    - Identify positive and negative polarity residues using two conditions,
- (Optional) Extract a quality or a weighting factor from the wrapped phase map or insert user supplied weights,
- Use any branch-cut algorithm to:
    - Group residues into dipoles of opposite polarity residues or group residues into a tree form with a net charge of zero,
        - Place branch-cuts between the dipoles or tree branches,
- Create a branch-cut mask,
- Unwrap all the pixels in the wrapped phase map using flood-fill algorithm and avoid crossing branch cut barriers in the inter-pixel network.

Fig. 2.10. A general summary of any branch-cut phase unwrapping algorithm.

### 2.3.6. Quality Guided or Reliability Ordering Methods

Quality-guided methods are region growing methods that start from a pixel of highest quality or weighting factor then moves to unwrap all pixels in a descending quality order until they end with the lowest quality pixel [Ghiglia and Pritt (1998)]. These methods do not find residues or create branch-cuts. The success of the unwrapping process relies on how good the quality or weight map is.

Meanwhile, reliability ordering methods are similar to quality-guided methods but they rely on sorting all the pixels in the wrapped phase map with respect to their quality and then unwrapping the wrapped phase map [Herra'ez *et al.* (2002)]. These methods' success also relies on how good the quality or weight map is.

It is important to point out that if the quality mask (i.e., thresholded quality map) is used then these methods will be equivalent to the branch-cut placement methods. Moreover, if the quality map used is extracted from the phase gradient information in the wrapped phase map, then, these methods will be gradient-following methods [Ghiglia and Pritt (1998)]. In essence, these methods' results vary completely from one quality map to

another and the quality map itself defines the nature of these methods, for example. gradient-following methods [Lim, Xu and Huang (1995)] for maximum gradient quality map and maximum correlation path methods [Xu and Cumming (1996)] for correlation coefficient quality maps.

The advantage of these methods is that they have low memory overheads and they are relatively simple to implement. If the phase unwrapping application contains contiguous data and the application is not critical or important then these algorithm may be useful. The disadvantage of these methods is that they rely completely on the quality or weight map without any intelligent or problem specific approach to the phase unwrapping problem. In essence, most quality or weight maps are extracted from the wrapped phase map using general image processing techniques without any knowledge of the cause preventing straightforward unwrapping. These weights guide the way unwrapping is done on the wrapped phase map with a high probability of phase unwrapping failure especially at high levels of noise and the existence of under-sampling and object discontinuity features in the wrapped phase map. These quality-guided processes don't consider the residue-vector which this thesis shows to be critical in understanding the causes of failure of unwrapping. Moreover, a thresholded quality map can be equivalent to the branch cut method depending on the threshold.

## 2.4. Minimum Norm Methods

Minimum-norm methods are completely different than path-following methods. They are divided into three different types: unweighted least-squares, weighted least-squares and $L^p$-norm methods. These methods in general minimize up to a certain degree (least-square to the 2 degree order and $L^p$-norm raised to the p degree order) the difference between the gradients of the wrapped and the gradient of the unwrapped solution in both x and y direction. However, these methods do still indirectly deal with the residue problem because their solution is obtained by integrating over the residues to minimize the gradient differences [Ghiglia and Pritt (1998)]. Minimum-norm methods have the advantage that they are more noise tolerant and they achieve the global smoothness of the unwrapped solution. Minimum-norm approaches are mainly applied to single contiguous phase distributions (albeit with gaps and noise).

### 2.4.1. Unweighted Least-Squares Method

This method is the first minimum-norm method to be developed for phase unwrapping. Its mathematical formulation was developed by Hunt [Zebker and Lu (1998)] where he developed a matrix formulation suitable for general phase reconstruction problems. The unweighted least-squares method minimizes the distance between the phase gradient estimate (unwrapped phase) and the true gradient in the least-square sense as presented in Eq. (2.9) [Ghiglia and Romero (1994)]:

$$\varepsilon^2 = \sum_{i=0}^{M-2}\sum_{j=0}^{N-1}\left|\Psi_{i+1,j} - \Psi_{i,j} - \hat{\nabla}\Phi^x_{i,j}\right|^2 + \sum_{i=0}^{M-1}\sum_{j=0}^{N-2}\left|\Psi_{i,j+1} - \Psi_{i,j} - \hat{\nabla}\Phi^y_{i,j}\right|^2 \qquad (2.9)$$

Therefore, the solution that minimizes Eq. (2.9) is the unweighted least-squares solution. This equation can be further modified to the form presented in Eq. (2.10) [Ghiglia and Romero (1994)]:

$$\Psi_{i+1,j} + \Psi_{i-1,j} + \Psi_{i,j+1} + \Psi_{i,j-1} - 4\Psi_{i,j} = \hat{\nabla}\Phi^x_{i,j} - \hat{\nabla}\Phi^x_{i-1,j} + \hat{\nabla}\Phi^y_{i,j} - \hat{\nabla}\Phi^y_{i,j-1} \qquad (2.10)$$

Eq. (2.10) can be further modified to the following partial differential equations in the form presented in Eq.(2.11) [Ghiglia and Pritt (1998)]:

$$(\Psi_{i+1,j} - 2\Psi_{i,j} + \Psi_{i-1,j}) + (\Psi_{i,j+1} - 2\Psi_{i,j} + \Psi_{i,j-1}) = \rho_{i,j} \qquad (2.11)$$

where $\rho_{i,j} = (\hat{\nabla}\Phi^x_{i,j} - \hat{\nabla}\Phi^x_{i-1,j}) + (\hat{\nabla}\Phi^y_{i,j} - \hat{\nabla}\Phi^y_{i,j-1})$.

Eq. (2.11) is a discretization of Poisson's equation on a rectangular grid presented in Eq. (2.12) [Ghiglia and Romero (1994)]:

$$\frac{d^2}{dx^2}\Psi(x,y) + \frac{d^2}{dy^2}\Psi(x,y) = \rho(x,y) \qquad (2.12)$$

Eq. (2.11) can be transformed into matrix vector form to transform the problem into a linear sytem as in Eq. (2.13) [Ghiglia and Romero (1994)]:

$$\boldsymbol{Q\Psi} = \boldsymbol{\rho} \qquad (2.13)$$

Where $\boldsymbol{Q}$ is a matrix that performs the discrete Laplacian operation on the vector $\Psi$ as shown on the left hand side of Eq. (2.11). $\boldsymbol{\Psi}$ is a column vector containing the

unwrapped phase values which is the solution and $\rho$ is a column vector containing the discrete Laplacian operation on the wrapped phase differences as shownin Eq. (2.11) [Ghiglia and Romero (1994)].

The unweighted least-squares method is well defined mathematicly. However, these methods generate a very large number of linear equations to be solved equivalant to the total number of pixels in the phase map. There are many methods developed to solve the linear system in Eq. (2.13). In essence, such methods are directly based on the fast Fourier transform (FFT) or discrete cosine transform (DCT) or the unweighted mutigrid algorithm by Ghigilia [Ghiglia and Romero (1994)]. The FFT and DCT are non-iterative methods, however, the unweighted multigrid method applies Gauss-Seidel relaxation on different grid sizes where each solution serves as an initial solution to the next grid size solution.

Unfortunately, if the wrapped phase map contains residues, this will prevent successful phase unwrapping using the unweighted least-squares method. Because the unweighted least-square algorithms will tend to solve through the residues not around them. The error presented by residues is then spread all over the phase map leading to a solution that is not congruent to the wrapped phase data.

### 2.4.2. Weighted Least-Squares Method

The weighted least-squares method requires weights to achieve better results than the unweighted counterpart. These weights are user defined weights generated from quality-maps used to isolate corrupted areas with residues by masking them out of the wrapped phase data to diminish their effect on the unwrapped solution. The weighted least-squares method is a modification of the unweighted one where Eq. (2.9) is modified to the equation presented in Eq. (2.14) [Ghiglia and Romero (1994)]:

$$\varepsilon^2 = \sum_{i=0}^{M-2}\sum_{j=0}^{N-1} w_{i,j}^{x}\left|\Psi_{i+1,j} - \Psi_{i,j} - \hat{\nabla}\Phi_{i,j}^{x}\right|^2 + \sum_{i=0}^{M-1}\sum_{j=0}^{N-2} w_{i,j}^{y}\left|\Psi_{i,j+1} - \Psi_{i,j} - \hat{\nabla}\Phi_{i,j}^{y}\right|^2 \quad (2.14)$$

where weights are defined as following in Eq. (2.15) [Ghiglia and Romero (1994)]:

$$w_{i,j}^{x} = \min(w_{i+1,j}^{2}, w_{i,j}^{2}), \; w_{i,j}^{y} = \min(w_{i,j+1}^{2}, w_{i,j}^{2}) \tag{2.15}$$

where $0 \le w_{i,j}^{x}, w_{i,j}^{y} \le 1$.

The weights are squared because of the matrix operation ($W^T W$) of the weighted least-squares method. This can be seen when the matrix representation of the unweighted least-squares method presented in Eq. (2.13) is modified to the weighted form shown in Eq. (2.16) [Ghiglia and Romero (1994)]:

$$\boldsymbol{W}^T \boldsymbol{W} \boldsymbol{Q} \boldsymbol{\Psi} = \boldsymbol{W}^T \boldsymbol{W} \boldsymbol{\rho} \tag{2.16}$$

where $\boldsymbol{W}$ is a matrix of weight values of every pixel in the phase map.

Furthermore, due to the weighting matrix included to the linear equations in Eq. (2.13) that are used to solve the unweighted least-squares problem makes it impossible for FFT and DCT algorithms to solve directly the weighted least-squares problem. In essence, iterative methods have been developed to solve the weighted least-squares problem presented in Eq. (2.16). Such iterative methods are the Picard iteration method, the preconditioned conjugate gradient (PCG) and the weighted multigrid algorithms by Ghiglia *et al.*.[ Ghiglia and Romero (1994)]. The three algorithms require zero-weights or quality masks explained in section 2.6. However, the Picard iteration method has the disadvantage that it does not guarantee to always converge to a solution. PCG produces solutions that are not congruent (unwrapped data are not phase matched to the wrapped data, it is defined in chapter 5 section 5.7) to the wrapped phase map, so a postprocessing operation need to be accomplished similar to the phase matching technique in chapter 5. On the other hand, the weighted multigrid is superior to PCG in giving better results and converges much faster. Recently, a new method has been developed to solve the weighted least-squares problem. This method is the biorthogonal wavelet transform (BWT) which is used to transfer the original linear system in Eq. (2.16) into the new equivalent system in the wavelet domain with the low-frequency and high-frequency portion decomposed [Kim (2005)]. This method converges more quickly and even produces better results than the weighted multigrid method [Kim (2005)].

A drawback with these methods is if some residues are not masked out, they will cause the unwrapped phase to be severely corrupted depending on the density of the unmasked residues.

### 2.4.3. Minimum $L^p$-Norm Method

A more advance method developed by Ghiglia [Ghiglia and Romero (1996)] is minimum $L^p$-norm method which uses similar techniques to the two previous least-squares methods to solve the phase unwrapping problem. However, this method does not compute the minimum $L^2$-norm but the general minimum $L^p$-norm. In essence, by computing the minimum $L^p$-norm where $p \neq 2$; this method can generate data dependent weights unlike the weighted least-square method.

Eq. (2.11) can be further modified to the following non-linear partial differential equations in the form presented in Eq.(2.17) [Ghiglia and Romero (1996)]:

$$\begin{aligned}&(\Psi_{i+1,j} - \Psi_{i,j})U(i,j) - (\Psi_{i,j} - \Psi_{i-1,j})U(i-1,j) \\ &+ (\Psi_{i,j+1} - \Psi_{i,j})V(i,j) - (\Psi_{i,j} + \Psi_{i,j-1})V(i,j-1) = \rho_{i,j}\end{aligned} \tag{2.17}$$

Where $\rho_{i,j} = \hat{\nabla}\Phi^x_{i,j}U(i,j) - \hat{\nabla}\Phi^x_{i-1,j}U(i-1,j) + \hat{\nabla}\Phi^y_{i,j}V(i,j) + \hat{\nabla}\Phi^y_{i,j-1}V(i,j-1)$

$U(i,j)$ and $V(i,j)$ are data dependent weights defined in Eqs. (2.18) and (2.19).

The data-dependent weights can eliminate, iteratively, the presence of the residues in the unwrapped solution. These weights are defined in Eqs. (2.18) and (2.19) [Ghiglia and Romero (1996)]:

$$U(i,j) = \frac{\varepsilon}{\left|\Psi_{i+1,j} - \Psi_{i,j} - \nabla\Phi^x_{i,j}\right|^{2-p} + \varepsilon} \tag{2.18}$$

$$V(i,j) = \frac{\varepsilon}{\left|\Psi_{i,j+1} - \Psi_{i,j} - \nabla\Phi^y_{i,j}\right|^{2-p} + \varepsilon} \tag{2.19}$$

Where $\varepsilon$ is the degree at which the solution gradients match the measured gradients and $p \neq 2$.

This method can also optionally use weights provided by quality maps that can be combined with the data-dependant weights. Eq. (2.17) can be represented in a matrix form similar to the weighted least-squares Eq. (2.16). This method solves non-linear partial differential equations (Eq. (2.17)) using iterative schemes similar to those used in the weighted least-squares method [Ghiglia and Romero (1996)]. This method is more robust than the previous mentioned least-squares methods but, unforunately, it is more computationally intensive. Moreover, another disadvantage of this method that it only finds solutions that are locally minimum.

### 2.4.4. Other Global Integration Methods

A global integration two-dimensional phase unwrapping method using Green's formulation was developed by Fornaro *et al.* [Fornaro *et al.* (1997), Fornaro *et al.* (1996), Ghiglia and Romero (1994)]. This method uses Green's first identity in its two-dimensional form. It was proved that Green's formulation method is mathematically equivalent to the least-square solution. This method is a direct solution for a global phase unwrapping problem once the phase at the image boundary has been estimated which can be achieved by solving a Fredholm equation of the second kind. It uses FFT techniques within this method which makes it computationally efficient. A discrete version of the Green's formulation method for phase unwrapping was later introduced in 2002 [Marano *et al.* (2002)].

Another global integration method for phase unwrapping estimates the parameters of a polynomial that best-fit the suface (unwrapped) solution to the wrapped phase data using regression [Slocumb and Kitchen (1994)]. This method was introduced first on one-dimensional phase unwrapping and then was later developed to two-dimensional phase unwrapping. The early two-dimensional method estimates the parameters of fringe lines in order to unwrap the wrapped phase map [Lin *et al.* (1994)]. Then, a more complex method was later developed by Schwarz in 1999 which is mentioned in section 2.5.3 [Schwarz (2004)].

## 2.5 Hybrid Methods

The former two methods have their own advantages and disadvantages, therefore, a hybrid model was introduced to use their merits and exclude their demerits. In essence, research had shown that there is a connection between methods using local and global

integration. In other words, the least-squares solution (global method) at a given point is the average of all the solutions obtained by simple path-following (local method) radial paths from the point to the boundary [Ghiglia and Pritt (1998)].

It was also found that in some cases, the global and the local ones were finding the same solution by minimizing the same error measure even though using completely different methods.

Moreover, another link is when Goldstein's branch cut algorithm minimizes the branch cut lengths in the local method; it is actually equivalent to minimizing the number of discontinuities in the global method [Ghiglia and Pritt (1998)]. Therefore, one way to achieve the merits of both methods is by using a joint method. Several hybrid methods have been suggested; however, in this section only two hybrid algorithms will be presented which are the synthesis algorithm and the hybrid phase unwrapping with overlapping windows.

### 2.5.1. Synthesis Algorithm

Weighted least-square algorithms suffer from a drawback, if some residues are not masked out, they will cause the unwrapped phase to be severely corrupted depending on the density of the unmasked residues. Hence, Zebker [Zebker and Lu (1998)] proposed a hybrid algorithm that uses a local path-following branch-cut algorithm to generate a mask that contains branch-cuts that balance all the residues in the wrapped phase map. This mask is zero-weighted and also a zero-weighted quality map was added to this mask. In this way, all residues are balanced and masked out and also all corrupted regions are masked out. Then, this mask is provided as zero-weights to a global weighted least-squares algorithm. In essence, better results were achieved than the previous weightings of the weighted least-square algorithm [Zebker and Lu (1998)].

However, Zebker had another problem even though with his synthesis algorithm all residues are balanced and masked out the algorithm will generate errors if the branch-cuts defining the zero weights are placed in error [Zebker and Lu (1998)]. In other words, if the branch-cuts were placed incorrectly even though the branch-cuts balance all the residues in the wrapped phase map; it will create $2\pi$ phase error in the unwrapped phase map. As result of the weighted least-squares solution, this phase error will be

smoothed out all over the unwrapped phase map, thus, spreading the error over the whole unwrapped phase map.

Zebker made the following statement about this phase unwrapping problem: "*Unfortunately, no completely reliable algorithm for placing the weights has yet been demonstrated. Simply balancing the number of positive and negative residues along the cuts is an ambiguous method of cut identification. The search for improved methods of cut definition will have to continue if algorithms such as those presented here are to be assuredly effective*" [Zebker and Lu (1998)].

### 2.5.2. Hybrid Phase Unwrapping with Overlapping Windows

This hybrid algorithm uses both local and global integration methods. This algorithm divides the wrapped phase map into overlapped windows. The algorithm uses a polynomial approximation to estimate the unwrapped phase by using complex parabolic polynomials.

The polynomial approximation is considered the main method of phase estimation which is used to process every window in the phase map. If the quality of the results in a particular window is poor especially in the areas of discontinuity; a second method is performed which utilizes a two-dimensional discrete Fourier transformation [Schwarz (2004)]. An example of the regions where the Fourier transform method is likely to be used by this algorithm is shown in Fig. 2.11.

The polynomial approximation method is used first because it is faster and uses less memory than the Fourier transform method but the second method guarantees to converge at all times. Also, on every window, a local integration method is performed using the minimum-cost-matching branch-cut algorithm developed by Huntley [Buckland *et al.* (1995), Collaro *et al.* (1997), Schwarz (2004)].

Then, after processing all the windows in the wrapped phase map; these unwrapped overlapping windows are joined globally using a region growing strategy followed by a fast global spline approximation used to match all the overlapping windows in one contiguous smooth result. This algorithm also uses phase matching or congruency because of the use of the global integration method in order to force the unwrapped

phase map to be congruent to the wrapped phase map. This is a good algorithm but it still suffers from the same problem as the synthesis algorithm which is the problem of the branch-cut placement [Schwarz (2004)].

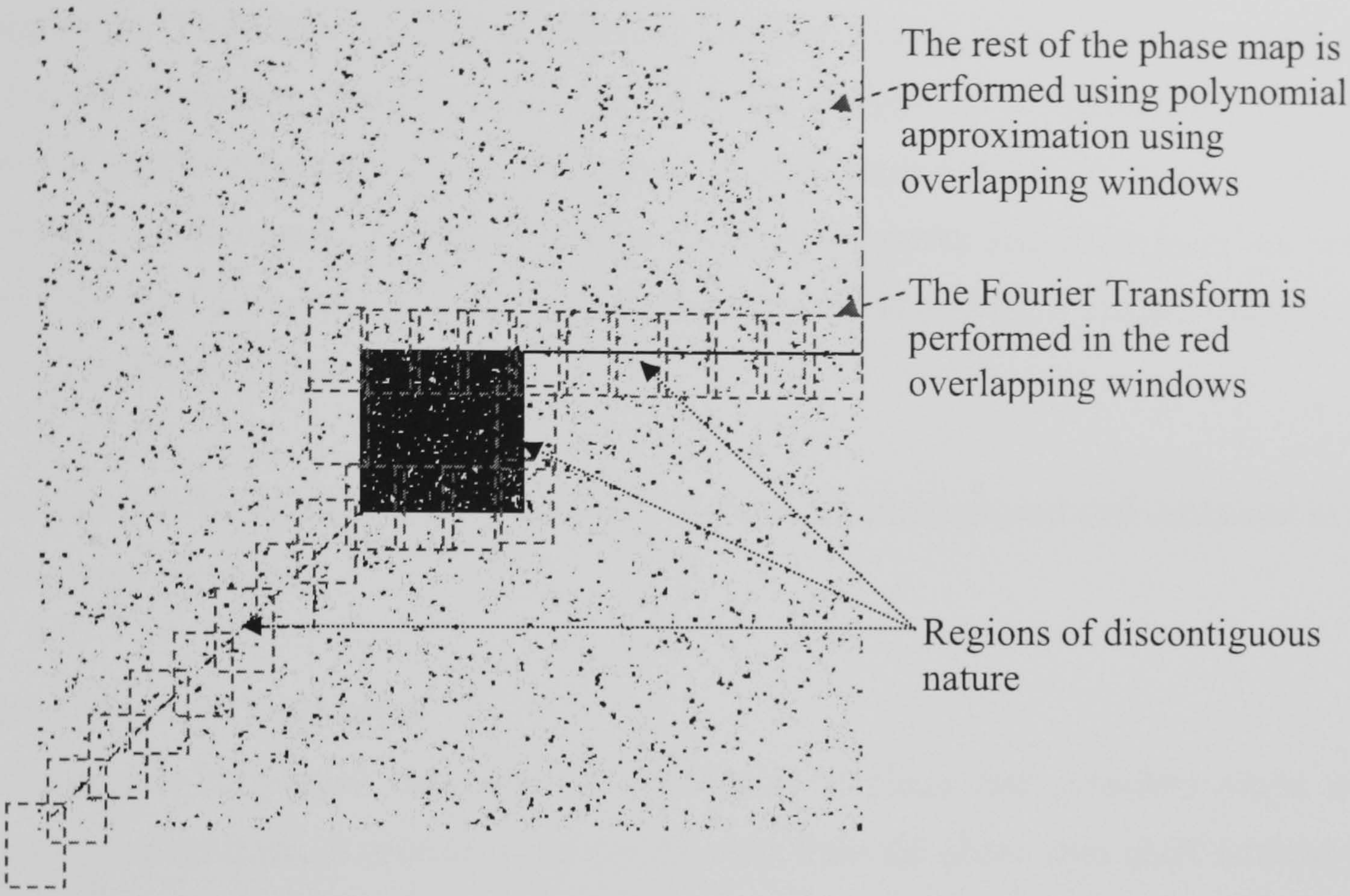



Fig. 2.11. A quality map with overlapping windows in regions of discontinuous nature marking where the Fourier Transform method is likely to be used by the hybrid algorithm.

## 2.6. Quality Maps or Weighting Factors and Masks

In a wrapped phase map, there exists information other than phase derivates (gradients) and residues, such information can be extracted from the wrapped phase by means of general image processing techniques adapted for the phase unwrapping problem. This additional information is quality maps or weighting factors. This information provides a weight or measure of quality of how good a particular pixel is in the wrapped phase map. In essence, a quality map is a map of all the quality values for all the pixels in a phase map. Quality map is an essential factor in generating good unwrapped solutions and the performance of the unwrapping algorithms rely on how good the quality map is or the weight extraction method is. Thresholding the quality map will provide a new form of data called a quality mask. Quality masks are masks that contain binary data, for instance a zero in the quality mask represents low quality data or pixel and a one

represents a high quality data or pixel. These zero-weight masks can be used to exclude certain regions from the unwrapping process.

Moreover, researchers have recognized the importance of quality maps in phase unwrapping, especially, after the introduction of the minimum cost flow network algorithm. They have developed different kinds of quality maps or weight extraction methods. However, there exists no standard that defines the weighting factors which would lead to acceptable phase unwrapping results [Gens (2003)]. In essence, different existing quality maps generate different unwrapped results and there is no universal procedure of generating a unique quality map that could provide a unique solution that is successful in phase unwrapping.

Several existing quality or weight extraction methods are discussed and explained in the following sections:

### 2.6.1. Pseudo-Correlation

The pseudo-correlation measures the correlation of phase and considers phase with uniform magnitude. It estimates the quality map from the phase data itself without the need for an extra set of data.

The 2D pseudo-correlation quality map defines the goodness of each pixel using the Equation:

$$q_{m,n} = \frac{\sqrt{\left[\sum \cos(\Phi_{i,j})\right]^2 + \left[\sum \sin(\Phi_{i,j})\right]^2}}{k^2} \tag{2.20}$$

Where $q_{m,n}$ is the quality of the pixel $(m,n)$,

$\Phi_{m,n}$ is the wrapped phase value of the pixel $(m, n)$,

$k$ is the window size (odd number).

The quality of every pixel in the wrapped phase map is calculated in the $k \times k$ neighbourhood of each pixel $(m, n)$ in the phase map. In essence, a $k \times k$ window is centred at pixel $(m, n)$. Fig. 2.12 shows a 3×3 window centred at pixel $(m, n)$.

The summation in Eq. (2.20) is carried out over the pixels in the window.

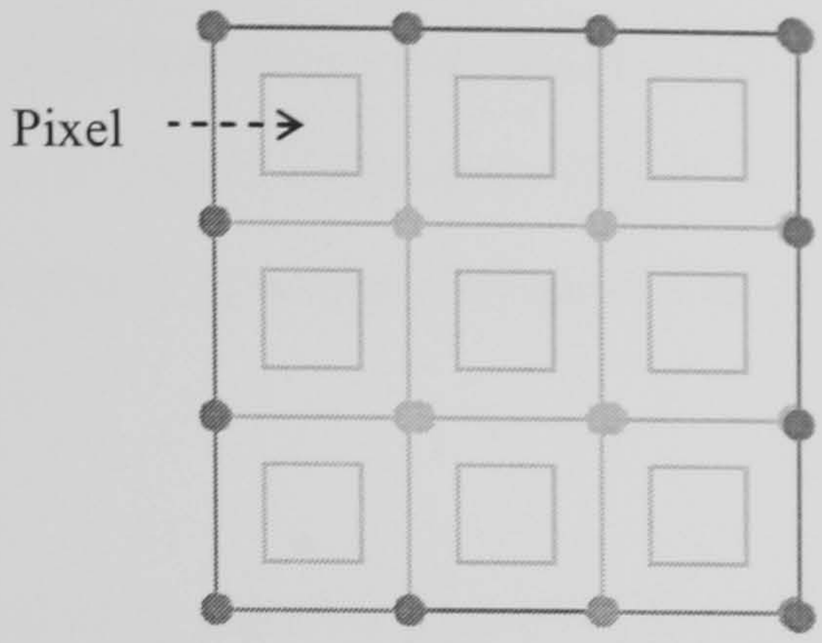



Fig. 2.12. A 3×3 window centred at pixel *(m, n)*.

### 2.6.2. Phase Derivative Variance

The phase derivative variance measures the statistical local variance of the wrapped phase derivatives. It is the most reliable quality map extracted from the wrapped phase itself. Actually, phase derivative variance indicates the badness rather than the goodness of the phase data.

The 2D phase derivative quality map defines the goodness of each pixel using Eq. (2.21) below:

$$q_{m,n} = \frac{1}{B_{m,n}} \tag{2.21}$$

Where $q_{m,n}$ is the quality of the pixel *(m, n)*.

$B_{m,n}$ is the badness of the phase derivative variance of pixel *(m, n)*, which is given by the Eq. (2.22):

$$B_{m,n} = \frac{1}{k^2} * [\sqrt{\sum \left( \hat{\nabla}\Phi^x_{i,j} - \overline{\hat{\nabla}\Phi^x_{i,j}} \right)^2} + \sqrt{\sum \left( \hat{\nabla}\Phi^y_{i,j} - \overline{\hat{\nabla}\Phi^y_{i,j}} \right)^2} \tag{2.22}$$

Where $\hat{\nabla}\Phi^x_{i,j}$ is the wrapped phase gradients in the horizontal direction,

$\hat{\nabla}\Phi^y_{i,j}$ is the wrapped phase gradients in the vertical direction,

$k$ is the window size (odd number),

$\overline{\hat{\nabla}\Phi^x_{i,j}}$ is the mean value of $\hat{\nabla}\Phi^x_{i,j}$ in $k \times k$ window,

$\overline{\hat{\nabla}\Phi^x_{i,j}}$ is the mean value of $\hat{\nabla}\Phi^y_{i,j}$ in $k \times k$ window.

### 2.6.3. Maximum Phase Gradient

The maximum phase gradient measures the magnitude of the largest phase gradient *i.e.*, partial derivative or wrapped phase difference in a $k \times k$ window. Like phase derivative variance quality map, maximum gradient indicates the badness rather than the goodness of the phase data.

The measure of badness for each pixel is given by the Eq. (2.23):

$$B_{m,n} = \max \left\{ \begin{array}{l} \max \left\{ \left| \hat{\nabla} \Phi^{x}_{i,j} \right| \right\}_{in\ \ k \times k\ \ window} \\ \max \left\{ \left| \hat{\nabla} \Phi^{y}_{i,j} \right| \right\}_{in\ \ k \times k\ \ window} \end{array} \right\} \qquad (2.23)$$

The good data in the phase map can be achieved by using the reciprocal of badness as define in Eq. (2.21).

### 2.6.4. Second Phase Difference

Another quality map extraction method is the second difference quality map [Herra'ez *et al.* (2002)]. This quality map measures the badness of each pixel in a $k \times k$ window using Eq. (2.24):

$$B_{m,n} = \sqrt{h^2(i,j) + v^2(i,j)} \qquad (2.24)$$

Where $h(i,j) = W[\Phi_{i-1,j} - \Phi_{i,j}] - W[\Phi_{i,j} - \Phi_{i+1,j}]$,

$$v(i,j) = W[\Phi_{i,j-1} - \Phi_{i,j}] - W[\Phi_{i,j} - \Phi_{i,j+1}],$$

$h(i, j)$ and $v(i, j)$ are the horizontal and vertical second differences, respectively.

After determining the badness of each pixel, the quality of each pixel $q_{m,n}$ can be easily calculated using Eq. (2.21).

### 2.6.5. Quality Map Extraction Using Weighted Window

All the quality map extraction methods described previously use fixed windows around each pixel to extract quality information. However, in this method, an adaptive weighted window is used to sharpen the shape of the extracted quality map and to

minimize the influence of noise [Cho (2004)]. This extraction method is summarized as following:

1. Extract the quality map $B_{m,n}$ using phase derivative variance method described in section 2.6.2.
2. Design a $k \times k$ weighted window using Eqs. (2.25) and (2.26) [Cho (2004)]:

$$w(i,j) = A_0 \alpha e^{-\alpha t}, \quad -(k-1)/2 \le i,j \le (k-1)/2 \tag{2.25}$$

$$\alpha = K \frac{\sigma_I}{\bar{I}} \tag{2.26}$$

Where $t$ is the distance from the centre pixel,

$I$ is the quality map in the $k \times k$ window,

$\bar{I}$ is the average value of $I$,

$\sigma_I$ is the standard deviation of I,

$A_0$ is the normalised constant,

$K$ is the damping factor, where this parameter is determined by the quality value $B_{m,n}$ calculated by in the previous step at $\text{pixel}(m,n)$.

3. The modified weighted quality map is then extracted using Eq. (2.27) but restricted with weights as in Eq. (2.25):

$$B_{m,n} = \frac{1}{w(i,j)^2} * \left[ \sqrt{\sum \left[ w(i,j) \cdot \left( \hat{\nabla}\Phi^x_{i,j} - \overline{\hat{\nabla}\Phi^x_{i,j}} \right) \right]^2} + \sqrt{\sum \left[ w(i,j) \cdot \left( \hat{\nabla}\Phi^y_{i,j} - \overline{\hat{\nabla}\Phi^y_{i,j}} \right) \right]^2} \right] \tag{2.27}$$

4. Proceed to step 2 unless the available execution time is exhausted or the difference between the present and previous solution is very small [Cho (2004)].

The good data in the phase map can be identified by using the reciprocal of badness as defined in Eq. (2.21). This quality map extraction is intelligent but it is not problem-specific to phase unwrapping. It is still a general image processing technique that is robust and intelligent, but unfortunately computationally exhaustive.

### 2.6.6. Other Quality Map Sources

There exist other quality or weighting map sources such as: correlation coefficient (only for SAR application), residue density, flatness of the unwrapped phase. smoothness of the unwrapped phase (generated by the sum of absolute values of the phase gradient) and statistically derived values (this method introduced for the MCF algorithms) [Chen and Zekber (2000)].

## 2.7. Other Forms of Phase Unwrapping

There are other forms of phase unwrapping such as 3D-phase unwrapping and Multi-wavelength/ Temporal phase unwrapping.

Three-dimensional phase unwrapping is still a new area of research and only a few algorithms have been proposed. In 2001, Huntley proposed a three-dimensional noise immune phase unwrapping algorithm that extended the two-dimensional residue-balancing method into three dimensions [Huntley (2001)]. Cusack *et al.* proposed another robust three-dimensional phase unwrapping algorithm that was used to unwrap MRI data [Cusack and Papadakis (202)]. This algorithm uses a quality measurement to guide the final unwrapping path. Two novel three-dimensional phase unwrapping algorithms were proposed by [Abdul-Rahman (2007)]. The first algorithm attempts to find the best unwrapping path in the three-dimensional wrapped-phase volume. The second algorithm follows a best path approach to unwrap the phase volume, but takes into account the effect of singularity loops. Singularity loops are defined here as the source of noise that must be avoided during unwrapping.

Multi-wavelength/ Temporal phase unwrapping requires a projection and acquisition of a sequence of fringe maps [Huntley and Saldner (1993)]. In this phase unwrapping method, the phase at each pixel is measured as a function of time. Unwrapping is then carried out along the time axis for each pixel independently of the others. Thus boundaries and regions with poor signal-to-noise ratios do not adversely influence good data points [Huntley and Saldner (1993)].

## 2.8. Conclusion

This chapter has presented all the major theory of the phase unwrapping problem. A definition of the problem has been presented and the phase unwrapping integration

methods have been explained with a review of some of the most popular existing phase unwrapping algorithms to date.

Moreover, this chapter has defined the quality map technique and presented a review of the major existing quality maps or weighting factors.

The problems that face many phase unwrapping algorithms have been briefly described. The major problem for all phase unwrapping algorithms is the residue problem and its corresponding ghost discontinuity lines. Many statements by researchers in phase unwrapping have pointed to this issue. Researchers have tried to solve the residue problem either directly or indirectly.

The optimum phase unwrapping algorithm has to rely on three major techniques to achieve optimum results. The first technique needs to deal with the residue problem directly. The second one needs to find an optimum weighting factor or quality map. Finally, it is necessary to create the masks or branch-cuts that balance residues of opposite polarity and mask out residues and their corresponding ghost discontinuity lines. The third point is the most important point which relies on identifying the residue ghost discontinuity lines. Without the third point, any phase unwrapping result is not robust.

In summary, none of the previous methods address the underlying causes of unwrapping failure. Such causes are specifically addressed in this thesis through the development of the residue vector approach.

# Chapter 3

# Artificial Intelligence

# Chapter 3

## 3. Artificial Intelligence

### 3.1. Introduction

Many Artificial Intelligence (AI) methods have been created and a lot of research has been invested in developing its techniques.

Artificial Intelligence has been used for the last two decades in many areas of science and technology especially image processing. In image processing, AI has been used for shape detection, feature extraction, object and pattern recognition, filtering, calibration, phase recovery, visual signature and optimization. It has proved to be efficient, robust, noise tolerant and intelligent. AI has found many image processing applications in the real world especially in MRI, optical metrology, system calibration, computer vision, virtual reality, *etc*..

These are some of the reasons why there is interest in AI technology for the benefit of phase unwrapping research. This chapter presents a brief description of some of the most important AI algorithms to date especially the stochastic search algorithms such as Simulated Annealing (SA), Neural Networks (NN) and Genetic Algorithms (GA). The AI stochastic search is a technique that uses probabilistic methods to solve hard combinatorial problems. Stochastic search algorithms rely on randomized decisions while searching for solutions to a given problem. These algorithms will be the basis of the work presented in later chapters. Moreover, this chapter presents some of the existing AI methods developed for phase unwrapping.

### 3.2. Artificial Intelligence Algorithms

#### 3.2.1. Simulated Annealing

Simulated Annealing is a global stochastic search algorithm that is capable of solving NP-hard problems in polynomial time [Cusack *et al.* (1995)]. It is easy to implement and can have many general applications. This algorithm relies on a random search using a Monte-Carlo method of acceptance to the modification of the existent solution controlled by a temperature mechanism. It is not a local search algorithm, but a global search algorithm that uses some problem specific local search algorithms. However, the

annealing process of the solution means very long slow cooling, which if speeded up will probably lock the solution into a local minimum. The reason for this slow cooling is that it relies on one solution. Although it seems memory efficient, it lacks the knowledge of different possible solutions at every iteration it executes. Therefore, the possibility of fast convergence of one solution to the global optimum is very small due to this lack of knowledge.

SA is a method that minimizes a cost function and defines a corresponding probability distribution then proceeds by sampling the probability distribution as the temperature is reduced to zero [Stramaglia *et al.* (1999)]. A simulated annealing algorithm is summarized by the following steps in Fig. 3.1.

Set an initial starting temperature; $T_{start}$
Generate an initial solution, $s_i$
Set a temperature reduction function, $\alpha$
Evaluate the quality of the initial solution, $E_i=f(s_i)$

Repeat
  Repeat
    Generate a new solution, $s_{i+1}$
    Evaluate the quality of the initial solution, $E_{i+1}=f(s_{i+1})$
    Evaluate, $\Delta E_k = E_{i+1} - E_i$

    If ($\Delta E_k$ <0), then, $s_i = s_{i+1}$;
    Else
      Generate a random number, $p \in [0,1]$;
      If $p < \exp(-\Delta E_k / T)$, then, $s_i = s_{i+1}$;
    EndElse
    $k=k+1$
  Until $k$=*number of repetitions*
  $T = \alpha(T)$
Until stopping condition ($T=T_{stop}$)

Fig. 3.1. A simple Simulated Annealing Algorithm.

SA consists of the following techniques:

- **Cost function** is used to evaluate the quality of the solution,
- **Solution generation** is a mechanism used to generate new solutions,
- **Acceptance criterion** is used to decide whether new solutions are accepted,

- **Temperature control parameter** is used to control the speed of the temperature cooling process,
- **Temperature schedule** is used to lower the temperature,
- **Stopping Criterion** is used to decide whether the algorithm has reached the optimum solution.

### 3.2.2. Reversed Simulated Annealing

A more advanced SA method is reverse simulated annealing (RSA) [Gutmann (1999)]. Unlike the very slow convergence of simulated annealing in converging from a randomly created solution that is far from the global optimum, RSA starts from an initial solution that is close to the global optimum using a local search method. If the cooling mechanism starts from this initial solution, it might trap the solution in a local minimum, thus, the initial solution is heated to a certain point and cooled back again to the reach global optimum. Even though RSA is about three times faster than SA, it was found that it is still slow compared to a local search algorithm.

### 3.2.3. Genetic Algorithms

A Genetic Algorithm is an artificial intelligence method that mimics the evolution of genes throughout human generations, leading to better ones. A Genetic Algorithm is a stochastic search technique that uses a global random or problem-specific search controlled by a set of different operators [Fornaro *et al.* (1997)]. It relies on a probabilistic search mechanism and it uses self-adapting strategies for searching based on random exploration of the solution space coupled with a memory component. This enables the algorithm to learn the optimal search path from experience [Steeb (2002)]. Genetic algorithms are specifically designed to solve non-deterministic polynomial problems called NP-hard problems which involve large search spaces containing multiple local minima [Steeb (2002)]. It has also been applied to many combinational optimisation problems and has proved its robustness and speed. It has one advantage over NN that it is an unsupervised AI, this means that it does not need training. Also, the GA strategy reduces the possibility of falling in a local optimum. One disadvantage of this method is that it needs well-chosen setting parameters in order to achieve a satisfactory result and to increase convergence speed [Cuevasa *et al.* (2002)].

Genetic algorithm iterations consist of a set of chromosomes where every chromosome represents a candidate solution for the problem to be solved. Chromosomes consist of a number of genes that can be considered as variables to the problem. Manipulating these genes (variables) will result in creating new solutions. To evaluate how much the solution has been improved, a fitness function (evaluation function) is tailored to the problem.

Such a method uses three natural techniques:

- **Natural selection**, which allows the best genes (*e.g.*, healthy in human terms) to be selected thus giving them more possibility of moving on to the next generation,
- **Crossover**, which is responsible for producing new chromosomes (offsprings) from the original chromosomes (parents),
- **Mutation**, applies deliberate changes to a gene at random, to maintain variation in the genes and increase the probability of not falling into a local minimum solution. It involves exploring the search space for new better solutions.

Genetic algorithms can be summarized as following in Fig. 3.2:

```
Randomly generate initial population, size Pop,
Evaluate the fitness of every chromosome,

Do
        For chromosome = 1 to Pop/2
                Use selection operator to select to chromosomes,
                If rand_no. < Px
                        Apply the crossover operator,
                If rand _no. < Pm
                        Apply the mutation operator,
        End
        Replace the new population with the old population,
        Evaluate the fitness of all new chromosomes,
        Preserve the most-fit chromosomes so far and copy into the new
        population,

Repeat while not converged or not at maximum iteration.
```

Fig. 3.2. A simple genetic algorithm, where $P_x$ and $P_m$ are the probabilities of performing crossover and mutation, respectively.

### 3.2.4. Hybrid Genetic Algorithms

Hybrid genetic algorithms (HGA) use both local search heuristics and genetic algorithm global search methods. This is of great interest in combinational theory especially in solving NP-hard or NP-complete problems like the Travelling Salesman problem. It has out performed other AI methods. Many journal papers have been published in this subject especially in developing new operators such as the selection, crossover and mutation operators. It is much faster than any other AI method and it can achieve results that have never been achieved with other graph theory or AI methods within a reasonable time.

This artificial intelligent method relies on both problem-specific and general search methods to achieve global optimum results.

Hybrid Genetic algorithms can be summarized as following in Fig. 3.3:

Generate initial population, size Pop, using a local search method like nearest neighbour,
Evaluate the fitness of every chromosome,

Do
  For chromosome = 1 to Pop/2
    Use advanced selection operator to select to chromosomes,
    If rand_no. < $P_x$
      Apply problem-specific crossover operator,
    If rand _no. < $P_m$
      Apply the mutation operator based on a local search method like nearest neighbour,
  End
  Replace the new population with the old population,
  Evaluate the fitness of all new chromosomes,
  Preserve the most-fit chromosomes so far and copy into the new population,

Repeat while not converged or not at maximum iteration.

Fig. 3.3. A simple hybrid genetic algorithm, where $P_x$ and $P_m$ are the probabilites of performing crossover and mutation, respectively.

### 3.2.5. Neural Networks

Neural Networks (NN) are a huge subject in modern science, they have been implemented in many areas of technology and especially in image processing, pattern recognition and phase unwrapping. There are different kinds of NN's. It can be defined as a network of neurons (a computational unit), which mimics the thinking and intelligence of human brain cells. This network is trained by different sets of training pattern where the error between the actual and the desired output of the network is used to update the weights of the neurons.

### 3.2.6. Fuzzy Logic

Fuzzy Logic (FL) is an AI method that has been used intensively in control engineering and it has some application in image processing. It is used in image processing for clustering and classification [Hui and Chai (2001)]. It is more efficient when used alongside other AI methods because it is very good in decision making. This method can be defined as a way that mimics human logic for decision-making. Unlike the decision making performed by computers which uses only a true or false cases, it uses a degree of how much a parameter is true or false. A drawback is that it cannot be an algorithm by itself – i.e. it must be used in conjunction with other techniques.

### 3.2.7. Mean-Field Annealing

Mean-Field Annealing (MFA) is a technique derived from statistical physics for the study of phase transitions [Stramaglia *et al.* (1999)]. It is very similar to SA; in fact, it uses the same steps as those used by SA. Instead of sampling the probability distribution, MFA attempts to approximate the mean of the probability distribution while the driving temperature mechanism is decreased [Stramaglia *et al.* (1999)].

## 3.3 Artificial Intelligence for Phase Unwrapping

### 3.3.1. Introduction

Due to the recent advances in artificial intelligence (AI) techniques and their ability to solve problems, AI is finding more and more applications, which gives an impetus to use these advances in the case of the phase unwrapping problem. PU has been implemented using AI techniques such as: neural networks, genetic algorithms, simulated annealing, mean-field annealing, Bayesian approaches, Markovian

approaches Monte-Carlo Metropolis and hybrid forms made up of a combination of these systems.

Even though AI has been used to solve the phase unwrapping problem, these algorithms have not been assessed or challenged by the variety of vexing phase unwrapping situations occurring in actual data analysis which more conventional algorithms have confronted [Zebker and Lu (1998)]. Meanwhile, the science of AI is evolving fast. New more efficient techniques have been developed and implemented. Besides, various sectors of this technology which have not been looked at previously in the context of PU, or did not have so much interest, are now well developed sciences which cannot be ignored. Such is the case with genetic programming, genetic algorithms, fuzzy logic and hybrid models.

There are different kinds of AI technique, some of which have been mentioned previously, which can be summarized in terms of their application to the phase unwrapping problem as follows:

### 3.3.2. Neural Network

Back-propagation NN was investigated on PU and the outcome was it detected phase wraps in %95 of the occasions in one-dimensional PU and in two dimensions unwrapping using NN have taken several hours to unwrap even simple images [Tipper *et al.* (1996), Hamzah *et al.* (1997)]. Also, Hopfield and Kohonen NNs have been experimented with and detected phase wraps on approximately 50% of the occasions [Hamzah *et al.* (1997)]. Due to the limited success with BP NN in one dimension, and their characteristic of being fast alongside their trait of having the ability to generalize input data and ignore noise, it was then applied to 2D PU and achieved 90% success in identifying phase wraps [Schwartzkopf *et al.* (2002)]. General regression and radial basis NN's have been explored, but with very minimal success. One disadvantage of NN's is that they require intensive training on a considerable number of wrapped phase maps.

### 3.3.3. Genetic Algorithms

This intelligent method was applied to PU as a local integration path-following technique and showed very promising results. This GA method is designed to find an

optimum path with the least number of discontinuities. It was mentioned by Fornaro that unwrapping using this GA method would result in some under sampled phase values [Collaro *et al.* (1997), Collaro *et al.* (1998)]. But it is very different from any other local or global integration method as the error remains strictly confined to the under sampled area. It does not allow the propagation of error out of the corrupted regions and that is achieved without fences and/or blanking weights [Collaro *et al.* (1997), Collaro *et al.* (1998)]. However, this method suffers from two main disadvantages. The first is that it uses a normal genetic algorithm which is very costly in terms of computation time. The second is this method attempts to locate an integration path with a minimum number of discontinuities from a very large number of possible paths. It is even worst with the increase of image size. This method has a very high computational cost and given that most of the iterations are meaningless if there is no residues present in the phase map its general efficiency is questionable.

### 3.3.4. Simulated Annealing

Several simulated annealing algorithms have been developed for solving the phase unwrapping problem. One of these algorithms uses a local phase unwrapping integration method which identifies residues, then, groups these residues in branch-cuts with minimum total length. The simulated annealing was used in the residue grouping phase such that it achieves the minimum total length of branch-cuts. It has proved to be robust but inefficient. Even for a phase unwrapping problem with few residues this algorithm still consumes excessive time [Cusack (1995)].

The other SA algorithm developed for phase unwrapping is based on global integration. This method fits a surface (unwrapped solution) to the wrapped phase data by minimizing an energy function that is the sum of two functions using SA. The first function measures the discrepancy between the surface (unwrapped solution) and the wrapped phase map and the second function is a "membrane model" that measures the departure from a contiguous surface. Moreover, this SA unwrapping method is still computationally exhaustive and the method itself what require further development in order to be practical. [Ghiglia and Pritt (1998)].

A more advanced phase unwrapping method was developed that uses simulated annealing with a constraint technique. In this method phase unwrapping is formulated as

a constraint minimization problem with a vector field of integers similar to the MCF and Flynn's algorithms in section 2.3.5. This method reduces the number of $2\pi$ discontinuity errors between the gradient of the estimated unwrapped solution and the wrapped gradient of the wrapped phase map [Guerriero (1998)]. The minimization problem is solved using simulated annealing with a constraint technique [Guerriero (1998)]. It would seem to be a sound approach to the phase unwrapping algorithm though it lacks weighting factors. It can give good results, but using simulated annealing causes this method to be time consuming and computationally intensive.

### 3.3.5. Mean-Field Annealing

Mean-Field Annealing (MFA) has been applied on the same basis as the phase unwrapping method described in section 3.3.4 that formulates phase unwrapping as a constraint minimization problem with a vector field of integers [Stramaglia *et al.* (1999)]. In essence, a mean-field annealing algorithm was used instead of a simulated annealing algorithm. This method uses a cost function based on the second-order differences and it can be applied for any other cost function similar to simulated annealing [Stramaglia*et al.* (1999)]. It is also a very good technique to be used as a phase unwrapping algorithm but again it lacks weighting factors. It achieves similar results to that of the simulated annealing algorithm; however, it consumes less computational time than the simulated annealing algorithm described in section 3.3.4.

### 3.3.6. Reverse Simulated Annealing

This artificial intelligent method has been used to solve the branch-cut minimization problem in phase unwrapping. It can be considered as a hybrid simulated annealing method that uses two local search heuristics and a global search method. The first local search heuristic uses a nearest-neighbour branch-cut solution as an initial starting solution to save the computational burden required by a normal simulated algorithm when starting with a random initial solution. The second heuristic is a clustering method used to cluster residues in groups where each group is solved separately using the reversed simulated annealing in order to lower the computational burden on the algorithm. The global search method is the simulated annealing algorithm used to find the global optimum minimum cut length solution. Unfortunately, this algorithm is very expensive in terms of calculation time if the number of residues is high but it is generally about three times faster than a normal SA algorithm [Gutmann (1999)].

### 3.3.7. Fuzzy Logic

Fuzzy logic has been used in phase unwrapping for clustering purposes. As can be seen many phase unwrapping algorithms have been developed and each algorithm is suitable for specific features in wrapped phase map, however there is no universal phase unwrapping algorithm that could unwrap all kinds of wrapped phase maps. Thus this fuzzy logic clustering method was designed to decide which phase unwrapping algorithm suitable for a certain cluster in the wrapped phase map [Hui and Chai (2001)].

## 3.4. Conclusion

In this chapter, we have presented a brief description of the major artificial intelligence methods and some of the applications of the artificial intelligence methods in phase unwrapping. The brief review presented the interest of researchers in solving the phase unwrapping problem using artificial intelligence. Some of the phase unwrapping methods presented in this chapter are efficient in producing good results but they require extensive computation and still they lack the sets of weights that help in achieving the optimum unwrapping solution. On the other hand, artificial intelligence for phase unwrapping is a promising field of research that has to be explored. In the next two chapters, two phase unwrapping algorithms will be presented that use an artificial intelligent method, namely a hybrid genetic algorithm.

# Chapter 4

# Hybrid Genetic Algorithm for Branch-Cut Phase Unwrapping

# Chapter 4

## 4. Hybrid Genetic Algorithm for Branch-Cut Phase Unwrapping

### 4.1. Introduction

#### 4.1.1. Existing Dipole Branch-Cut Phase Unwrapping Methods

The branch-cut phase unwrapping method has been implemented using many different techniques to achieve the global optimum in minimizing the total cut length. Such techniques that use the dipole branch-cut method include: the nearest-neighbour algorithm [Cusack *et al.* (1995)], the modified nearest-neighbour algorithm [Cusack *et al.* (1995)], the simulated annealing algorithm [Cusack *et al.* (1995)], the minimum-cost matching algorithm [Buckland *et al.* (1995)] and reverse simulated annealing algorithm aided with a clustering technique [Gutmann (1999)].

Nearest-neighbour and modified nearest-neighbour algorithms are local heuristic search techniques that use a set of nearest neighbour heuristics suitable for the branch-cut phase unwrapping problem [Cusack *et al.* (1995)]. Although these algorithms are fast, they might end up with large branch-cuts embedded in the phase map, thus, causing the unwrapped phase map to lose its smoothness.

A more advanced graph theory algorithm called 'minimum-cost matching' (MCM), developed by Buckland *et al.* uses the Hungarian algorithm to find the minimum total cut length of branch-cuts [Buckland *et al.* (1995)]. This algorithm finds the minimum of the total cut length for the branch-cuts within a time approximately proportional to the square of the number of residues. This algorithm is a powerful one in finding an optimum solution, but it suffers from several disadvantages. It requires a matched number of opposite polarity residue sources in the phase map. Also, it needs a reconstructed graph of all possible branch-cuts between nearest-neighbours. It can be hard to find the optimal graph to solve the problem from the first run of the algorithm. In other words, the optimum number of edges (branch-cuts) in a graph that will lead the minimum-cost matching algorithm to the global optimum, without needing to add excessive numbers of border pixels to the problem (yet more edges) is hard to find, and more border pixels mean a slower and more complicated problem.

Another forms of search are by means of artificial intelligence in the form of simulated annealing (SA) [Buckland *et al.* (1995)] and reverse simulated annealing (RSA) [Gutmann (1999)]. These two methods have been explained in chapter 3.

### 4.1.2. Branch-Cut Phase Unwrapping Problem and the Travelling Salesman Problem

The travelling salesman problem is a graph-theoretical combinational optimization problem. It is a very famous problem and a well-known profound research subject. The problem can be summarized as following: "A travelling salesman has to visit an $N_c$ number of cities, $\mathbf{C} = \{c_1, c_2, c_3 \ldots c_{Nc}\}$ covering the shortest route passing through all the cities only once and returning back to the starting city";

$$\text{Total\_Dist} = \sum d(c_i, c_{i+1}) \qquad \text{where } c_i \neq c_{i+1}. \tag{4.1}$$

where d(., .) is a method that calculates the distance between two points.

An example of the travelling salesman tour is shown in Fig. 4.1. This problem cannot be solved in polynomial time execution as the number of cities increase, since the complexity of the problem increases exponentially. If it is required to explore all possible solutions, the number of iterations is infinite or very large depending on the size of the problem. In essence, if N-cities exist, it will take $N_c!$ iterations to explore every solution. This problem belongs to a class of non-deterministic polynomial time problems called NP-hard or NP-complete. No polynomial time algorithm exists to solve this problem. Moreover, an algorithm that can solve an NP-hard problem can solve any class of problems [Jung and Moon (2002)]. In order to develop an algorithm to solve this problem, the algorithm must be intrinsically non-linear as it only generates a series of approximate guesses at a possible solution.

The TSP can be defined as a Graph **G** = (V, E, W), which consists of a set of vertices 'V' corresponding to the cities in the TSP, a set of edges 'E' representing the trip covered from one city to another and 'W' a set of edge weights specifying the distance of the trip between two cities.[Nagata and Kobayashi (1999)].


Fig. 4.1. An example of a travelling salesman tour, where the circles represent the cities the travelling salesman has to visit and the dashed lines are the trip from one city to another and on the dashed lines there is a set of weights corresponding to the distance covered. The start and the end city are marked by a star.

To achieve an efficient unwrapping, the cut length between residues has to be minimized. Also, for a global optimization of the unwrapping algorithm, the total cut length between all the residues and boundary pixels, if there are any, has to be minimized. The minimization of the total cut length can be achieved by graph theory in the form of TSP. The TSP is a method that has been used to formulate many problems and applications. Thus, once the problem in question is formulated into a TSP form, it can use the knowledge and advances used for solving TSP to be optimized. One problem that can be formulated into the TSP is the branch-cut problem. The branch cut problem cannot be considered as finding a minimum weight spanning tree (MWST) as

there is no connectivity between the vertices, hence the efficient algorithms for finding MWSTs are not applicable here.

It is found that the branch-cut problem (BCP) is equivalent to the TSP. In essence, residues $\mathbf{R} = \{r_1, r_2, r_3 \ldots r_{Nc}\}$ are considered to be the cities in the TSP and the nodes or vertices in a Graph. The branch-cuts $\mathbf{B} = \{b_1, b_2, b_3 \ldots b_{Nc-1}\}$ are the trips from one city to another in the TSP and the edges in a Graph. The cut length between two residues, $L(r_i, r_j)$ where $r_i$ and $r_j$ must be positive and negative residues respectively, is the distance of the trip between two cites and the edge weight in the Graph. Hence, the main objective is to reduce the total distance covered by the salesman in the TSP, which can be implemented in phase unwrapping as the reduction of the total cut length between all residues. In other words, the algorithm achieves a global minimum optimum solution of cuts before unwrapping. The only difference between the BCP and the TSP is that BCP is a matching problem, where its nodes have to be matched and connected in pairs, and TSP is a tour construction problem, where nodes have to be connected sequentially. The similarities between TSP and BCP are summarized in Fig. 4.2.

| **Graph** | **TSP** | **BCP** |
|---|---|---|
| **Complexity** | NP-Hard | NP-Hard |
| **Weights** | Tour of the salesman | Branch-cut Length |
| **Vertices** | Cities | Residues |
| **Edge** | Trip from one city to another | Branch-cut |

Fig. 4.2. Similarities between the Travelling salesman problem and the branch-cut problem and their relation to a typical graph.

A vast number of algorithms have been used to solve the TSP. One method of solving the TSP involves the use of heuristics, which are local search algorithms that can be characterized as linear. The simplest heuristic algorithm for solving the TSP is the nearest-neighbour heuristic. This algorithm is a local search algorithm that finds the nearest-neighbour city by calculating the distance separating two cities [Nagata and Kobayashi (1999)]. Once the nearest city is found, it is joined in the tour and the search

for a new city commences until returning to the starting city. Heuristics are widely used even though they produce sub-optimal solutions because they converge in polynomial time. They have interesting properties such as exploring all the possible nodes or cities close to the current city and they are also greedy because heuristics always choose solutions that are best at the moment. These algorithms are fast but they often have the potential of not converging to an optimum solution [Tsai *et al.* (2004)].

On the other hand, another method for solving the TSP is non-linear utilizing artificial intelligence. Artificial intelligence algorithms include simulated annealing and genetic algorithms [Jayalaksmi *et al.* (2001)]. These algorithms are global search methods that are robust but often converge more slowly than local search techniques. A powerful artificial intelligence algorithm is the genetic algorithm. This algorithm searches for a global optimum solution. It uses a global search with methods that are general and not problem specific, unlike heuristics. GAs do not utilize the knowledge of how to search for the solution [Mansour *et al.* (2004)], thus, a certain percentage of GAs iterations are not needed because they do not lead to any improvement to the solution. Hence, GAs require longer periods of time to converge to an optimal solution.

A compromise was made to use the advantages of both global and local search methods and this has resulted in a new algorithm called a hybrid genetic algorithm. The HGA produces very efficient solutions to the TSP in a promising period of time and it is the most researched method for solving the TSP. The hybrid approach possesses both the global optimality of the genetic algorithm as well as the convergence of a local search. The latest advances in HGAs offer much better results for TSP. These methods rely on creating problem-specific genetic operators that make use of every iteration by the genetic algorithm to produce a better new solution. This results in speeding up the algorithm. Moreover, problem-specific operators are more intelligent methods than general operators. Also, heuristics have a share in improving or introducing new solutions in HGA. The use of problem-specific operators in HGAs will lead to searching more possible good solutions in the solution search space.

Therefore, HGA seems to be a very suitable method for solving the branch-cut phase unwrapping problem, because of their capabilities in solving the TSP. Moreover, any operator developed for solving the TSP can solve the branch-cut problem. This

advantage will provide the branch-cut problem with better optimum solutions, leading to improved phase unwrapping results.

### 4.1.3. HGA for Solving the Dipole Branch-Cut Phase Unwrapping Problem

In this chapter, a hybrid genetic algorithm (HGA) is used to optimize the total cut length in a phase map globally before unwrapping. Phase unwrapping in this proposed algorithm is presented in the form of the travelling salesman problem (TSP) with the exception of the matching concept instead of the tour concept. Therefore, most of the advances in solving the TSP will be used to the benefit of the branch-cut phase unwrapping problem. The newly developed genetic algorithm is then tested on simulated and real wrapped phase maps to verify its characteristics and the results are compared with three existent branch-cut phase unwrapping algorithms, which are: simulated annealing, reverse simulated annealing and the minimum-cost matching algorithms.

## 4.2. Coding the Phase Unwrapping Problem in GA Syntax Form

Any optimization problem using a GA requires the problem to be coded into GA syntax form, which is the chromosome form. The chromosome can be used to represent a graph or an equation or a system. Moreover, every chromosome will consist of a number of genes corresponding to nodes in a graph, variables to an equation, or control parameters in a system. In essence, coding a problem plays a major role in the performance of the genetic algorithm. Inefficient coding will result in achieving a poor solution or the algorithm being trapped in local minima. However, good coding will push the GA to have a high tendency to achieve global optimum results but with one drawback, which is the fact that the time needed to execute the algorithm will be very long. Thus, to achieve good results at high speed, the problem has to be studied in many forms and a set of limits has to be deduced to achieve an efficient coding that will fulfil the requirements of the problem. On this basis, two coding schemes were developed to code the branch-cut phase unwrapping problem as a genetic algorithm. One coding scheme used was the all residues and all corresponding boundaries (*ARACB*) coding, which is simple to implement but will result in unnecessary added complexity to the genetic algorithm and hence slow convergence. A compromise on a certain probability stated by Gutmann [Gutmann (1999)] was made, based on the understanding of the problem and this resulted in another coding scheme, which is all residues and minimum

corresponding boundaries *(ARMCB)* coding. This coding scheme reduces the complexity of the problem, which results in speeding up the convergence and reducing memory usage.

### 4.2.1. *ARACB* Coding

This coding scheme calculates all the residues in the wrapped phase map and using the nearest-neighbour algorithm, for every residue it finds its corresponding boundary pixel, which will be considered as a fictitious opposite polarity residue to its corresponding real residue. This problem setting can be clearly seen in the example illustrated in Fig. 4.3.

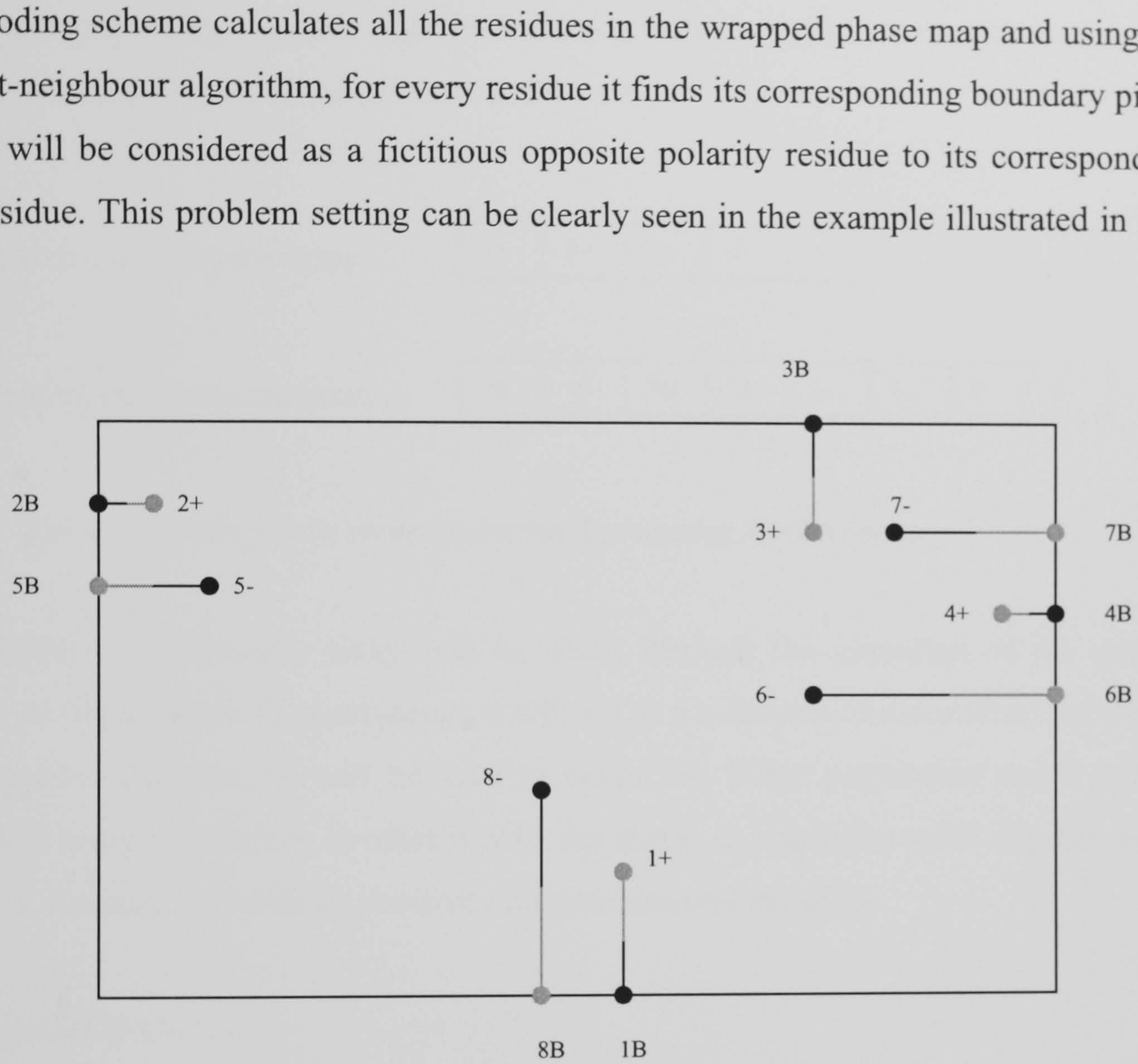



Fig. 4.3. The residues and their corresponding opposite polarity residue boundary pixels in the masked image.

This coding scheme represents the whole branch-cut phase unwrapping problem but adds many boundary pixels, which might not be needed to reach a global optimum solution. It results in chromosomes whose length is twice the number of residue sources, in essence, twice the complexity.

The coding scheme can be summarized in the following steps:

- Calculate residues in the wrapped phase image by integration of the gradient of a 2×2 closed loop path.

- Insert the Indexes of residues calculated from the wrapped phase map in two arrays:
    1. Positive polarity residue Array
    2. Negative polarity residue Array
- Create the initial chromosome by inserting the indexes of all residues and their corresponding opposite polarity residue boundary pixels in two opposite polarity chromosomes in the order seen in Fig. 4.4.

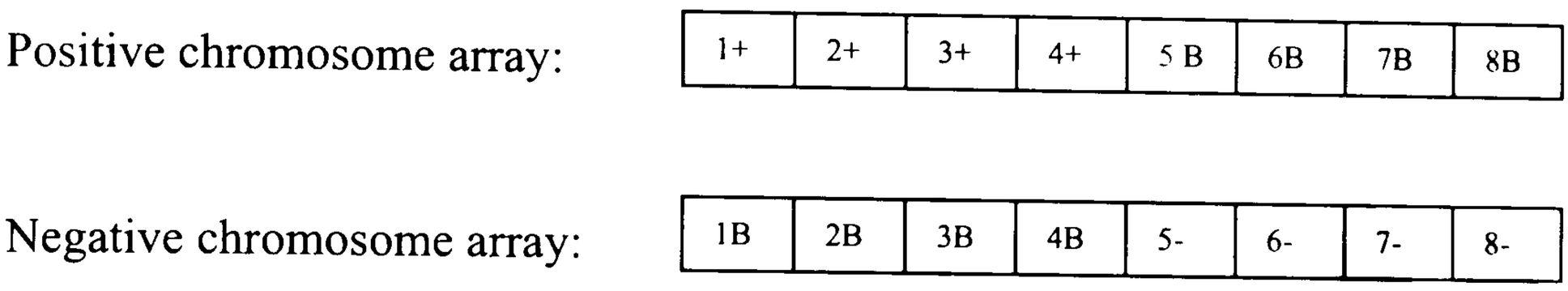



Fig. 4.4. The two opposite polarity chromosomes configured using *ARACB* coding.

The positive chromosome array will be fixed through the alteration of the genetic algorithm (throughout all generations; it will act as a reference chromosome). However, the negative chromosome will be used to create the initial population and it will be altered in every generation. In other words, the aim is to match the order of genes in the negative chromosome with its positive chromosome counter parts.

### 4.2.2. *ARMCB* Coding

Due to the excessive amount of boundaries in the chromosomes, complexity becomes a big factor in slowing down the above algorithm. Thus, a major reduction of boundary pixels in the chromosomes must be achieved in a way that hopefully will not affect the convergence of the GA to the global optimum solution.

One way to reduce boundary pixels is achieved by only considering boundary pixels that have less than twice the distance of the closest opposite polarity residue. This concept of choosing boundary pixels was implemented before in Gutmann [Gutmann (1999)]. An example of this configuration is shown in Fig. 4.5.

The two opposite polarity chromosomes configured using this method for the example in Fig. 4.5 are shown in Fig. 4.6. It is important to note that the size of the chromosomes

and their genes will not change throughout the iterations of the algorithm. The only changes to the chromosome that are permitted are the order of genes in the negative chromosome array.

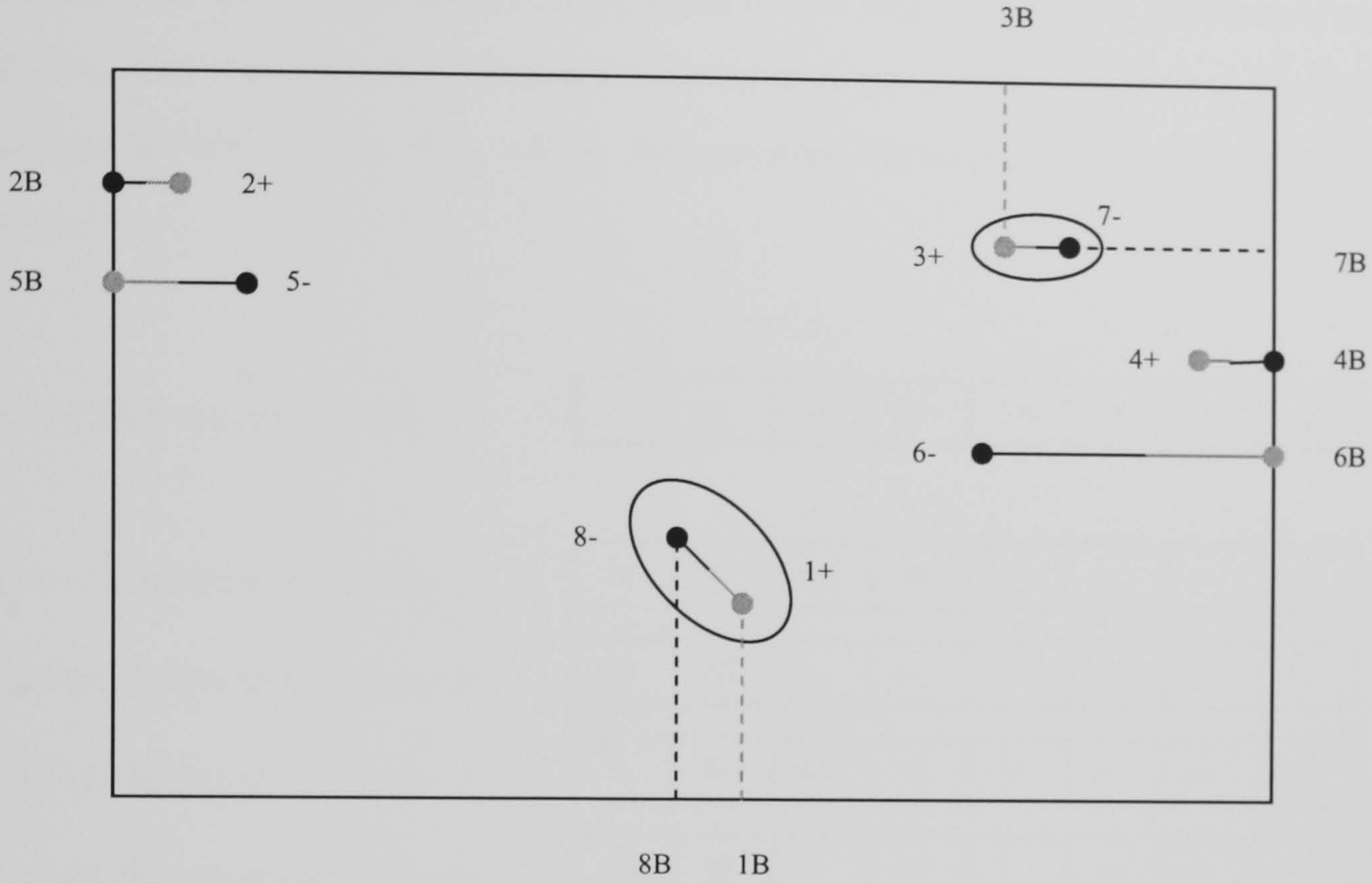



Fig. 4.5. The residues and their corresponding opposite polarity residue boundary pixels in the masked image with an emphasis on boundary calculation.

Positive chromosome array:

| 1+ | 2+ | 3+ | 4+ | 5 B | 6B |
|---|---|---|---|---|---|

Negative chromosome array:

| 8- | 2B | 7- | 4B | 5- | 6- |
|---|---|---|---|---|---|

Fig. 4.6. The two opposite polarity chromosomes configured using *ARMCB* coding.

## 4.3. Creating the Initial Population

A GA requires an initial population of chromosomes where each chromosome represents a possible solution. From this initial population, the GA starts using a stochastic search to achieve the global optimum solution. The method that is used to create the initial population will determine the speed of convergence to an optimum solution, as well as the size of the population (the number of chromosomes in the population). In essence, the size of the population depends greatly on the method used to create the initial population. Moreover, as the size of the population increases, the

complexity and memory usage increase, but on the other hand, the tendency to converge to a global optimum solution increases as well.

Thus, it is required to have an initial population that has the necessary information and gene possibilities for the GA to converge, without the huge amount of chromosomes in a population. The way chromosomes are ordered in a typical population can be seen in the example shown in Fig. 4.7, where the positive chromosome is not part of the population.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Positive chromosome array: | 1+ | 2+ | 3+ | 4+ | 5 B | 6B | 7B | 8B |
| Negative chromosome array 1: | 1B | 2B | 3B | 4B | 5- | 6- | 7- | 8- |
| Negative chromosome array 2: | 1B | 4B | 8- | 6- | 2B | 3B | 7- | 5- |
| Negative chromosome array 3: | 5- | 8- | 1B | 4B | 3B | 6- | 7- | 2B |
| Negative chromosome array 4: | 2B | 3B | 6- | 4B | 5- | 1B | 7- | 8- |
| Negative chromosome array 5: | 5- | 4B | 7- | 6- | 1B | 2B | 3B | 8- |

Fig. 4.7. An example of an initial population consisting of 5 negative chromosomes, where each chromosome represents a solution for the branch-cut phase unwrapping problem.

Because it is an NP-hard problem, a global solution cannot be assured, however, it has as high a probability of achieving a global solution as any existing algorithm. Moreover, this algorithm starts searching from a good initial solution and it has an elitism operator that ensures only a better solution, if found, is to be considered as a final solution. So, in the worst case, the proposed algorithm will return back the good initial solution, if no better solution is found. Three different ways were used to create the initial solution, which are presented as following:

### 4.3.1. Random Initialization (*RI*)

This method randomizes the genes of the initial negative chromosome to create a new chromosome. This method can be implemented using a uniform random number generator or a Gaussian random number generator with variable seed number. This

method will provide the GA with a starting position, generating chromosomes randomly without using an initial solution to create other chromosomes. Thus, it will take the GA a long period of time to reach convergence, *i.e.*, global optimum.

### 4.3.2. Nearest Neighbour and Random Initialization (*NNRI*)

This method uses the nearest-neighbour algorithm to generate a good initial solution for the Branch-cut phase unwrapping. In this manner edges will be ordered with respect to the reference positive chromosome to create a negative chromosome that represents the nearest-neighbour solution. Starting the nearest neighbour algorithm from a different starting residue creates another nearest neighbour solution. Several nearest neighbour solutions are used to create a number of chromosomes, then, the rest of the initial population is generated using a random initialization *RI*. This method of initialization is quite powerful and gives the GA a good start to reach convergence. It is more intelligent and problem-specific than *RI* and it also gives the GA the option of fewer chromosomes in a population than that required using the *RI* method. Moreover, it speeds up the convergence of the GA to an optimal solution.

### 4.3.3. Nearest Neighbour and Random 2-opt Initialization (*NNR2OPTI*)

A faster and more efficient way to create an initial population is by using the nearest neighbour algorithm to create the first chromosome in the population in the same way as that performed in *NNRI*. However, generating the rest of the chromosomes in the initial population is carried out in a different manner, using a *Random 2-opt Heuristic*. This heuristic is applied to the first nearest-neighbour chromosome a random number of times not exceeding the total number of genes in the chromosome in order to create a new chromosome.

The Random 2-opt Heuristic can be summarized as follows:

- Choose two genes (*i.e.*, negative residues) in the first negative chromosome randomly.
- Calculate the summed cut lengths of the corresponding edges and store in $d_a$.
- Swap the two genes (*i.e.*, negative residues).
- Recalculate the summed cut lengths of the corresponding new edges and store in $d_b$.
- If $d_a < d_b$, swap the genes back to their old configuration.

This method can be seen in the example illustrated in Fig. 4.8.

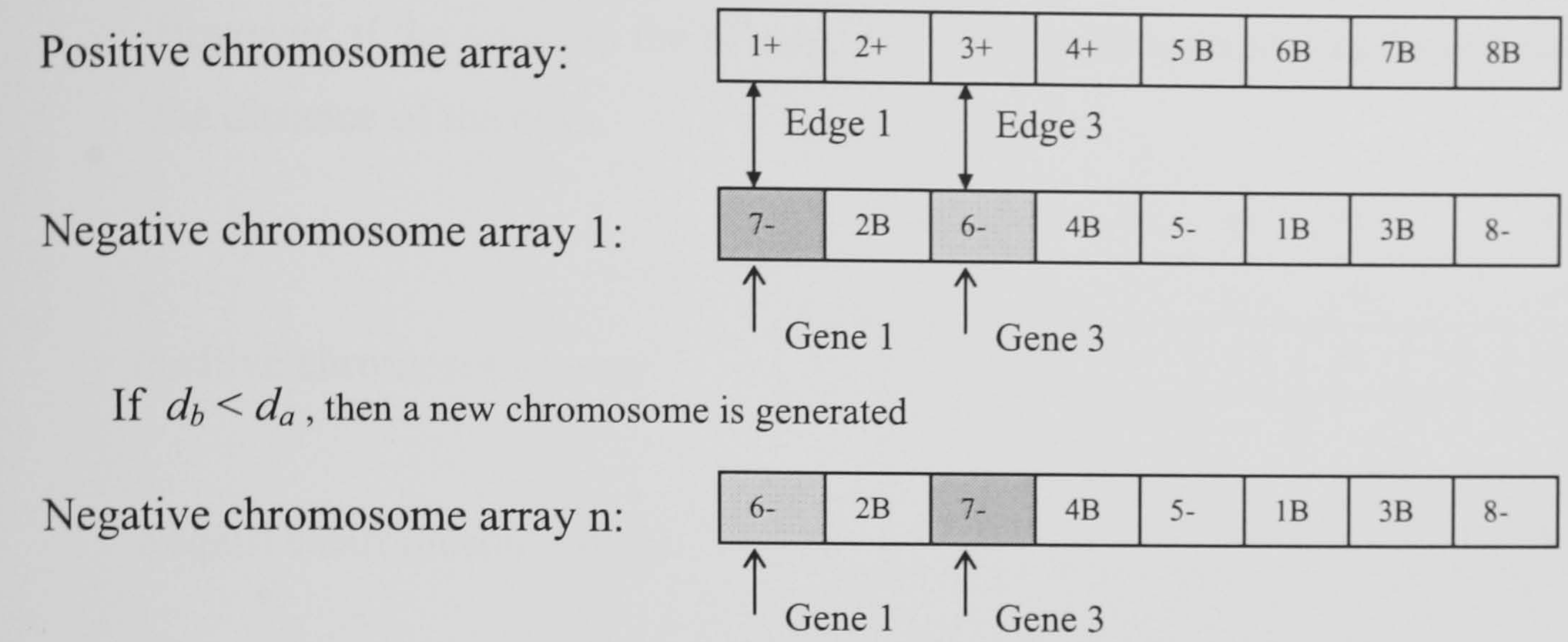



Fig. 4.8. The 2-opt heuristic method.

## 4.4. Chromosome Fitness Evaluation (Total Sum of Cut Distances)

To find the global optimum solution to the branch-cut phase unwrapping problem, the quality of the solution must be evaluated at every generation in order to inform the genetic algorithm of how good its current solution is at each stage. The evaluation will increase the knowledge of the quality level of the solution. This can be achieved by using a problem-specific fitness function. The fitness function corresponding to the problem of branch-cut phase unwrapping must calculate the total cut length of branch-cuts in the wrapped phase map. Thus, a distance measure must be employed. The Euclidian distance was employed to evaluate the distance of every edge at every gene in the chromosome. Then, the total distance presented in Eq. (4.2) is calculated by summing the distance of all edges in the chromosome.

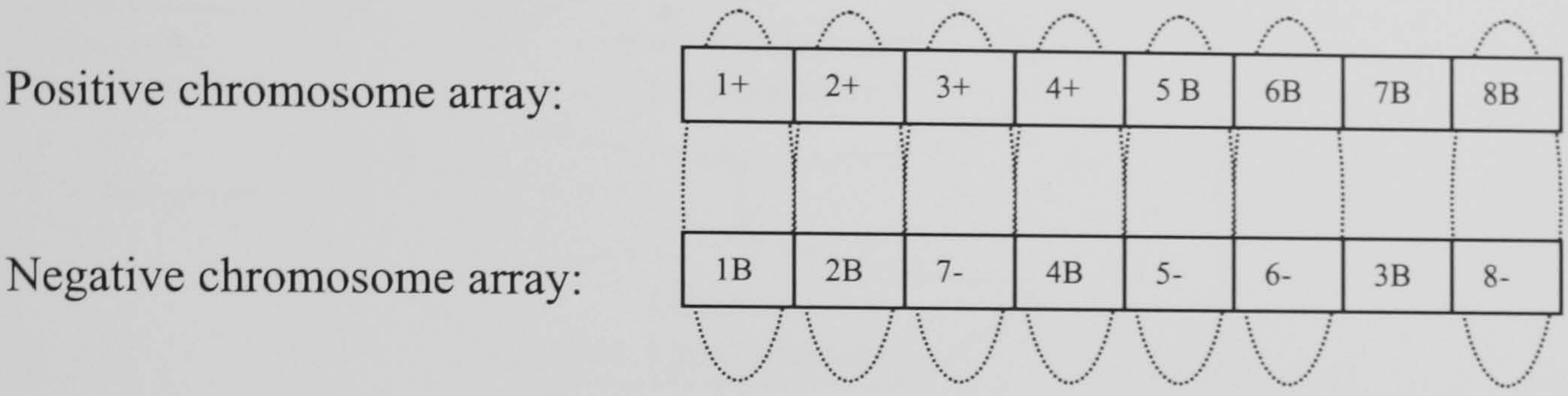
$$Fitness = \sum_{i}^{N} \left[ \left( x_i^+ - x_i^- \right)^2 + \left( y_i^+ - y_i^- \right)^2 \right]^{1/2} \tag{4.2}$$

where $x_i^+$ and $y_i^+$ are the horizontal and vertical index of positive gene $i$, $x_i^-$ and $y_i^-$ are the horizontal and vertical index of negative gene $i$, respectively.

This total distance represents the total cut length in the wrapped phase image. There are three rules that must be obeyed while calculating the fitness of a chromosome:

1. If the genes in the $n^{th}$ edge are positive residue and negative residue genes; then calculate the distance of the edge.

2. If the genes in the $n^{th}$ edge are positive residue and boundary genes or negative residue and boundary genes; then calculate the distance of the edge.
3. However, if the genes in the $n^{th}$ edge are both border genes; then do not calculate the distance of the edge.




Fitness = $d_1 + d_2 + d_3 + d_4 + d_5 + d_6 + d_7$ = Total sum of cut distances.

Fig. 4.9. The selection of edges used to calculate the fitness of the negative chromosome where the positive chromosome is a reference chromosome.

An example of fitness evaluation can be seen in Fig. 4.9. The reason why border genes and other border genes are not evaluated is because they will not be joined together in a cut. In another words, joining boundary and boundary genes together will not improve the branch-cut solution.

### 4.5. Selection Operator

The selection operator is an important step in a genetic algorithm. This reproduction operator selects the fittest chromosomes from the current population and copies them to the new chromosomes in the next generation. It applies the natural concept of evolution, which states that: "the most fit individual survives to the next generation".

Selected parent chromosomes must be suitable for crossover (mating) to generate new child chromosomes, *i.e.*, new solutions that have a high tendency to be better (more fit) than their parent chromosomes [Tsai *et al.* (2003)].

The selection operator is required to be intelligent and problem-specific in order to speed convergence and to avoid trapping the solution in local minima. It is required that the selection operator avoids causing a loss of population diversity and also avoids the ineffective execution of crossover operation [Goldstein *et al.* (1988)].

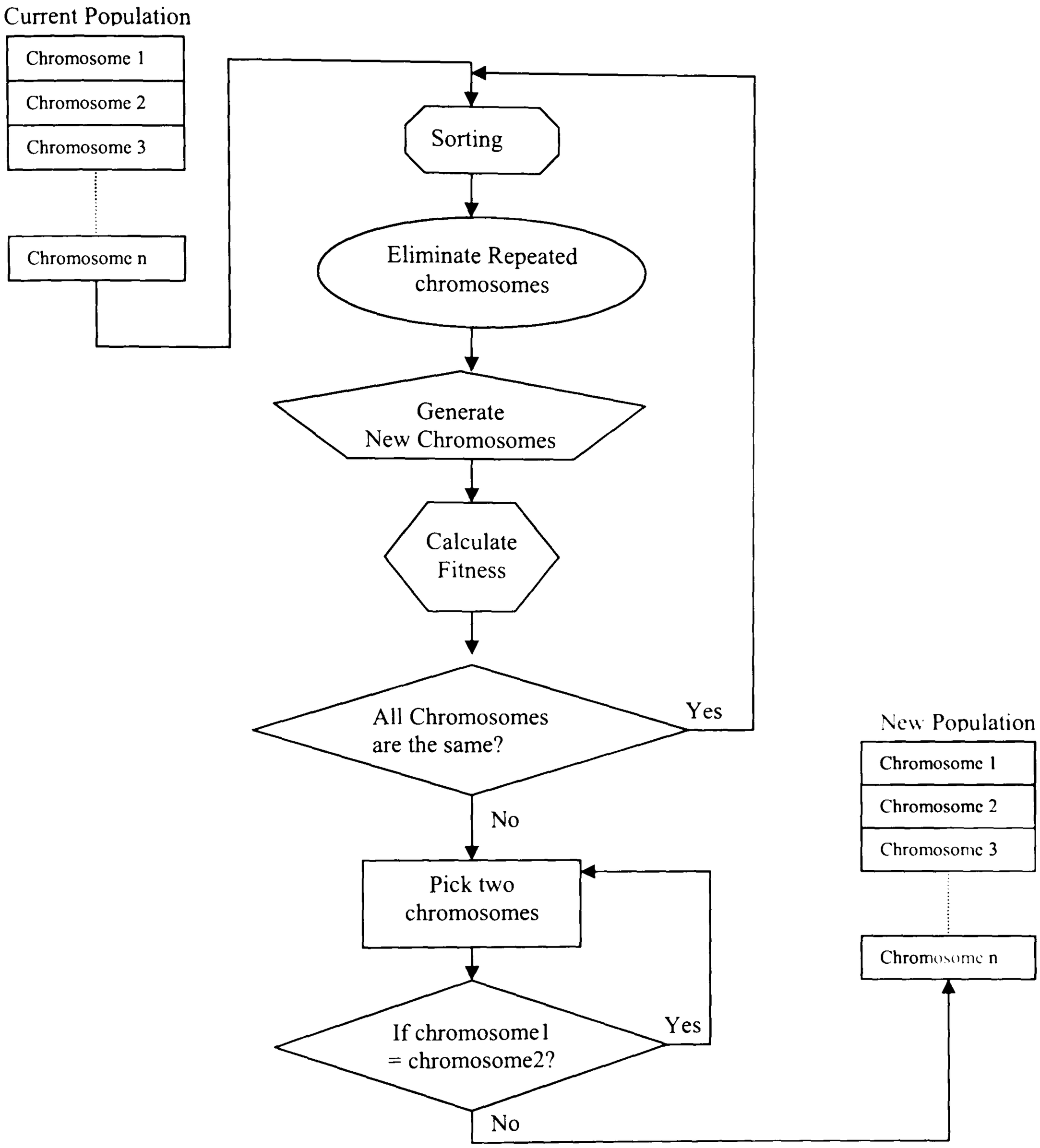



Fig. 4.10. The main steps of the selection operator.

The selection operator used in this proposed algorithm can be summarized as following and a flowchart for this operator is presented in Fig. 4.10:

1. Sort all chromosomes of the current population with respect to their fitness (total branch-cut length).

2. Search for chromosomes that have an equal fitness or have an approximately equal fitness within a very small error margin, such that: $f_n - f_k < e$. Repeated or equivalent chromosomes (after a gene similarity check is made) are then deleted.
3. Generate new chromosomes using the *NNR2OPTI* method to replace repeated or equivalent chromosomes. As mentioned before, this function takes in a good chromosome and generates another. Thus, *NNR2OPTI* will be given the best-fit chromosome to create new chromosomes. The number of chromosomes created equals the number of chromosomes deleted.
4. Calculate the fitness of the new generated chromosomes.
5. Repeat steps 1, 2, 3 and 4 until there are no repeated or equivalent chromosomes.
6. Pick the good-fit chromosomes in descending order of their fitness and randomly pick another chromosome from this modified current population.
7. Check if the chosen chromosomes are the same. If this is the case repeat step 6.
8. Copy both chromosomes to the new generation, which then may be fed to the crossover and mutation operators depending upon the probability factor.

An example of this selection method can be seen in Fig. 4.11 where a population of 8 chromosomes is used.

| Current Population | Value of fitness | Sorted Population | Value of fitness | New Sorted Population | Value of fitness |
|---|---|---|---|---|---|
| Chromosome 1: | 7000 | Chromosome 1: | 7000 | Chromosome 3: | 8500 |
| Chromosome 2: | 4500 | Chromosome 3: | 7000 | Chromosome 1: | 7000 |
| Chromosome 3: | 7000 | Chromosome 7: | 6400 | Chromosome 7: | 6400 |
| Chromosome 4: | 5300 | Chromosome 4: | 5300 | Chromosome 4: | 5300 |
| Chromosome 5: | 1000 | Chromosome 2: | 4500 | Chromosome 2: | 4500 |
| Chromosome 6: | 2900 | Chromosome 6: | 2900 | Chromosome 8: | 4000 |
| Chromosome 7: | 6400 | Chromosome 8: | 1000 | Chromosome 6: | 2900 |
| Chromosome 8: | 1000 | Chromosome 5: | 1000 | Chromosome 5: | 1000 |

Fig. 4.11. An example of the main steps of the selection operator on an 8-chromosome population.

The proposed selection operator gives a chance for 'poor fitness chromosomes' to propagate to the next generation, but also ensures that 'good fitness chromosomes' are propagating to the next generation. However, in terms of avoiding premature convergence in the GA iterations, the proposed selection operator creates new chromosomes instead of identical chromosomes in order to avoid premature convergence.

### 4.6. Smallest Edge Crossover (SCX) Operator

This crossover is a problem-specific operator, which was designed specifically for the phase unwrapping problem. It preserves good edges from both parent solutions, as well as creating new edges in the solution search space [Tsai *et al.* (2004)]. This operator uses a local search method, which chooses a candidate edge that has the smallest cut-length distance. This may sound critical in terms of creating a local optimal solution, however, the percentage chance of leading a created child solution to a local optimal solution is minimal. The reason for this is due to the fact that the variety of genes in different parent solutions in the same population along with the conditional statement at the end of the crossover algorithm may not allow the child solution to be in the next population unless it is more fit than one of the parent solution [Nagata *et al.* (1999)].

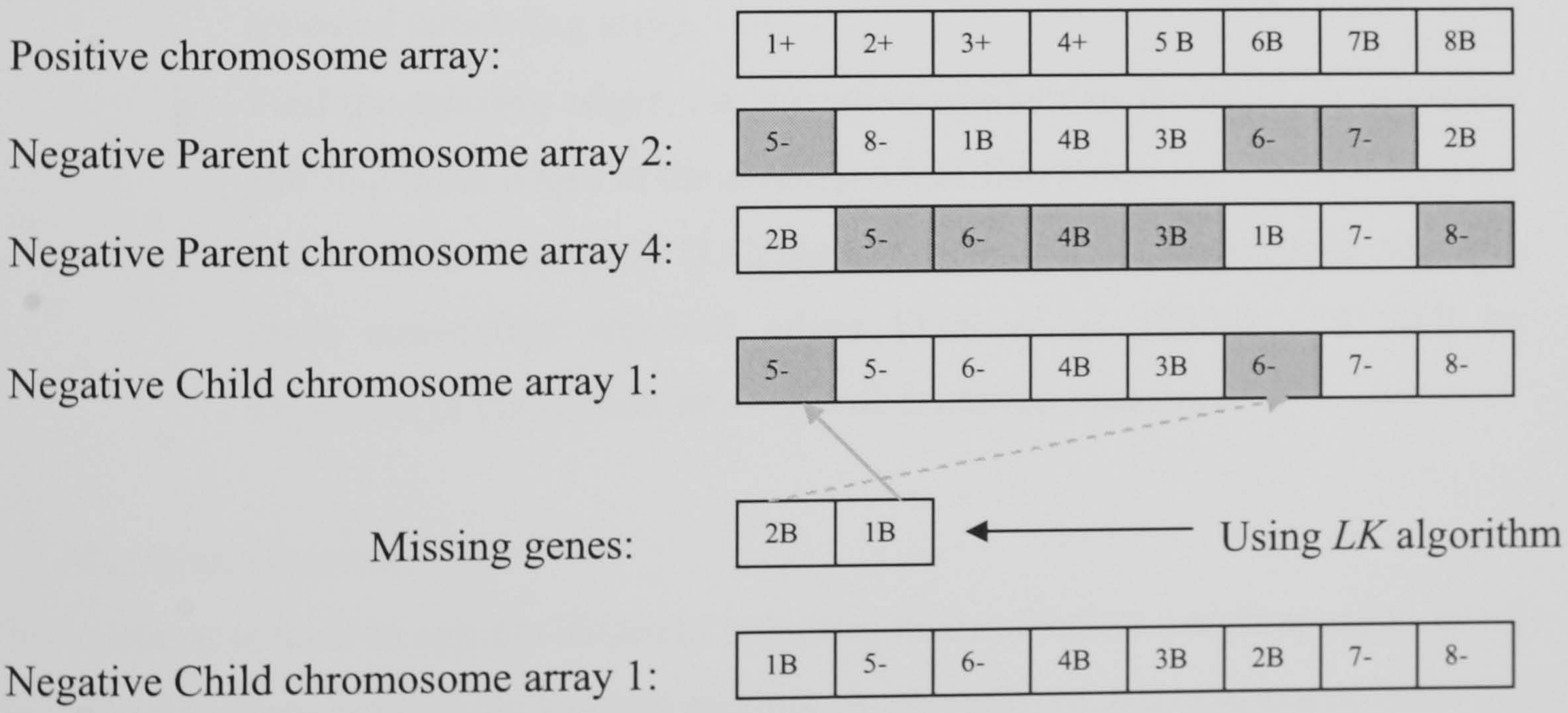



Fig. 4.12. An example of SCX crossover operator developed for the HGA.

This crossover was tested before including it in this phase unwrapping algorithm. It showed very promising results against several well known TSP crossovers. It was found

that using this crossover could cause the genetic algorithm to reach the global optimal solution more rapidly and intelligently than any other crossover operators used for TSP. It has an exponential reduction of the best solution's fitness throughout each generation. It was also found that this operator is very efficient and fast in reaching an optimal solution even without an initial solution. An example of SCX operations is shown in Fig. 4.12.

This crossover can be described as follows:

1. Choose two parent chromosome solutions from the current population.
2. Generate an intermediate solution by:
    - Choosing two edges that have the same positive reference gene from each parent,
    - Comparing both edges with respect to their distances,
    - Choosing the edge with the smaller distance,
    - Putting the new edge in the child chromosome.
3. The new child chromosome may have a set of repeated edges, thus the crossover is responsible for creating new edges not in both parent chromosomes and ensuring no repeated edges in the child solution.
    - Find repeated edges in the child chromosome and flag these edges in the repeated edges flag array.
    - Find the missing edges, *i.e.*, negative residues in the child chromosome and flag these edges in the missing edges flag array.
    - Use the Lin-Kernighan (*LK*) algorithm to arrange the missing edges in more convenient repeated edges [Tsai *et al.* (2004)]. *LK* will be explained in the section dealing with mutation operators.

## 4.7. Mutation Operator

This operator is used to explore different solutions in the solution search space to avoid the algorithm from converging to local minima. A powerful mutation operator leads the GA to better solutions. In this algorithm, a local search method is used as the mutation operator, called the Lin-Kernighan (*LK*) algorithm [Tsai *et al.* (2004), Baraglia *et al.* (2001)]. This algorithm is very efficient and powerful in preserving good edges and introducing new ones to the solution. It has also proved its capabilities in solving the

travelling salesman problem. It uses the concept of exchanging edges only if the generated solution is better than the current solution [Tsai *et al.* (2004)].

The algorithm can be summarized as following:

1. Select randomly $L$ number of edges to exchange (mutate), where $L$ is a variable number varying between $\{2, N_{ng}\}$ and $N_{ng}$ is the total number of negative genes (it was chosen by trial and error).
2. Calculate the total summed distance of the selected edges, $d_1$.
3. Swap the corresponding negative genes to generate a new arrangement of edges as shown in Fig. 4.13.
4. Calculate the total summed distance of the new generated edges, $d_2$.
5. Compare the current and the new total distances.
6. If $d_1<d_2$ then, swap the negative genes back to the original positions in the chromosome and repeat steps 3, 4 and 5.
7. Repeat step 6 until $d_2<d_1$ or until execute until the number of executions equals k.

An example to this algorithm can be seen in Fig. 4.13.

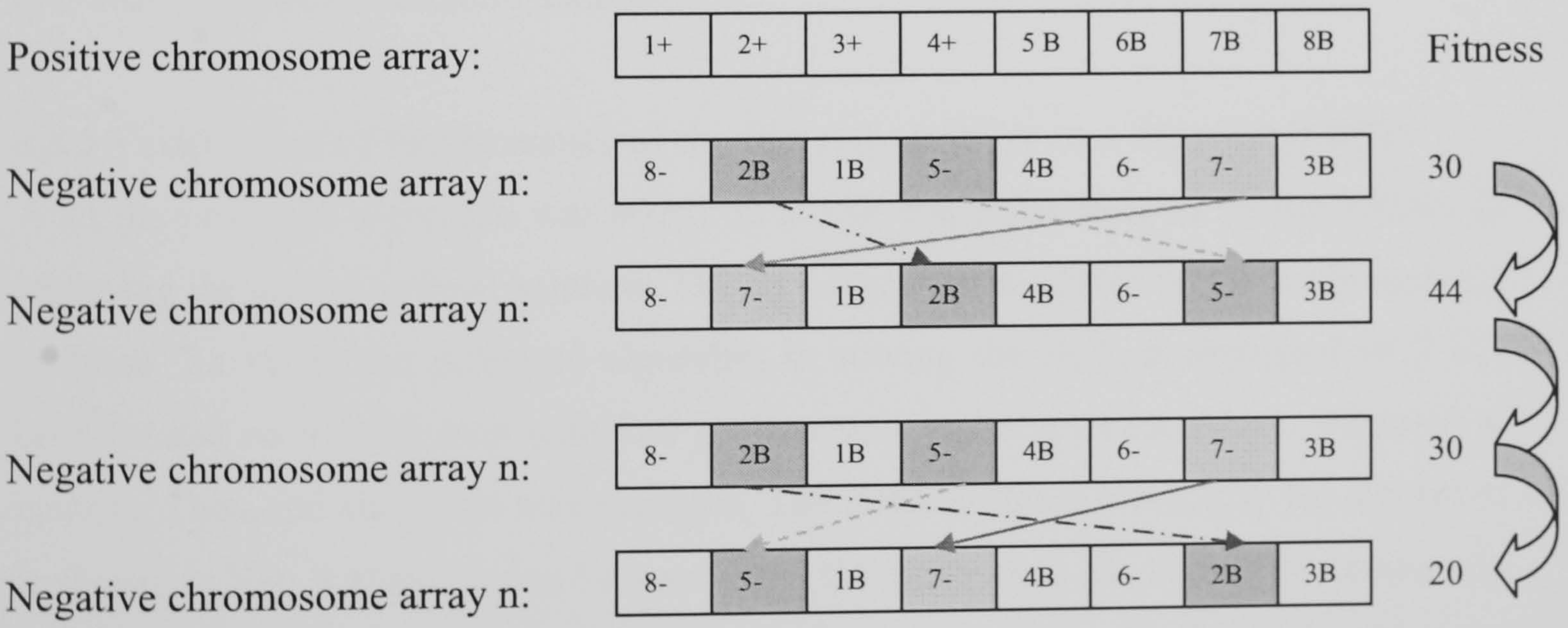



Fig. 4.13. The *LK*- mutation operator on an 8 genes chromosome performing 3 gene mutations.

## 4.8. Results and Discussion

The proposed HGA was implemented for the phase unwrapping application, in order to solve the branch-cut problem. Two sets of wrapped phase maps were used to verify the

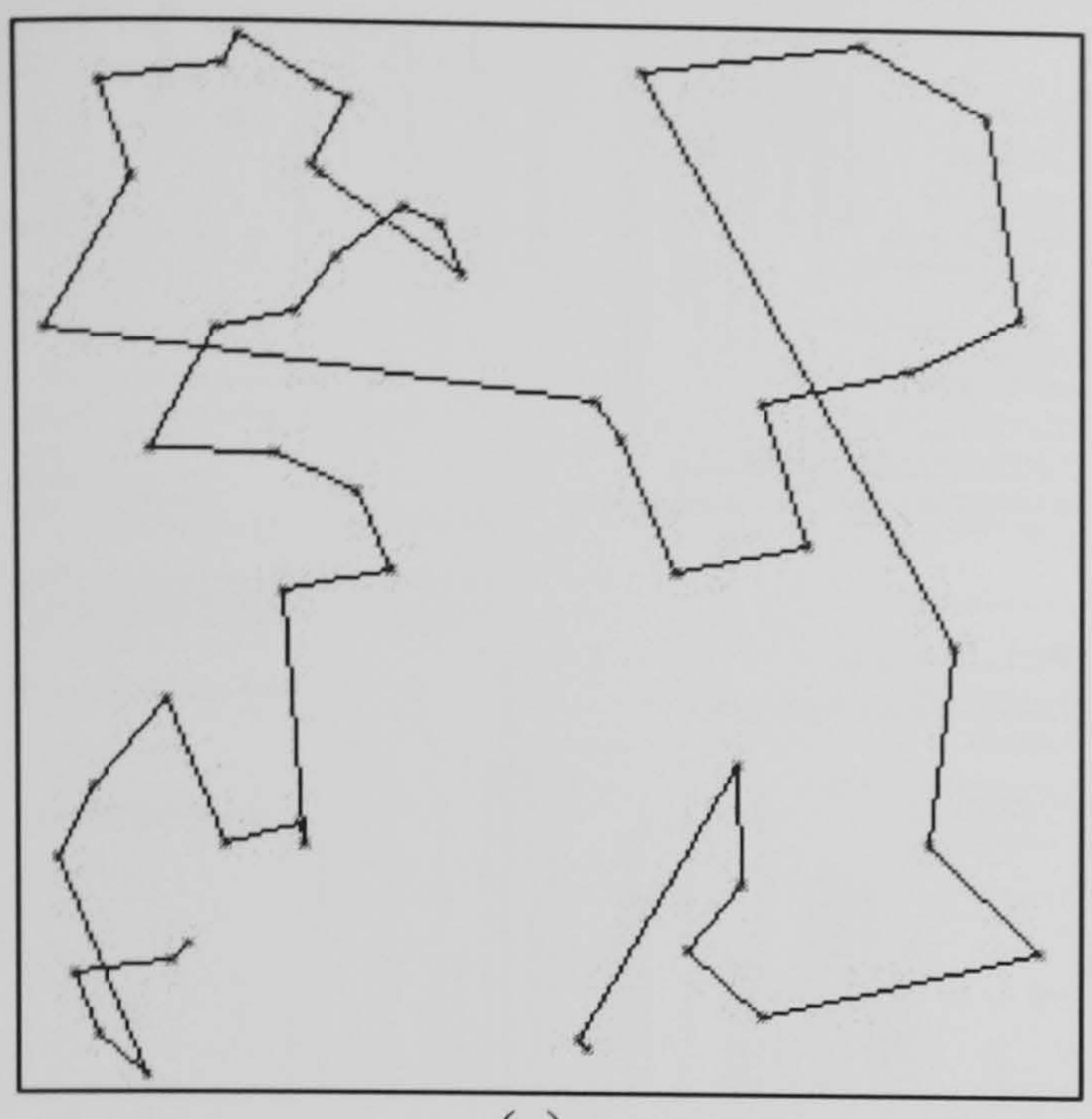
performance of the proposed algorithm; i.e., simulated and real wrapped phase maps. The results were also compared with three existing branch-cut algorithms: i.e., simulated annealing, reverse simulated annealing and minimum-cost matching. The results of all these stated algorithms were executed on a Pentium IV- 3.0 GHz computer. The hybrid genetic setting was chosen by trial and error.

### 4.8.1 Verification of Performance of the Developed HGA on a TSP

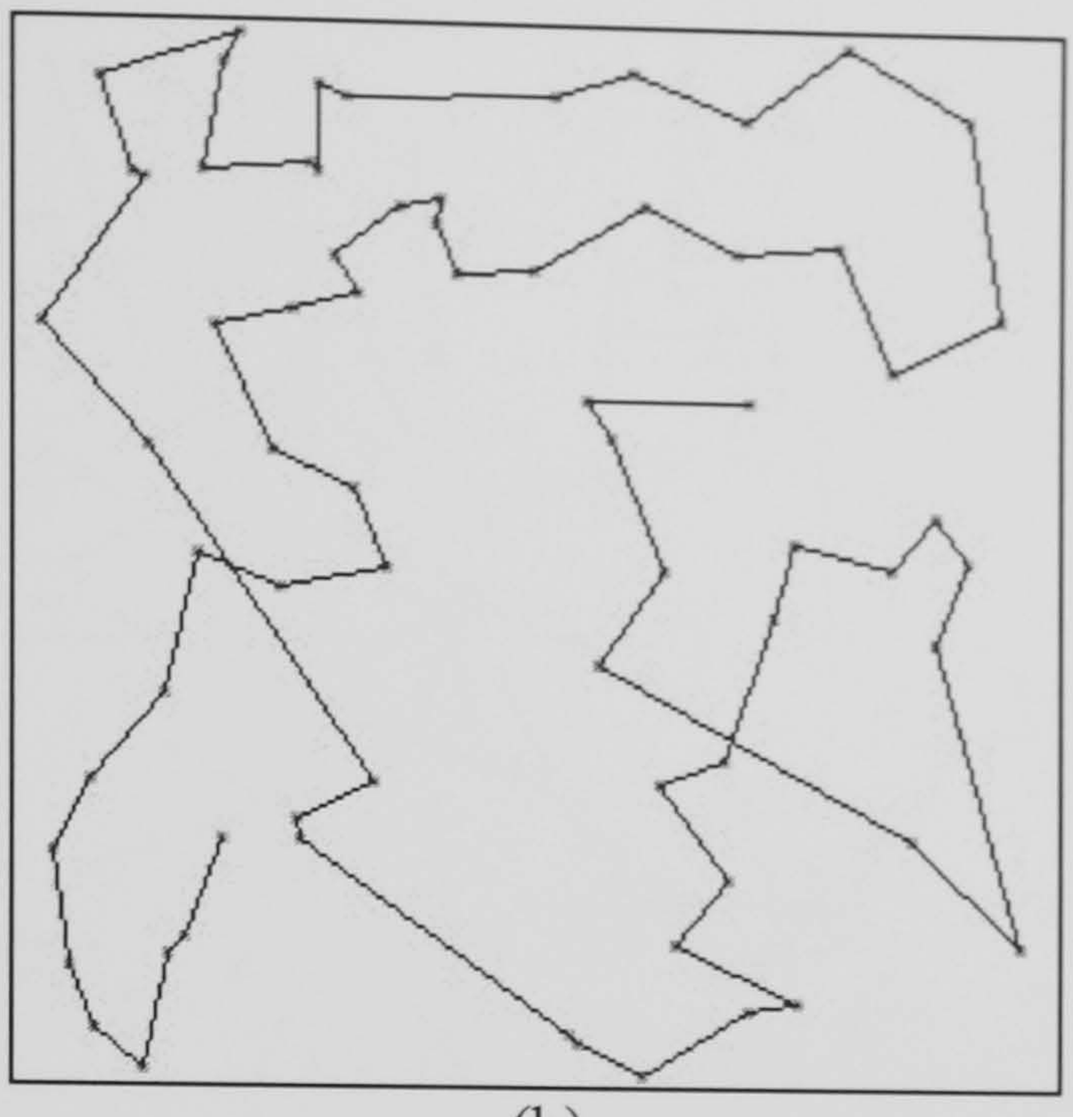
To verify the capabilities of the proposed algorithm, the algorithm was tested on a typical 50 cities travelling salesman problem. The capability of the algorithm in solving this problem indicated that all the HGA operators and selection methods used were ready to be applied to the branch-cut problem. In essence, the proposed HGA will be tested to verify its capability in achieving global optimum solutions. The 50 cities positions were chosen at random. Then, the proposed HGA was executed. In the initial generation, the chromosome with best fitness is shown in Fig. 4.14(a). The chromosome with the best fitness solutions at several algorithm convergences are shown in Fig. 4.14(b), (c) and (d). It can be seen that the algorithm is capable of creating different solutions and is further capable of converging to what could be said a better solution. In Fig. 4.14(e), the optimal solution is shown where the algorithm converges to the best tour with a minimal travelling distance.

### 4.8.2 Verification of Performance of the Developed HGA on a Simulated BCP

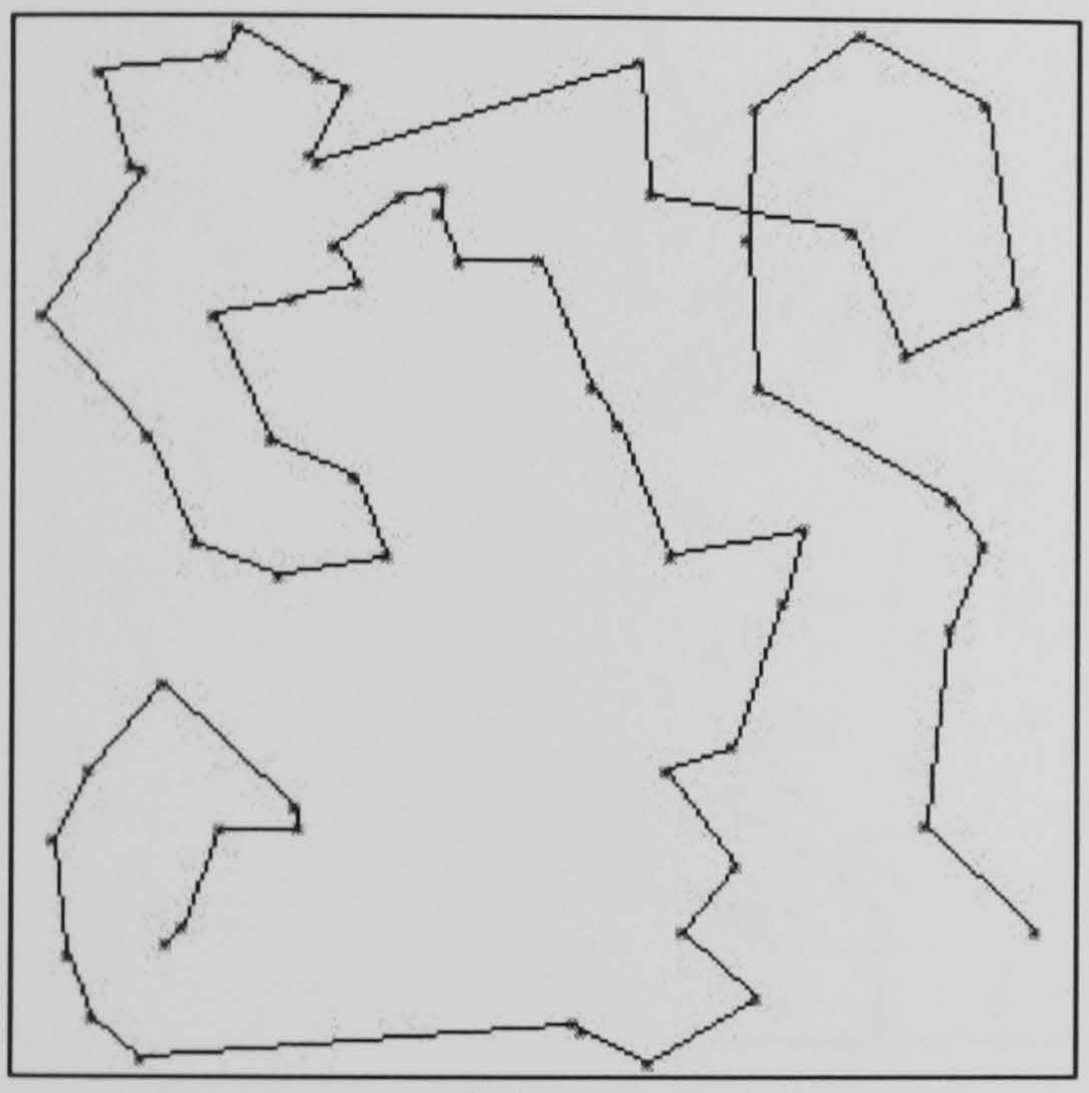
After the proposed algorithm was tested on a TSP and it has showed its capabilities in achieving the global optimal solution. The algorithm was modified to solve a branch-cut problem. To verify the proposed algorithm in solving the BCP, a simulated BCP has been created on a mask map with 100 positive and 100 negative residues generated at random. Then, the algorithm was executed. The initial solution created by the algorithm is shown in Fig. 4.15(a). It can be seen from the figure that the algorithm creates the initial solution by connecting all the existing residues to the border in branch-cuts. Moreover, several algorithm convergences are shown in Figs. 4.15(b) and (c). In Fig. 4.15(d), the global optimum solution is displayed where the global minimal branch-cuts length is achieved. This simulated example demonstrated the algorithms capability to solve the BCP.



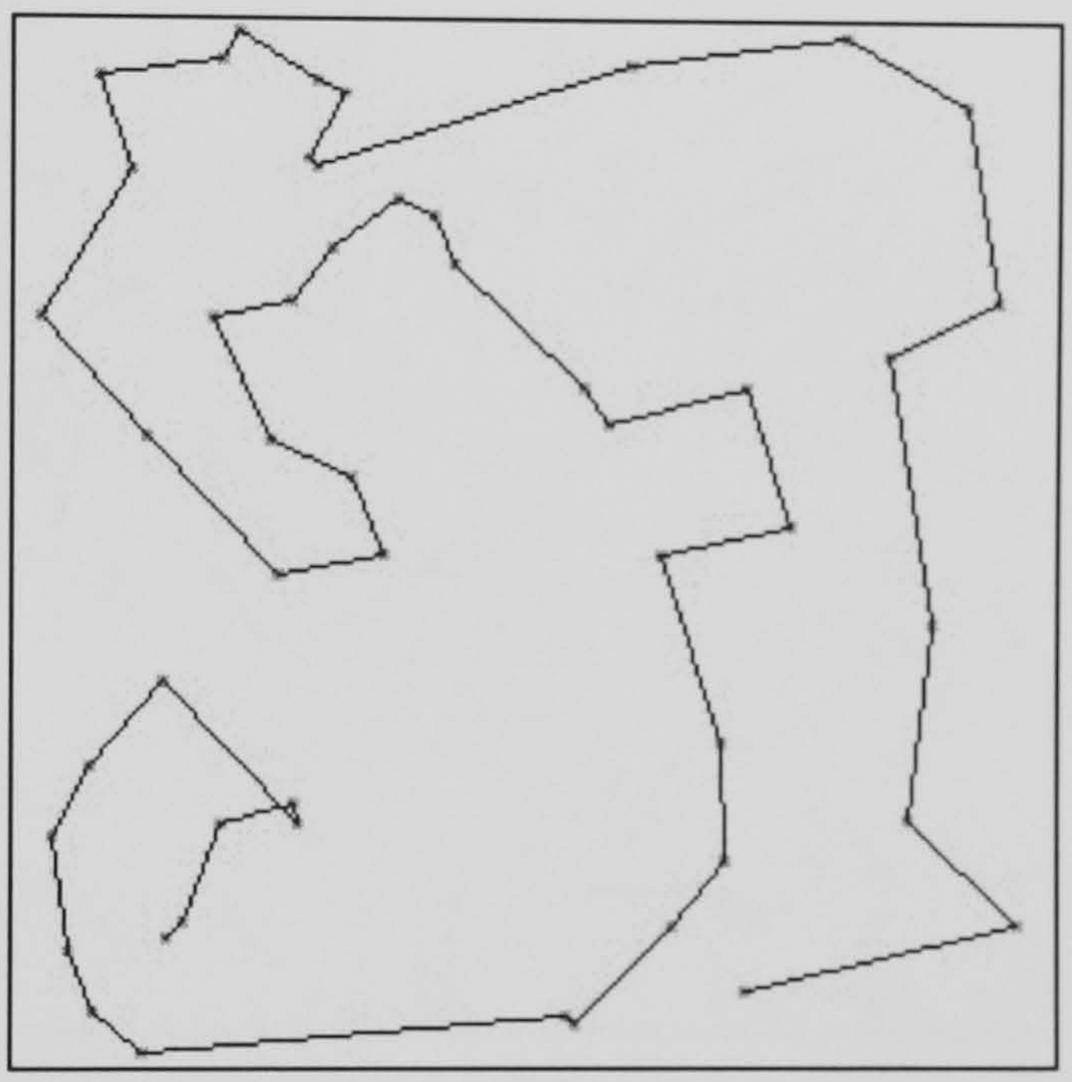
(a)



(b)



(c)



(d)



(e)

Fig. 4.14. Travelling salesman tour of 50 cities generated at random solved by the proposed HGA (a) initial solution; (b), (c) & (d) several convergences of the hybrid genetic algorithm leading to (e) the optimum solution.

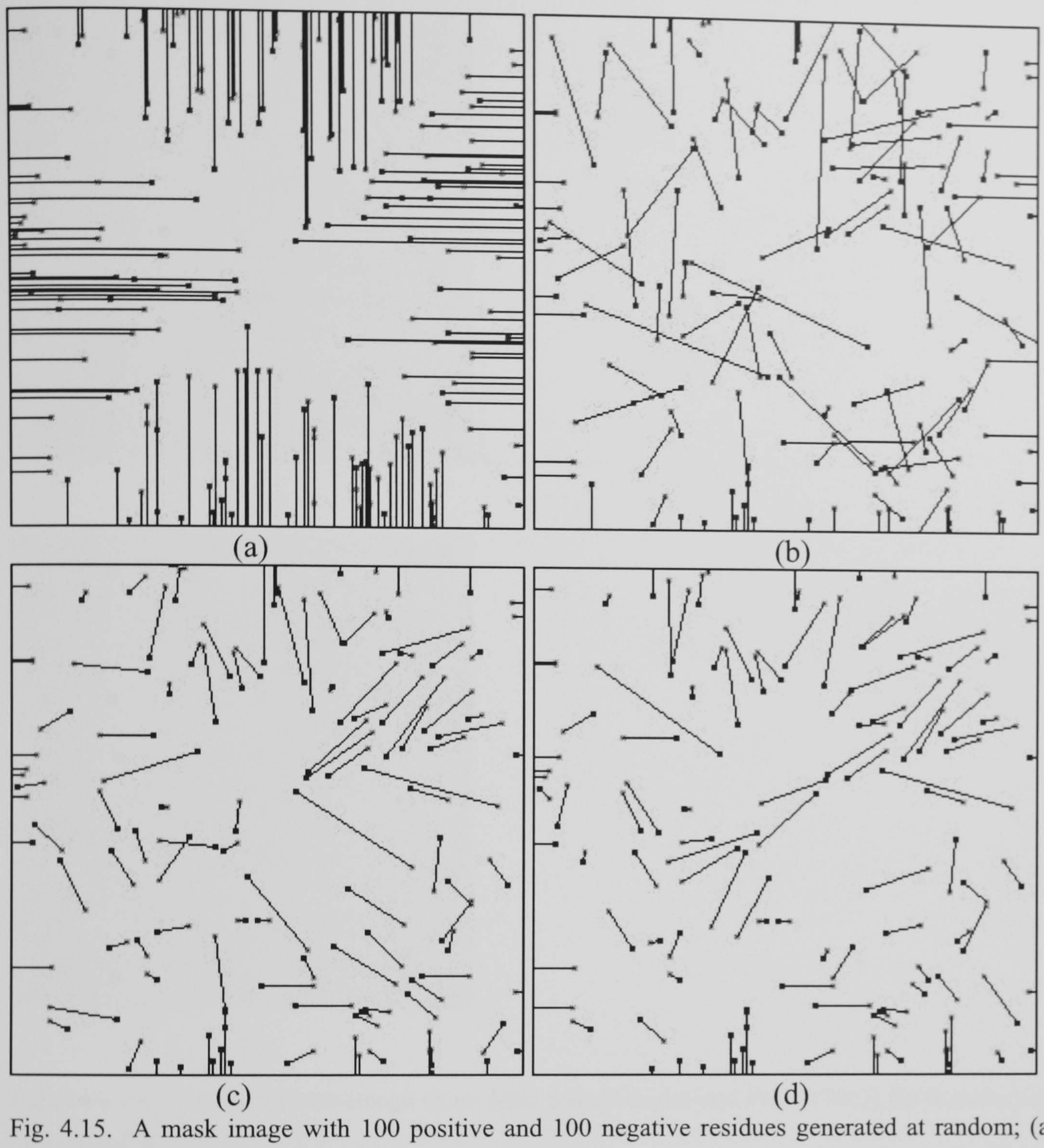
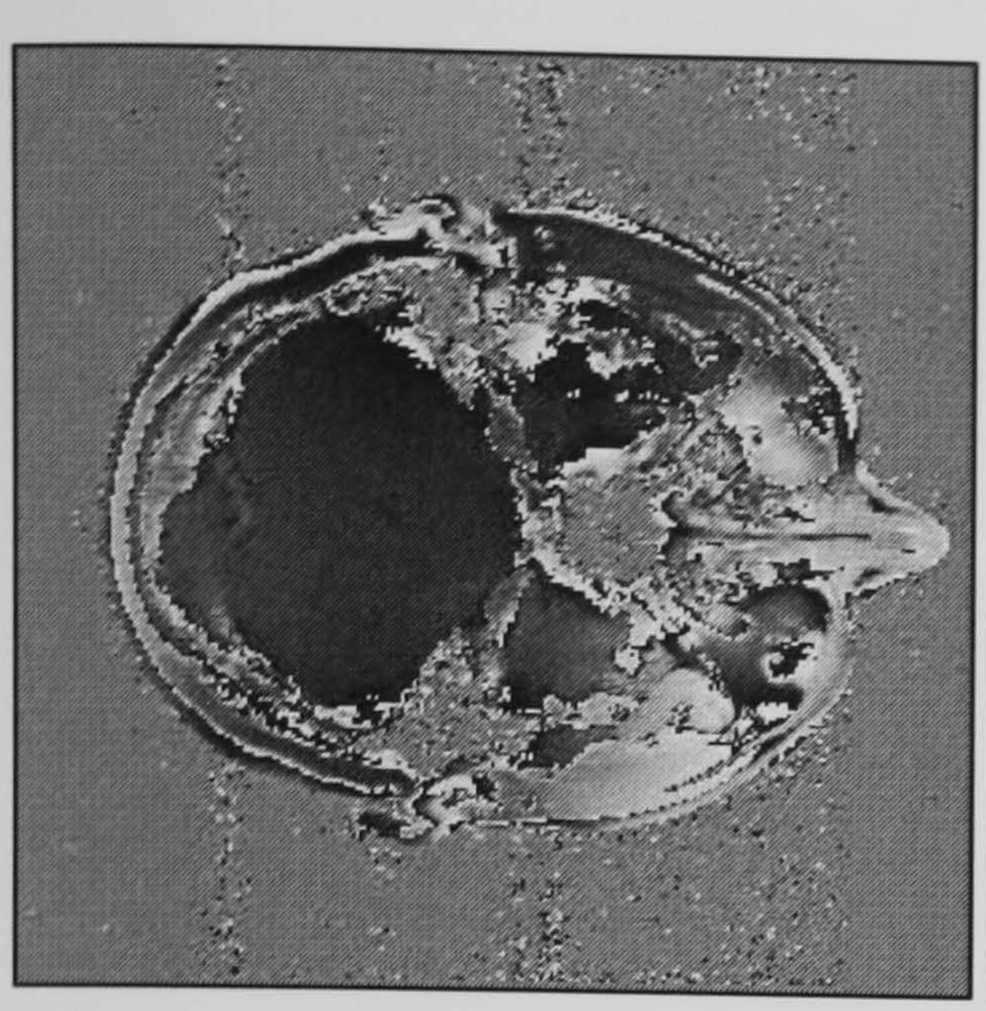


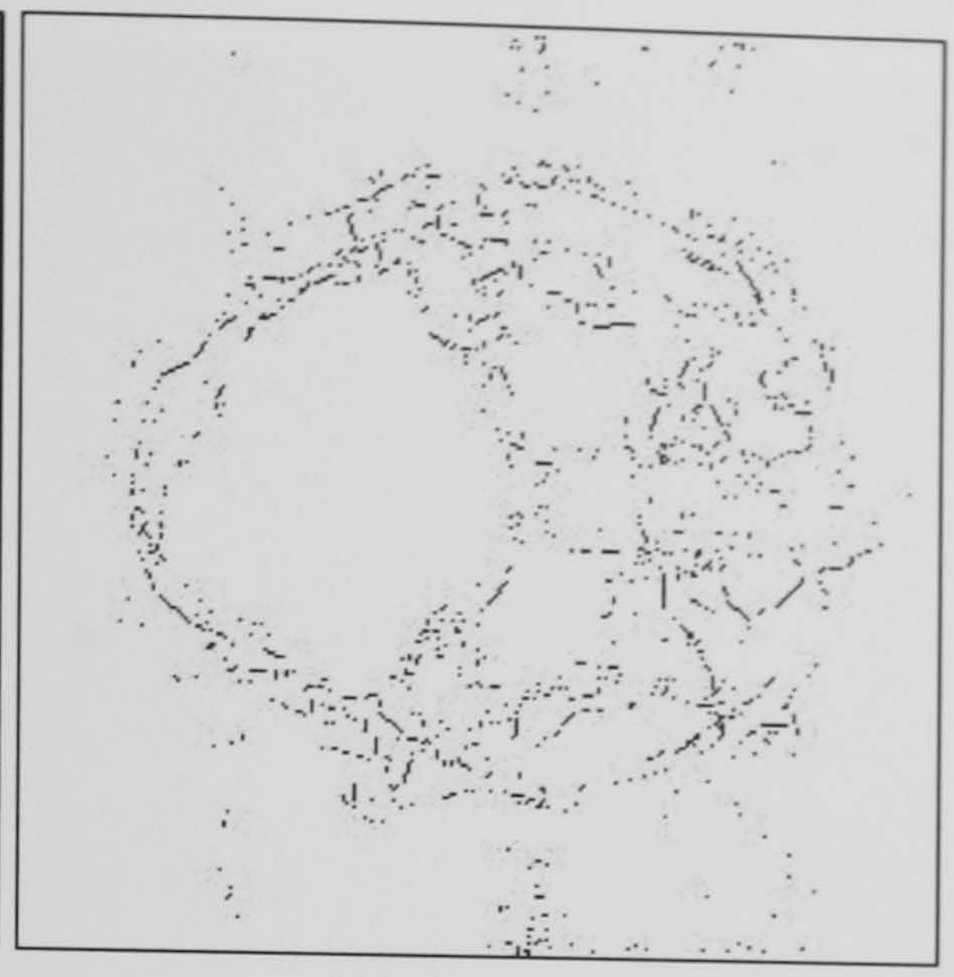

Fig. 4.15. A mask image with 100 positive and 100 negative residues generated at random; (a) initial solution; (b), (c) & (d) several convergences of the genetic algorithm leading to the optimum solution.

### 4.8.3. Real Phase Unwrapping Problem and the Algorithm Performance Graphs

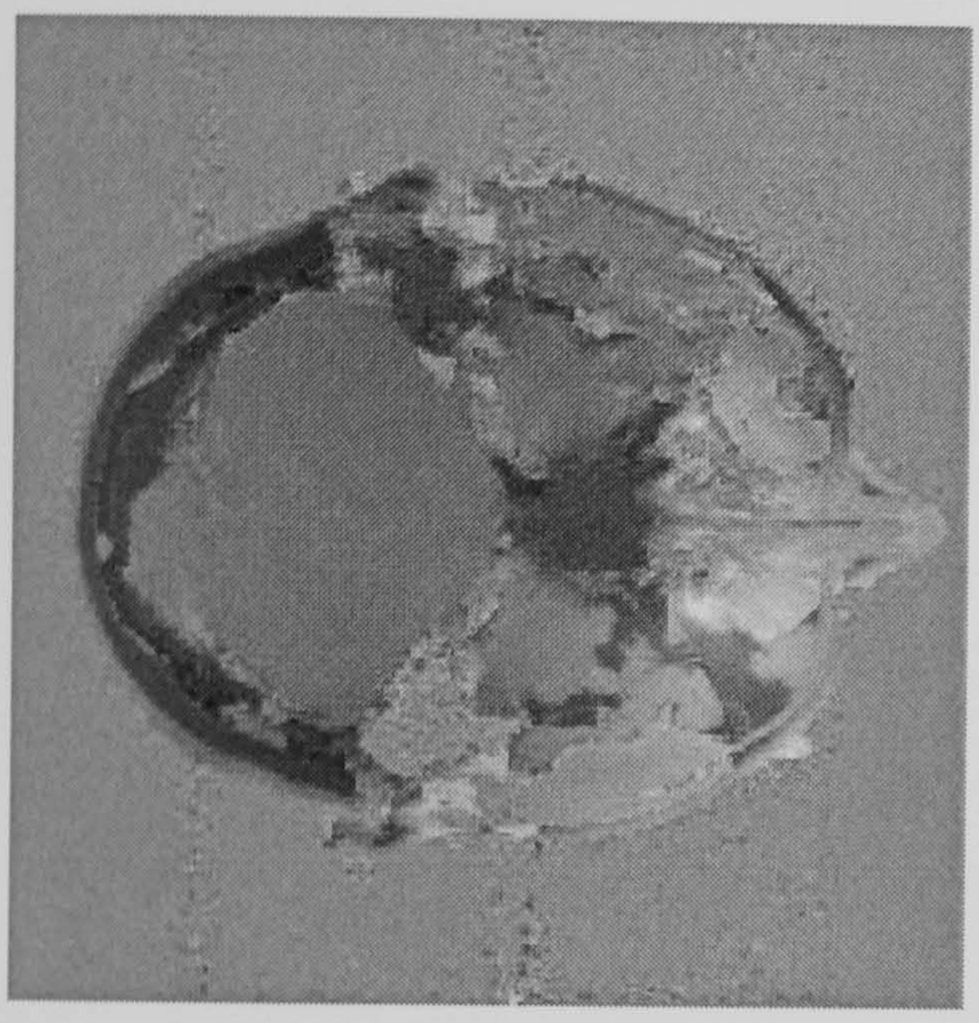
After the success of the proposed algorithm in the two previous stages, the proposed algorithm is tested on a real BCP. This problem is used to verify the algorithm's capability in solving the actual branch-cut phase unwrapping problem. This section will test the algorithm and explain the algorithm's behaviour through out the generations while solving the BCP. This will be shown in GA performance graphs.



(a)



(b)

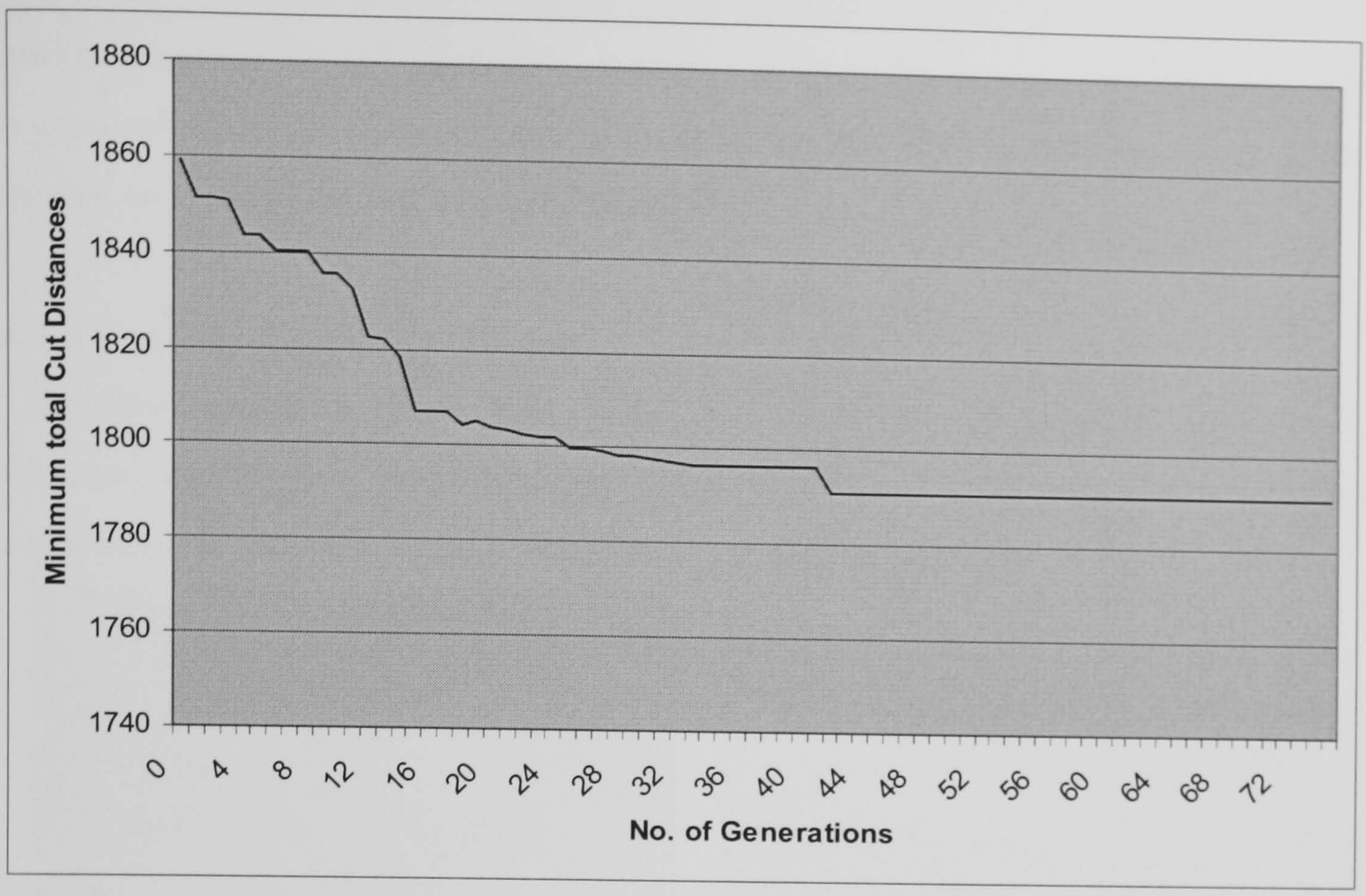


(c)

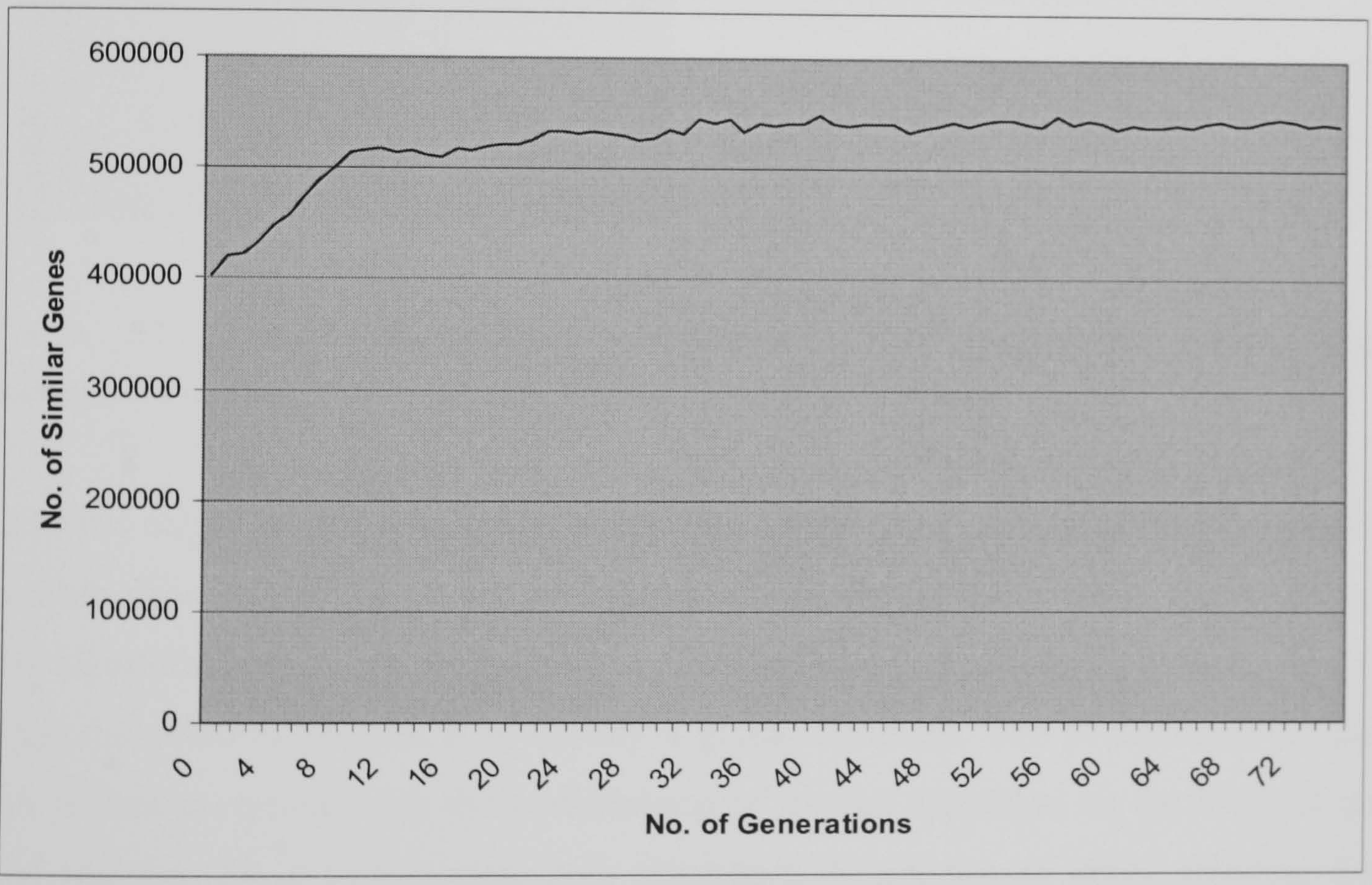
Fig. 4.16. (a) Wrapped phase image of an MRI scan [Ghiglia and Pritt (1998)] (b) branch-cuts and (c) unwrapped phase image.

The algorithm was tested on an MRI wrapped phase map scan in Fig. 4.16(a) [Ghiglia and Pritt (1998)]. The optimum solution with minimal branch-cut length that the hybrid genetic algorithm has converged on is shown in Fig. 4.16(b) and its corresponding unwrapped phase map is shown in Fig. 4.16(c). With this real example the algorithm demonstrated its capabilities to solve the actual branch-cut phase unwrapping problem, achieving a optimum solution and also generating successful phase unwrapping result.

Figs. 4.17(a) and (b) are the performance graphs of the algorithm. Fig. 4.17(a) shows the fitness of the best solution at each generation. The fitness decreases sharply for the first16$^{th}$ generations. Then, it starts to slow down until it reaches the convergence level. The reason for this behaviour is explained in Fig. 4.17(b) where it shows the number of similar genes in all the chromosomes in each generation.




(a)




(b)

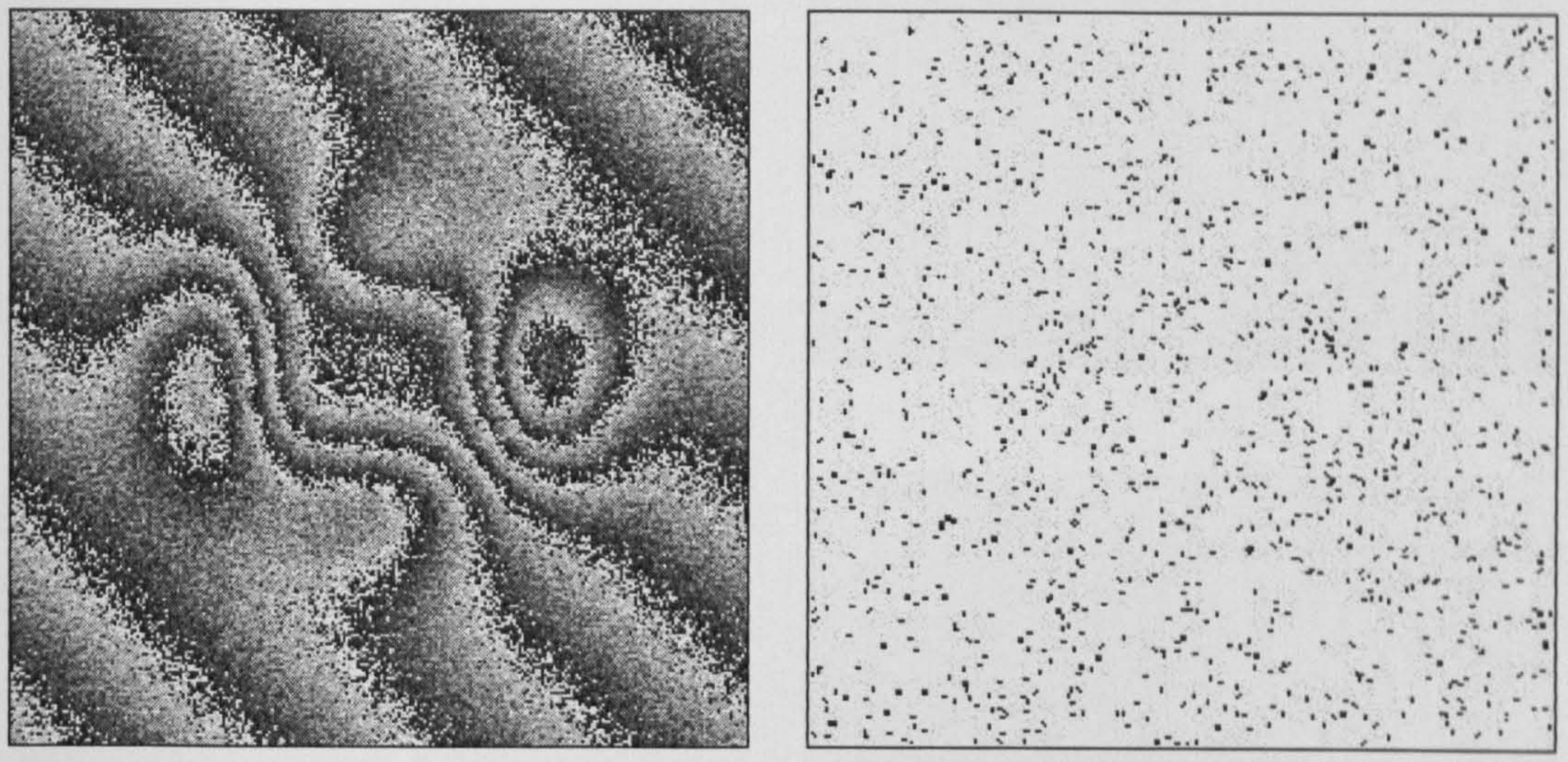
Fig. 4.17. (a) The reduction of the minimum total Cut Distances and (b) the number of similar genes in the population.

The sharp decrease in the fitness of the best solution is due to sharp increase of similar genes in all the chromosomes in the first 16 generations. This indicates that the algorithm is saving and generating good genes, in other words, good solutions. As the

algorithm exceeds the 16$^{th}$ generation, it starts searching for the optimum solution out of the good solutions that it has created so far reaching the convergence point at the global optimum solution in the last few generations.

### 4.8.4. Computer Simulation Results

A simulated wrapped phase map with 2501 residues was used to evaluate the performance of the proposed algorithm. The wrapped phase map and its corresponding residue map are shown in Figs. 4.18(a) and (b) respectively.



(a) (b)

Fig. 4.18. (a) A 256×256 noisy simulated wrapped phase map, (b) its corresponding residue map containing 2501 residues; 1255 positive residues and 1246 negative residues.

Due to the nature of the distribution of residues in the phase map, a nearest-neighbour algorithm can achieve the global optimum solution. On the other hand, the RSA and HGA algorithms use a nearest-neighbour initial solution, thus, both algorithms do not change the solution because it is already a global optimum. The performance of the HGA is then compared with the performance of the SA algorithm on the basis of no initial solution. Since in this case, both algorithms do not use an initial solution, the performance of HGA with respect to SA can be clearly seen in Fig. 4.20. HGA achieves the global optimum in only one tenth of the time required by SA. The proposed HGA (without an initial solution) was implemented on the wrapped phase map shown in Fig. 4.18(a) and the resultant branch-cut distribution has a total cut length of 1345.32. However, the minimum-cost-matching algorithm achieved a total cut length of 1346.97. The corresponding unwrapped phase maps for the phase map in Fig. 4.18(a) are shown

in Figs. 4.19(a), (b), (c) and (d) for SA, RSA, minimum-cost-matching and HGA (without initial solution) respectively.

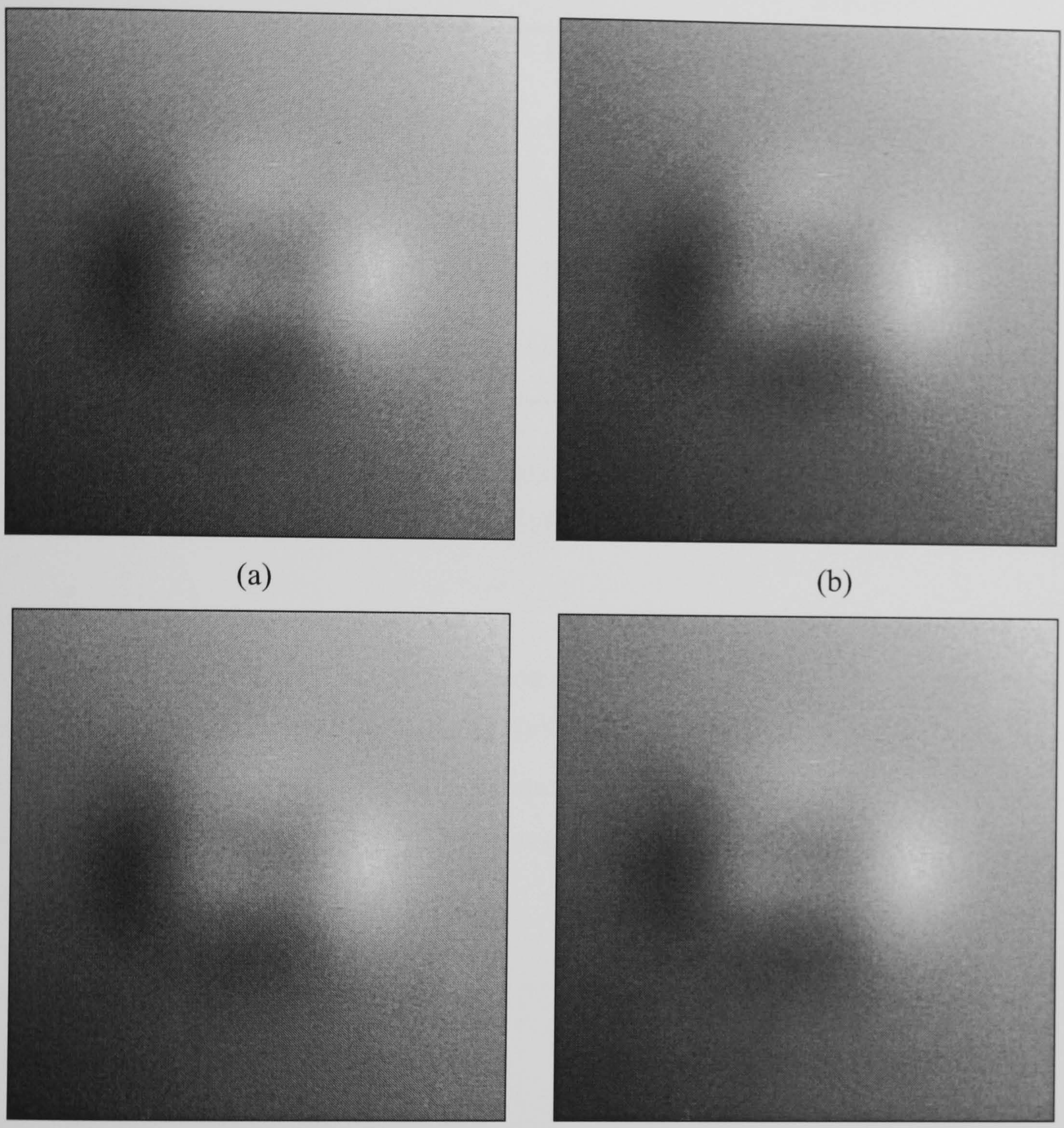



Fig. 4.19. The unwrapped phase map for the simulated wrapped phase map in Fig. 14(a) achieved using (a) SA, (b) RSA (without heating), (c) minimum-cost-matching algorithm and (d) HGA (without an initial solution).

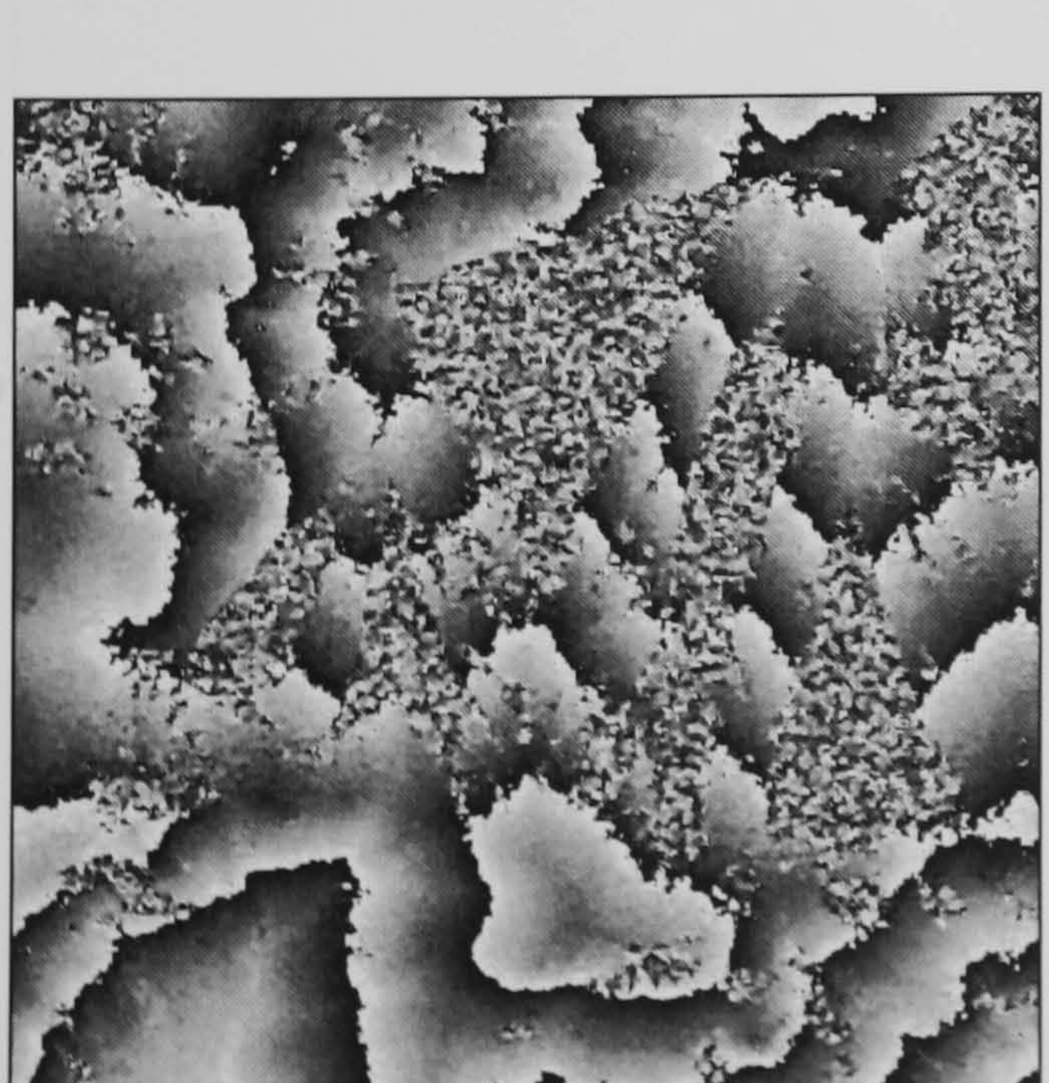
The HGA result in Fig. 4.19 (d) was achieved using a population of 100 chromosomes, $P_x = 0.9$, $P_m = 0.01$ and in 20 generations. The HGA algorithm uses RI to create the initial population based on ARMCB coding. It can be seen that the HGA is a very robust algorithm for phase unwrapping, with a very competitive execution time in comparison to the SA algorithm. Execution time results of the HGA and other phase unwrapping algorithms are shown in Fig. 4.20.

| Algorithm | Time (sec) | Total Cut Distance | Unweighted $L^0$-Measure |
|---|---|---|---|
| SA | 1870 | 1346.31 | 0.021199 |
| HGA (*RI*) | 159 | 1345.32 | 0.021199 |
| RSA | 2 | 1345.32 | 0.021167 |
| MCM | 1 | 1346.97 | 0.021167 |
| HGA (*NNR2OPTI*) | 2 | 1345.32 | 0.021167 |

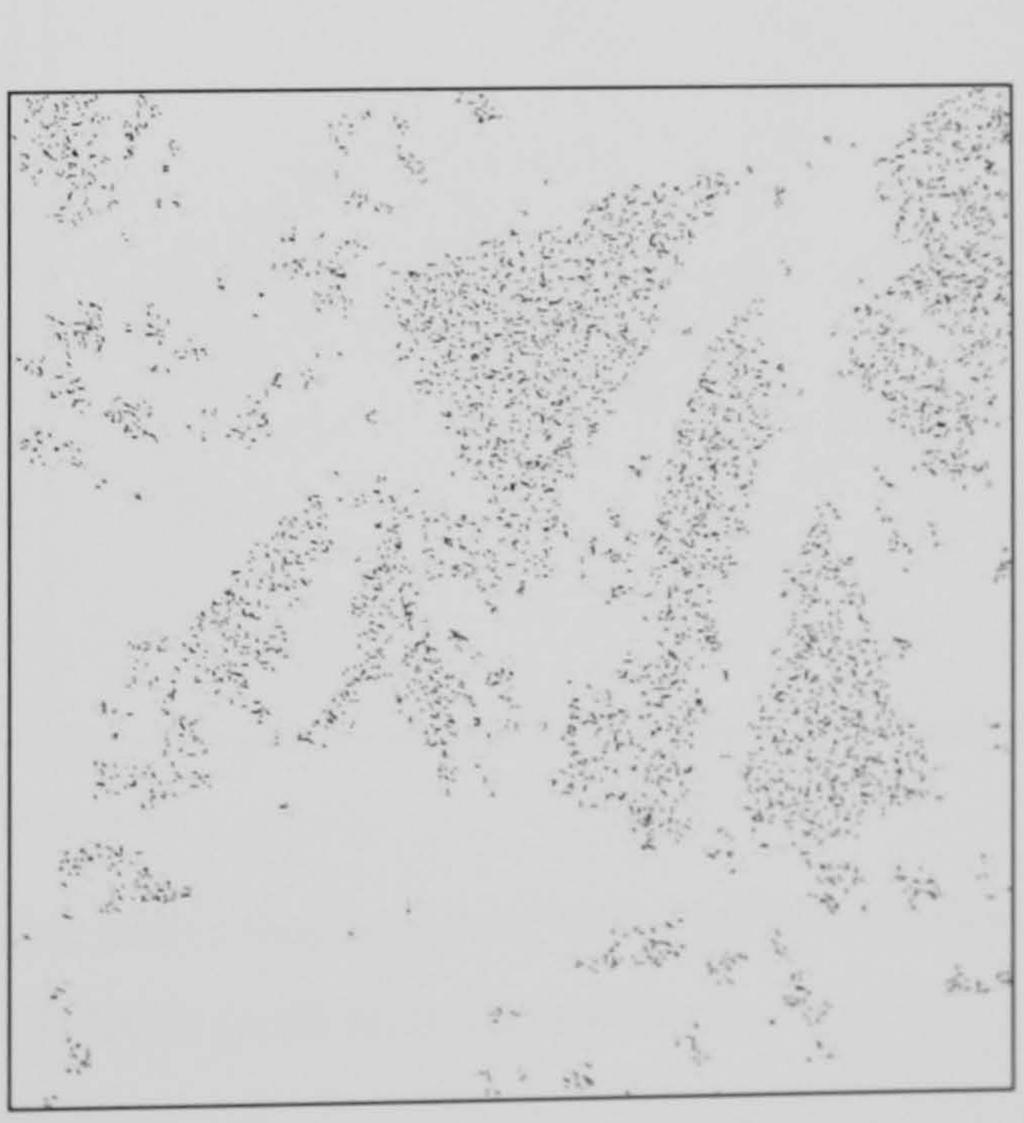
Fig. 4.20. A comparison of the HGA execution time, Total cut-length distance and unweighted $L^0$-measure with other algorithms for the noisy wrapped phase map.

### 4.8.5. Experimental Results

The proposed HGA was also implemented on a real wrapped phase map generated from Interferometric Synthetic Aperture Radar (IFSAR) data [Ghiglia and Pritt (1998)]. The IFSAR wrapped phase map and its corresponding residue distribution are shown in Figs. 4.21 (a) and (b) respectively. This wrapped phase map contains 5877 residues located in noisy patches.



(a)



(b)

Fig. 4.21. (a) A 512×512 noisy IFSAR wrapped phase map obtained from [Ghiglia and Pritt (1998)] and (b) its corresponding residue map containing 5877 residues; 2940 positive residues and 2937 negative residues.

The HGA algorithm uses *NNR2OPTI* to create the initial population and 100 chromosomes with a crossover probability $P_x$ and mutation probability $P_m$ set to the values 0.9 and 0.01 respectively. The chromosomes are coded using *ARMCB* coding method. For the presented experimental results, only four nearest neighbour solutions were inserted into the initial population.

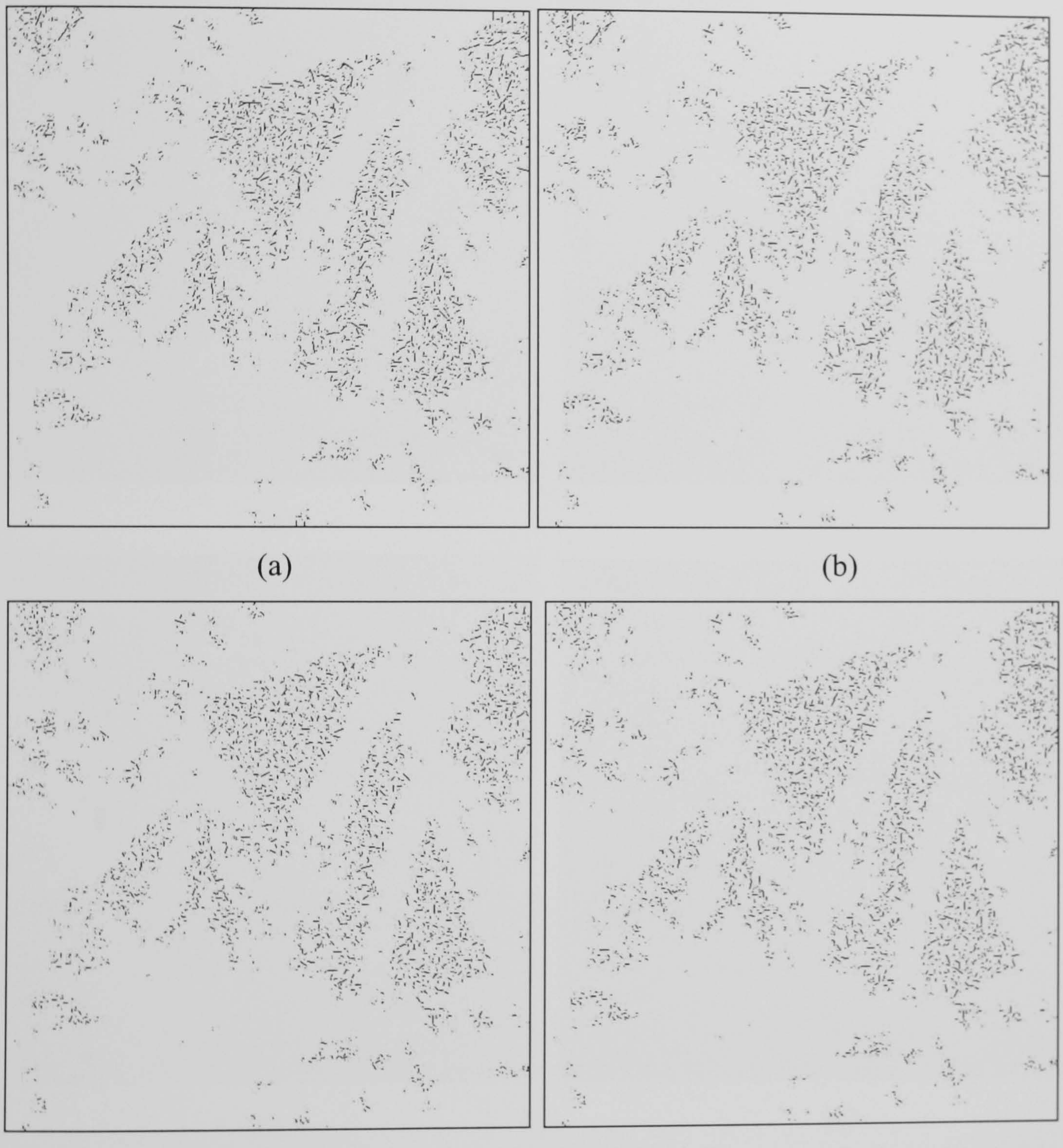



Fig. 4.22. The distribution of Branch-cuts in the residue phase map achieved using (a) SA, (b) RSA (with heating), (c) minimum-cost-matching algorithm and (d) HGA (with an initial solution).

The HGA was implemented on the IFSAR wrapped phase map. The branch-cut distributions of the SA, RSA, minimum-cost-matching and HGA are shown in Figs. 4.22(a), (b), (c) and (d) respectively. Both SA and RSA algorithms were altered by

adding a mutation operator used in the HGA to speed up their execution time. The corresponding unwrapped phase maps for the SA, RSA, Minimum-cost-matching and HGA are shown in Figs. 4.23(a), (b), (c) and (d) respectively. The execution time results of the HGA and other phase unwrapping algorithms are shown in Fig. 4.24.

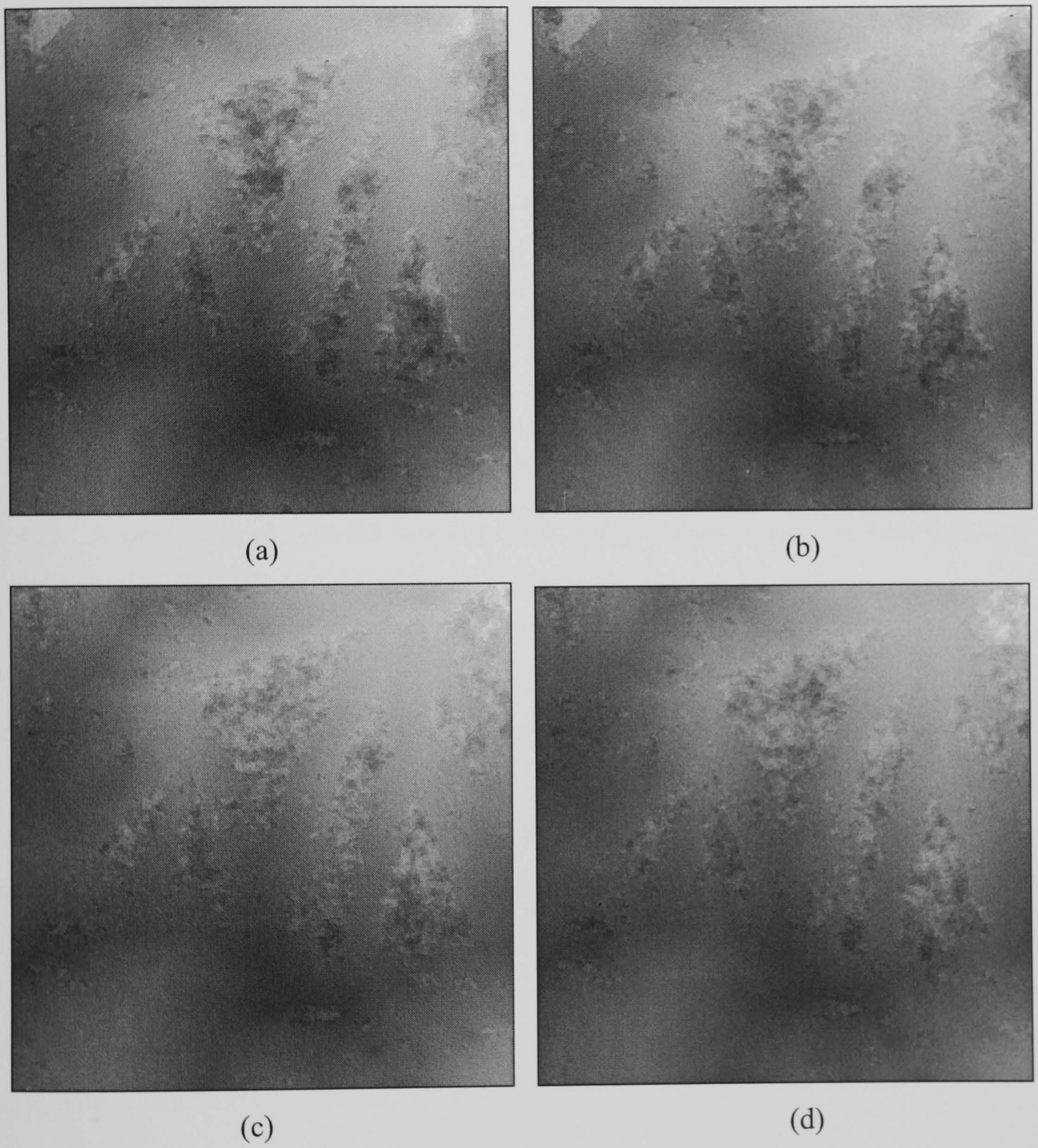


Fig. 4.23. The unwrapped phase map for the noisy IFSAR wrapped phase map in Fig. 17(a) achieved using (a) SA, (b) RSA (with heating), (c) minimum-cost-matching algorithm and (d) HGA (with an initial solution).

As can be seen from Fig. 4.24, HGA achieves better total cut distance and $L^0$-measure results than SA and RSA. However, it did not find a better solution than that of the Minimum-cost matching algorithm. This is due to the low number of chromosomes with

respect to the number of genes present. On the other hand. the low number of chromosomes was used to achieve an optimum solution using the HGA in a very short period of time as demonstrated in table 2. HGA has the shortest execution time of all the algorithms tested.

| Algorithm | Time (sec) | Total Cut Distance | Unweighted $L^0$-measure |
|---|---|---|---|
| SA | 1128 | 7042.87 | 0.028843 |
| RSA | 60 | 6831.32 | 0.028219 |
| MCM | 859 | 6611.61 | 0.027170 |
| HGA (*NNR2OPTI*) | 42 | 6704.39 | 0.027654 |

Fig. 4.24. A comparison of the HGA execution time, total cut-length distance and unweighted $L^0$-measure with other algorithms for the IFSAR wrapped phase map.

## 4.9. Conclusion

In conclusion, it can be seen from the experiments presented that the developed HGA as an artificial intelligent method is more robust and faster than the SA and RSA AI methods. This is due to the memory factor (chromosomes) in the form of solving more than one solution at a time that HGA has. However, SA and RSA lack this memory factor instead they solve only using one solution. The memory factor is efficient in achieving better results than other AI methods in a shorter execution time that is what makes HGA a better method for solving the phase unwrapping problem than any other best to date AI method like SA and RSA even the GA method.

On the other hand, it was also found that the proposed HGA is faster in reaching a global optimum solution than the most robust graph theory algorithm such as MCM algorithm. This is due to the stochastic search that HGA uses and due to the hybridism between the local and global searches used by this proposed algorithm.

# Chapter 5

# Hybrid Genetic Algorithm using a Parametric Method to Solve the Two-Dimensional Phase Unwrapping Problem

# Chapter 5

## 5. Hybrid Genetic Algorithm using a Parametric Method to Solve the Two- Dimensional Phase Unwrapping Problem

### 5.1. Introduction

In this chapter, a global phase unwrapping algorithm is proposed that uses a genetic algorithm (GA) to estimate the parameter coefficients of an $n^{th}$-order polynomial used to create the unwrapped phase solution that minimizes the $L^p$-norm error between the gradient of the solution and the gradient of the wrapped phase map. This method is similar in concept to least-squares and $L^p$-norm phase unwrapping methods developed by Ghiglia [Ghiglia and Pritt (1998)] except it does not rely directly on the wrapped phase data to construct the unwrapped solution. Instead, it uses a polynomial to construct the unwrapped surface solution. The reason for this is that in this method we approach the wrapped and the unwrapped phase maps as if completely independent from each other, such that the only relation between them being the $L^p$-norm error minimization. The other advantage of the proposed algorithm is that it generates a noise-free unwrapped phase map and achieves a global smoothness constraint.

The proposed algorithm as mentioned previously relies on estimating the parameters of an $n^{th}$ order-polynomial to approximate the unwrapped surface solution from the wrapped phase data. The proposed algorithm uses a genetic algorithm to obtain the coefficients of the polynomial that best unwrap the wrapped phase map. However, by providing the genetic algorithm with an initial population created by randomly choosing a number between two limits will cause the algorithm to take a very long time to converge to the global optimum solution. A faster way was achieved by obtaining an initial solution that is used to create the initial population.

In this proposed algorithm, the complexity of the problem relies on the order of the polynomial used to reconstruct the unwrapped surface solution. By increasing the order of the polynomial, more precision and a lower minimum $L^p$-norm error are achieved. The number of coefficients of the polynomial also increases with the order of the polynomial. This proposed algorithm is mainly applicable to contiguous phase distributions (albeit with gaps).

The proposed algorithm is summarised in Fig. 5.1:

- Calculate the quality map of the wrapped phase map using maximum phase gradient quality map,
- Unwrap the wrapped phase using quality guided algorithm [Ghiglia and Pritt (1998)],
- Surface-fit the unwrapped solution with a polynomial using weighted least-square multiple regression controlled by the quality map weights.
- Coefficients of the surface-fitted polynomial are given to the genetic algorithm as an initial solution to lower execution time,
- Genetic Algorithm minimizes the $L^p$-norm error between the gradient of the polynomial unwrapped surface solution and the gradient of the original wrapped phase map.

Fig. 5.1. A summary of the proposed parameter estimation genetic algorithm.

### 5.1.1. Polynomial Surface-fitting Weighted Least-Square Multiple Regression

Surface-fitting using polynomials is a very well established subject used to best fit a polynomial to set of data. One way to surface fit a polynomial to a set of data is by weighted least-square multiple regression. This method of regression minimzes the sum of residuals (least square error or $L^2$-norm) controlled by a set of weights. It involves smoothing of the data or identifying an apparent trend in the data. The weights used define how good the data to be fitted is and how much they can contribute to the fitting of the polynomial to the data. The number of coefficients of the polynomial specifies the size of the matrices used to solve the least-square problem.

The best fitted polynomial surface has the minimum weighted least square error defined in Eq. (5.1):

$$S = \sum_{i=1}^{NM} w_i \left[ z_i - f(x_i, y_i) \right]^2 \tag{5.1}$$

where $w_i$ is the weight at at pixel i.

$z_i$ is the unwrapped data pixel to be fitted,

$f(x_i, y_i)$ is the pixel value evaluated using the polynomial in Eq. (5.1) at the coordinates $x_i$ & $y_i$ in the x and y direction respectively.

The coeficients of the n[th]-order polynomial are the unknowns which need to be evaluated to construct the surface. However, the known parameters are the data to be fitted $z_i$ and the two control points defined by the $x_i$ & $y_i$ coordinates of the polynomial surface in the x and y directions. To obtain the weighted least-square error, the unknown polynomial coeficients must yield zero first derivatives.

$$\begin{cases} \dfrac{dS}{da_0} = 2\sum_{i=1}^{NM} w_i\left[z_i - f(x_i, y_i)\right]\dfrac{df}{da_0} = 0 \\ \dfrac{dS}{da_1} = 2\sum_{i=1}^{NM} w_i\left[z_i - f(x_i, y_i)\right]\dfrac{df}{da_1} = 0 \\ \dfrac{dS}{da_2} = 2\sum_{i=1}^{NM} w_i\left[z_i - f(x_i, y_i)\right]\dfrac{df}{da_2} = 0 \\ \vdots \\ \dfrac{dS}{da_m} = 2\sum_{i=1}^{NM} w_i\left[z_i - f(x_i, y_i)\right]\dfrac{df}{da_m} = 0 \end{cases} \tag{5.2}$$

By expanding Eq. (5.2) will result in the set of linear equations presented in Eq. (5.3):

$$\begin{cases} a_0\sum_{i=1}^{NM} w_i + a_1\sum_{i=1}^{NM} w_i x_i + a_2\sum_{i=1}^{NM} w_i y_i \cdots\cdots + a_m\sum_{i=1}^{NM} w_i y_i^n = \sum_{i=1}^{NM} w_i z_i \\ a_0\sum_{i=1}^{NM} w_i x_i + a_1\sum_{i=1}^{NM} w_i x_i^2 + a_2\sum_{i=1}^{NM} w_i x_i y_i \cdots\cdots + a_m\sum_{i=1}^{NM} w_i x_i y_i^n = \sum_{i=1}^{NM} w_i x_i z_i \\ a_0\sum_{i=1}^{NM} w_i y_i + a_1\sum_{i=1}^{NM} w_i x_i y_i + a_2\sum_{i=1}^{NM} w_i y_i^2 \cdots\cdots + a_m\sum_{i=1}^{NM} w_i y_i^{n+1} = \sum_{i=1}^{NM} w_i y_i z_i \\ \vdots \\ a_0\sum_{i=1}^{NM} w_i y_i^n + a_1\sum_{i=1}^{NM} w_i x_i y_i^n + a_2\sum_{i=1}^{NM} w_i y_i^{n+1} \cdots\cdots + a_m\sum_{i=1}^{NM} w_i y_i^{2n} = \sum_{i=1}^{NM} w_i y_i^n z_i \end{cases} \tag{5.3}$$

Eq. (5.3) can be expressed in the matrix form in Eq. (5.4):

$$\begin{bmatrix} \sum_{i=1}^{NM} w_i & \sum_{i=1}^{NM} w_i x_i & \sum_{i=1}^{NM} w_i y_i & \cdots & \sum_{i=1}^{NM} w_i y_i^n \\ \sum_{i=1}^{NM} w_i x_i & \sum_{i=1}^{NM} w_i x_i^2 & \sum_{i=1}^{NM} w_i x_i y_i & \cdots & \sum_{i=1}^{NM} w_i x_i y_i^n \\ \sum_{i=1}^{NM} w_i y_i & \sum_{i=1}^{NM} w_i x_i y_i & \sum_{i=1}^{NM} w_i y_i^2 & \cdots & \sum_{i=1}^{NM} w_i y_i^{n+1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \sum_{i=1}^{NM} w_i y_i^n & \sum_{i=1}^{NM} w_i x_i y_i^n & \sum_{i=1}^{NM} w_i y_i^{n+1} & \cdots & \sum_{i=1}^{NM} w_i y_i^{2n} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{NM} w_i z_i \\ \sum_{i=1}^{NM} w_i x_i z_i \\ \sum_{i=1}^{NM} w_i y_i z_i \\ \vdots \\ \sum_{i=1}^{NM} w_i y_i^n z_i \end{bmatrix} \quad (5.4)$$

Eq. (5.4) can be simplified to the following matrix form in Eq. (5.5):

$$\boldsymbol{XA} = \boldsymbol{Z} \quad (5.5)$$

where $\boldsymbol{A}$ is the matrix representing the polynomial coefficients,

$\boldsymbol{X}$ is the matrix representing the left side of Eq. (5.4),

$\boldsymbol{Z}$ is the matrix representing the right side of Eq. (5.4).

By reordering the matrices, the coeficient matrix can be evaluated from Eq. (5.6):

$$\boldsymbol{A} = \boldsymbol{X}^{-1}\boldsymbol{Z} \quad (5.6)$$

This can be solved by Gaussian Elimination to calculated the values of the coefficient of the best fit polynomial surface that can minimises the weighted least-square error. Then, the coefficients are optimized by the hybrid genetic algorithm.

## 5.2. Coding the Phase Unwrapping problem in GA Syntax Form

Any optimisation problem using a GA requires the problem to be coded into GA syntax form, which is the chromosome form. In this problem, the chromosome consists of a number of genes where every gene corresponds to a coefficient in the $n^{th}$-order surface fitting polynomial as described int Eq. (5.7) and Fig. 5.2.

$$f(x,y) = a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2 + \ldots + a_m y^n \quad (5.7)$$

where $a_{[0..m]}$ are the parameter coefficients that will be estimated by the genetic algorithm to approximate the unwrapped phase that can achieve the minimum $L^p$-norm,

$x$ and $y$ are indices of the pixel location in the image respectively, $m$ is the number of coefficients.

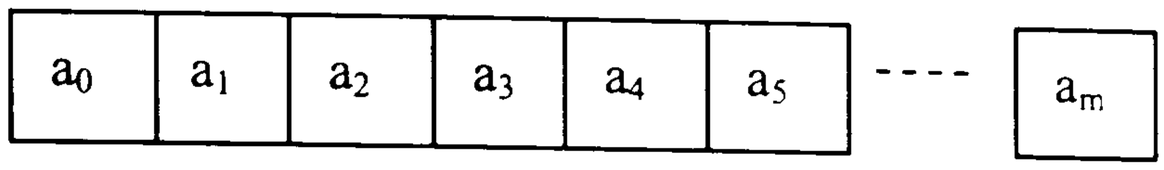



Fig. 5.2. Coding scheme of the coefficients of the $n^{th}$-order surface fitting polynomial into the chromosome syntax form.

## 5.3. Initial Population

A GA requires an initial population of chromosomes where each chromosome represents a possible solution. The method that is used to create the initial population will determine the speed of convergence to an optimum solution, as well as the size of the population (the number of chromosomes in the population).

### 5.3.1. Initial Solution

The initial population is generated by creating an initial solution using one of the simple phase unwrapping algorithms such as 'Quality guided phase unwrapping algorithm' [Ghiglia and Pritt (1998)]. The initial solution is approximated using a 'polynomial Surface-fitting weighted least-square multiple regression' method presented in section 5.1.1. The number of coefficients required to represent a surface relies on the polynomial order which can be estimated by solving the regression with an increasing polynomial order such that the polynomial order with the minimum $L^2$-norm error (not exceeding a user defined error threshold) is used in the genetic algorithm. In essence, using the multiple regression method, the initial coeficients of the polynomial will be generated. These coefficients are, then, inserted into the first chromosome of the genetic algorithm initial population. This method for creating an initial solution gives the GA a good start to reach convergence. It is more intelligent and problem-specific than random initialization of polynomial coeficients.

### 5.3.2. Generating an Initial Population Based on the Initial Solution

Once the initial solution coefficients are calculated, the coefficients of the initial solution are inserted in the first chromosome in the initial population. The rest of the population is generated using the following method: for every gene in each chromosome in the population, a small number, depending on the precision of the gene, is added or subtracted to the value of the gene as in Eq. (5.8):

$$a_k = a_k + (\pm 1)\left\{10^{[\log(a_k)+rand]}\right\} \tag{5.8}$$

where $a_k$ is the coefficient parameter stored in gene '$k$', *rand* is a random number generated between the values $rand \in [0,17]$. This is because the random numbers are represented in the computer using double precision which has 17 decimal points.

## 5.4. Fitness Evaluation

To find the global optimum solution to the parameter estimation $L^2$-norm phase unwrapping problem, the quality of the solution must be evaluated at every generation in order to inform the genetic algorithm of how good its current solution is at each stage. The evaluation will increase the knowledge of the GA of the quality level of the solution. The genes of a chosen chromosomes are substituted as coeficients in Eq. (5.7) to evaluate the approximated phase value at coordinate $(x, y)$. Then, the obtained unwrapped phase solution is phase matched with wrapped phase to force the unwrapped surface to be congruent to wrapped phase using Eq. (5.9). The phase matched unwrapped solution is substituted in Eq. (2.14) to evaluate the error in the $L^2$-norm sense.

## 5.5. Greedy 2-Point Crossover

The crossover is a natural operator used to generate new chromosomes from original chromosomes. It is an important operator in genetic algorithms because it introduces diversity into chromosomes. It is capable of combining schemas (important genes) located in the original chromosomes. It avoids destroying the schemas in the original chromosomes, in essence, good schemas are conserved and propagated into the new chromosomes. Thus, it has a tendency to create a better chromosome rather than weaker one. A greedy method was also used in this crossover operator to ensure only best fit chromosomes are allowed to propagate into the new generation.

The greedy 2-point crossover is summarized as follows:

- Choose two chromosomes from the initial population for mating with a crossover probability; $P_x$,
- Choose randomly two crossover points,
- Swap the genes lying between these two points of one the chromosomes with the genes lying between the same points in the other chromosome; and visa versa.

- Evaluate the fitness of both new generated chromosomes,
- Compare the fitness of the new generated chromosomes with that of the original chromosomes,
- Choose the two best fit chromosomes from the original and the new chromosomes to be added into the new population.

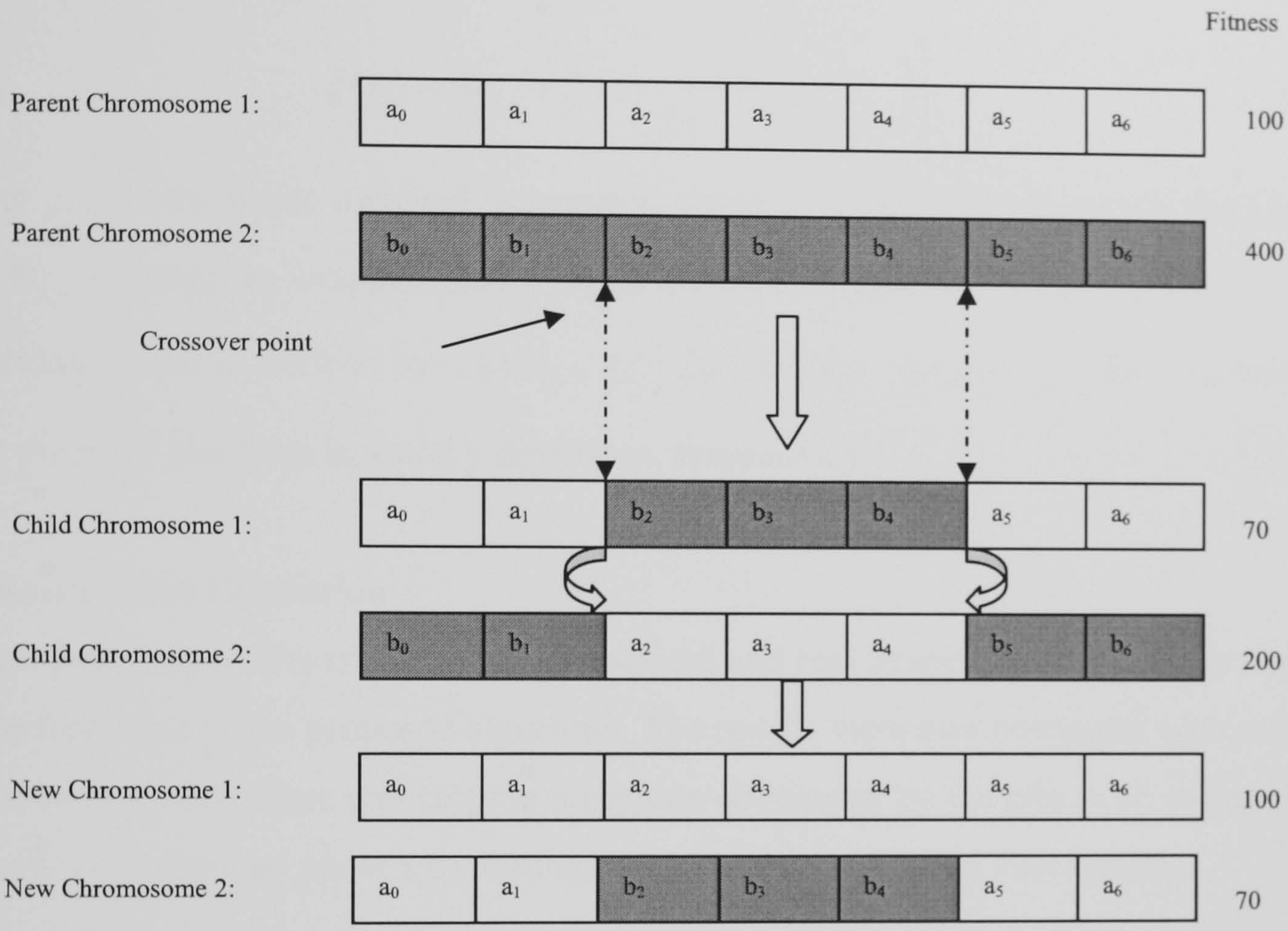



Fig. 5.3. The basic steps in the greedy 2-point crossover.

## 5.6. Mutation Operator

Mutation operator applies deliberate changes to a gene at random, to keep variation in genes and to increase the probability of not falling into a local minimum solution. It involves exploring the search space for new better solutions. This proposed operator uses a greedy technique which ensures only the best fit chromosome is allowed to propagate to the next generation.

This operator can be summarized as follows:

- Choose a chromosome from the current population at a probability; $P_m$,
- For every gene in the chromosome a random number is added to the value of the gene as in Eq. (5.8).

## 5.7. Phase Matching

This method matches the phase of the wrapped phase with approximated unwrapped phase to establish the best representation of the unwrapped phase. The phase matching step extracts the small details embedded in the wrapped phase data which are lost in the global phase unwrapping [Schwarz (2004)]. This step is performed using Eq. (5.9).

$$\xi(p) = \Phi_{i,j} + 2\pi\rho\left[\frac{1}{2\pi}\left(\Psi_{i,j} - \Phi_{i,j}\right)\right] \tag{5.9}$$

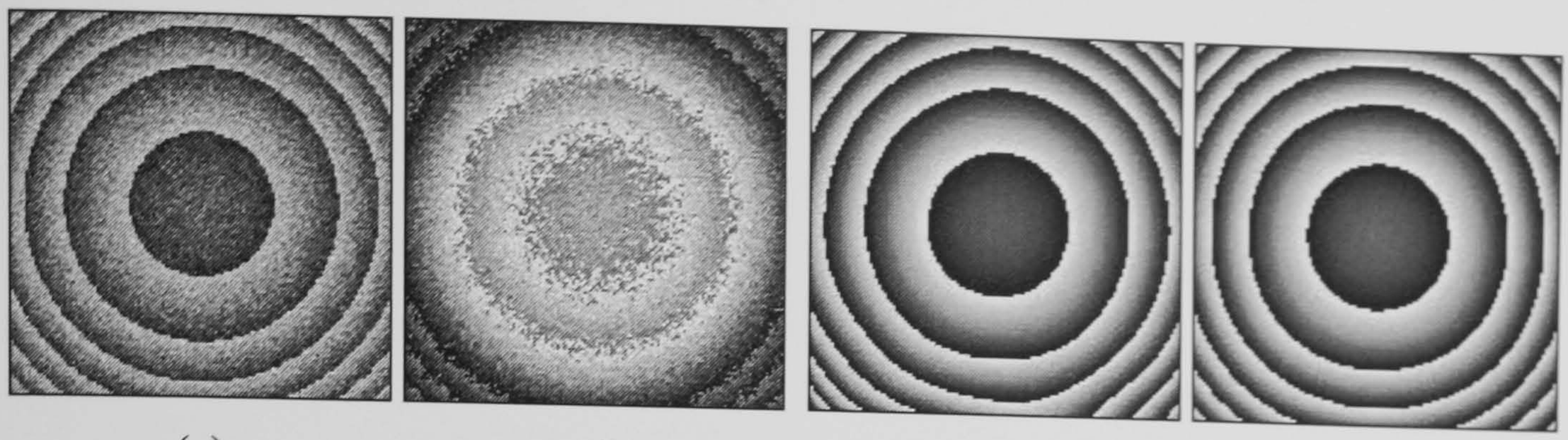
Where $\xi(p)$ is the phase matched unwrapped phase, $p$ is the pixel position in the phase map, $\Phi_{i,j}$ is the given wrapped phase, $\Psi_{i,j}$ is the approximated unwrapped phase, $\rho[.]$ is a rounding function defined by $\rho[t] = \lfloor t + 1/2 \rfloor$ for $t \geq 0$ and $\rho[t] = \lfloor t - 1/2 \rfloor$ for $t < 0$ and '$i$, $j$' are the pixel positions in x and y directions, respectively.

## 5.8. Results and Discussion

The proposed algorithm is tested using simulated and real wrapped phase maps to verify the performance of the proposed algorithm. The results were also compared with a very well known global phase unwrapping algorithm developed by Ghiglia *et al.* called '$L^p$-norm two-dimensional phase unwrapping algorithm' [Ghiglia and Pritt (1998)].

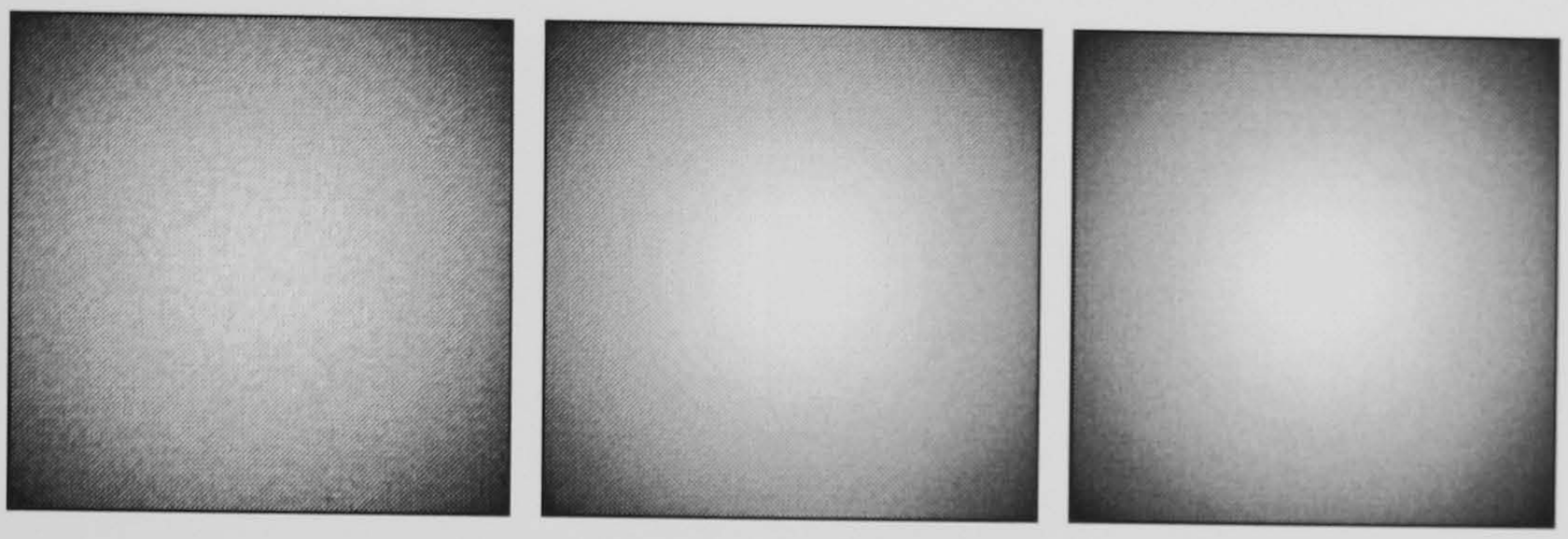
### 5.8.1. Grid Method Computer Simulated Results

The proposed algorithm was first developed based on an initial generation obtained from grid line fitting to the surface of the initial solution in the horizontal and vertical directions. In this method, every grid line polynomial coefficients are inserted as a chromosome in the initial generation. To demonstrate this technique, 32×32 grid line coefficients estimated from the surface of the initial solution of the 128×128 simulated wrapped phase map were generated spaced by 4 pixels in each grid line direction. This technique was first developed to generate the initial generation of the HGA. The proposed HGA has estimated the unwrapped surface solution in a very efficient way such that its rewrapped result best matches the wrapped phase map unlike the $L^p$-norm algorithm and the polynomial surface-fitting weighted least-square algorithm as shown in Figs. (a), (b), (c) and (d).



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

Fig. 5.4. The simulated noisy object 128×128 (a) wrapped phase map, rewrapped phase map using (b) $L^p$-norm algorithm, (c) polynomial surface-fitting weighted least-square algorithm, (d) polynomial surface-fitting using HGA.

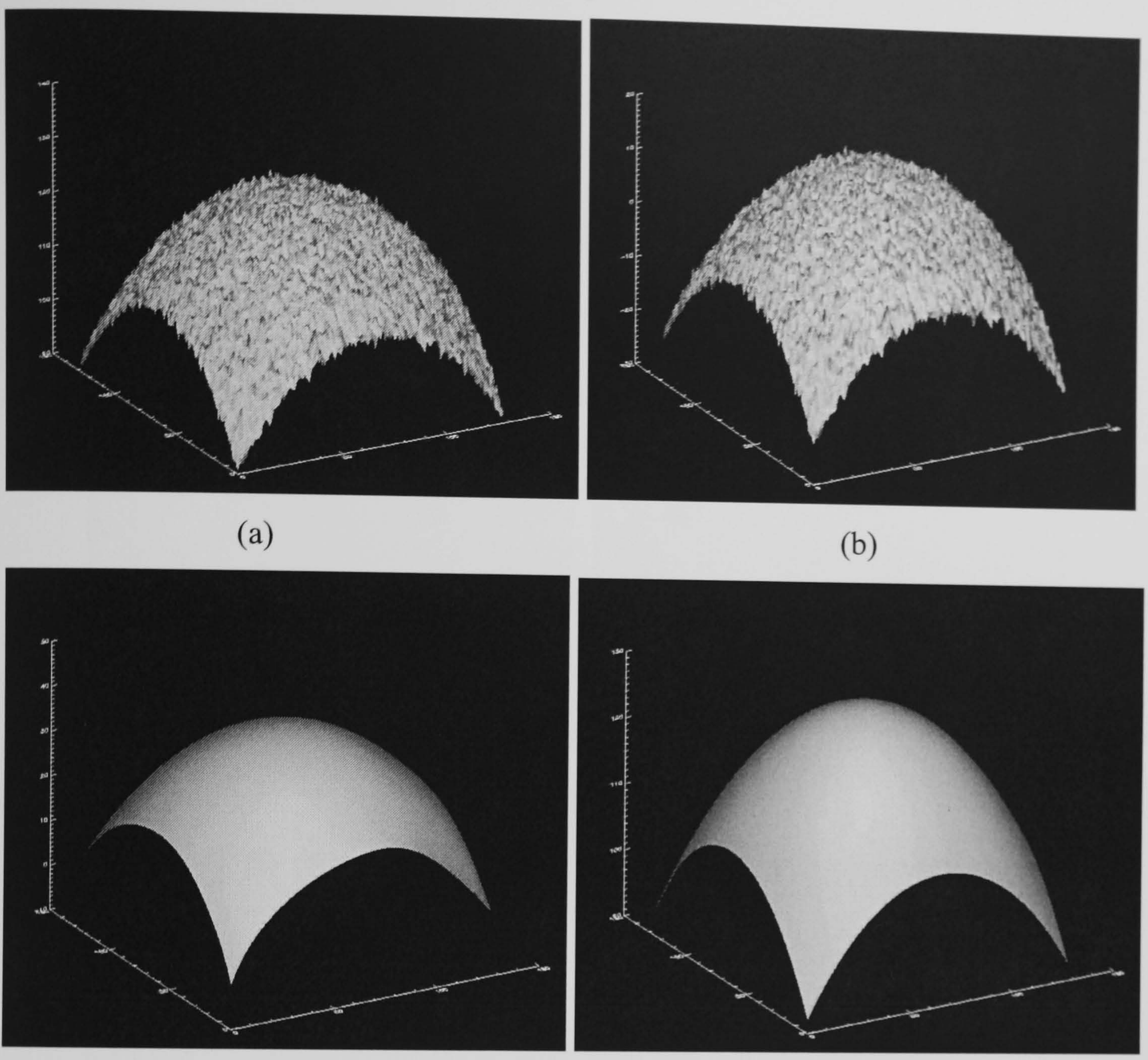


(a) (b) (c)

Fig.5.5. The simulated noisy object 128x128 unwrapped phase map using (a) $L^p$-norm algorithm, (b) polynomial surface-fitting weighted least-square algorithm, (c) polynomial surface-fitting using HGA.

Also, Figs. 5.5(a), (b) and (c) show the unwrapped solution of the $L^p$-norm algorithm, the polynomial surface-fitting weighted least-square algorithm and the polynomial surface-fitting using HGA, respectively. Moreover, Figs. 5.6(a), (b), (c) and (d) show the original surface before wrapping, the unwrapped $L^p$-norm surface, the unwrapped polynomial surface-fitting weighted least-square surface and the unwrapped polynomial surface-fitting using HGA surface, respectively. It can be seen from the unwrapped surface results that the proposed algorithm has the best matching unwrapped surface.

However, the grid method technique used to develop the HGA initial generation cannot be applied on complex surfaces but only simple surfaces. Using this method will cost the HGA an extensive amount of time to estimate and optimise unwrapped phase solutions. Thus, the grid method can only be used on simple and small wrapped phase maps using the proposed HGA. Thus, this has lead to the technique of direct surface estimation using weighted least-square regression to generate the initial solution of the proposed HGA and the results of using this method are presented and discussed in the next two sections.



(a) (b)

(c) (d)

Fig. 5.6. The simulated noisy object 128×128 (a) original 3D-unwrapped phase map using (b) $L^p$-norm algorithm, (c) polynomial surface-fitting weighted least-square algorithm, (d) polynomial surface-fitting using HGA.

### 5.8.2. Computer Simulated Results

The proposed algorithm was tested on computer simulated object with high noise and the result was compared with that of the $L^p$-norm algorithm. The order of the polynomial used to estimate the unwrapped surface was the $9^{th}$ order. The wrapped phase map and the rewrapped results of the $L^p$-norm algorithm and the proposed algorithm before and after phase matching are shown in Figs. 5.7(a), (b), (c) and (d), respectively. The proposed algorithm best matches the original wrapped phase with an advantage of smoothing the noise embedded in the wrapped phase map if the algorithm was used without phase matching.

The original surface and the unwrapped $L^p$-norm surface and the proposed algorithm unwrapped surface are presented in Figs. 5.8(a), (b) and (c), respectively. The unwrapped surface of the proposed algorithm with phase matching best matches the original object surface.

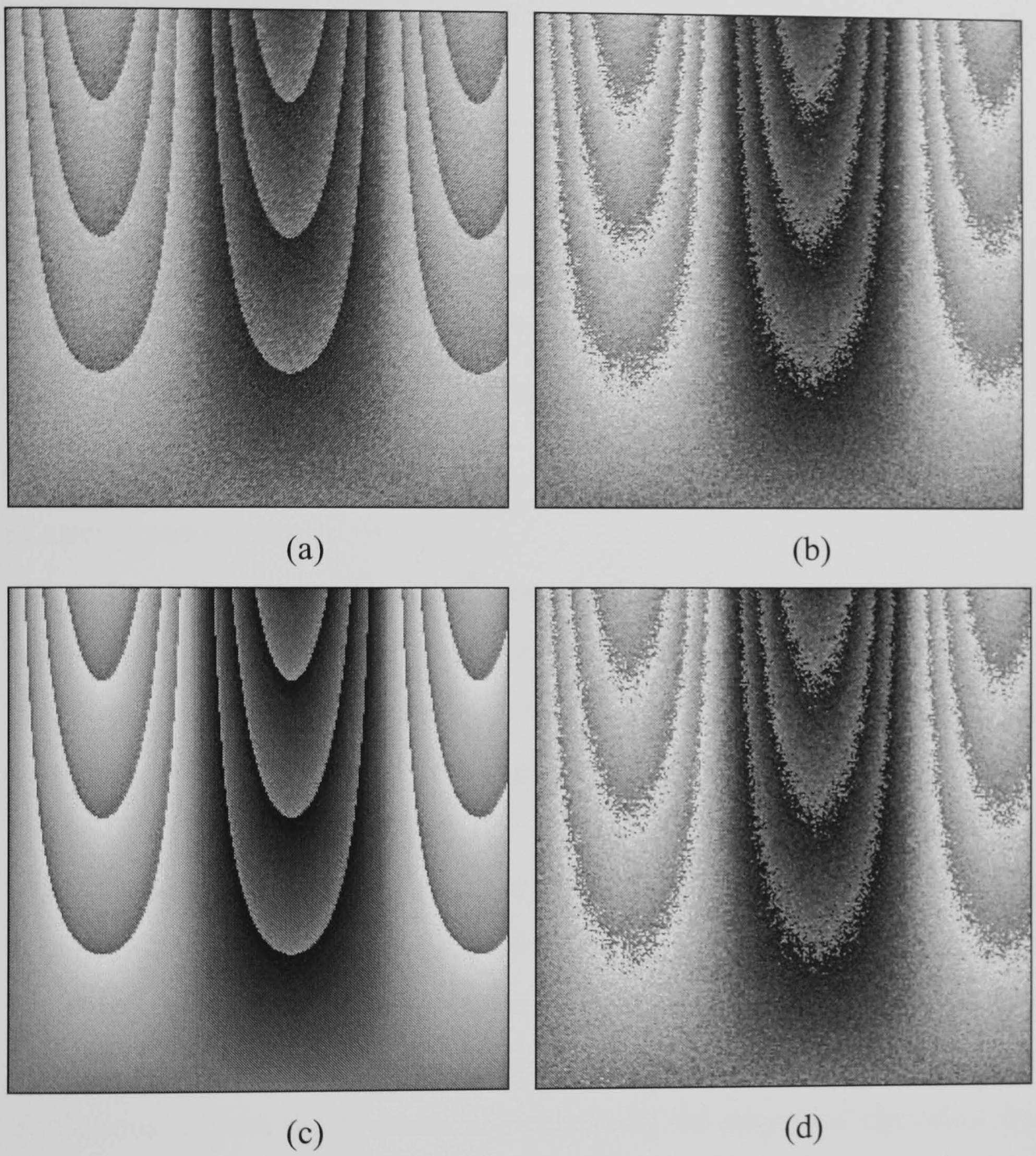
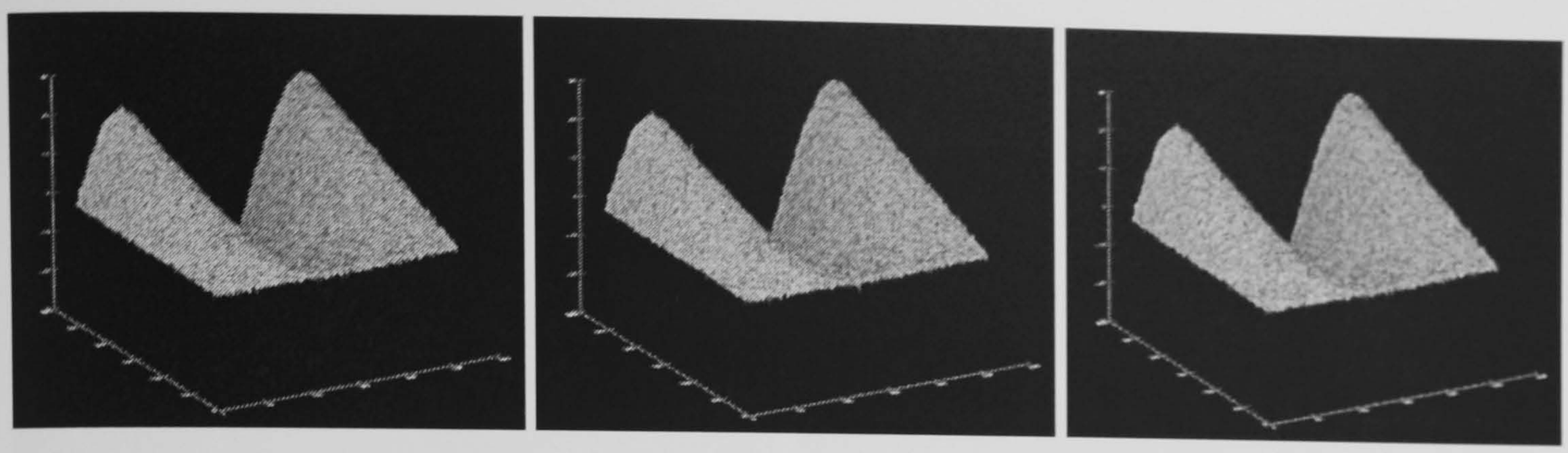



Fig. 5.7. The simulated noisy object 256×256 (a) wrapped phase map, rewrapped phase map using (b) $L^p$-norm algorithm and the proposed algorithm (c) before and (d) after phase matching.

This demonstrates the capabilities of using the weighted least-square surface-fitting regression to provide the coefficients of the initial solution to the HGA for complex object wrapped phase maps. In essence, this surface regression method is better than the grid method which was incapable of estimating the coefficients of more complex surfaces. Thus, the initial generation was easily generated and the HGA was left the task of optimizing the unwrapped solution that minimizes the $L^2$-norm error measure.



(a) (b) (c)

Fig. 5.8. Simulated noisy object 256×256 (a) original 3d-suface, unwrapped phase map using (b) $L^p$-norm algorithm and (c) the proposed algorithm.

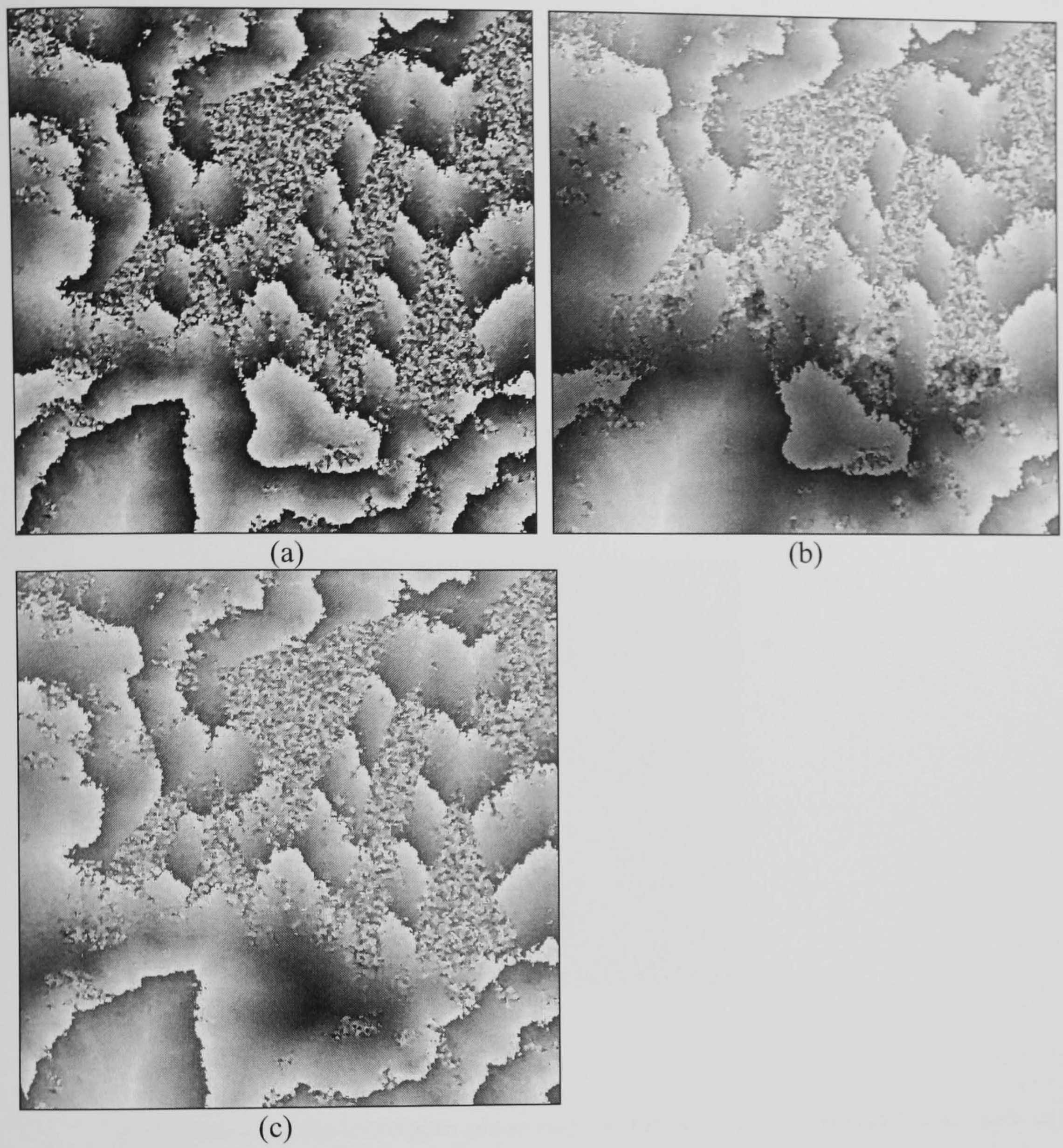
The $L^2$-norm error achieved by the proposed method on the wrapped phase map in Fig. 5.7(a) was 7.90992e+006. However, the $L^2$-norm error achieved by the $L^p$-norm method on the same wrapped phase map in Fig. 5.7(a) was 7.91294e+006.

### 5.8.3. Experimental Results

The proposed algorithm was also implemented on a real wrapped phase map generated from Interferometric Synthetic Aperture Radar (IFSAR) data [Ghiglia and Pritt (1998)]. The IFSAR wrapped phase map and the rewrapped results of the $L^p$-norm algorithm and the proposed algorithm are presented in Figs. 5.9(a), (b) and (c), respectively.

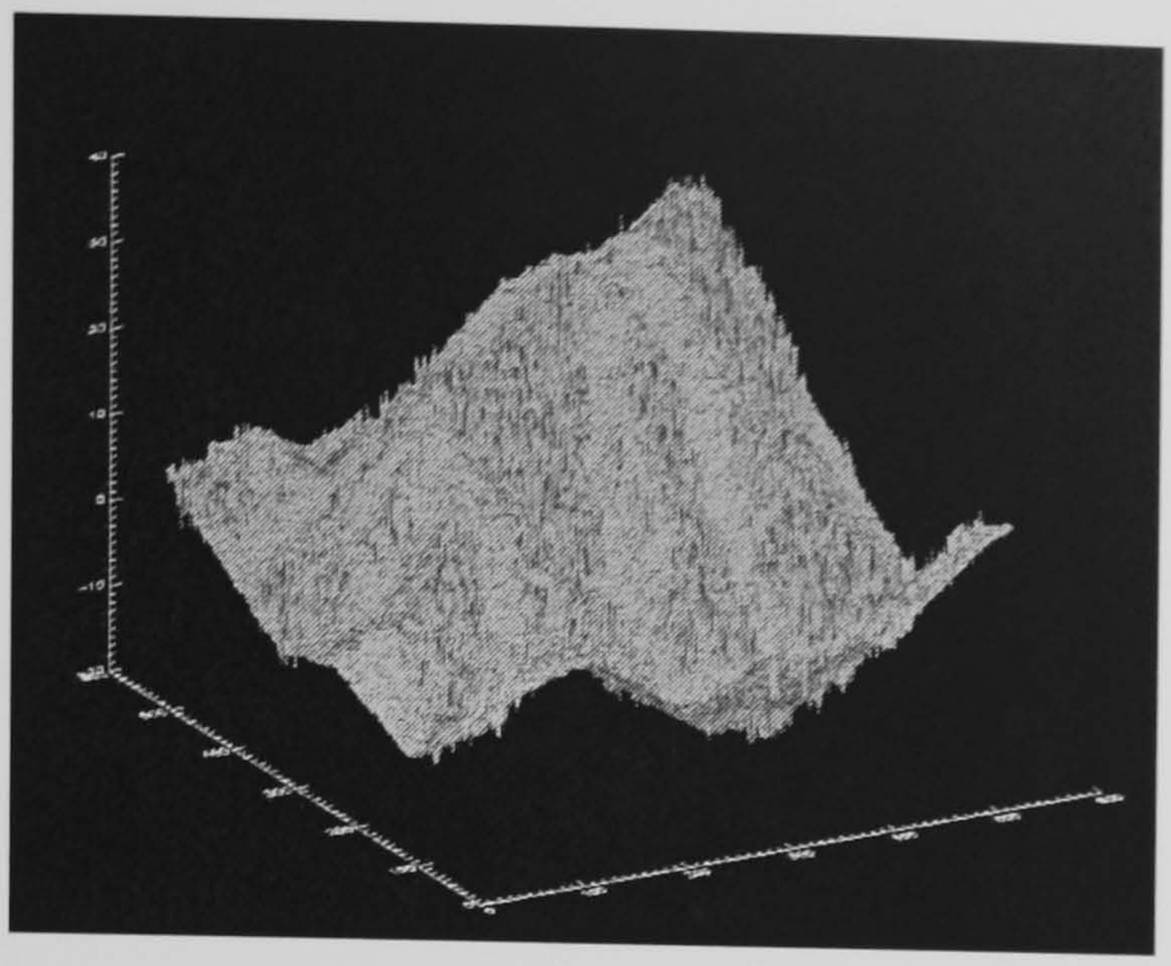
The proposed algorithm proved to be very robust in unwrapping the wrapped phase map. It achieves a solution that is the best matching rewrapped phase map with that of the original wrapped phase map. However, the $L^p$-norm algorithm did not retrieve as much of the original wrapped phase when its solution was rewrapped.The unwrapped surface solutions of both the $L^p$-norm algorithm and the proposed algorithm are shown in Figs. 5.10(a) and (b), respectively.

The $L^2$-norm error achieved by the proposed method on the wrapped phase map in Fig. 5.9(a) was 1.72796e+007. However, the $L^2$-norm error achieved by the $L^p$-norm method on the same wrapped phase map in Fig. 5.9(a) was 2.79040e+007.

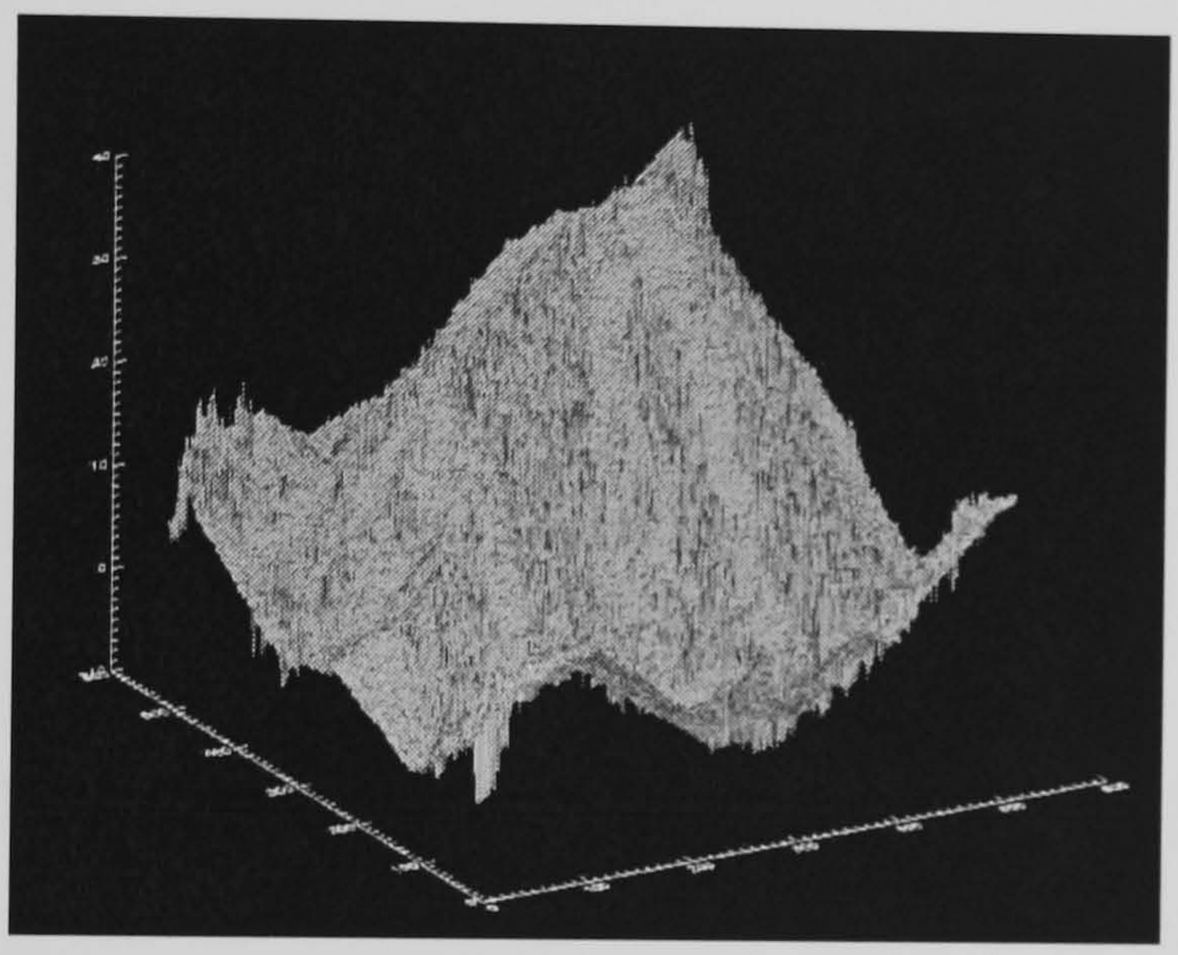


(a) (b) (c)

Fig. 5.9. (a) A 512×512 noisy IFSAR wrapped phase and the rewrapped phase map using (b) $L^p$-norm algorithm and (c) the propsoed algorithm with phase matching



(a)



(b)

Fig. 5.10. 3D-surface of the unwrapped phase map for the noisy IFSAR wrapped phase map in Fig. 5.9(a) achieved using (a) $L^p$-norm and (b) the proposed algorithm with phase matching.

## 5.9 Conclusion

In the case of complex object wrapped phase maps, the coefficients generated by the regression method as an initial solution for HGA are very small resulting to a very large search space for the HGA to find the optimum solution. In this proposed method, as the polynomial order increases and the object surface become more complex, the polynomial coefficients become more smaller as the polynomial order increase. In essence, the difference between coefficients in the polynomial is big thus creating a large search space. In conclusion, the proposed algorithm is robust and efficient but it has high computation time burden. This could be improved by using overlapping windows that divide the wrapped phase map into regions that can be processed separately by the proposed algorithm.

# Chapter 6

# Branch-Cut Placement in Phase Unwrapping Using Residue-Vectors

# Chapter 6

## 6. Branch-Cut Placement in Phase Unwrapping Using Residue-Vector

### 6.1. Introduction

A new method for branch-cut placement is proposed which relies on information provided by what we will term the residue-vector. This method is expected to lower the computation complexity while at the same time improving the quality of the unwrapped phase map. Moreover, it will speed up phase unwrapping algorithms such as the minimum cost flow. It also could be considered as a weighting factor generator for a different range of existing phase unwrapping algorithms such as those outlined in Fig. 6.1.

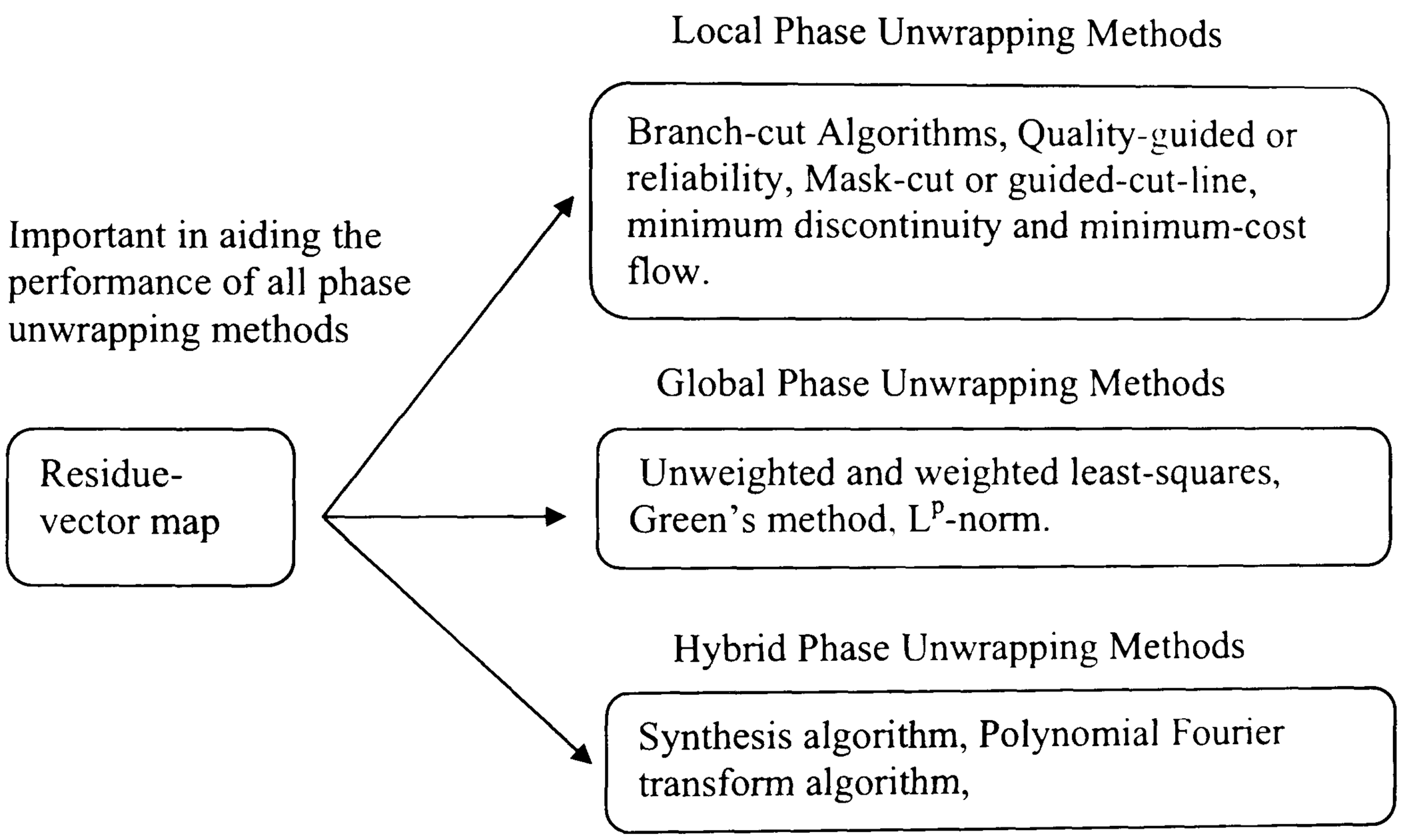



Fig. 6.1. A diagram illustrating that the proposed residue-vector map will be an aid to all existing phase unwrapping algorithms to create an optimum unwrapped solution.

The residue-vector can be used as a weighting factor to optimize the results of algorithms employing local, global and hybrid phase unwrapping methods. It can optimize the results of the phase unwrapping to a degree never previously achieved by existing weighting quality map methods such as: phase variance, maximum gradient, pseudo-correlation, correlation, *etc*. This residue-vector map created by residue-vector extraction is more problem-specific to the phase unwrapping problem. It combines both the concepts of quality maps and the knowledge of the residues and their branch-cuts based

on the fact that residues are the cause of the problems of the phase unwrapping in the first place..

It can also be a post-processing technique to algorithms such as minimum discontinuity and minimum cost flow. This post-processing technique is called residue-vector matching. By post-processing, it will further optimize generated solutions to include more details embedded in the wrapped phase map.

## 6.2. The Residue-Vector

### 6.2.1. Definition

Residues in a wrapped phase map generate a vector which can be viewed by calculating and displaying the first derivative in x and y directions (*i.e.*, dx and dy). This residue-vector can be seen in the dx and dy corrected gradient phase maps in Figs. 6.2(c) and (d), respectively extracted form the wrapped phase map in Fig. 6.2(a) of the simulated spiral object shown in Fig. 6.2(e) and its residue distribution is shown in Fig. 6.2(b). Figs. 6.2(c) and (d) show the behaviour of the residue-vector in the x and y directions. In Fig. 6.2(f), an enlarged single residue-vector is shown to illustrate how the residue-vector appears in the gradient phase map. Usually residue-vector orientation follows edges, shadows, areas with under-sampling and phase noise.

A ***residue-vector*** is a vector generated by a residue in the phase map that has a certain orientation pointing out to the balancing residue of opposite polarity.

In essence, residues have vector fields that are very directional in nature. This directional vector field can only point to the balancing opposite polarity residue. These vector fields are very visible in the case of noise and under-sampling. With the increase of phase noise, the residue vector grows smaller but it stays strong at the residue. At very high noise, residues will get closer until they become dipoles of one sample apart. The behaviour of the residue-vector in the wrapped phase map under different conditions can give useful information on how to solve the branch-cut problem.

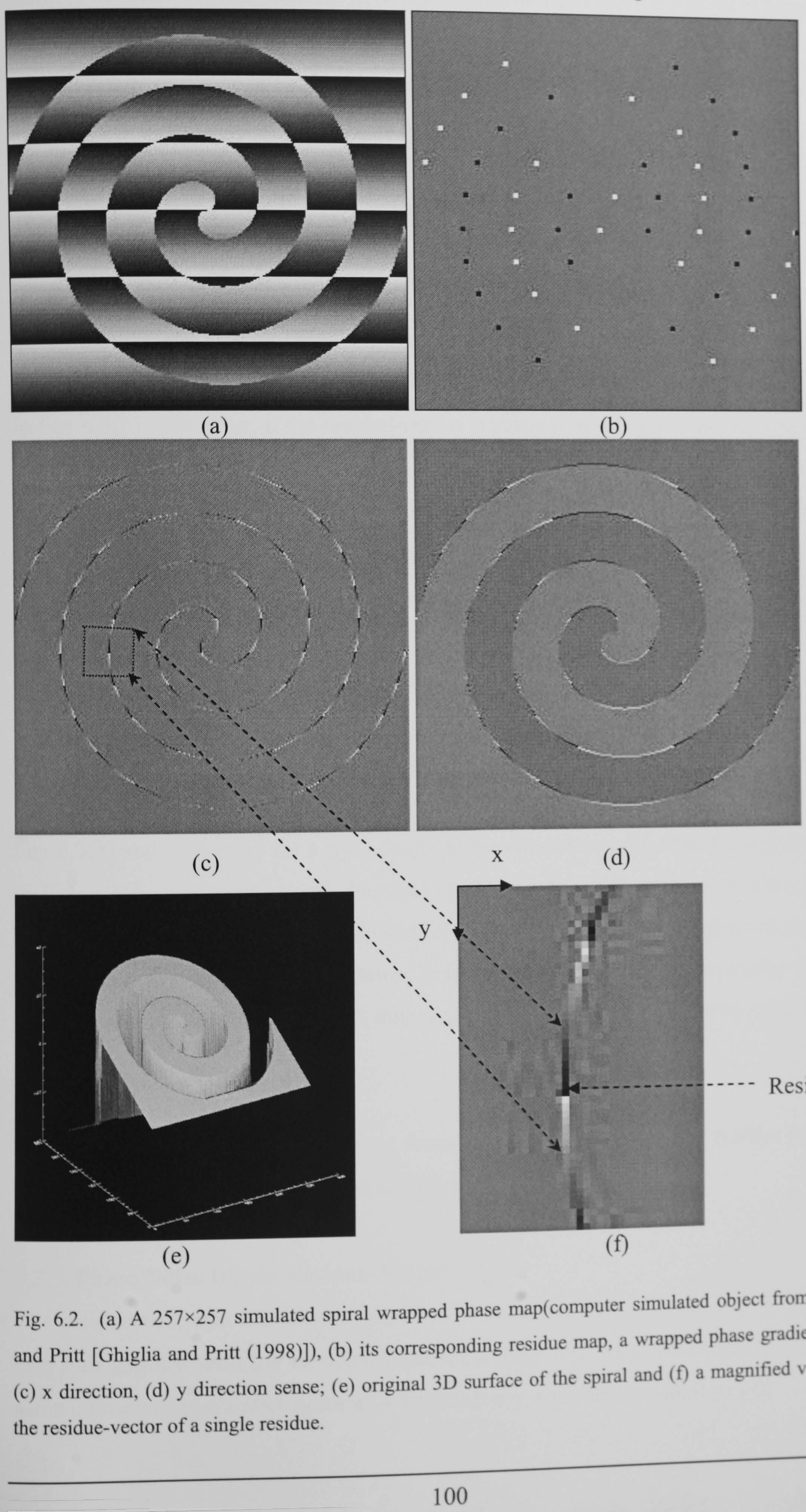



Fig. 6.2. (a) A 257×257 simulated spiral wrapped phase map(computer simulated object from Ghiglia and Pritt [Ghiglia and Pritt (1998)]), (b) its corresponding residue map, a wrapped phase gradient in the (c) x direction, (d) y direction sense; (e) original 3D surface of the spiral and (f) a magnified version of the residue-vector of a single residue.

### 6.2.2. A Method to Differentiate between Negative and Positive Residues from the Residue-Vector

In all types of wrapped phase maps, it is possible to differentiate between negative and positive resides from the residue-vector in the gradient phase map. Positive and negative residues have a unique residue-vector behaviour in the gradient phase depending on their polarity and their residue-vector direction in the horizontal and vertical senses.

In the case of a positive residue having a horizontal residue vector, its residue-vector will go from low (left) to high (right) at the positive residue as demonstrated in Fig. 6.3(a) and visa versa for the negative residue in the horizontal residue-vector sense also shown in Fig. 6.3(b). However, if the residue-vector is vertical, it will go from low (top) to high (down) at the positive residue and visa versa for the negative residue again shown in Fig. 6.3(c) and (d), respectively. In the case of the horizontal residue vector, the positive and negative residues have no pattern in the dx gradient phase map and visa versa for the case of the vertical residue vector as presented in Fig. 6.3(a), (b), (c) and (d).

A diagonally orientated residue-vector has a pattern on both dx and dy gradient phase maps. A right diagonal residue-vector of a positive residue has both horizontal and vertical patterns in dy and dx gradient phase maps as explained previously in addition it also has different or opposite orientation in both dx and dy gradient phase map as seen in Fig. 6.3(e) and visa versa for a right diagonal residue-vector of a negative residue shown in Fig. 6.3(f). Moreover, a left diagonal residue-vector of a positive residue has both horizontal and vertical patterns in dy and dx gradient phase maps as explained previously in addition it also has similar orientation in both dx and dy gradient phase map as seen in Fig. 6.3(g) and visa versa for a left diagonal residue-vector of a negative residue shown Fig. 6.3(h).

The residue-vector also takes different forms depending on the kind of residue causing it. These vector forms are summarized as follows:

### 6.2.3. Phase Noise Dipole Residue-Vector

The residue vector in this kind of dipole is very localized and small. Fig. 6.4(a) shows two such dipoles with the residue-vector extending vertically, while in Fig. 6.4(b) the same arrangement is shown somewhat more clearly in a diagrammatic format.

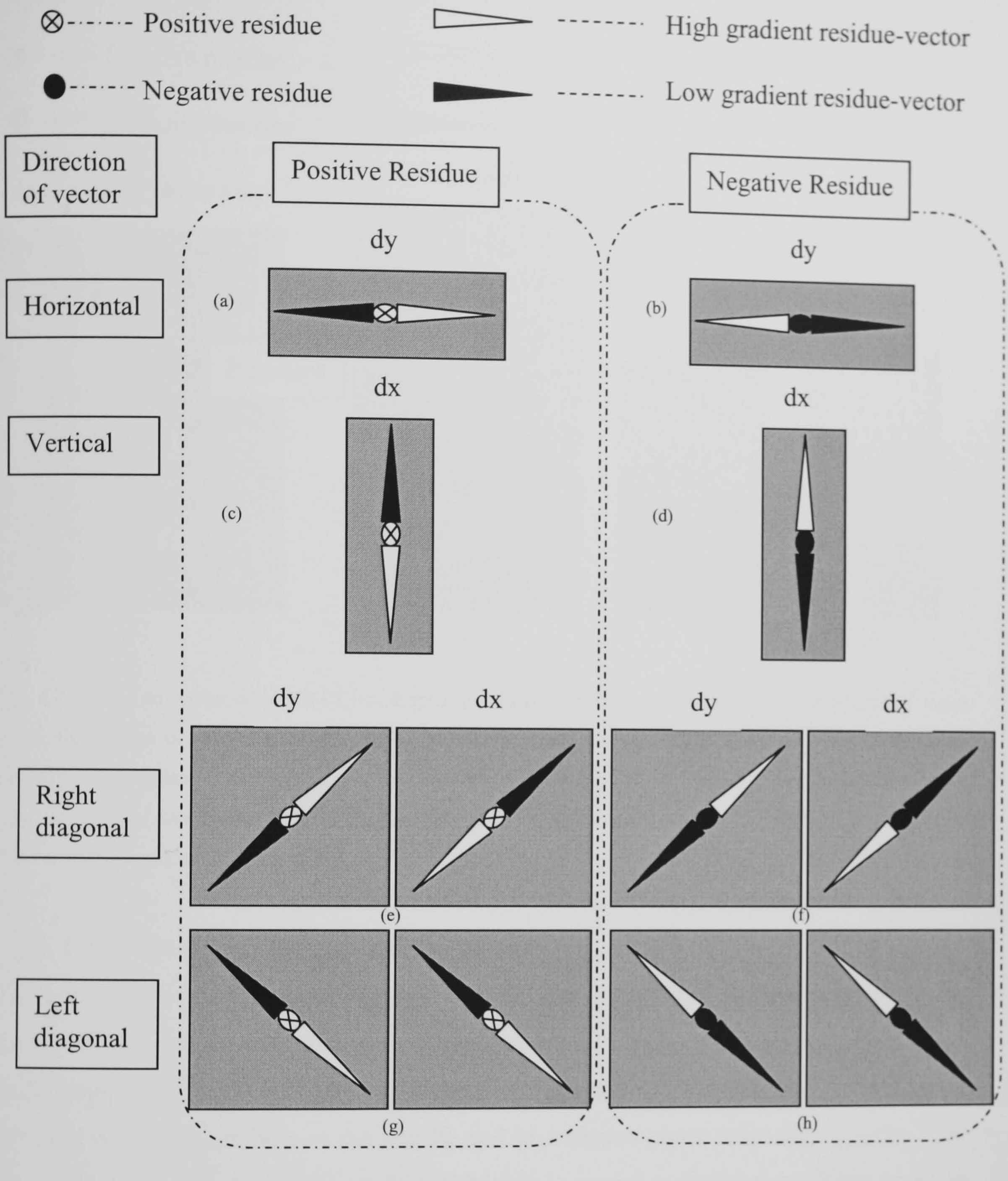



Fig. 6.3. An illustration in how to distinguish between positive and negative residues for residue-vector direction (a) horizontal positive residue-vector, (b) horizontal negative residue-vector, (c) vertical positive residue-vector, (d) vertical negative residue-vector, (e) right diagonal positive residue-vector, (f) right diagonal negative residue-vector, (g) left diagonal positive residue-vector and (h) left diagonal negative residue-vector.

In Figs. 6.4(c) and (d), the position of both positive (white pixel) and negative (black pixel) residues are presented. The residue-vector in this case generated by phase noise is local to the dipole. It does not need to extend further than the dipole because the residues are close to each other.

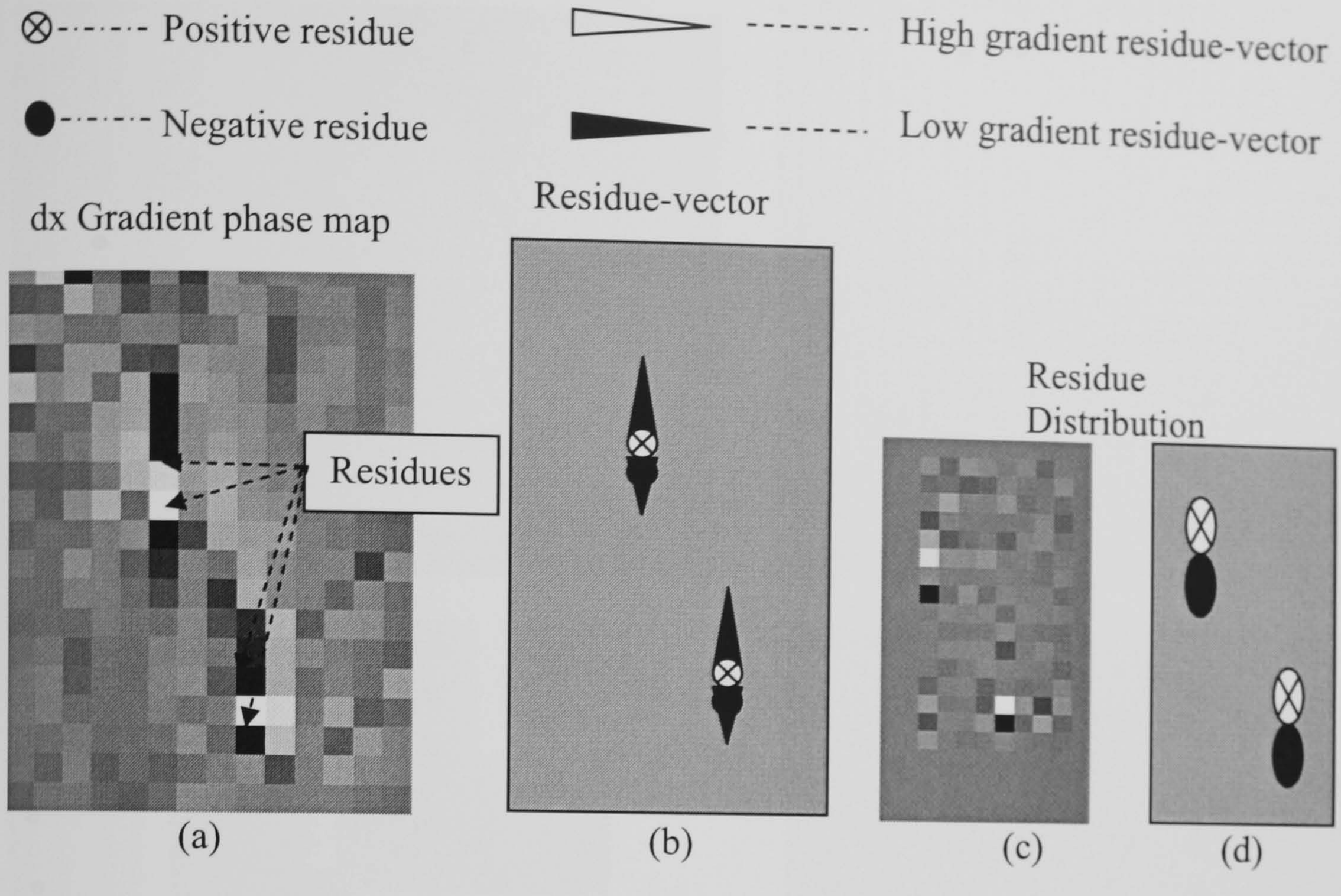
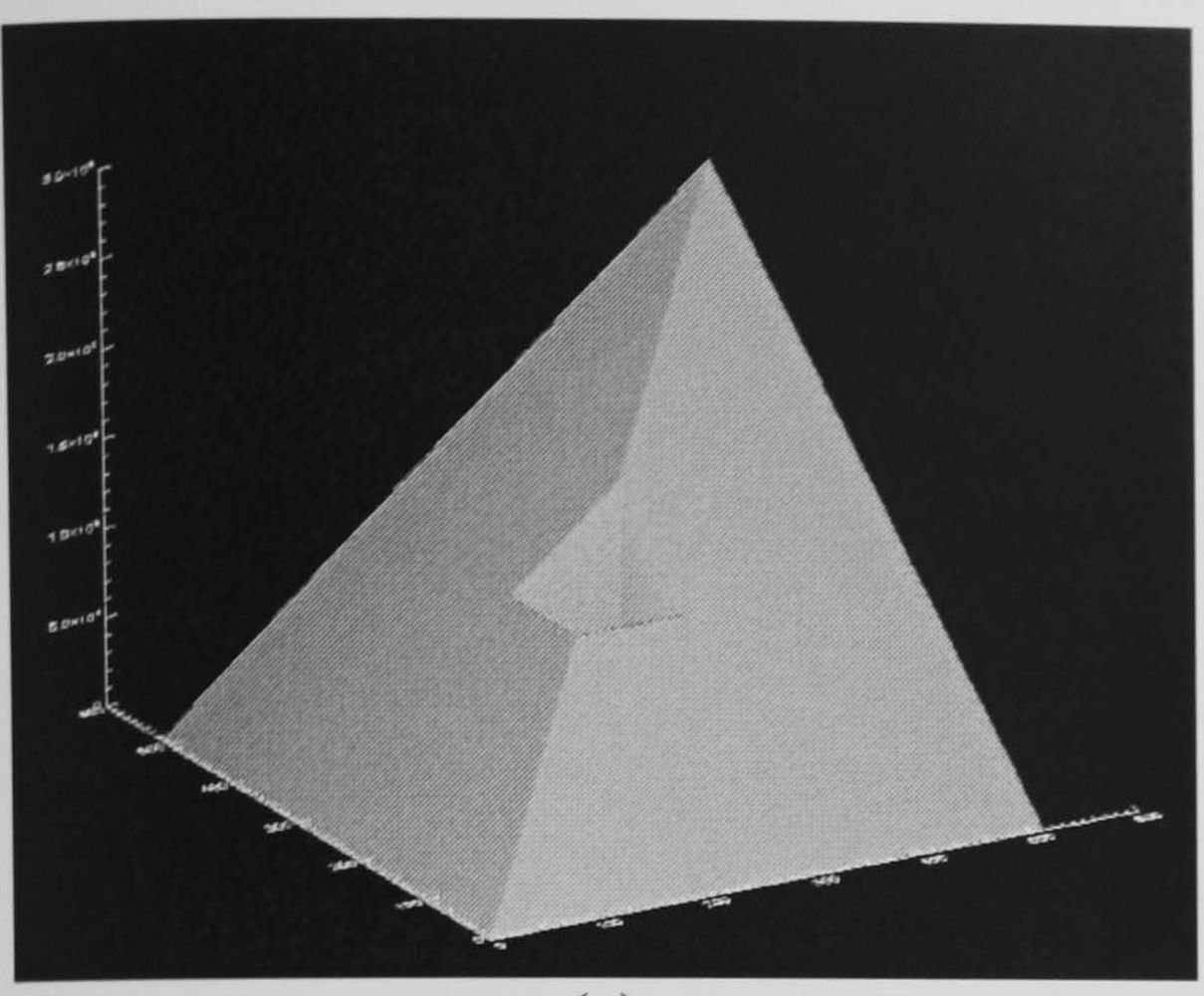


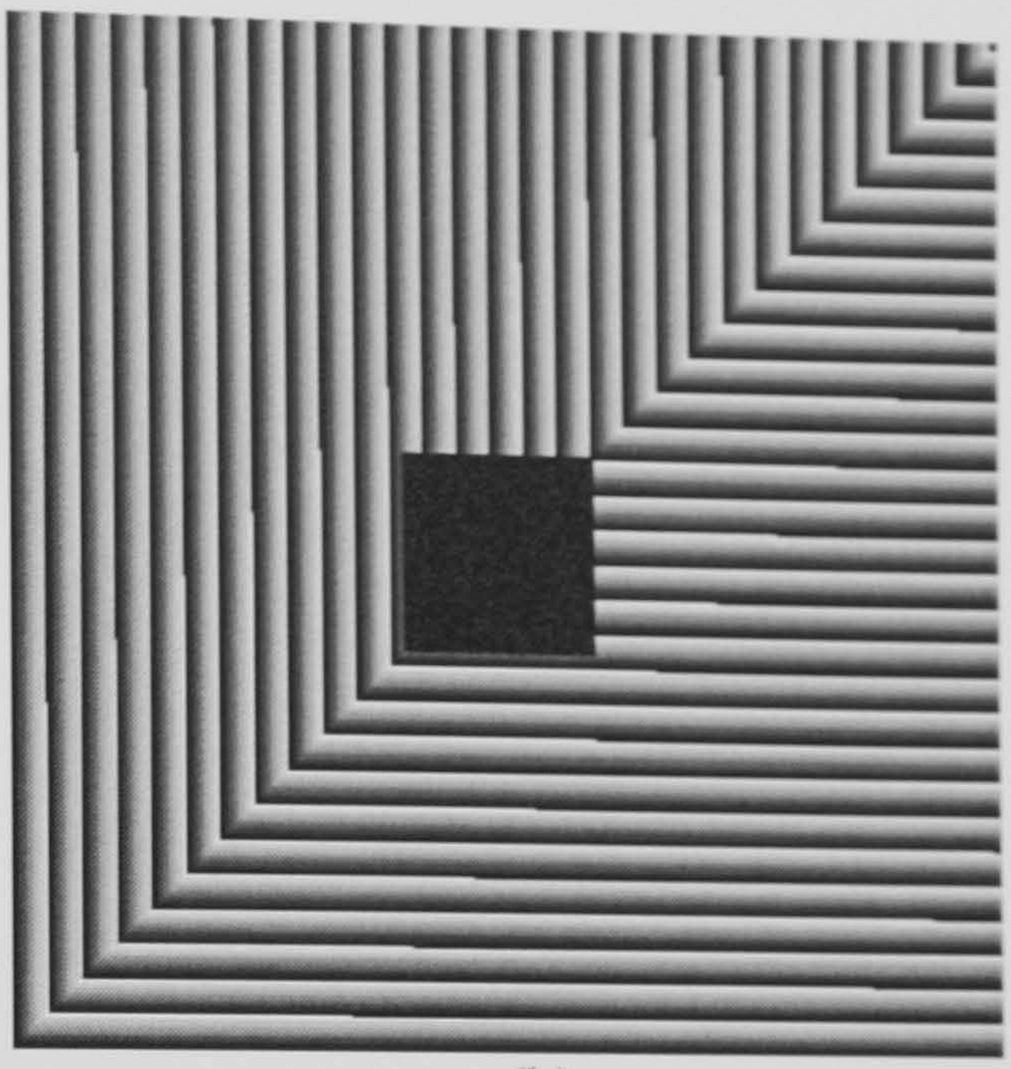

Fig. 6.4. Residue-vector of a pair of two opposite polarity dipole residues caused by high level of phase noise, this image is created from the IFSAR wrapped phase map in Ghiglia and Pritt [Ghiglia and Pritt (1998)]; (a) dx gradient phase map, (b) an emphasis of how the residue-vector in the dx gradient phase map, (c) residue distribution map of the same dx gradient phase map and (d) an emphasis of the pair of dipoles in the residue distribution.

### 6.2.4. Under-Sampling Dipole Residue-Vector

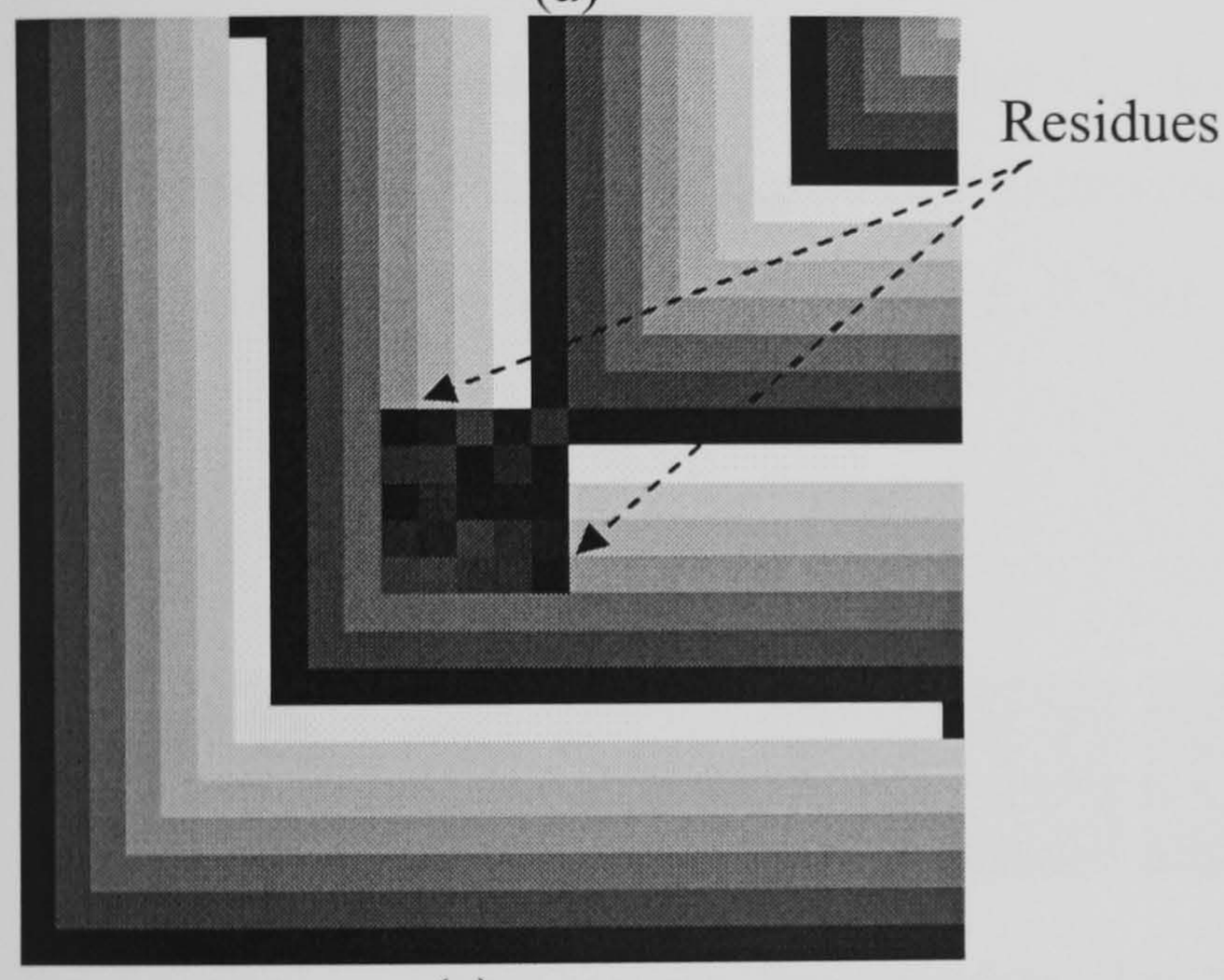
The under-sampling dipole residue-vector is similar in form to the discontinuous object dipole residue-vector described in the next section. To emphasize the behaviour of an under-sampling dipole residue-vector, Figs. 6.5(a) and (b) show a simulated quarter pyramid with a square hole in the middle and its wrapped phase map; respectively. This wrapped phase map is down-sampled 20 pixels in x and y directions and the result is presented in Fig. 6.5(c). Moreover, Fig. 6.5(d) shows a sketch that emphasizes the behaviour of the residue-vector in a down-sampled wrapped phase map, which summarizes the residue-vector in each of Figs. 6.5(e) dx and 6.5(f) dy gradient phase map of the wrapped phase map of Fig. 6.5(c), respectively. It can be seen from Fig. 6.5(d) that the residue-vector at the residue is strong and its power decreases in steps until it reaches the balancing vector of its counter part balancing dipole residue. Even with the large amount of discontinuity that exists in the down-sampled wrapped phase map of Fig. 6.5(c), it is still evident from the under-sampling dipole residue-vector how to branch cut successfully this wrapped phase map prior to unwrapping.



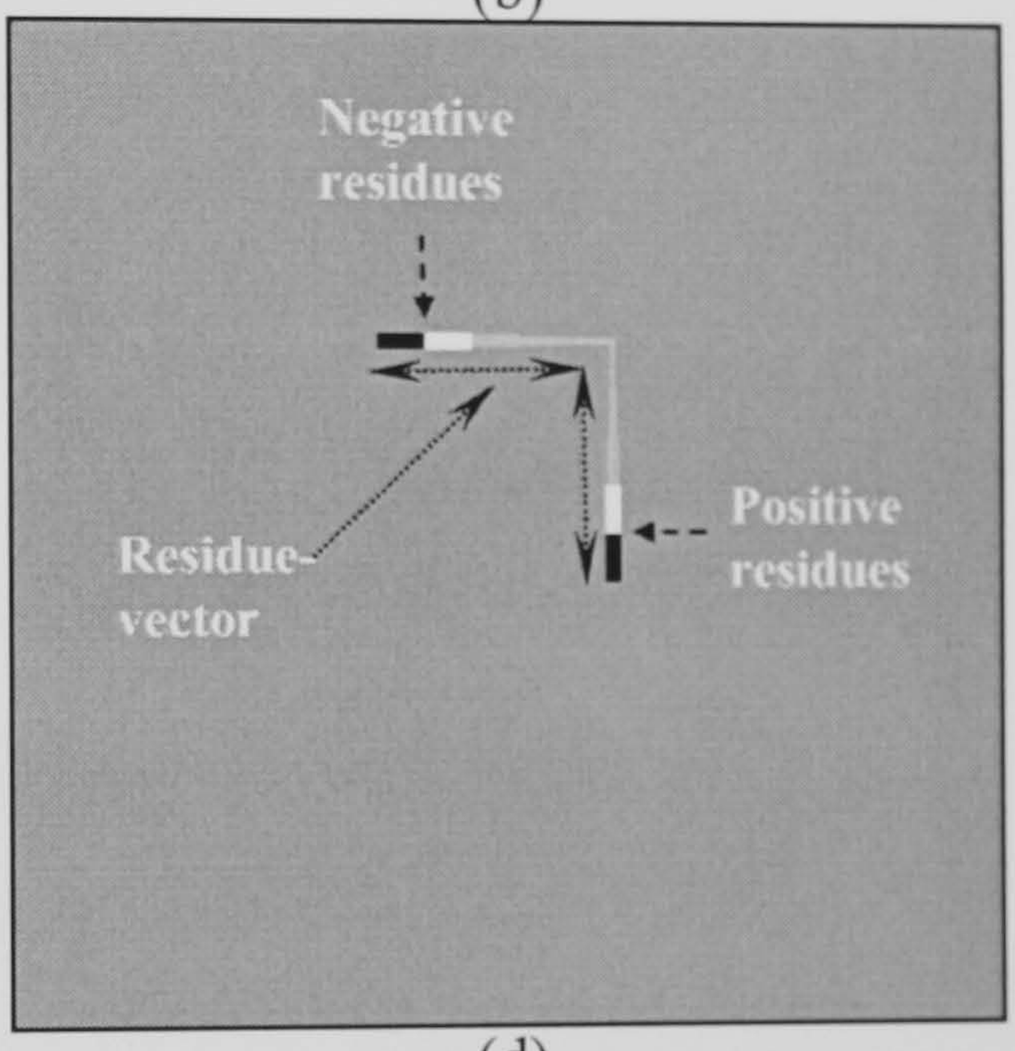
(a)



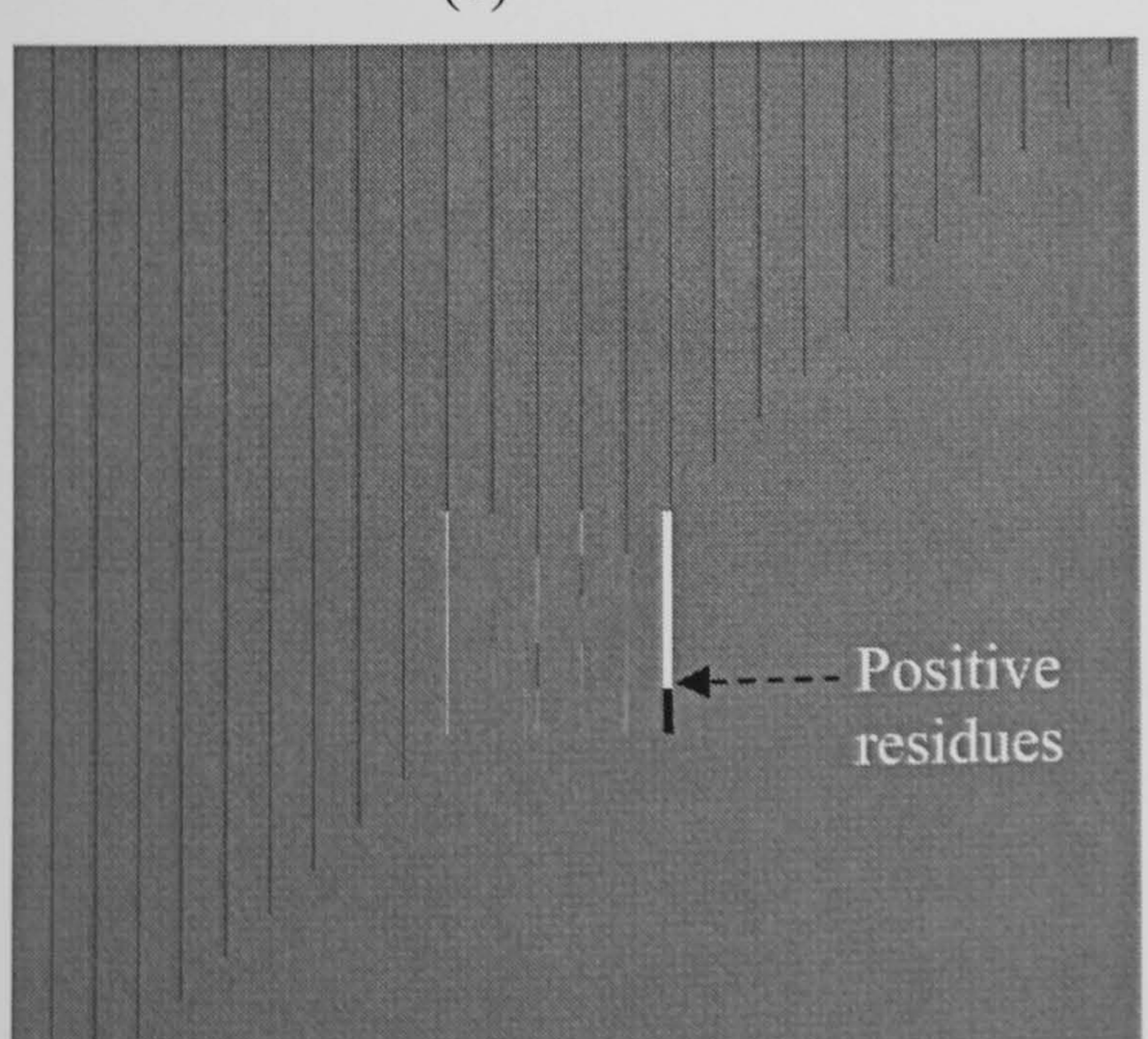
(b)



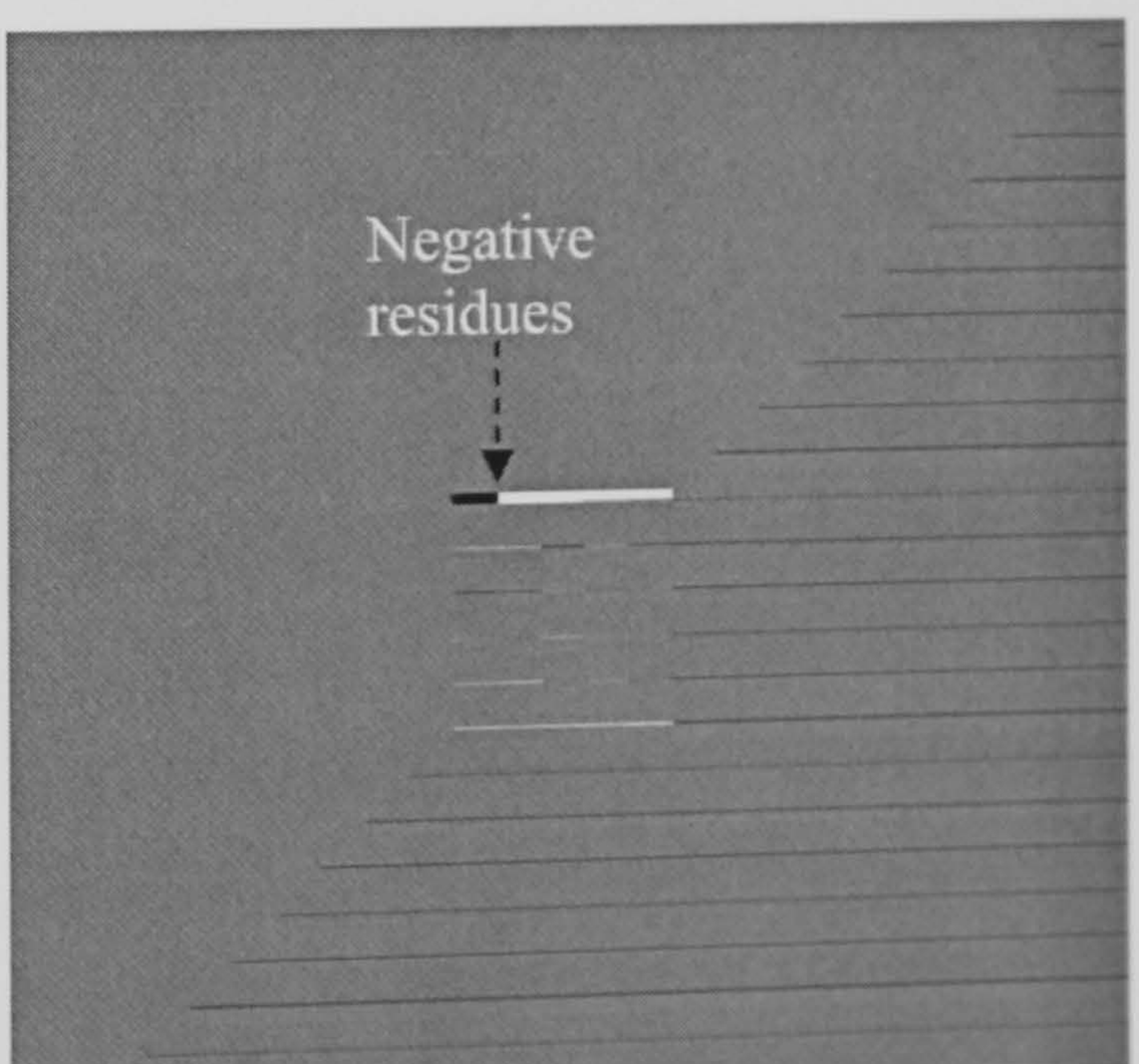

(c)




(d)




(e)




(f)

Fig. 6.5. (a) Original 3D simulated object of a quarter pyramid with a square hole in the middle, (b) its wrapped phase map, (c) its down-sampled wrapped phase, (d) an illustrative emphasis of the behaviour of the residue-vector in the (e) dx and (f) dy gradient phase map.

This presents a considerable advantage to the phase unwrapping problem by relying on residue-vector it is possible to distinguish discontinuities caused by down-sampling (down-sampling is caused by providing incomplete data intentionally to study the concept behaviour in the case of under-sampling) and discontinuities caused by under-sampling residues (under-sampling is caused by incomplete data to represent a feature or an information).

### 6.2.5. Discontinuous Object Dipole Residue-Vector

Residues in this case generate a residue-vector that is very large due to the tendency of these dipole residues to lie far apart from each other, as described previously. If the residue-vector is not disrupted with other dipoles along the same edge or discontinuity, the dipoles would share the constant residue-vector between them as in the example in Figs. 6.6(a) and (b) and illustrated in Fig. 6.7(b).

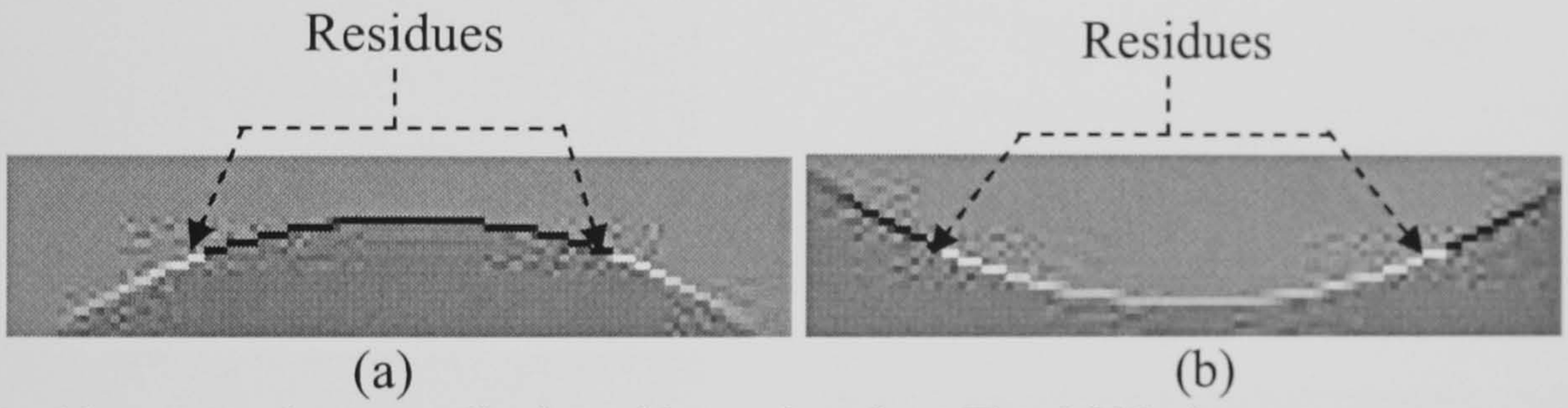



Fig. 6.6. Residue-vector between dipole residues taken from Fig. 6.2(d) shows a constant vector charge shared between the residues of the dipole (a) and (b) present the residue-vector charge varying with the nature of discontinuity whether descending or ascending.

However, once the residue-vector is disrupted by a residue or a dipole lying between them and having the same residue vector direction, it will not have the necessary power to overcome the vector of the local residue or the dipole lying between it and the balancing residue. This case is called the ***zero-vector*** *or null or neutral-vector* and is illustrated in Fig. 6.7(c). This zero-vector makes it difficult to extract the original contiguous vector of the first dipoles. This case can be seen in Figs. 6.7(a), (d) and (e).

### 6.2.6. Object Discontinuity and Phase Noise

Residue-vectors generated by dipoles of object discontinuity are not greatly affected by phase noise as shown in the dy gradient phase map of Fig. 6.8(c) extracted from the wrapped phase map in Fig. 6.8(b) of a simulated object shown in Fig. 6.8(a), where it stays clear of noise and undisturbed. However at very high level of phase noise that

does not exceed the level that creates phase noise residues, as demonstrated in Fig. 6.8(d), the residue-vector is disrupted to an extent that the vector becomes localized at the residue. Even in this case, the object discontinuity residue-vector can be still traceable in the gradient phase map.

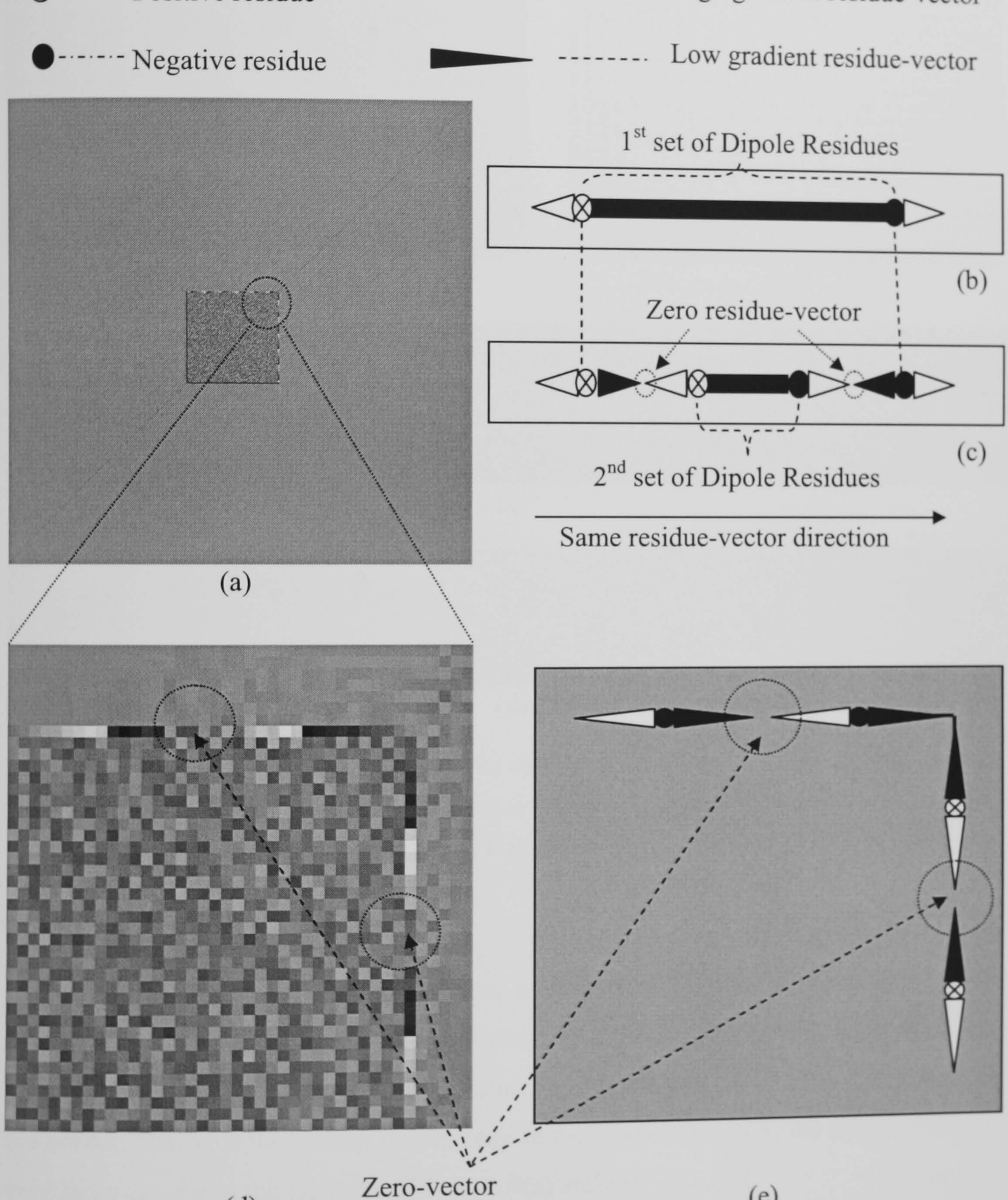
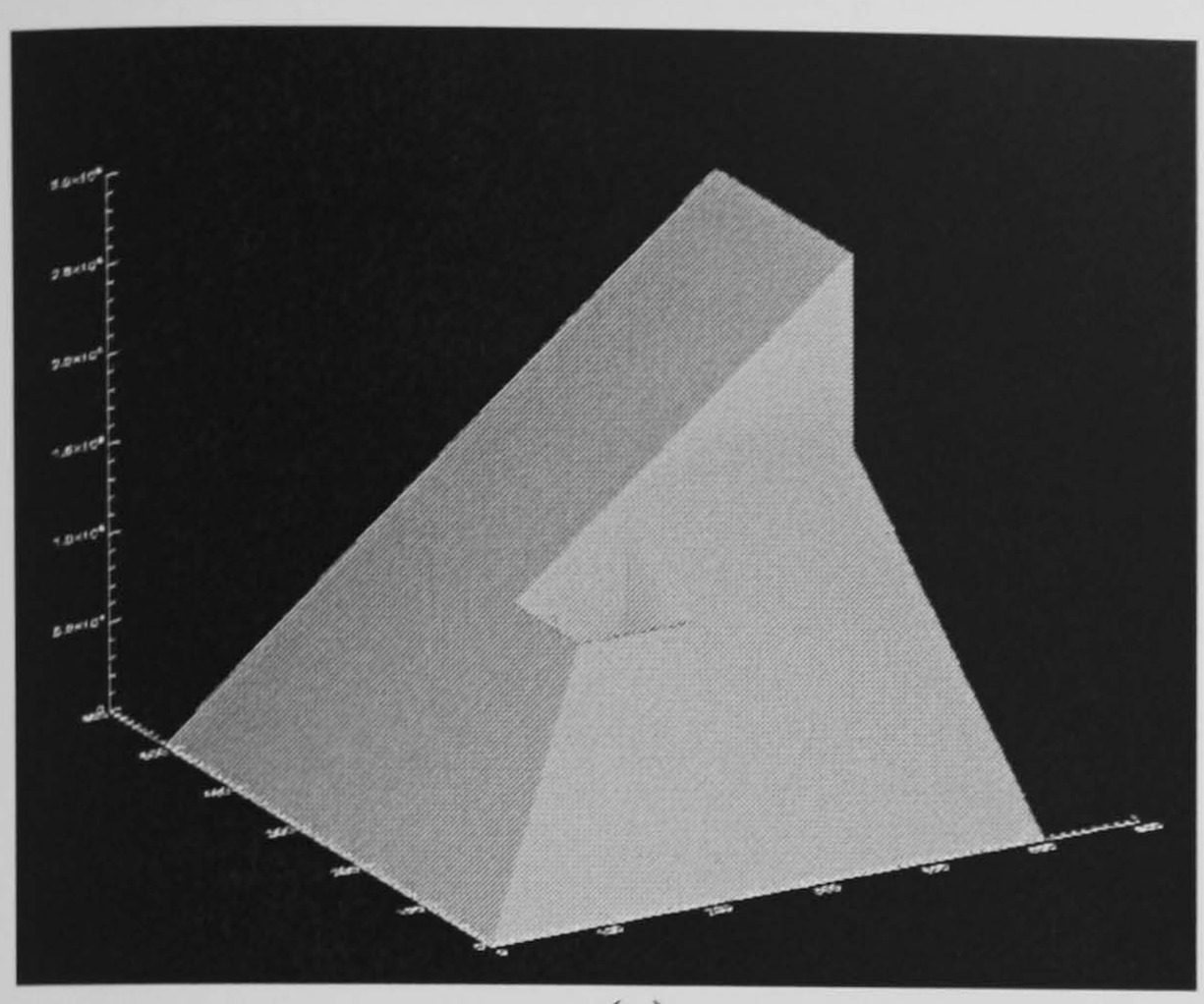
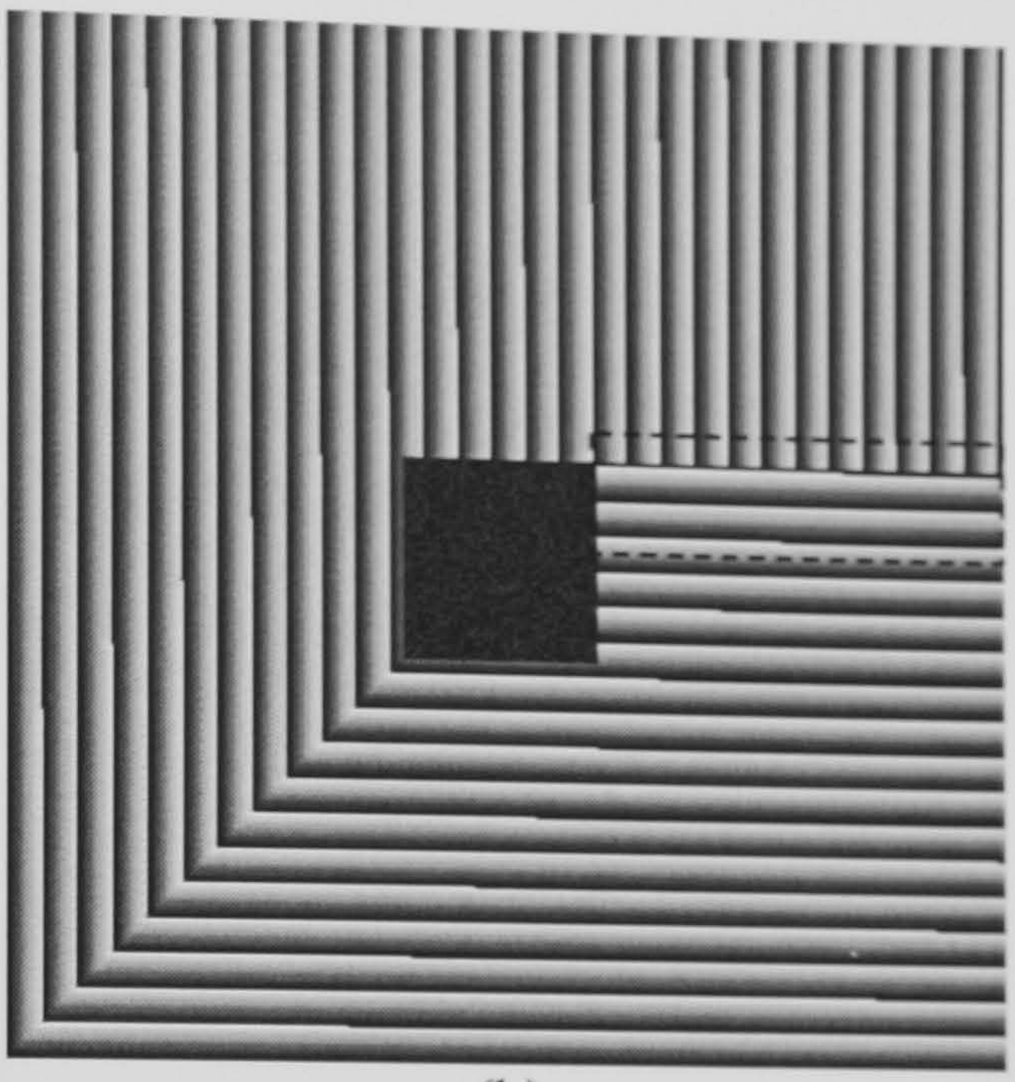



Fig. 6.7. (a) The overlapped wrapped phase gradient of dx and dy calculated from the wrapped phase map in Fig. 6.5(b), (b) a schematic showing a constant vector charge shared between the residues of the dipole, (c) a schematic showing how the zero-vector is created, (d) residue-vector orientation of the four upper right corner residues in Fig. 6.7(a), (e) a schematic of the residue-vector orientation of the four upper right corner residues.



(a)



(b)



(c)



(d)

Fig. 6.8. (a) Original 3D simulated object, (b) its wrapped phase map, a section of dy wrapped phase gradient map of Fig. 6.8(b) marked by a box is used to show the (c) residue vector behaviour at low phase noise and the (d) residue-vector behaviour at very high phase noise.

### 6.2.7. Monopole Residue-Vector

Residue-vectors generated by monopoles extend from the monopole residue to the border. The size of the residue-vector depends on the position of the monopole in the phase map and on the nature of the discontinuity extending from the monopole residue to the border. Hence, if the monopole lies deep in the phase map; then, the residue-vector is very large as seen in the example of Figs. 6.9(a) and (b).

In Figs. 6.9(a) and (b), the monopole residue-vector decreases as it approaches the border due to the distance covered in the phase map to reach the border and due to the phase noise and the nature of the discontinuity. Figs. 6.9(c) and (d) shows the original fairy's

image (the fairy is a complex statue that contains sharp edges and discontinuities employed as a challenging test piece in this work) and its corresponding wrapped phase map, respectively, to illustrate the position of the fairy's hand with the monopole residue. The importance of Fig. 6.9(b) is the monopole residue in the middle of the image. This is because the monopole residue is lying far away from the border that it should be linked to and not to the close borders that are not related to this monopole residue. This information was not known prior to the discovery of the residue-vector.

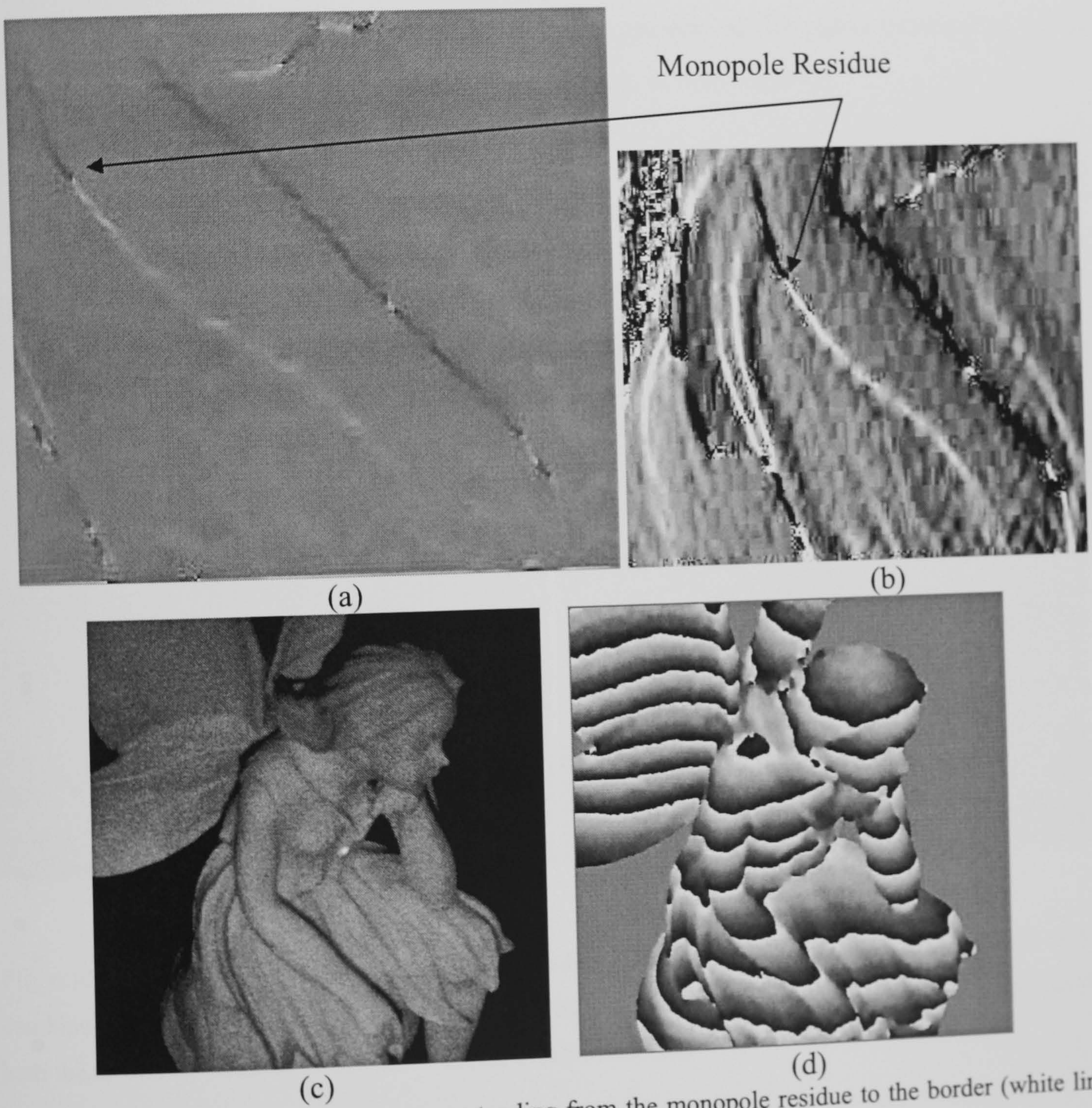



Fig. 6.9. (a) A monopole residue-vector extending from the monopole residue to the border (white line), (b) similar image but with increased contrast to make the residue-vector more visible, (c) the original fairy's image illustrating the position of the fairy's hand with the monopole residue and (d) its corresponding wrapped phase map.

### 6.3. Residue-Vector, High Gradients and the Effect of Under-Sampling

It should be noted that extracting information from the magnitude of the phase gradient as described by Chavez *et al.* [Chavez *et al.* (2002)] and Salfity *et al.* [Salfity *et al.* (2006)] cannot distinguish the difference between high phase gradient magnitudes generated by phase noise and edges; and high phase gradient magnitudes generated by residues in the form of residue-vectors. So, it is improper to rely on the magnitude of the phase gradient alone to generate branch-cuts as stated by Chavez *et al.* [Chavez *et al.* (2002)]and Salfity *et al.*[Salfity *et al.* (2006)]. Phase gradient magnitude does not reveal information about the causes of residue discontinuities as is the case with the residue-vector. The difference between relying on the magnitude of the phase gradient or on the original phase gradient dx and dy is evident in Figs. 6.10(a) and (b).

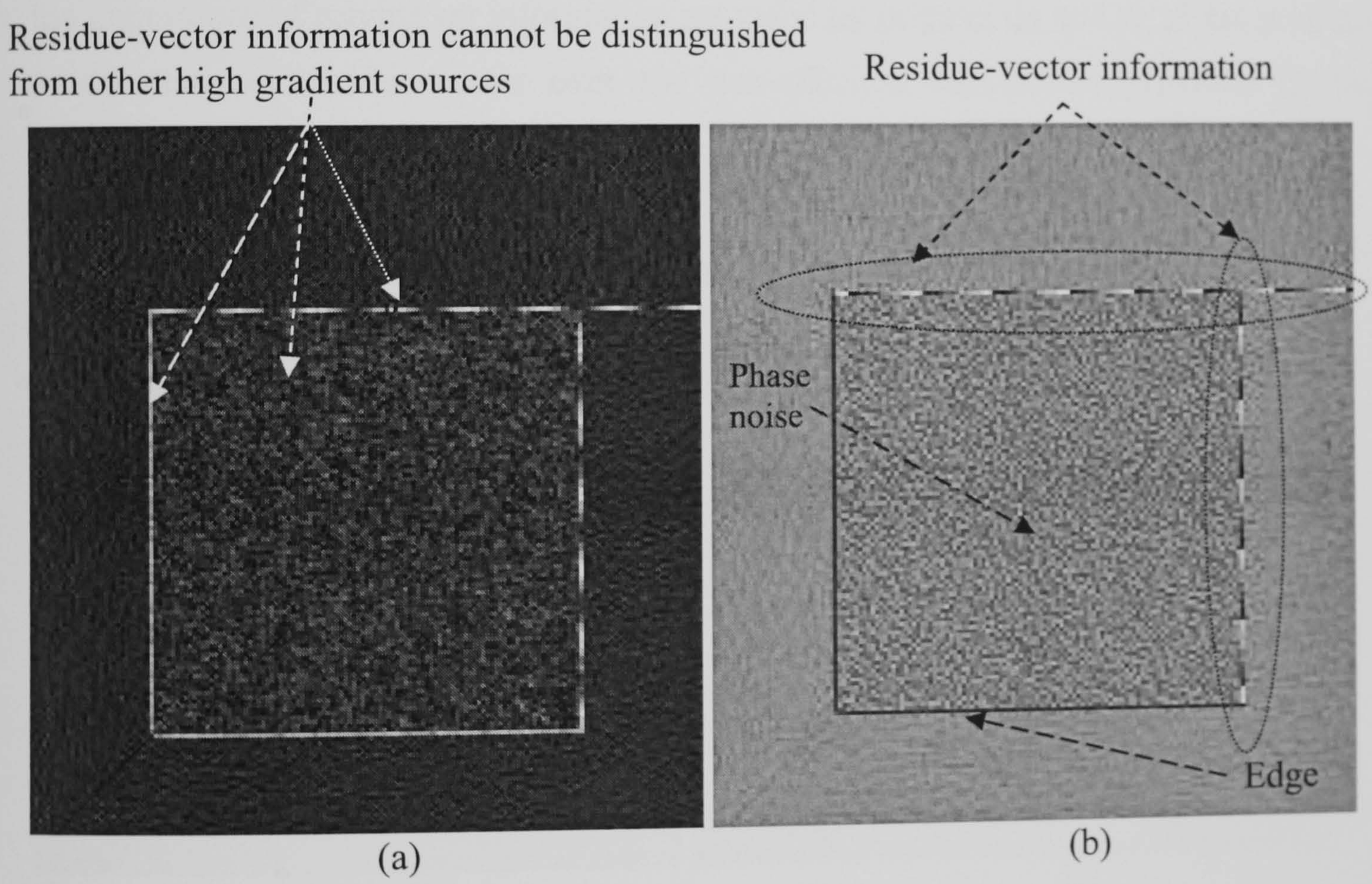
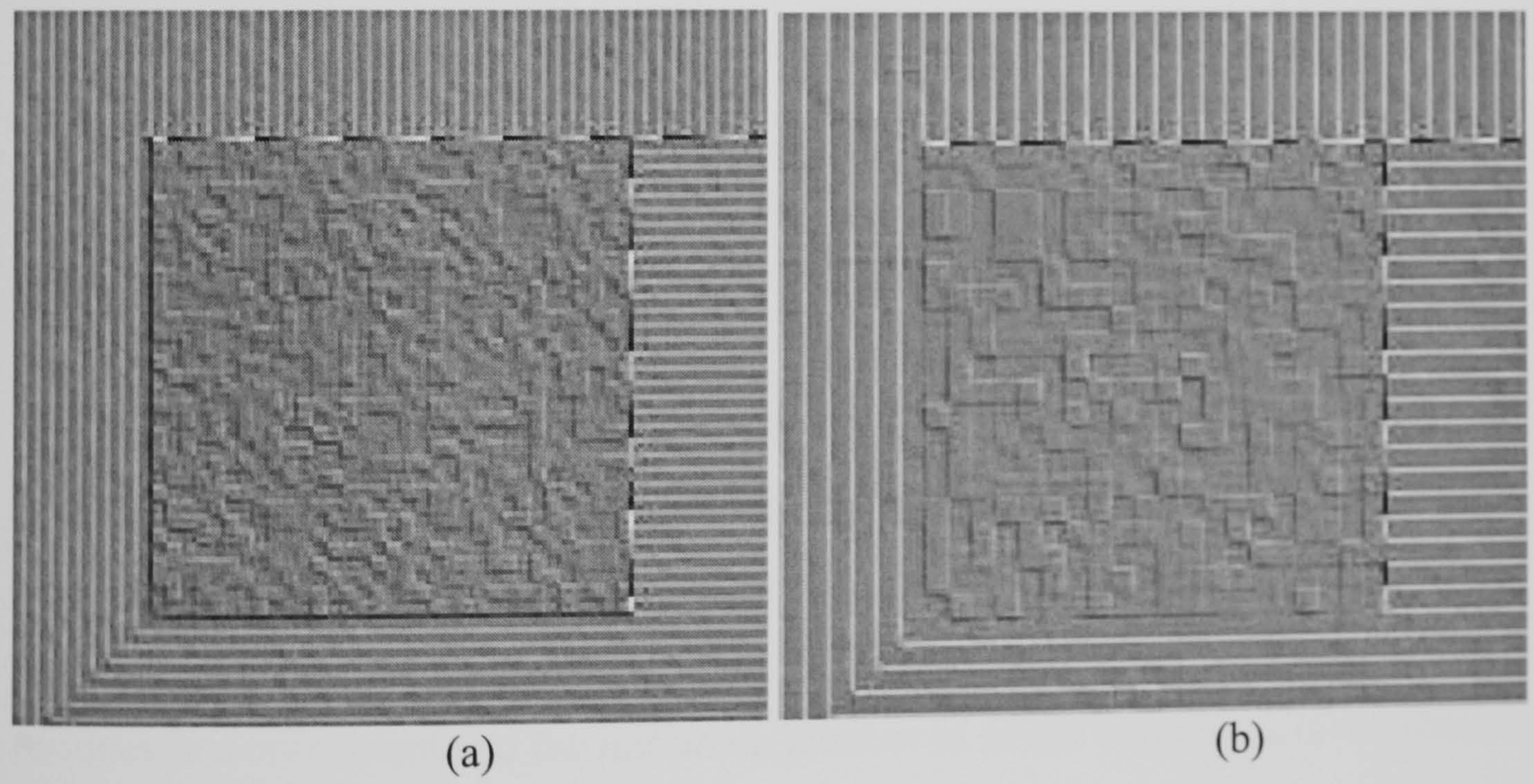



Fig. 6.10. (a) Magnitude of the wrapped phase gradient of the wrapped phase map in Fig. 6.8(b) and (b) combined wrapped phase gradient map of Fig. 6.8(b)(middle section of the image) using the maximum of both the dx and dy wrapped phase gradient (for illustration only).

In Fig. 6.10(a), the residue-vector information cannot be distinguished from other high gradient sources like edges and phase noise. However, in Fig. 6.10(b), it is clearly distinguishable from the residue-vector information whether high gradient phase originates from an edge or phase noise. In essence, more information is present in the

original dx and dy phase gradients than just their magnitude would leave us to understand.

The residue-vector is not affected by under-sampling and it can still provide all the necessarily information for the best placement of the branch-cuts. It can be distinguished from other high gradients caused by under-sampling or phase noise. Thus, perfect phase unwrapping can still be achieved from the information provided only by the wrapped phase map based on the residue-vector. This is contradictory to the assertion made by Salfity *et al.* stating that "any criterion based on the wrapped phase gradient alone would fail to decide the best placement of the branch-cut" [Salfity *et al.* (2006)]. While this statement is true if only the magnitude of the gradient phase map is considered, the use of the residue-vector brings new information into play. This proves the importance of relying on information provided by original dx and dy phase gradient in the form of residue-vector over the non-efficient information provided by the magnitude of the wrapped phase gradient.



(a) (b)

Fig. 6.11. Combined wrapped phase gradient map of the middle section of the wrapped phase map in Fig. 6.8(b) using the maximum of both the dx and dy wrapped phase gradient for illustration only scaled down by (a) 3×3 pixels and (a) 5×5 pixels.

Figs. 6.11(a) and (b) show clearly that even with down-sampling of 3×3 pixels and 5×5 pixels to generate different under-sampling resolutions, respectively; the residue-vector information is still present in the combined wrapped phase gradient map which is enough to adequately guide the optimum phase unwrapping.

## 6.4. Residue-Vector Branch-Cut

In the previous section, we have shown the existence of a residue-vector, the nature of which is characteristic of different types of dipoles and monopoles and which is detectable even in the presence of considerable phase noise. We will now move on to consider how this residue-vector can be used as a criterion for the placement of branch cuts. A neutral unwrapped phase map is defined by having all residues in the phase map neutralized by branch-cuts based on Eq. (2.6).

This can be defined in Eq. (6.1):

$$\sum_{b}^{N_{bc}} \sum_{p}^{N_{bp}} \hat{\nabla}\Phi(b,p) = 0 \tag{6.1}$$

Where $\hat{\nabla}\Phi(b,p)$ is the sum of gradient estimate in a 2×2 closed loop at pixel 'p' of branch-cut 'b',

$N_{bc}$ is the number of optimal branch-cuts in a wrapped phase map,

$N_{bp}$ is the number of optimal branch-cut pixels in a branch-cut 'b'.

The formula in Eq. (6.1) does not necessarily indicate that this set of branch-cuts will result in the correct phase estimate in the unwrapped phase.

This leads to a new concept which specifies that for a set of branch-cuts in a phase map to achieve the minimum error between the estimated gradient of the unwrapped phase solution and the gradient of the wrapped phase map; branch-cuts should follow the maximum number of residue-vector pixels separating each pair of opposite polarity residues in a branch-cut and the minimum number of pixels branch-cut in the phase map.

This concept is summarized in Eq. (6.2):

$$\min E \xleftrightarrow{\text{If and only if}} \begin{cases} \max N_{rf} \\ \min N_{br} \end{cases} \tag{6.2}$$

$N_{rf}$ is the total number of residue-vector pixels overlapped by branch-cut in the phase map,

$N_{br}$ is the total number of branch-cut pixels in the phase map.

$$E = \sum_{i}^{N}\sum_{j}^{M}\left|k_{i,j}^{x}\right| + \sum_{i}^{N}\sum_{j}^{M}\left|k_{i,j}^{y}\right| \tag{6.3}$$

where $E$ is the total number of errors (discontinuities) in the unwrapped phase map [Costantini (1998)].

In other words, the minimum cost flow [Costantini (1998)] and the Flynn minimum discontinuity [Flynn (1997)] algorithms attempt to identify the lines of discontinuity representing the branch-cuts between residues; however, they are effectively trying to identify the residue-vectors that result in such discontinuities – without explicitly being aware of their existence. This theorem is further explained using experimental results in section 6.5.2. Moreover, due to the mentioned algorithms complete reliance on the weights provided; they try to approximate the position of the residue-vector to branch-cut in the phase map. In essence, they do not have problem-specific knowledge of what is causing the discontinuity they are trying to identify. We can rectify this shortcoming by using the residue-vector to orientate the dipole branch-cuts. This method is a general form of branch-cut which could form a cut or a tree of cuts, which could make it very suitable to nearly all types and variants of branch-cut phase unwrapping algorithms.

The residue-vector Branch-cut method using dipole strategy can be summarized as following:

- Calculate the phase gradient in the x and y directions.
- Identify positive and negative residues in the wrapped phase map.
- Start from a random residue in the phase map.
    - Identify the residue-vector by locating consecutive pixels with high and low gradient values in the x and y phase gradient maps, respectively.
    - Follow the residue-vector on both sides of the residue until a balancing residue of opposite polarity is found or until the residue-vector reaches a zero-vector. If a zero-vector is found:
        - Identify the nearest residue-vector of another residue and include pixels of the highest and lowest gradient located between the two vectors.

- Iteratively locate the nearest residue-vector of another residue closer to the last residue-vector until a balancing residue of opposite polarity of the starting residue is found.

- Branch-cut all the included pixels of this pair of residues.

- Repeat the same procedure until all the residues are balanced by branch-cuts created by this method.

The residue-vector map can then be inserted as weights in different phase unwrapping algorithms such as Flynn and minimum cost flow or it may be optimized to satisfy the condition of Eq. (6.2) by minimizing the number of branch-cut pixels ensuring the maximum number of residue-vector pixels included and all residues are balanced. The successful branch cut placement should obey the residue-vector orientation in the gradient phase map to balance two dipole residues.

To demonstrate this fact, an example that shows the incorrect and correct branch-cut placement is displayed in Figs. 6.12(a), (b) and (c). Fig. 6.12(a) shows the dipole residue-vector with the residues positions and their polarity indicated by arrows. It can be seen from the figure that the residue-vector follows a curved path. Hence, it is incorrect to place a straight branch-cut as in Fig. 6.12(b) that only takes in consideration the distance of the branch-cut between the dipole residues and does not recognizes the residue-vector. On the other hand, Fig. 6.12(c) shows the correct branch-cut placement that completely follows the residue-vector lying between the two dipole residues with the minimum number of branch-cut pixels. Moreover, consider the case of Figs. 6.9(a) and (b), a monopole residue, which has a large visible vector connecting it to a border pixel. The residue-vector follows the edge of the fairy's hand until it reaches the nearest border pixel along that edge. This is an advantage to phase unwrapping because it specifies how the correct branch-cut should be placed.

Fig. 6.13 shows the optimum branch cut in this circumstance and therefore leads to the best possible unwrapped phase solution with the minimum possible discontinuity, which is compatible with the original data. It should be noted that a different branch-cut placement to the example stated above would distort the unwrapped phase map even though the residue is balanced by another residue of opposite polarity or it is balanced

with nearest border pixel. Residue-vector information lowers the search and guess complexity that was used by other algorithms to identify the monopole residue.

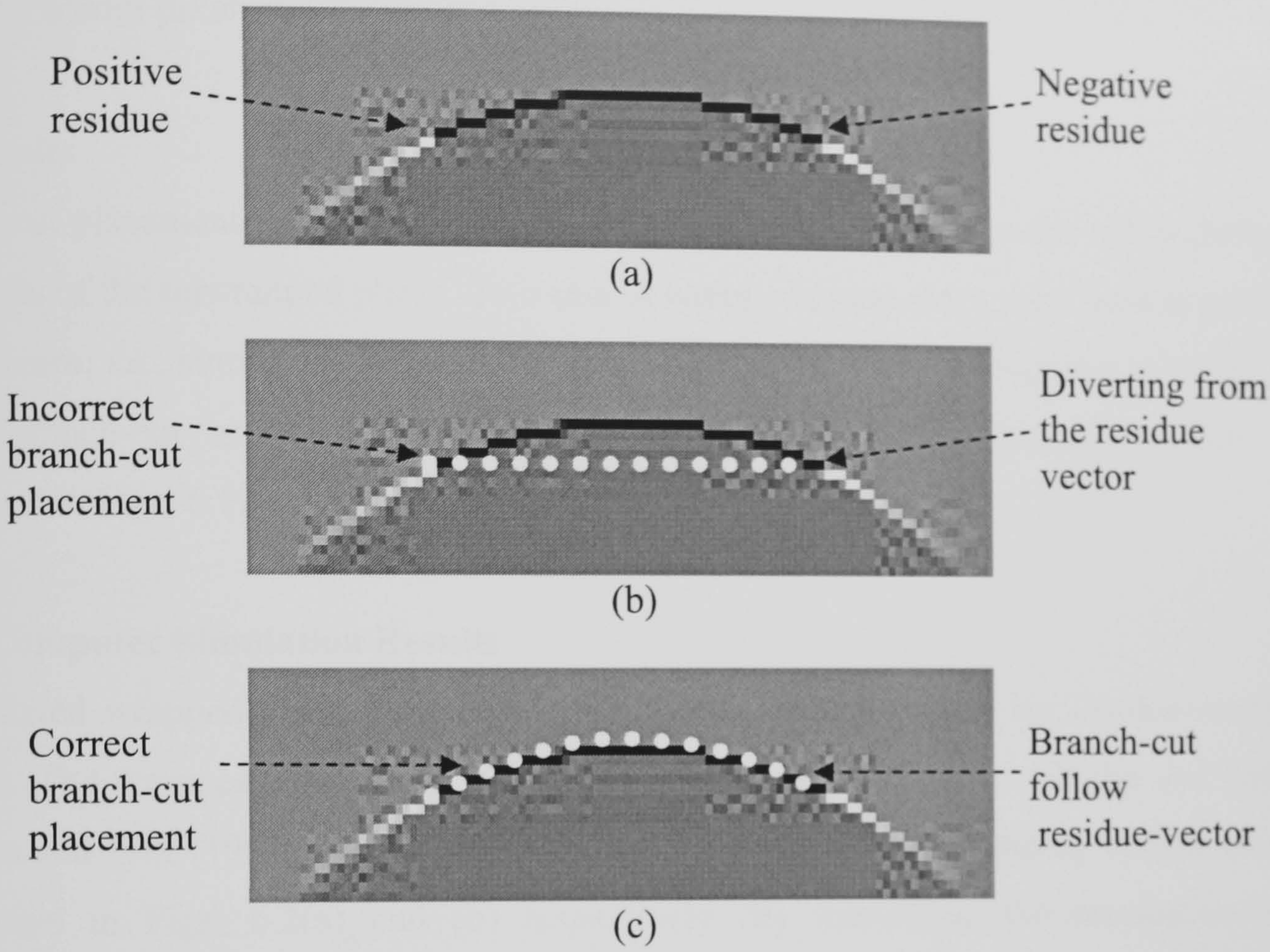



Fig. 6.12. Branch-cut placement methods used to connect two residues; (a) original dy gradient map taken from Fig. 6.2(d), (b) incorrect branch-cut placement using straight line cuts and (c) correct branch-cut placement obeying the residue-vector rule.

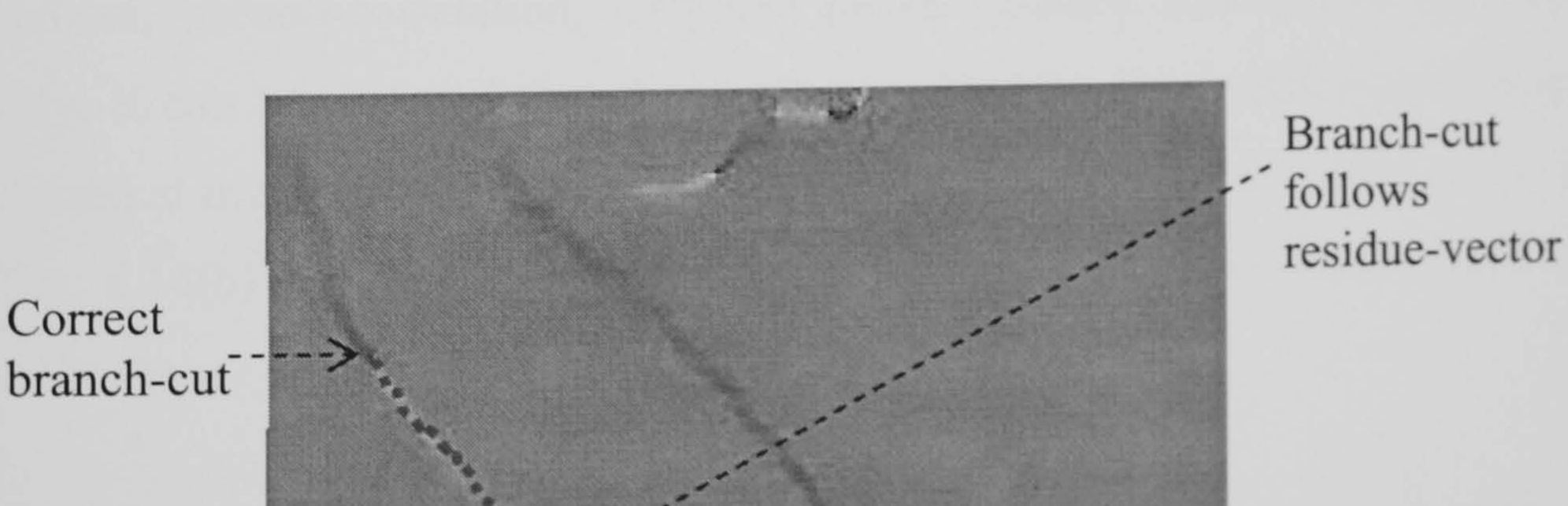



Fig. 6.13. Monopole residue correct branch-cut placement implemented on the dy wrapped phase gradient map of the wrapped phase map in 6.9(d).

Residue-vector gives all the information necessary to know how to place a branch-cut in order to balance the discontinuity in the residue. The information provided by the residue vector includes; direction, pixels to be branch-cut, and destination to an opposite polarity pixel or a border pixel.

## 6.5. Results

Branch-cut placement should obey the residue-vector orientation in order not to disturb the results of the unwrapped phase. Two sets of wrapped phase maps were used to justify this theorem; *i.e.*, simulated and real wrapped phase maps. This investigation is to verify that a branch-cut between two residues should follow the residue-vector. This is illustrated in Figs. 6.12 and 6.13.

### 6.5.1. Computer Simulation Results

A simulated wrapped phase map with residues was used to verify the residue-vector concept. The wrapped phase map is a spiral, which was taken from Ghiglia and Pritt [Ghiglia and Pritt (1998)]. The wrapped phase map and its corresponding residue map are shown in Figs. 6.2(a) and (b) respectively. By examining the residue-vector distribution in Figs. 6.2(c) and (d) of the spiral's phase gradient maps, it is shown in Fig. 6.14 (a), (b), (c), (d), (e) and (f) the masks of the quality maps generated using the following extraction methods: phase derivative variance, residue vector, maximum phase gradient, pseudo-correlation, weighted phase variance and second difference, respectively. It can be deduced that the mask generated by the residue-vector map is more localized at the residue-vector unlike the mask of the rest of the quality maps. The data in Fig. 6.14(b) was found by manually choosing the residue-vector pixels with respect to the theory presented about the residue-vector.

Moreover, the residue-vector map includes the zero-vector pixels in the mask unlike the rest of the quality maps that is not problem-specific to phase unwrapping.

Flynn's algorithm with phase variance quality map approximates the position of the residue-vector including the position of the zero-vector pixels and solves the problem of balancing the residues with minimum discontinuity or minimum branch-cut pixels as seen in Fig. 6.15(a). However, when Flynn's algorithm uses the residue-vector map, it

only solves the problem of balancing the residues with the minimum amount of discontinuities or branch-cut pixels as seen in Fig. 6.15(b).

However, Flynn's algorithm using the mask of the maximum gradient quality map approximates the position of the residue vector as seen in Fig. 6.15(c). Moreover, Flynn's algorithm fails in achieving the minimum total of branch-cut pixels. This failure is due to zero-vector pixels being not masked, thus creating a different solution than that of 6.15(a).

On the other hand, Figs. 6.15 (d) and (e) show the branch-cuts achieved by Flynn's algorithm using the masks of both pseudo-correlation and weighted phase variance quality maps, respectively. Both figures show the failure of Flynn's algorithm in locating the residue vector lines and in achieving minimum total of branch-cut pixels. Failure can be seen where branch-cuts cut the arms of the spiral in Fig. 6.15 (d) and (e).

Finally, the branch-cut result produced by Flynn's algorithm using the mask of the second difference quality map is quite similar to the results produced by Figs. 6.15(a) and (b) except for one branch-cut crossing the arm of the spiral causing a local failure of the algorithm.

By examining the unwrapped results of the spiral using the masks of the quality maps mentioned previously, it is clear from Fig. 6.16(a) that Flynn's algorithm with phase variance quality map cannot unwrap the spiral edges properly because of the random discontinuities or branch-cuts generated by the method to overlap the residue-vector even though it succeeds in unwrapping the spiral. On the other hand, Flynn's algorithm with residue-vector map, as seen in Fig. 6.16(b), produces successful unwrapped result with high precision in unwrapping at the spiral edges. This is because in this case it is able to solve the problem without random discontinuities or branch-cuts.

In the case of the unwrapped result by the Flynn's algorithm using the mask of the minimum gradient quality map shown in Fig. 6.16(c), the algorithm fails in unwrapping the spiral successfully because the algorithm did not branch-cut zero-vector pixels located in the wrapped phase map.

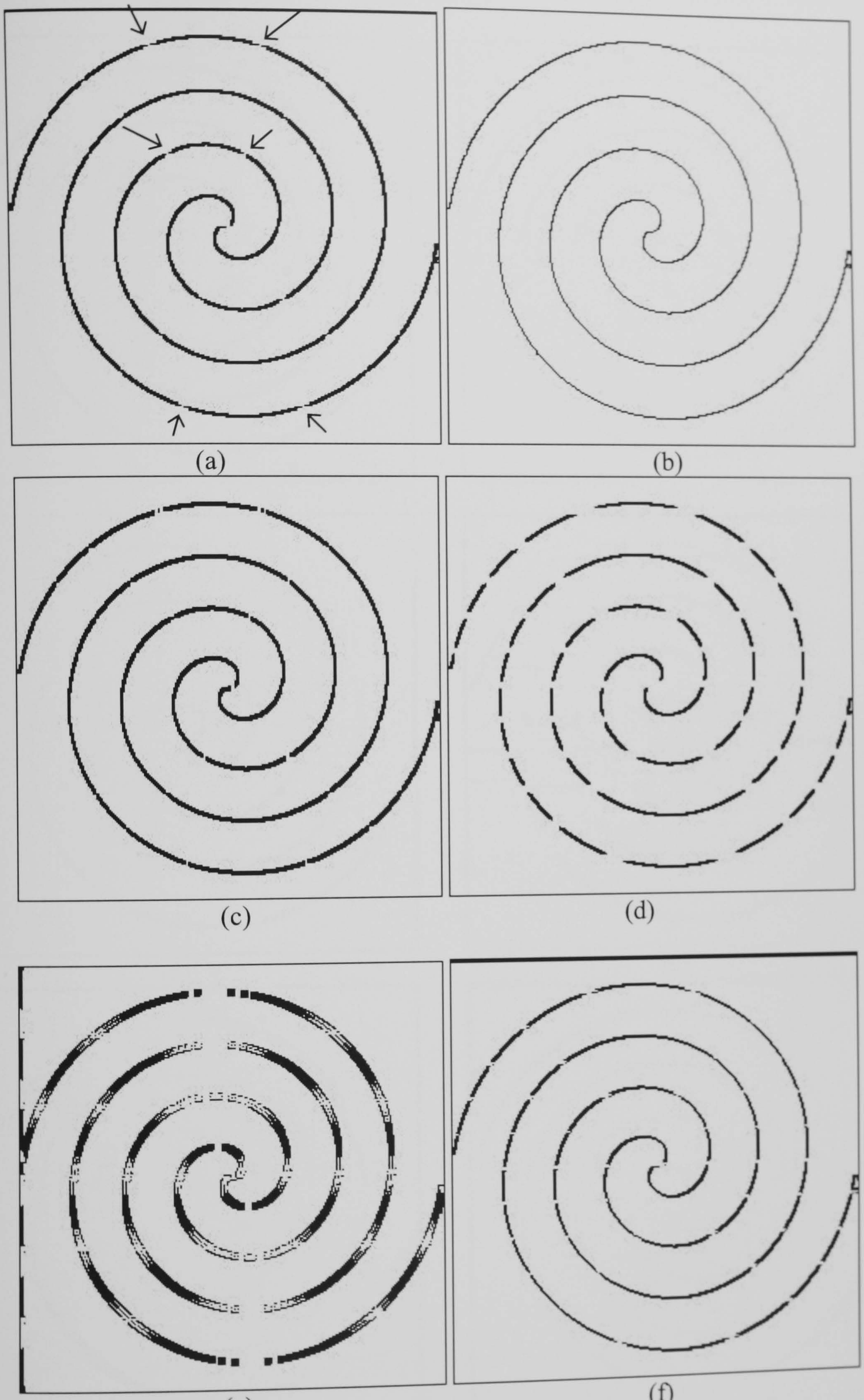
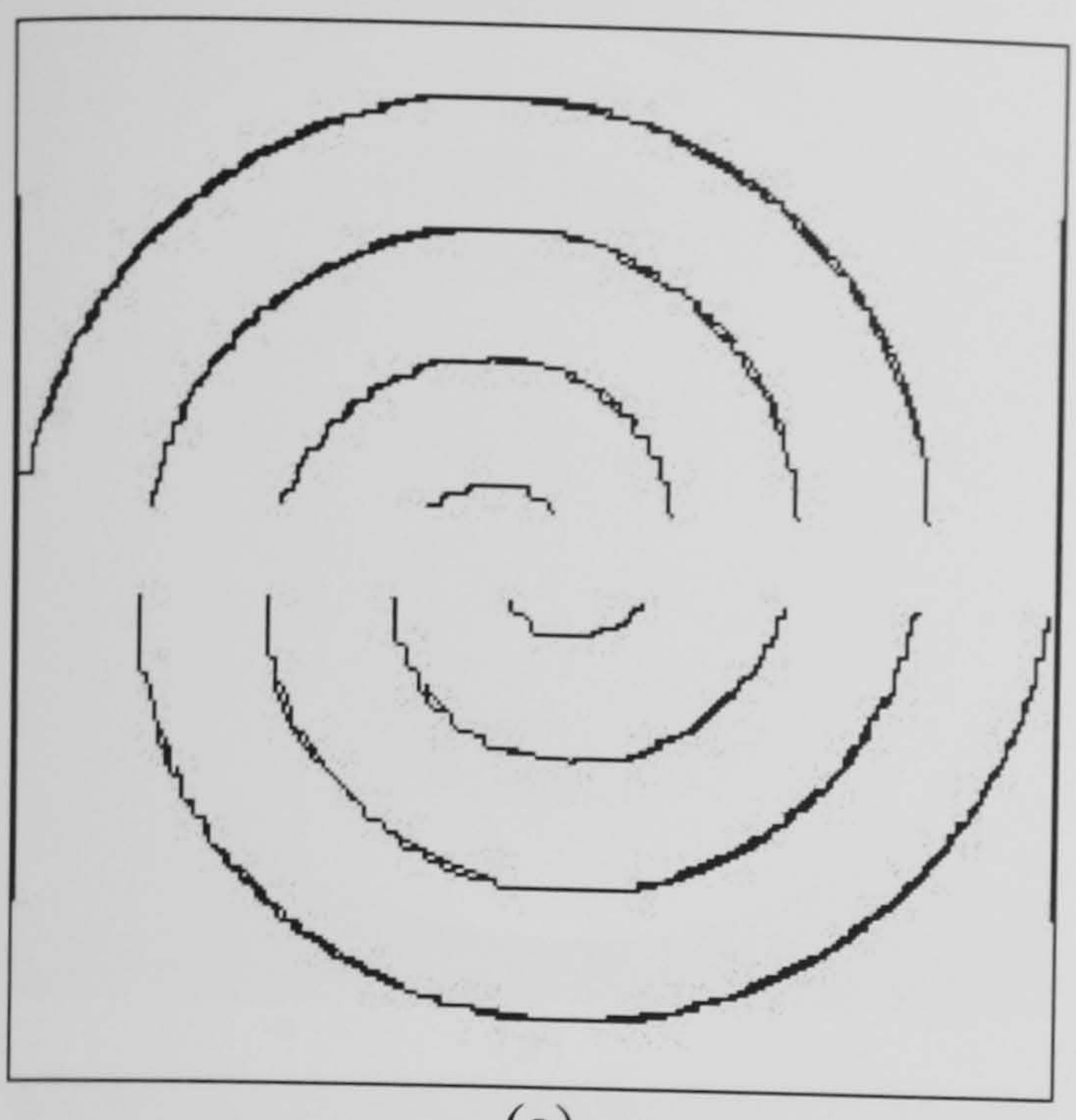
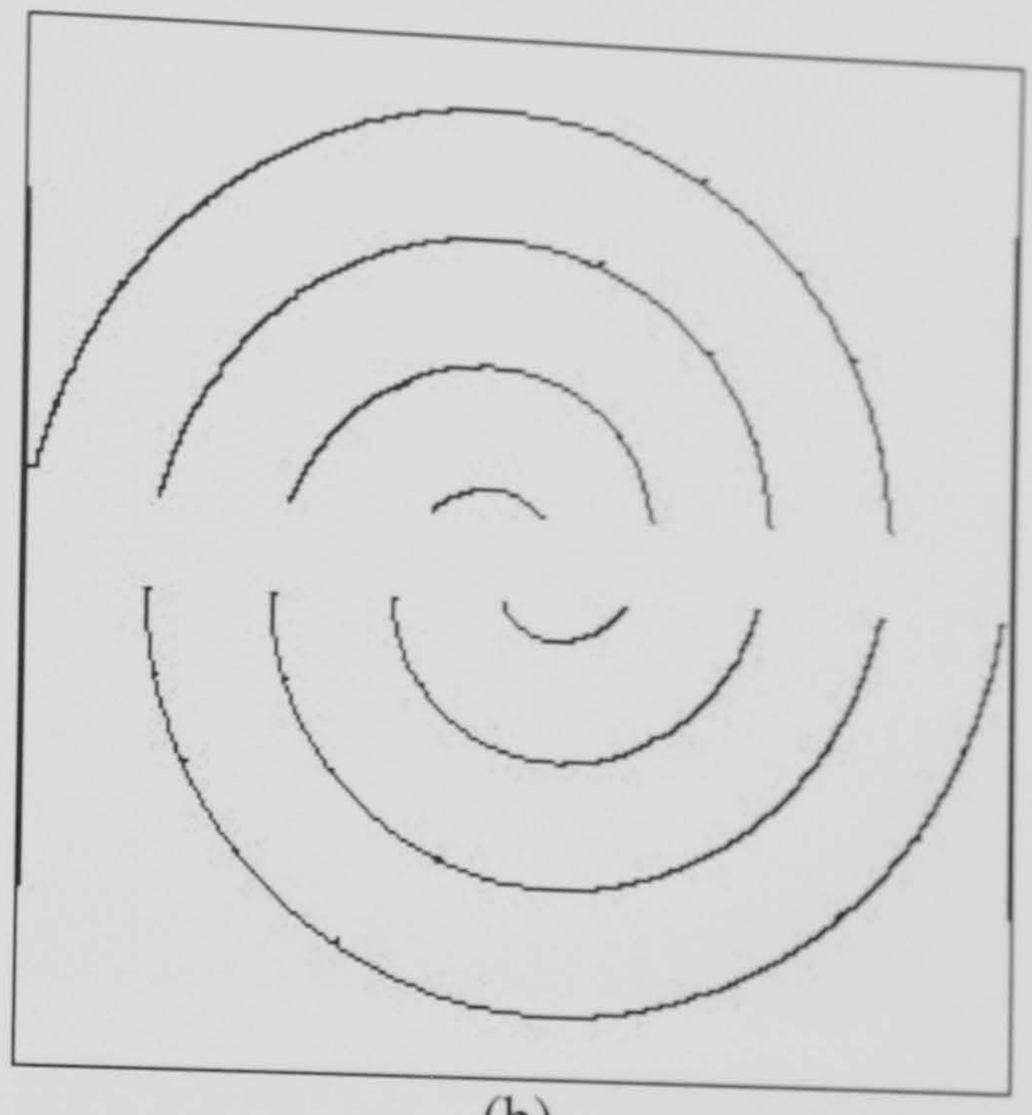


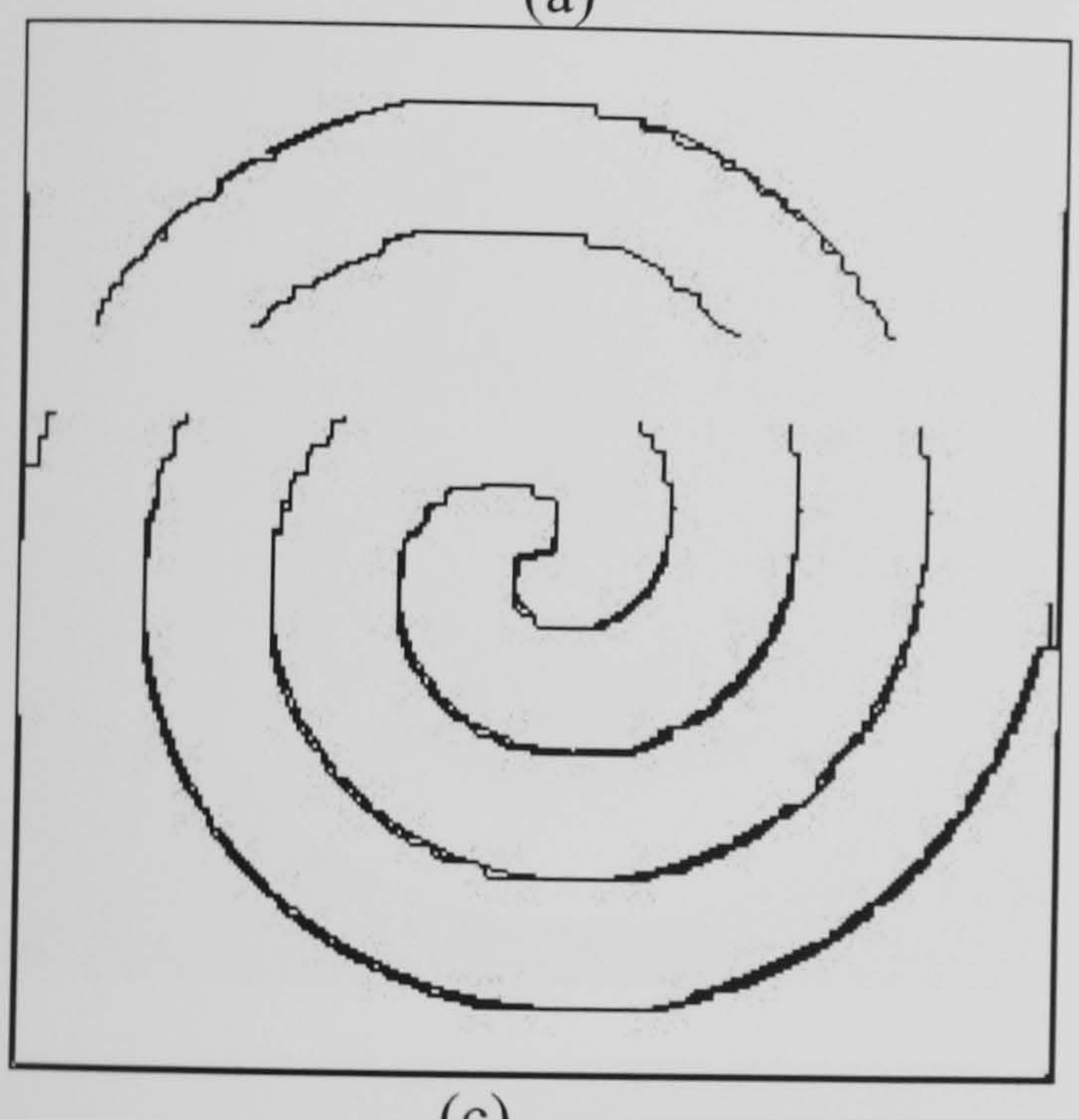

Fig. 6.14. Zero-weighted mask of the wrapped phase map in 6.2(a) using (a) phase variance quality map (arrows point at some of the non-masked zero-vector) and (b) residue-vector map showing the position of the extracted residue-vector pixels in the gradient phase map, (c) maximum phase gradient quality map, (d) pseudo-correlation quality map, (e) weighted phase variance quality map and (f) second difference quality map.



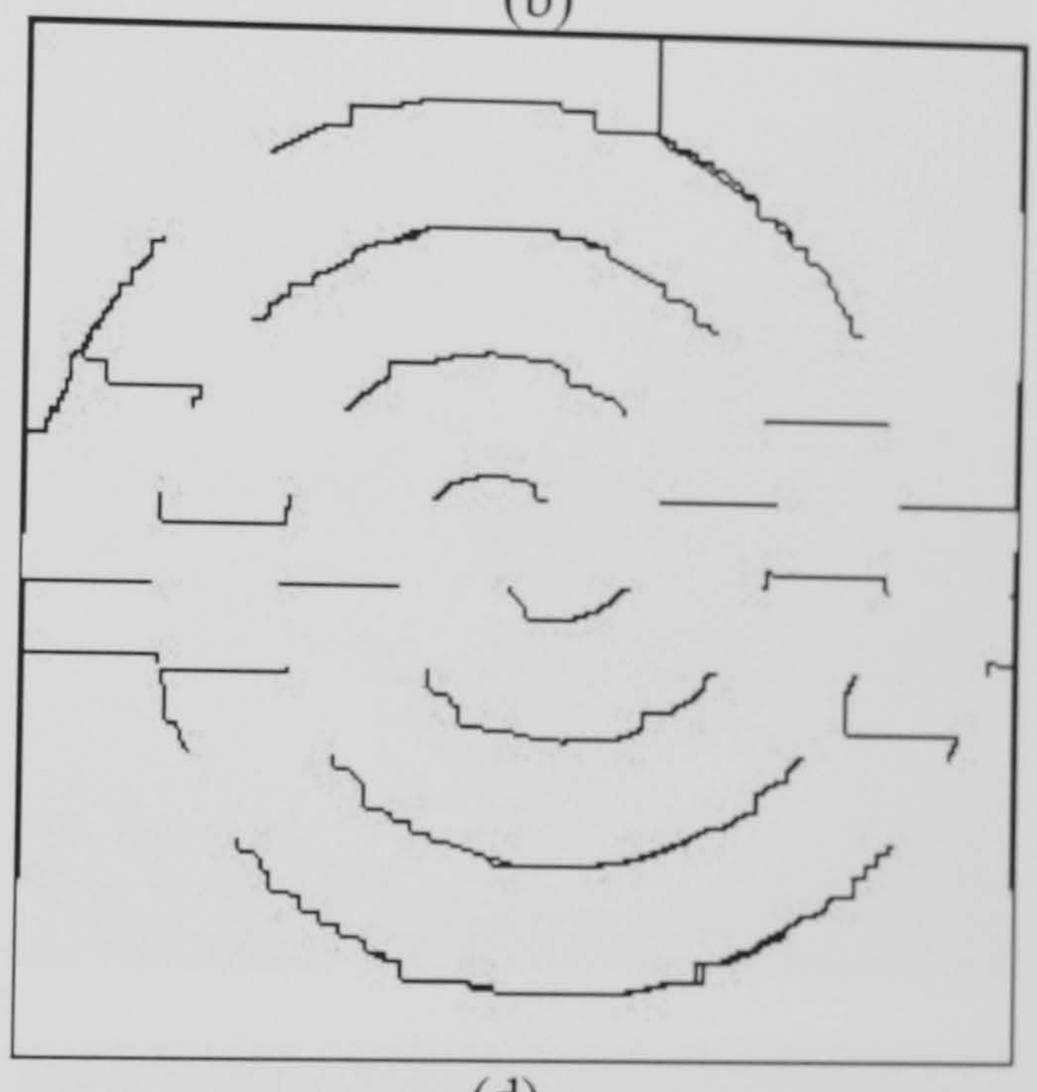
(a)



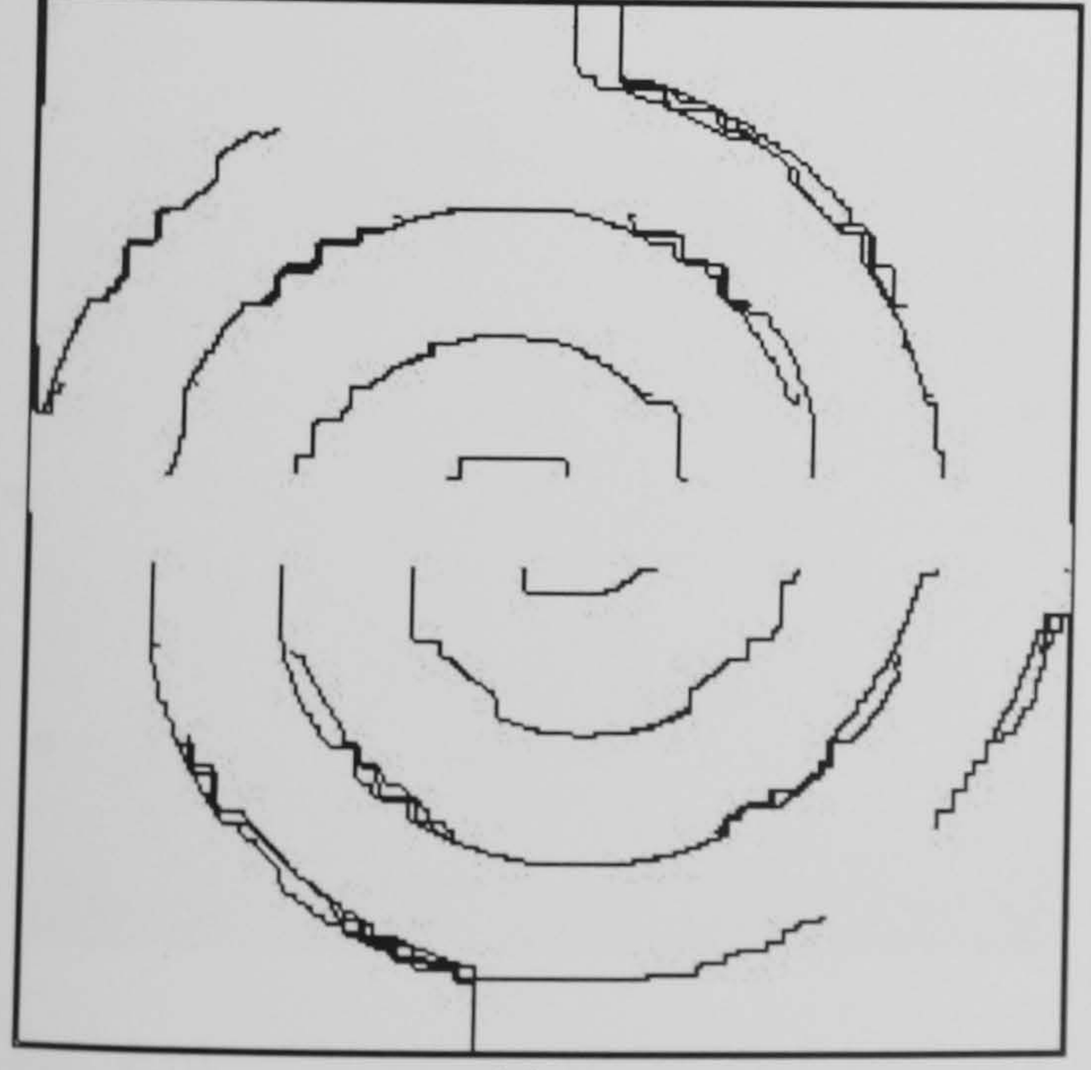
(b)



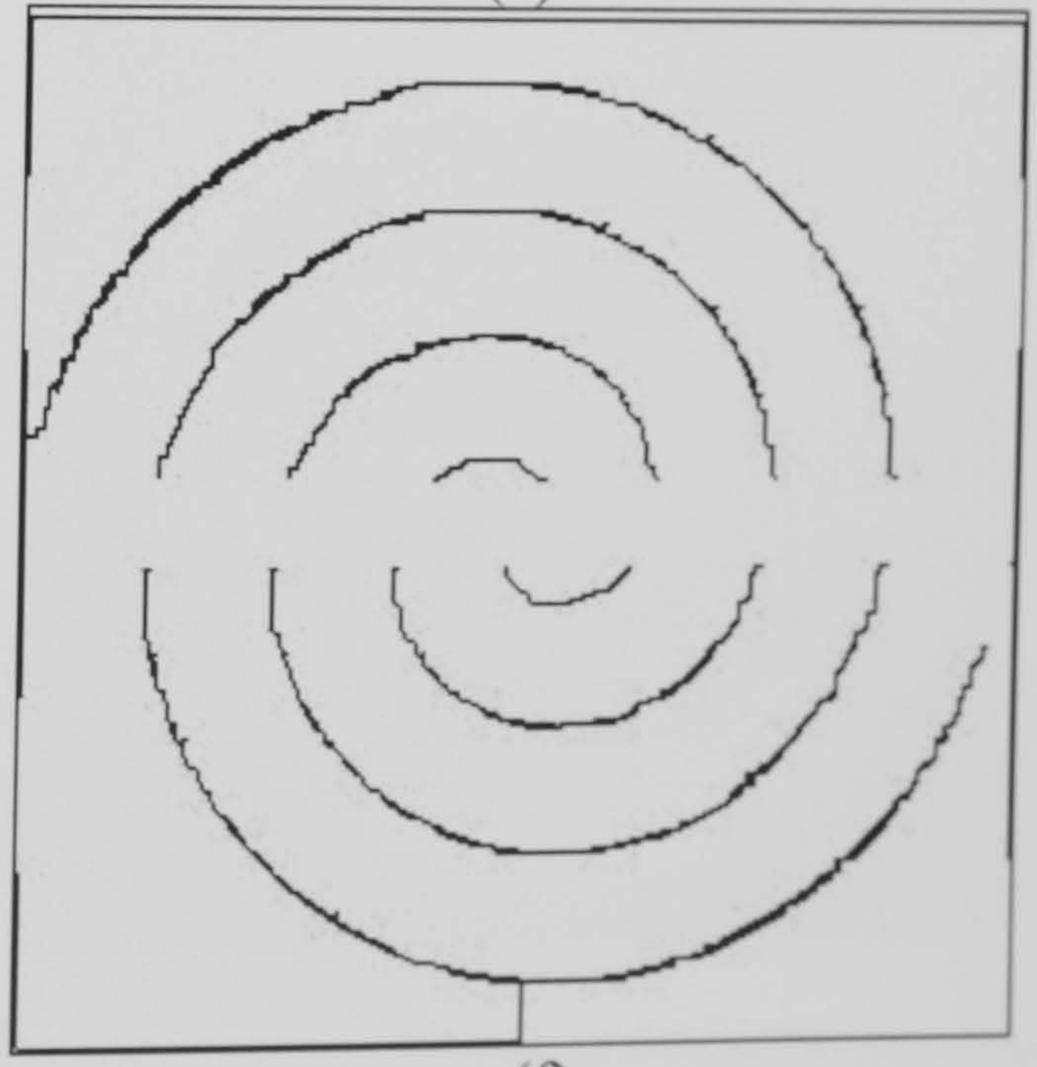
(c)



(d)

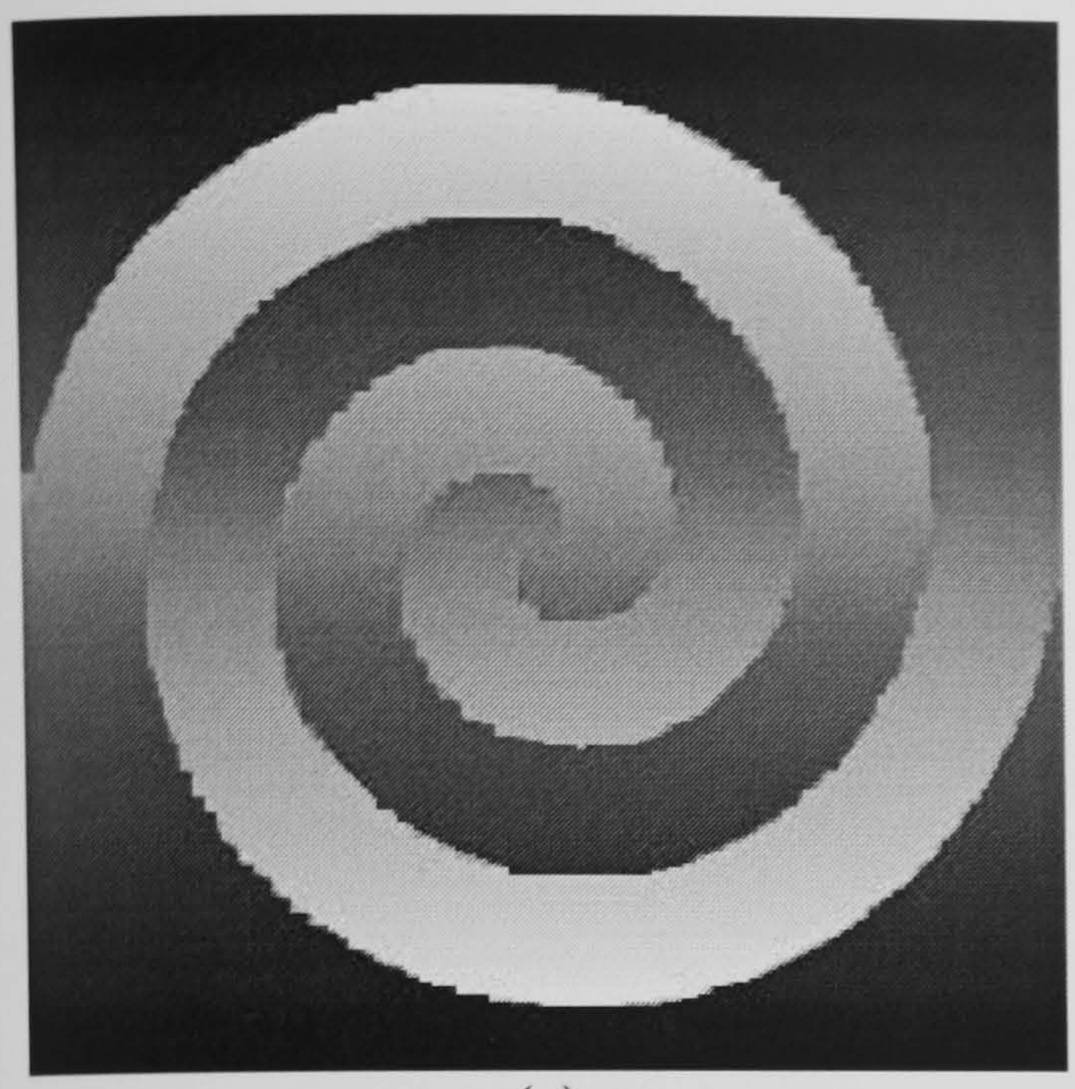


(e)

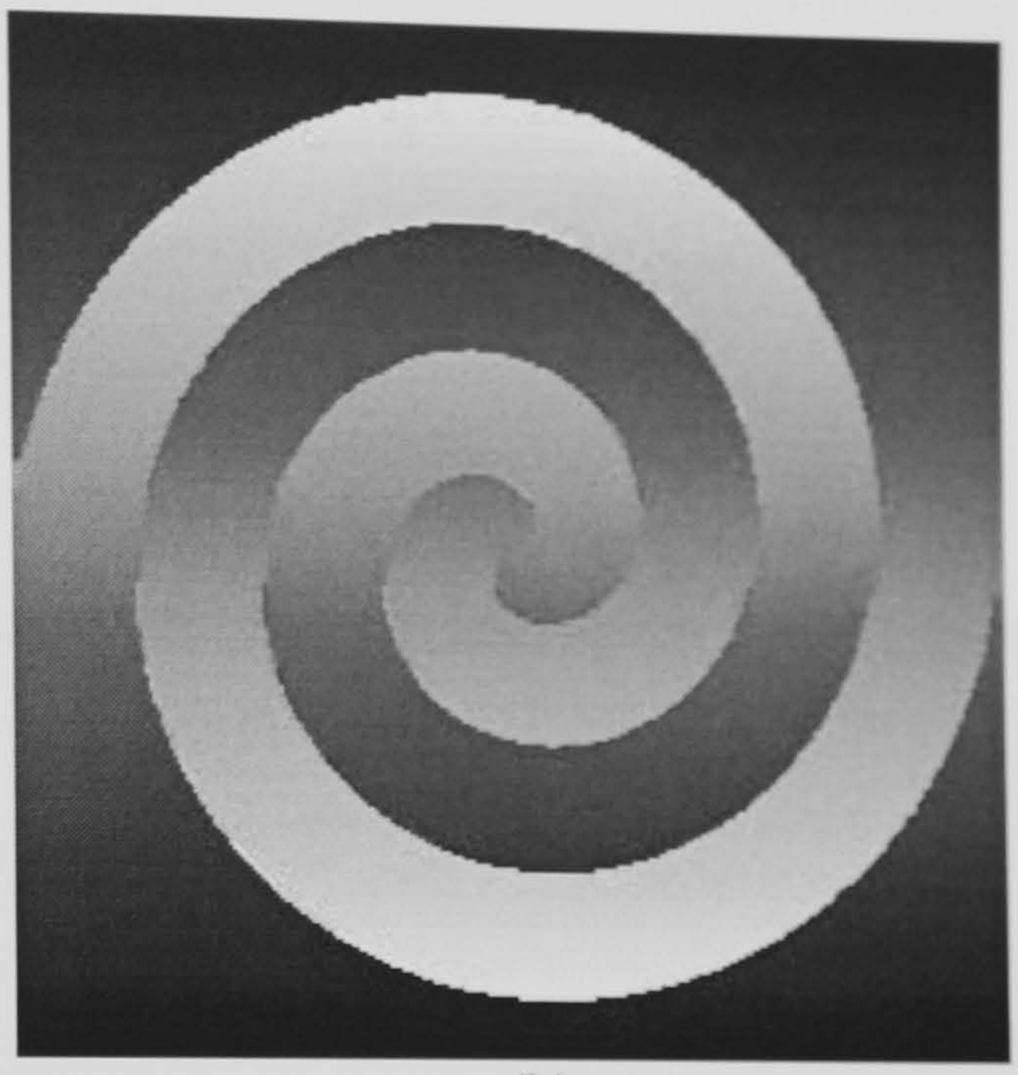


(f)

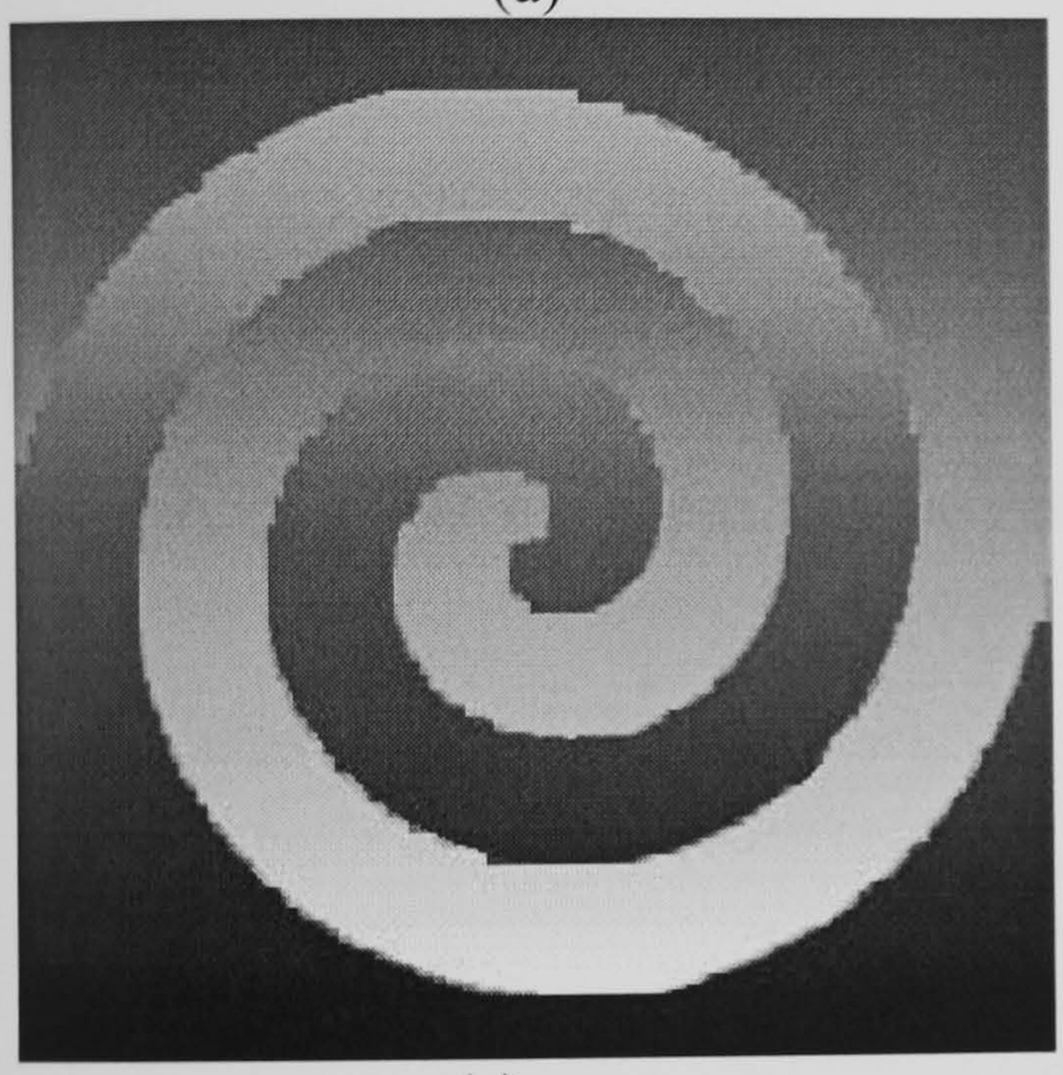
Fig. 6.15. Branch-cuts produced by Flynn's algorithm [Flynn (1997)] with zero-weights provided by the mask of the (a) minimum phase variance quality map (b) residue-vector map, (c) maximum phase gradient quality map, (d) pseudo-correlation quality map, (e) weighted phase variance quality map and (f) second difference quality map.



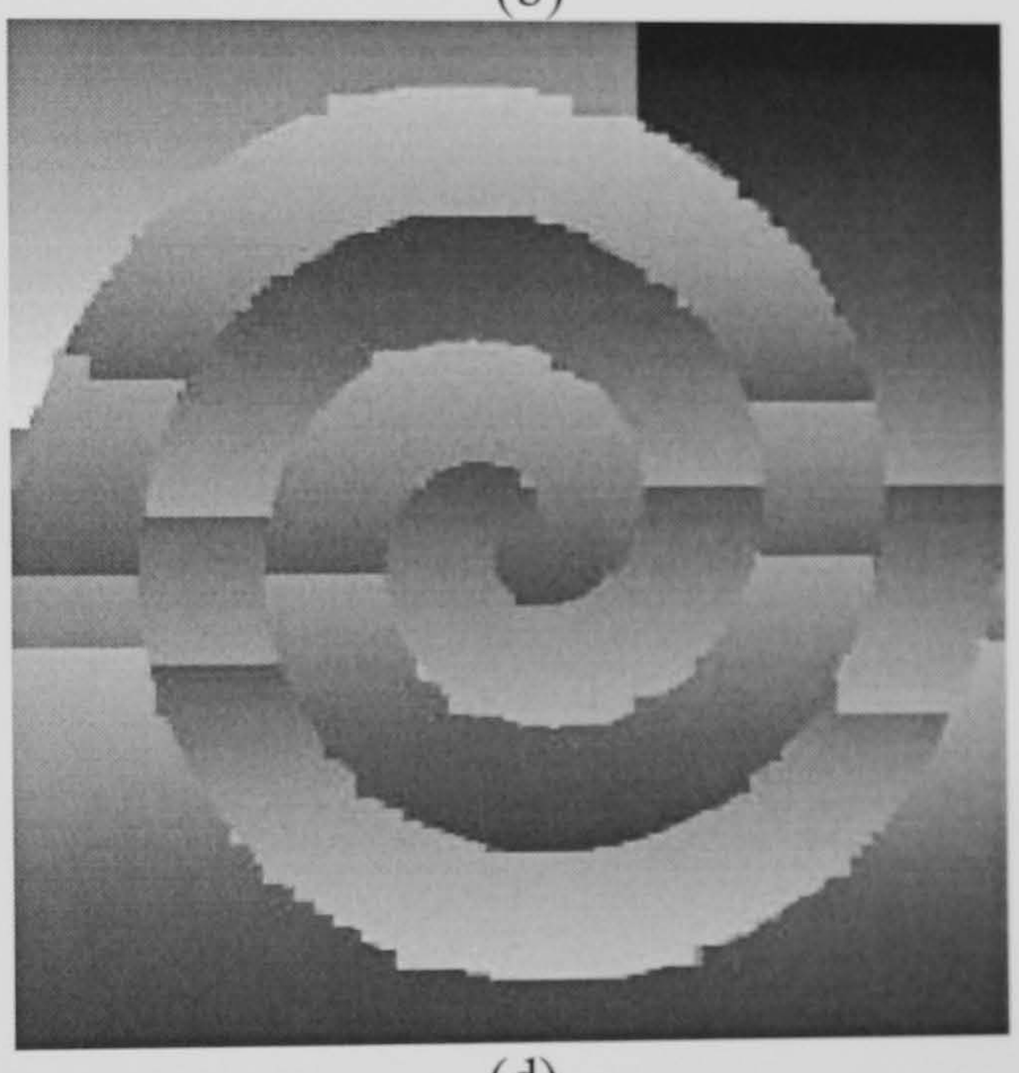
(a)



(b)



(c)



(d)



(e)



(f)

Fig. 6.16. Unwrapped phase map produced by Flynn's algorithm [Flynn (1997] with zero-weights provided by the mask of the (a) minimum phase variance quality map, (b) residue-vector map, (c) maximum phase gradient quality map, (d) pseudo-correlation quality map, (e) weighted phase variance quality map and (f) second difference quality map.

Moreover, the unwrapped results by Flynn's algorithm using the mask of both pseudo-correlation and weighted phase variance quality maps, respectively are shown in Figs. 6.16(d) and (e). The algorithm completely fails in unwrapping the spiral due to the same reasons as the results produced by the mask of the maximum gradient quality map.

On the other hand, the unwrapped result produced by Flynn's algorithm using the mask of the second difference quality map is partially successful except for the local failure due to a $2\pi$ discontinuity introduce by a misplaced branch-cut from the spiral arm to the border. But generally, this quality map produces a result similar to that of the phase derivative variance quality map.

An improved precision can be even achieved if the branch-cut map produced by Flynn's algorithm with zero-weights provided by the mask of the residue-vector map prior to unwrapping is matched with the mask of the residue-vector map. This will produce a prefect solution in unwrapping the spiral edges.

### 6.5.2. Experimental Results

The mask-cut method developed by Ghiglia and Pritt was implemented on the fairy image in Fig. 6.9(c) [Flynn (1996)].

The resulting branch-cut map is shown in Fig. 6.17(a) and its corresponding unwrapped phase map is shown in Fig. 6.17(b). Even though it uses the maximum gradient quality map, this method of branch-cut placement does not follow the residue-vector as a result the unwrapped phase map is distorted by the residue-vector. This can be seen in the 3D-surface of the unwrapped phase as in Fig. 6.17(c). The phase distortion is clearly seen on the 3D-surface causing a $2\pi$ discontinuity cut on the surface, thus, destroying the smoothness of the unwrapped surface.

On the other hand, Flynn's algorithm with zero-weights provided by all the quality maps mentioned in the previous section except the residue-vector map results in the failure of the algorithm in unwrapping the fairy properly. Flynn's algorithm using the mask of these quality maps generated branch-cuts that go across the unwrapped phase map as seen in Figs. 6.18 (a), (b), (c), (d) and (e), thus, introducing multiples of $2\pi$ discontinuities in the locations of the wrong branch-cuts.

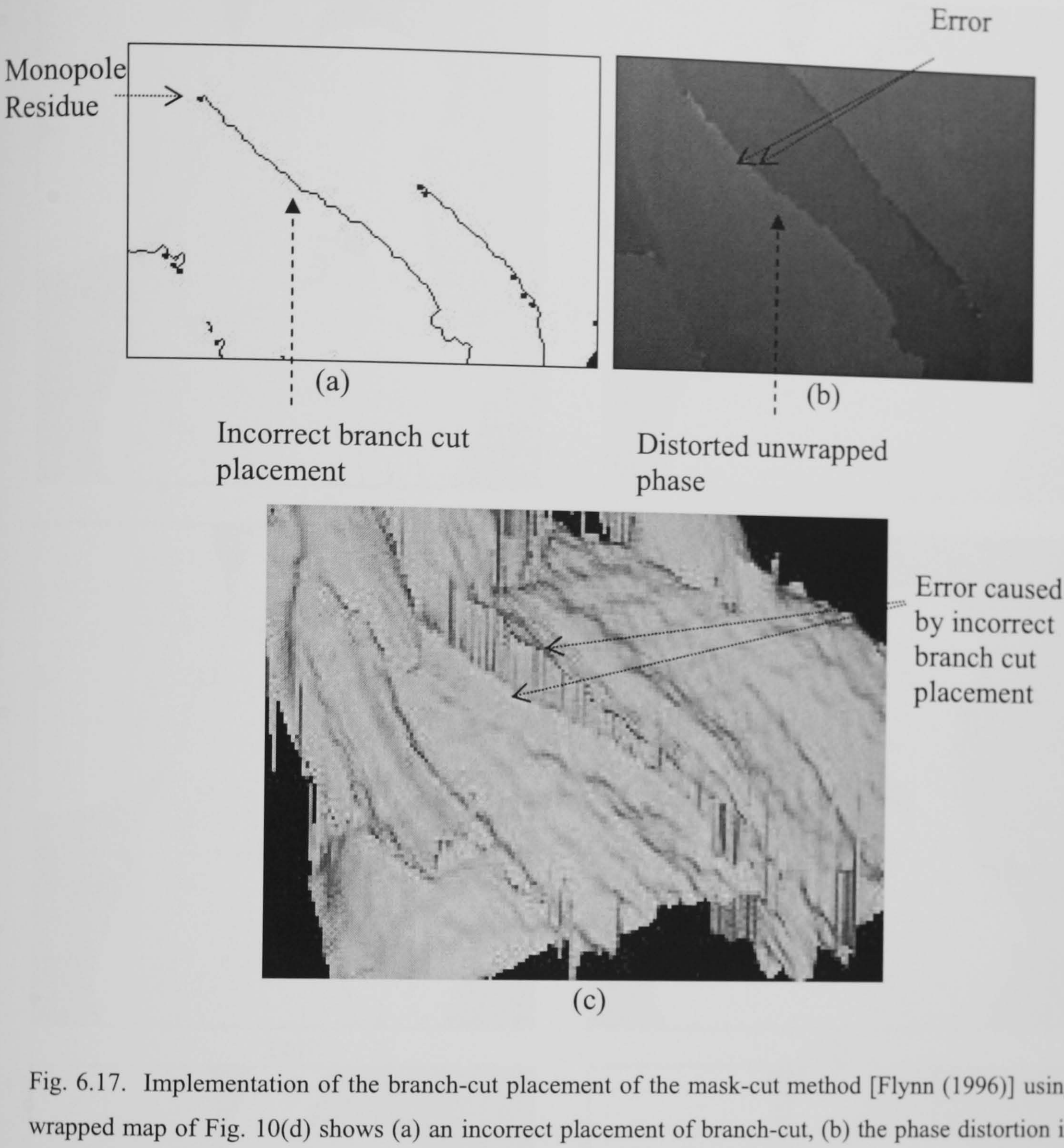
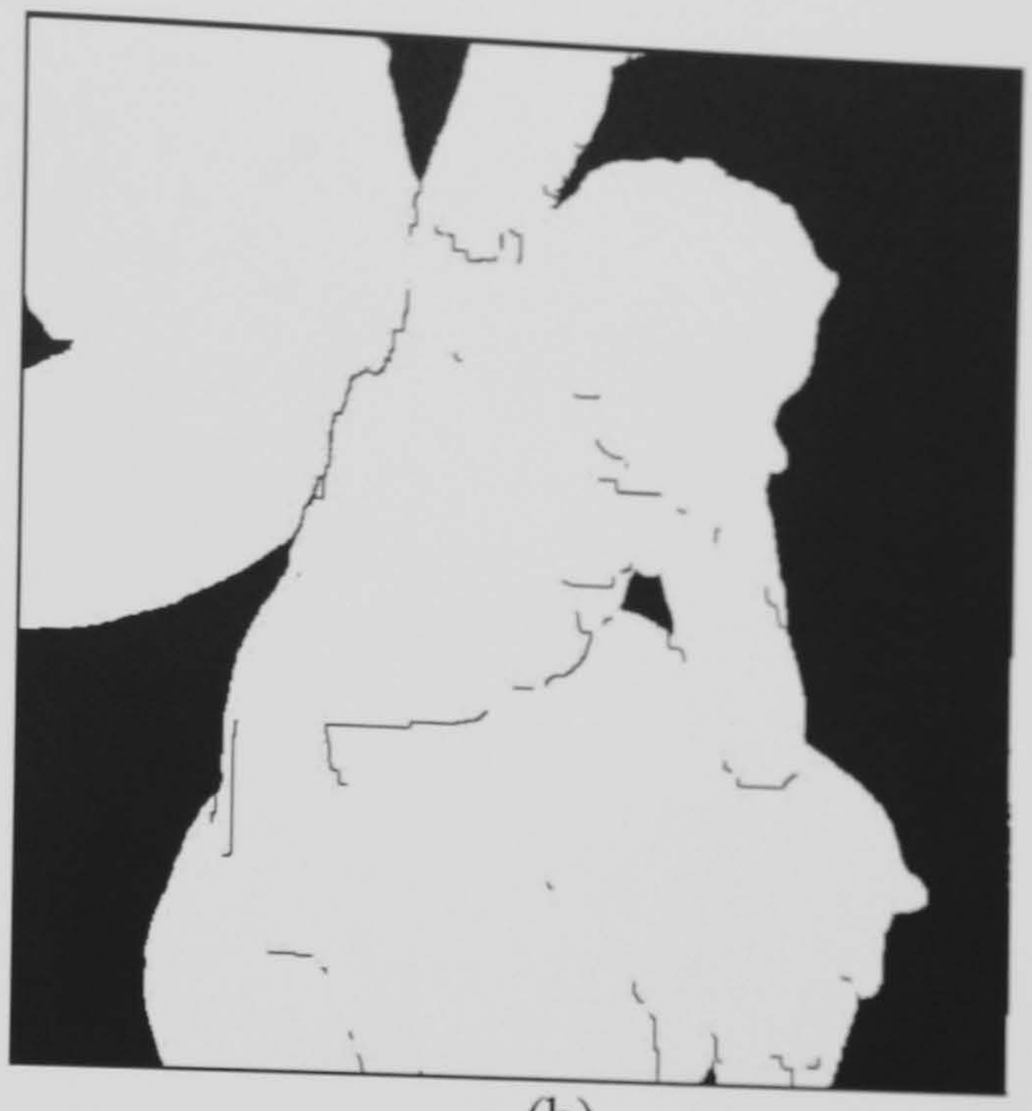



Fig. 6.17. Implementation of the branch-cut placement of the mask-cut method [Flynn (1996)] using the wrapped map of Fig. 10(d) shows (a) an incorrect placement of branch-cut, (b) the phase distortion in the unwrapped phase map and (c) the 3D-surface of the unwrapped phase with phase distortion.

However, when Flynn's algorithm was provided with the zero-weights from the residue-vector map; a successful and much improved result was achieved as in Fig. 6.18(f).

The set of branch-cuts generated by Flynn's algorithm when provided by the mask of the residue-vector map was as predicted by the author. It can be seen in Figs. 6.18 (f) that the branch-cuts that follow the residue-vectors also follow precisely the object edges in the wrapped phase map. The unwrapped results of the fairy shown in Figs. 6.19 (a), (b), (c), (d) and (e) were generated by Flynn's algorithm when provided by all the quality maps except the residue-vector map are approximately the same.

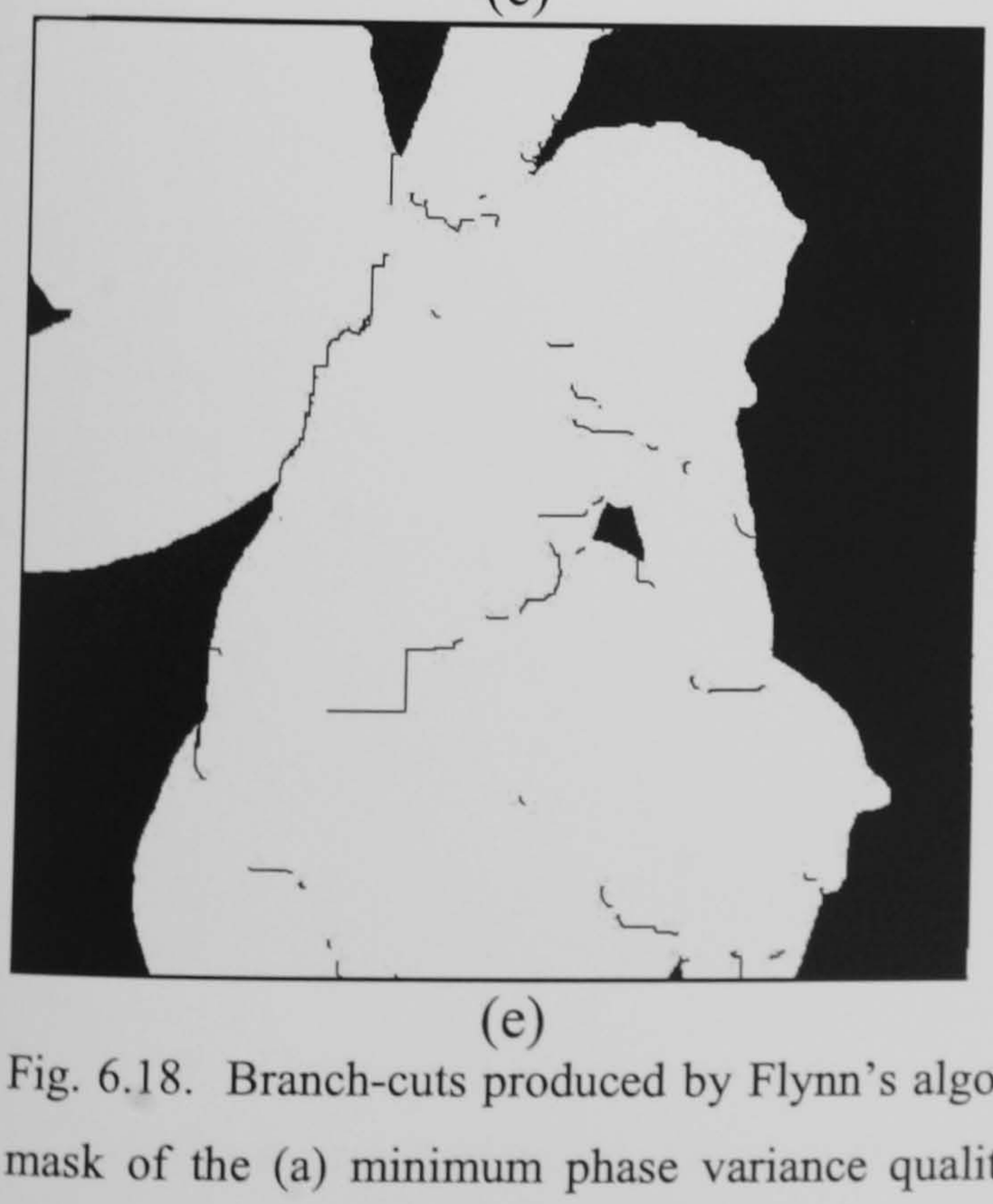
(a)



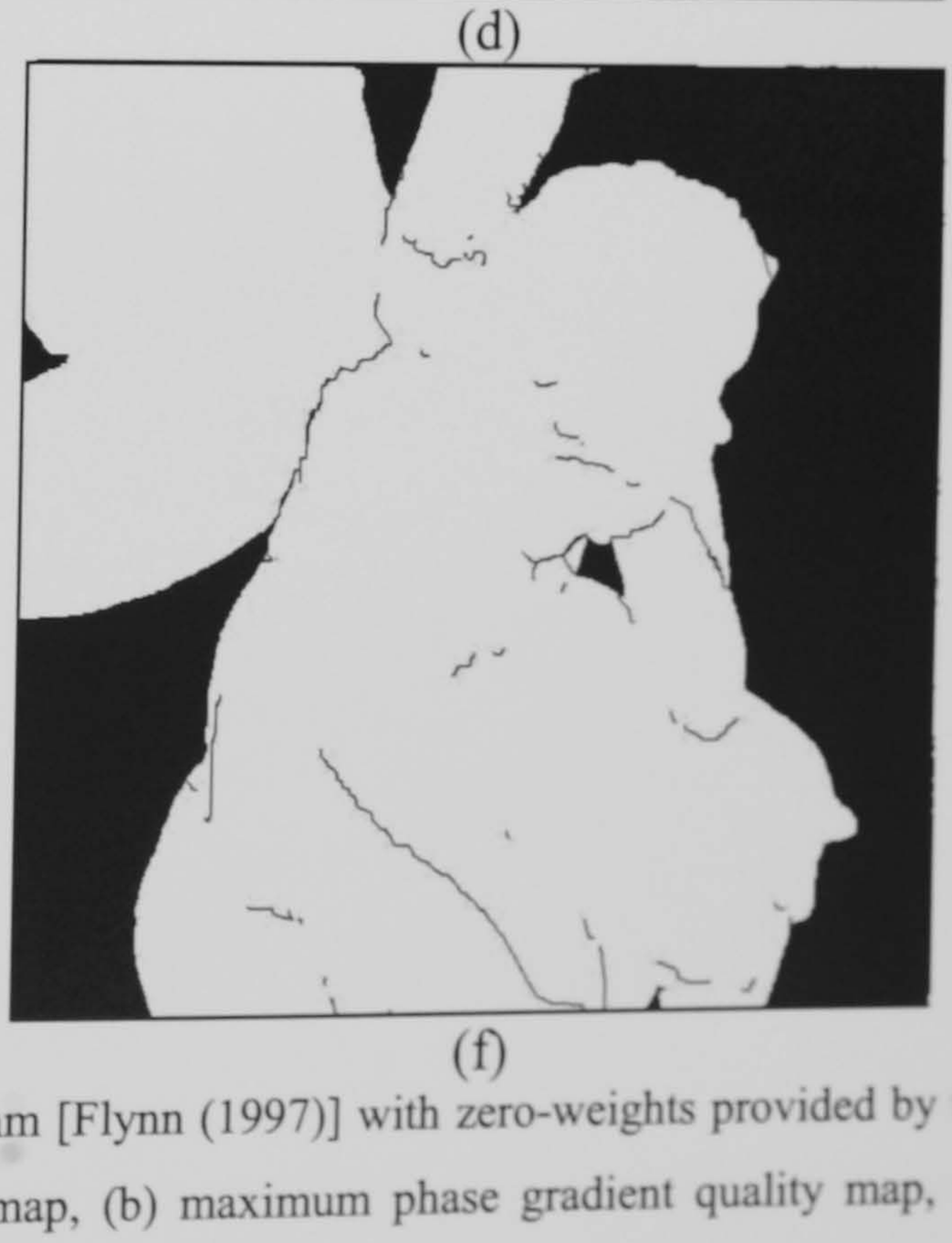
(b)

(c)

(d)

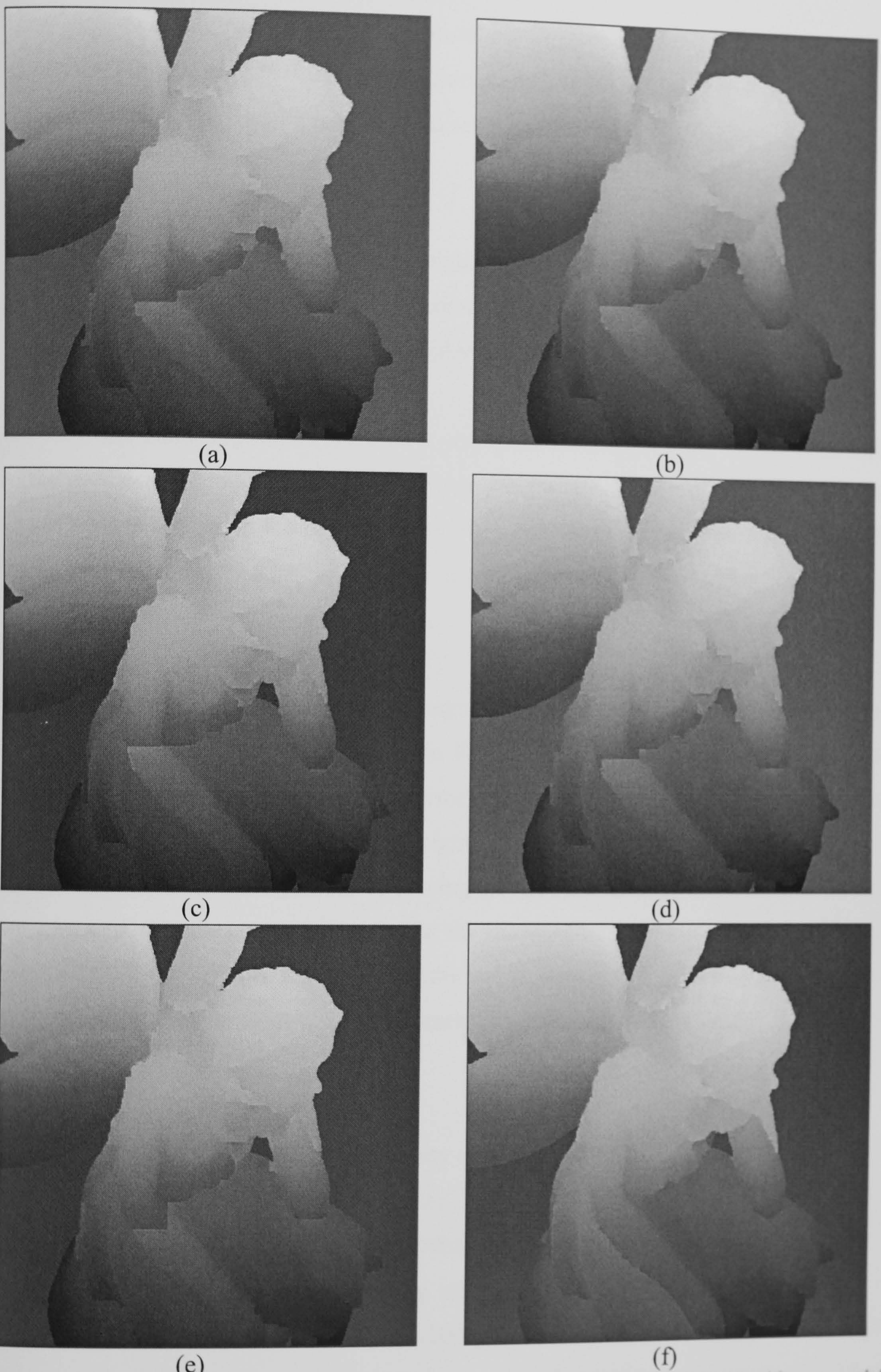


(e)



(f)

Fig. 6.18. Branch-cuts produced by Flynn's algorithm [Flynn (1997)] with zero-weights provided by the mask of the (a) minimum phase variance quality map, (b) maximum phase gradient quality map, (c) pseudo-correlation quality map, (d) weighted phase variance quality map, (e) second difference quality map and (f) residue-vector map.



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

Fig. 6.19. Unwrapped phase map produced by Flynn's algorithm [Flynn (1997)] with zero-weights provided by the mask of the (a) minimum phase variance quality map, (b) maximum phase gradient quality map, (c) pseudo-correlation quality map, (d) weighted phase variance quality map, (e) second difference quality map and (f) residue-vector map.

They also make the same unwrapping mistakes. In essence, due to branch-cuts placed across the fairy's hands, the unwrapped fairy results lose the fairy's uniform shape as a result of introducing multiples of $2\pi$ discontinuities in the locations of the wrong branch-cuts.

This is due to the lack of appropriate weights at the hand edges of the fairy. Therefore, residues were balanced on the basis of minimum branch-cut length only not either the residue-vector presence or taking in consideration the object edges.

On the other hand, Fig. 6.19 (f) shows a very good unwrapped fairy result. This is due to the use of the residue-vector map by Flynn's algorithm. In Fig. 6.19(f), it is clear that Flynn's algorithm using the residue-vector map does not violate object shape instead it preserves its characteristics. In essence, it produces unwrapped results with minimal discontinuity errors.

The experimental results presented demonstrate the validity of the theorem summarized in Eq. (6.2). This can be demonstrated in Flynn's optimization technique which relies on minimizing the total discontinuity error by minimizing the number of branch-cut pixels according to an optimum set of weights. Thus, if the algorithm lacks the optimum set of weights; then the algorithm is solving for the total minimum number of branch-cut pixels in the phase map to minimize the discontinuity error measure as shown in Figs. 6.20(a), (c) and (e) which shows the weights provided by the maximum phase gradient quality map and its resulting branch-cuts and unwrapped phase at the elbow of the fairy, respectively.

Thus, the minimization of the discontinuity error measure relies on minimizing the total minimum number of branch-cuts depending on maximizing the probability of the branch-cuts overlapping the residue-vector pixels. Moreover, the optimum set of weights for a given phase unwrapping solution is just the residue-vector itself as demonstrated in the results of Figs. 6.20(b), (d) and (f) which shows the weights provided by the residue-vector map and its resulting branch-cuts and unwrapped phase at the elbow of the fairy, respectively. This theorem was not realized by researchers who developed the minimization of discontinuity error for phase unwrapping. Their research has stopped at the lack of optimum weights for the phase unwrapping methods [Gens

(2003)]. The residue-vector information and this theorem in Eq. (6.2) answers the questions that most recent phase unwrapping researchers ask.

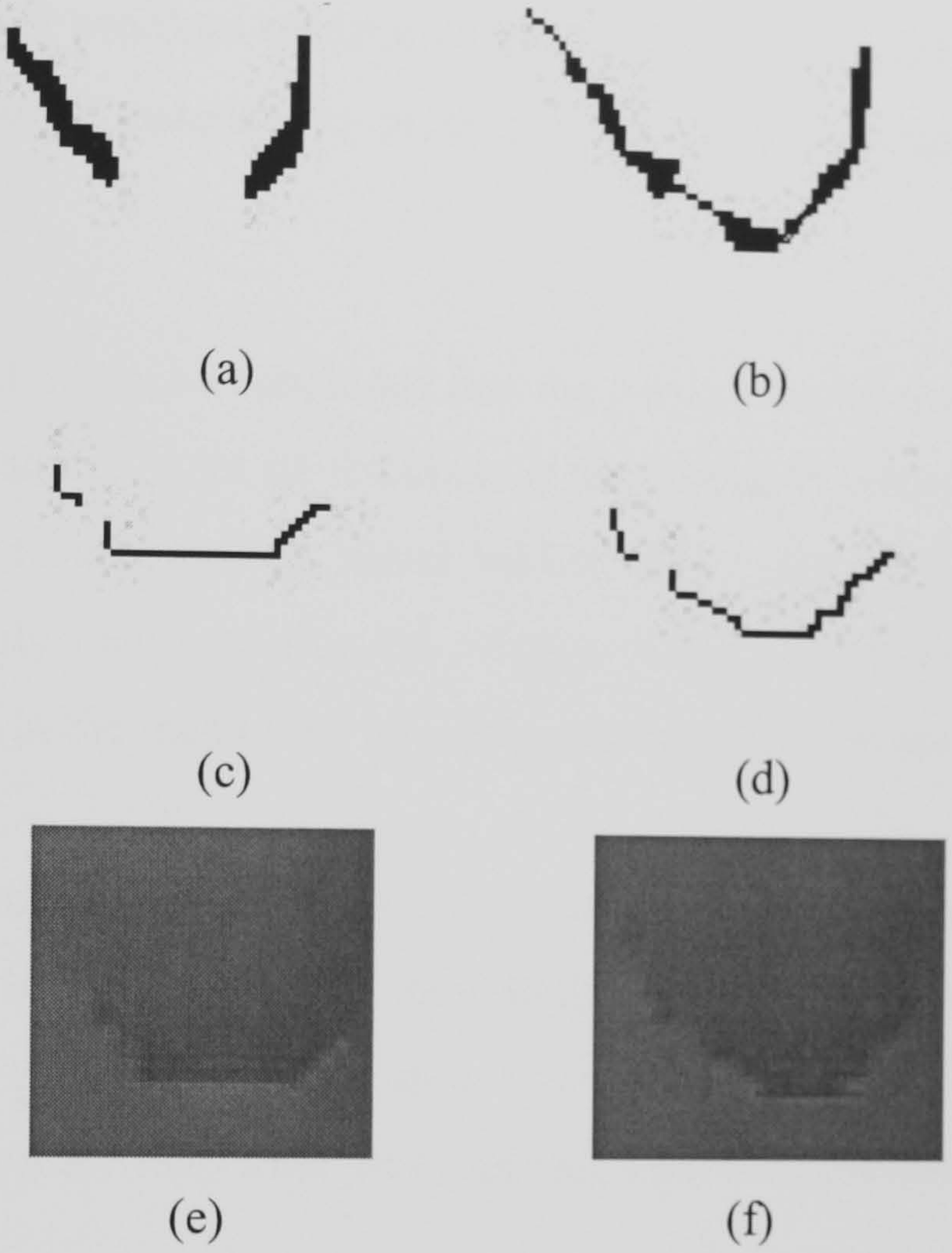



Fig. 6.20 (a) Mask of the maximum phase gradient quality map of the fairy's elbow, (b) Mask of the residue-vector map of the fairy's elbow, (c) the branch-cuts made at the elbow by Flynn's algorithm using the maximum phase gradient quality map, (d) the branch-cuts made at the elbow by Flynn's algorithm using the residue-vector map, (e) the unwrapped phase at the elbow by Flynn's algorithm using the maximum phase gradient quality map and (f) the unwrapped phase at the elbow by Flynn's algorithm using the residue-vector map.

A better result can be achieved by a better residue-vector extraction method and/or by residue-vector matching with the branch-cuts produced by Flynn's algorithm prior to unwrapping.

## 6.6. Conclusion

In conclusion, the newly developed theory of the residue-vector was presented in this chapter. It extends the phase unwrapping residue theory into a new perspective and lays the basis of future successful phase unwrapping technique.

The cause of failure of many unwrapping algorithms was presented in the examples introduced in this chapter. Especially, the presence of the zero-vector in the phase map which will work against robust phase unwrapping algorithms. Moreover, it was shown that existing quality maps not always produce successful results even the best of them, this is because they are not problem-specific to phase unwrapping unlike the residue-vector map that showed great potential in producing successful unwrapped results no matter the application.

From the results presented, it can be concluded that the residue-vector does not just give information on how to balance a set of residues; it also takes in consideration object edges and shapes. Thus, residue-vectors never violate object shape but follow object edges. In essence, a quality map based on the residue–vector information will always preserve object shape and in the mean time balanced residue effects in the phase map.

An improved precision can be even achieved if the branch-cut map produced by Flynn's algorithm or any other branch-cut phase unwrapping algorithm with zero-weights provided by the mask of the residue-vector map prior to unwrapping is matched with the mask of the residue-vector map after unwrapping. This will produce an optimum unwrapping solution.

# Chapter 7

# Conclusions and Future Work

# Chapter 7

## 7. Conclusion and Future Works

### 7.1. Conclusion

In conclusion to the thesis, a general review to the phase unwrapping problem was presented. Most of the methods that solve the phase unwrapping problem were explained and evaluated. Several quality map extraction techniques were also explained. Moreover, the importance of artificial intelligence techniques in modern applications is increasing and phase unwrapping is one problem that has been approached by solving it in artificial intelligence. However, most attempts are still immature. Thus, artificial intelligence techniques and their application to solve phase unwrapping were also presented in this thesis. The thesis presented two proposed new phase unwrapping methods using hybrid genetic algorithms. Furthermore, also presented the new discovery of the residue-vector information existing in wrapped phase maps. The newly developed theory of the residue-vector was explained. It extends the phase unwrapping residue theory into a new prospective and lays the basis of future successful phase unwrapping techniques.

A hybrid genetic algorithm for branch-cut phase unwrapping has been proposed. The proposed algorithm has been demonstrated to be very robust and computationally efficient. The branch-cut problem was formulated as a travelling salesman problem in order to benefit from all the advances in the TSP field of research using a hybrid genetic algorithm. It is now possible to solve large branch-cut problems with thousands of residues using genetic algorithms. The speed of convergence to the global optimum of the HGA was found to be comparable to that of local search algorithms.

The proposed algorithm was tested on both simulated and real wrapped phase maps. It was found that it is capable of achieving a global optimum solution for the branch-cut problem in a very short time. The results of the proposed algorithm were also compared to other branch-cut phase unwrapping algorithms. It is deduced that it is a better artificial intelligence algorithm than SA, GA and RSA in terms of speed of convergence and quality of results.

The HGA is more robust than the SA and RSA artificial intelligence algorithms. because of the memory factor (using several sets of solution stored in chromosomes as a population). Moreover, HGA also benefits from both the crossover and the mutation operators in creating and highlighting possible good solutions.

HGA is faster than graph theory methods in achieving an optimum solution, since it is a stochastic search algorithm (*i.e.*, generally it is a random search algorithm, although it has problem-specific operators). In other words, it does not rely on a step-by-step calculation and search like the graph theory minimum-cost matching algorithm and other local search algorithms.

It is important to point out that HGA is very fast and efficient in unwrapping wrapped phase maps with up to a certain number of residues. The complexity of the algorithm increases with the increase of the number of residues. This is due to the increase of the size of the chromosome, which requires a larger population size.

Even though the method of branch cutting used in this paper could limit the HGA algorithm from successfully unwrapping discontinuous objects, this algorithm has proved to be very robust in unwrapping contiguous objects. In chapter 4, the HGA was tested on a discontinuous wrapped object and produced acceptable results only because of the location of the dipole residues in the phase map being close to each other. Future modification of this algorithm will enable it to deal with both contiguous and discontinuous objects.

Another hybrid genetic algorithm using a parametric method to solve the two-dimensional phase unwrapping problem has been also proposed. This algorithm uses a genetic algorithm to estimate the coefficients of an $n^{th}$-order polynomial that best approximates the unwrapped phase map which minimizes the difference between the unwrapped phase gradient and the wrapped phase gradient. The genetic algorithm in this proposed method uses an initial solution to speed convergence. The initial solution is achieved by unwrapping using a simple unwrapping algorithm and estimating the parameters of the polynomial using weighted least squares multiple regression.

The algorithm was then tested on simulated and experimental data and it proved to be efficient and robust. The comparison of performance of this algorithm was made with powerful established phase unwrapping algorithms such as the $L^p$-norm. Based on the rewrapping of the solution, the newly proposed method gave improved results that best matched the original wrapped phase map. However, the complexity of the object surface in the wrapped phase map result in very small polynomial coefficients that causes a large burden on the algorithm and result in expensive execution time. This complexity could be lowered by using overlapping windows that divide the wrapped phase map into regions that could be processed separately by the proposed algorithm.

The problem of phase unwrapping that was presented and explained in the previous chapters can be summarized by the problem of residues and their branch-cuts. The residue and branch-cut problem has been approached by many researchers. Many algorithms have been also developed to solve this problem. Researchers such as Chavez *et al.* and Salfity *et al.* have identified and linked the problem of high wrapped phase gradient to the cause of discontinuity formed by residues which is identified as branch-cuts [Huntley (2001), Huntley (1989)]. However, they were not able to distinguish high wrapped phase gradient causing these branch-cuts from other sources of high wrapped phase gradient. Other good and computationally exhaustive algorithms, like Flynn and minimum cost flow, approximated the position of the perfect branch-cuts provided by a good zero-weight quality map.

It was found by the author that the discontinuity lines presented as branch-cuts are caused by a residue-vector. The residue-vector was defined as a vector generated by a residue in the wrapped phase map that has an orientation pointing out to the balancing residue of opposite polarity. Different kinds of residue-vectors were presented, studied and their causes were illustrated. Moreover, with the new knowledge of the residue-vector, branch-cut placement techniques were presented to demonstrate the method of achieving an optimum unwrapping.

It was found out that the residue-vector gives all the information on how to place a branch-cut to balance the discontinuity in the residue. Information ranges from direction, pixels to be branch-cut, and destination of an opposite polarity pixel or a border pixel. In the minimum cost flow technique [Ghiglia and Romero (1996), Goldstein *et al.* (1988)]

placing a branch-cut between two residues results in a network with a set of nodes and flows representing possible branch-cuts where there exits one optimal branch-cut. The time necessary to identify this optimal branch-cut is saved in the residue-vector method of branch-cut placement because it identifies the optimal branch-cut directly from the vector.

The residue-vector technique also identifies the possible residues that could be used to balance a residue unlike the minimum cost flow that has to solve the network flows until it identifies the optimum balancing residue, *i.e.*, all residues in the network are possible balancing residues which adds to the complexity of the problem. The same holds true with Flynn's algorithm that has to identify the minimum discontinuity measure to achieve good unwrapping.

It is important to note that the quality guided and reliability phase unwrapping methods will definitely fail if there exists a zero-vector in the wrapped phase map. Further it was shown that the occurrence of such zero-vectors can be common if the phase noise level is high.

It was found that the residue-vector information can provide the best weighting factors to a different range of phase unwrapping algorithms as demonstrated in the previous section, where it was implemented with Flynn's algorithm where unwrapping with the zero-weight residue-vector mask resulted in perfect precision to unwrap object details. Moreover, the optimum set of weights for a given phase unwrapping solution is just the residue-vector itself. The theorem summarized in Eq. (6.2) was not realized by researchers who developed the minimization of discontinuity error for phase unwrapping. Their research has stopped at the lack of optimum weights for the phase unwrapping methods [Gens (2003)]. The residue-vector information and this theorem in Eq. (6.2) answers a number of important and topical questions in phase unwrapping.

Using image processing and phase unwrapping knowledge, an optimum algorithm has to be developed to extract the residue-vector as it can be done by human eye. Once the optimum residue-vector extraction algorithm is achieved using the study that has been made by this thesis on residue-vectors, the problem of actually unwrapping the phase itself becomes trivial.

In summary, the advantages of using residue-vector information for solving the phase unwrapping problem are as follows:

- Problem-specific.
- Faster than any other efficient phase unwrapping method.
- It can unwrap any contiguous and discontinuous wrapped phase objects or features.
- It can unwrap successfully independent of the level of under-sampling present in the wrapped phase map.
- It is powerful in the presence of noise.
- Image size is not an issue.
- Unwrapping application is not an issue.
- This technology demonstrates that the phase unwrapping problem is not an NP-hard problem.
- Most important, it can provide very high precision in unwrapping especially unwrapping small details. This is not achieved by the most robust phase unwrapping algorithms.

## 7.2. Future Work and Suggestions

It is proposed for future work to develop the theory and to formulate the newly discovered 'residue-vector' for phase unwrapping. Also, this theory can be extend to be the basis of creating the necessary tools for the residue-vector extraction from the wrapped phase images and to develop its corresponding phase unwrapping method.

To achieve a universal phase unwrapping technique that can solve any kind of wrapped phase map; first, it is important to study the residue-vector behaviour in different types of wrapped phase maps. Then, theory and formulation will accompany this study. By means of enough knowledge of the residue-vector method, a method of extraction should be developed and implemented. An unwrapping method has to be developed based on the extracted residue-vector map, which consists of identifying the minimum amount of discontinuity that could lead to an optimum phase unwrapping solution by problem-specific residue-vector optimization. In essence, instead of providing the residue-vector map to a general phase unwrapping optimization algorithm which is time consuming, a problem-specific algorithm could save time to make this method

appropriate to high speed real time processing. Finally, the developed algorithm will be tested on many applications and on different known phase unwrapping problems.

This consideration of the residue-vector in phase unwrapping and future work and suggestion made by this thesis can be summarized in the following points:

1. To investigate the residue-vector deeply and to further develop its theory and formulations.
2. To develop an extraction method that can extract both the high-gradient residue-vector and the zero-vector. It is expected to employ both image processing and artificial intelligence.
3. To develop an optimization method based on the extracted residue-vector map in order to speed the unwrapping process by having an unwrapping algorithm adapted to the residue-vector theory.
4. To investigate the residue-vector behaviour and theory in 3D wrapped phase maps for volume unwrapping and to expand the 2D extraction and optimization technologies achieved to 3D.
5. To demonstrate in both theory and practice the residue-vector technology in number of application areas where current technologies would either fail or produce unsatisfactory results.
6. To produce a compact residue-vector phase unwrapping algorithm that could serve systems that requires unwrapping procedure no matter what the applications is.

# References

Abdul-Rahman, H., Gdeisat, M., Burton, D., Lalor, M., Lilley. F. and Moore. C., (2007), Fast and Robust Three-dimensional Best Path Phase Unwrapping Algorithm, Applied Optics (submitted and waiting for the final decision).

Baraglia, R., Hidalgo, J. I. and Perego, R., (2001), A Hybrid Heuristic for the Travelling Salesman Problem, IEEE Transactions on Evolutionary Computation, vol. 5, pp. 613-622.

Buckland, J. R., Huntley, J. M., and Turner, J. M., (1995), Unwrapping noisy phase maps by use of a minimum-cost-matching algorithm, Applied Optics, vol. 34, pp. 5100-5108.

Chavez, S., Xiang, Q. and An, L., (2002), Understanding phase maps in MRI: a new cutline phase unwrapping method, IEEE Transactions on Medical Imaging , vol. 21, no. 8, pp. 966-977.

Chen, C. and Zebker, H., (2000), Network approaches to the two-dimensional phase unwrapping: intractability and two new algorithms, Applied Optics, vol. 17, pp. 401-414.

Cho, B., (2004), Quality map extraction for radar interferometry using weighted window," Electronic Letters, vol. 40, no. 8.

Collaro, A., Fornaro, G., Franceschetti, G., Lanari, R., Sansosti, E. and Tesauro, M., (1997), Local, global and unconventional phase unwrapping techniques, 1997 International Geoscience and Remote Sensing Symposium. Remote Sensing - A Scientific Vision for Sustainable Development, pp. 433-435.

Collaro, A., Franceschetti, G., Palmieri, F. and Ferreiro, M. S., (1998), Phase unwrapping by means of genetic algorithms, Applied Optics, vol. 15, no. 2, pp. 407-418.

Costantini, M., (1998), A novel phase unwrapping method based on network programming, IEEE Transactions on geoscience and remote sensing, vol. 36, pp. 813-821.

Cuevasa, F.J., Sossa-Azuelab, J.H. and Servin, M., (2002), A parametric method applied to phase recovery from a fringe pattern based on a genetic algorithm, Optics Communications, vol. 203, pp. 213-223.

Cusack, R., Huntley, J. M., and Goldstein, H. T., (1995), Improved noise-immune phase-unwrapping algorithm, Applied Optics, vol. 24, pp. 781-789.

Cusack, R. and Papadakis, N. (2002) New Robust 3-D Phase Unwrapping Algorihtms: Application to Magnetic Field Mapping and Undistorting Echoplanar Images. *Nero Image,* 16, 754-764.

Flynn, T., (1996), Consistent 2-D phase unwrapping guided by a quality map, in IEEE Proceeding of the 1996 International Geoscience and Remote Sensing Symposium, Lincoln, vol. 4, pp. 2057-2059.

Flynn, T., (1997), Two-dimensional phase unwrapping with minimum weighted discontinuity, Applied Optics, vol. 14, pp. 2691-2701.

Fornaro, G. and Franceschetti, G., (1995), Two dimensional phase unwrapping based on the Laplace and Eikonal equations, 1995 International Geoscience and Remote Sensing Symposium, IGARSS '95. Quantitative Remote Sensing for Science and Applications, vol. 3, pp. 1828-1830.

Fornaro, G., Franceschetti, G. and Lanari, R., (1996), Interferometric SAR phase unwrapping using Green's formulation,'' IEEE Transaction Geoscience Remote Sensing, vol. 34, pp. 720–727.

Fornaro, G., Franceschetti, G., Lanari, R., Sansosti, E., and Tesauro, M., (1997), Global and local phase-unwrapping techniques: a comparison, Applied Optics, vol. 14, No. 10, pp. 2702-2708.

Gens, R., (2003), Two-dimensional phase unwrapping for radar interferometry: developments and new challenges, International Journal of Remote Sensing, Vol. 24, No. 4, pp. 703-710.

Ghiglia, D. C. and Romero, L. A., (1994), Robust two-dimensional weighted and unweighted phase unwrapping that uses fast transforms and iterative methods, Applied Optics, vol. 11, no. 1, pp. 107-117.

Ghiglia, D. C., and Romero, L. A., (1996), Minimum $L^p$-norm two-dimensional phase unwrapping, Applied Optics, vol. 13, No. 10, pp. 1999-2013.

Ghiglia, D. C., and Pritt, M. D., (1998), Two-Dimensional Phase Unwrapping: Theory, Algorithms and Software, John-Wiley & Sons.

Goldstein, R., Zebker, H. and Werner, C., (1988), Satellite radar interferometry: two-dimensional phase unwrapping, Radio Science, vol. 23, pp. 713-720.

Guerriero, L., (1998), New regularization scheme for phase unwrapping, Applied Optics, vol. 37, no. 14, pp. 3053-3058.

Gutmann, B., (1999), Phase unwrapping with the branch-cut method: clustering of discontinuity sources and reverse simulated annealing, Applied Optics, vol. 38, pp. 5577-5793.

Hamzah, S., Pearson, J.D., Lisboa, P.J. and Hobson, C.A., (1997). Phase unwrapping in 3-D shape measurement using artificial neural networks, IEE Conference Publication No. 443, pp. 680-683.

Herra´ ez, M.A., Gdeisat, M.A., Burton, D.R., and Lalor, M.J., (2002). Robust, fast, and effective two-dimensional automatic phase unwrapping algorithm based on image decomposition, Applied Optics, vol. 41, pp. 7445-7455.

Hui, P. T. H. and Chai, W. Y., (2001), Towards hybrid 2D phase unwrapping using fuzzy clustering and neuro-fuzzy learning for SAR images a case study on IFSAR phase image, Proceedings of 2001 international Symposium on Intelligent Multimedia, video and Speech Processing, (2001), pp. 271-274.

Huntley, J. M., (1989), Noise-immune phase unwrapping algorithm, Applied Optics, vol. 28, no. 15, pp. 3268-3270.

Huntley, J.M and Saldner, H., (1993), Temporal phase-unwrapping algorithm for automated interferogram analysis, Applied Optics, vol. 32, no. 17, pp. 3047-3052.

Huntley, J. M., (2001), Three-dimensional noise-immune phase unwrapping algorithm, Applied Optics, vol. 40, pp. 3901-3908.

Jayalakshmi, G. A., Sathiamoorthy, S. and Rajaram, R., (2001), A Hybrid genetic algorithm- a new approach to solve Travelling Salesman Problem," International Journal of Computational Engineering Science (Imperial College Press), vol. 2, pp. 339–355.

Jung, S. and Moon, B. R., (2002), Toward Minimal Restriction of Genetic Encoding and Crossovers for the Two-Dimensional Euclidean TSP, IEEE Transactions on Evolutionary Computation, vol. 6, pp. 557-564.

Kim, S.B. and Kim, Y.S., (2005), Least-squares phase unwrapping in wavelet domain, IEE Proceeding of Vision and image signal process, vol. 152, no. 3, pp. 261-267.

Lim, H., Xu, W. and Huang, X., (1995), Two new practical methods for phase unwrapping, Proceedings of the 1995 International Geoscience and Remote Sensing Symposium, Tokyo, Japan, August 18-21, IEEE, pp. 196-198.

Lin, Q., Vesecky, J. F. and Zebker, H. A., (1994), Phase unwrapping through fringe-line detection in synthetic aperture radar interferometry, Applied Optics, vol. 33, no. 2, pp. 201-208.

Mansour, N., Tabbara, H. and Dana, T., (2004), A genetic algorithm approach for regrouping service sites, Elsevier Computers & Operations research, vol. 31, pp. 1318-1333.

Marano, S., Palmieri, F. and Franceschetti, G., (2002), Discrete Greens methods and their application to two-dimensional phase unwrapping, Applied Optics, vol. 19, pp. 1319-1333.

Nagata, Y. and Kobayashi, S., (1999), An analysis of Edge Assembly Crossover for the Travelling Salesman Problem, in the Proceedings of IEEE International Conference on systems, man, and cybernetics (Institute of Electrical and Electronic Engineers), pp. 628-633.

Salfity, M., Ruiz, P., Huntley, J., Graves, M., Cusack, R. and Beauregard, D., (2006), Branch-cut surface placement for unwrapping of under-sampled three-dimensional phase data: application to magnetic resonance imaging arterial flow mapping, Applied Optics, vol. 45, pp. 2711-2721.

Schwartzkopf, W., Milner, T.E., Ghosh, J., Evans, B.L. and Bovik, A.C., (2002), Two-Dimensional Phase Unwrapping Using Neural Networks, 4th IEEE Southwest Symposium on Image Analysis and Interpretation, pp 274-7.

Schwarz, O., (2004), Hybrid phase unwrapping in laser speckle interferometry with overlapping windows, Shaker Verlag.

Slocumb, B. and kitchen, J., (1994), A polynomial phase parameter estimation phase unwrapping algorithm, IEEE Transaction Geoscience Remote Sensing, pp. 129-132.

Steeb, W.H., (2002), The Nonlinear Workbook, London: World Scientific Publishing.

Stramaglia, S., Guerriero, L., Pasquariello, G. and Veneziani, N., (1999), Mean-field annealing for phase unwrapping, Applied Optics, vol. 38, no. 8, pp. 1377-1383.

Tipper, D. J, Burton, D. R. and Lalor, M. J, (1996), A Neural Network Approach To The Phase Unwrapping Problem In Fringe Analysis, Nondestructive Testing and Evaluation, vol. 12, pp. 391-400.

Tsai, H., (2003), Heterogeneous Selection Genetic Algorithms for Travelling Salesman Problems, Taylor and Francis, Eng. Opt., vol. 35, pp. 297-311.

Tsai, H.K., Yang, J.M., Tsai, Y.F. and Kao, C.Y., (2004). An Evolutionary algorithm for Large Travelling salesman problem, IEEE Transactions on systems, man, and cybernetics-part B: Cybernetics, vol. 34, pp. 1718-1729.

Xu, W. and Cumming, I., (1996), A region growing algorithm for InSAR phase unwrapping, Proceedings of the 1996 International Geoscience and Remote Sensing Symposium, Lincoln, NE, May 27-31, IEEE, pp. 2044-2046.

Zebker, H. A. and Lu, Y., (1998), Phase unwrapping algorithms for radar interferometry residue-cut, least-squares, and synthesis algorithms, Applied Optics, vol.15, no. 3, pp. 586-598.

# Appendix

## Published Papers

S.Karout, M.Gdeisat, D.Burton and M.Lalor, "Two-dimensional phase unwrapping using a hybrid genetic algorithm", Applied Optics, Vol. 46, No. 5 (10 February 2007).

S.Karout, M.Gdeisat, D.Burton and M.Lalor, "Residue Vector a New Approach to Branch Cut Placement in Phase Unwrapping - the Theoretical Study", Applied Optics, Vol. 46, No. 21 (20 July 2007).

S.Karout, M.Gdeisat, D.Burton and M.Lalor, "Hybrid genetic algorithm using a parametric method to solve the two-dimensional phase unwrapping problem", Photon06 conference (September 2006).