Multilevel two-dimensional phase unwrapping

Iddit Shalem
Multilevel two-dimensional phase

unwrapping

Research Thesis

Submitted in Partial Fulfillment
of The Requirements for the Degree of Master of Science
in Computer Science.

Iddit Shalem

Submitted to the Senate of the Technion - Israel Institute of Technology

Adar, 5767 Haifa

March 2007
The Research Thesis Was Done Under the Supervision of Prof. Irad Yavneh in the Department of Computer Science

I would like to thanks my thesis supervisor, Professor Irad Yavneh, for being available whenever needed, for all the helpful ideas and for critically reading this thesis so many times.

I would also like to thanks my family. My parents for willing to watch over my kids anytime I asked. My father for being interested in this work, reading it and willing to share thoughts about it. I also thank my husband for the moral support and for pushing me into research. I couldn’t have done it without you all.

The Generous Financial Help of the Technion is Gratefully Acknowledged.
## Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Abstract</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>List of Symbols</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>1.1</td>
<td>Synthetic Aperture Radar</td>
<td>3</td>
</tr>
<tr>
<td>1.2</td>
<td>Phase unwrapping</td>
<td>5</td>
</tr>
<tr>
<td>1.3</td>
<td>Minimum norm methods</td>
<td>8</td>
</tr>
<tr>
<td>1.4</td>
<td>Path Following Methods</td>
<td>9</td>
</tr>
<tr>
<td>1.4.1</td>
<td>Residue Cut Tree Algorithms</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>Phase unwrapping using segmentation of the wrapped image</td>
<td>17</td>
</tr>
<tr>
<td>2.1</td>
<td>Motivation</td>
<td>17</td>
</tr>
<tr>
<td>2.2</td>
<td>Image Segmentation</td>
<td>18</td>
</tr>
<tr>
<td>2.3</td>
<td>Multi-level unwrapping algorithm</td>
<td>20</td>
</tr>
<tr>
<td>2.3.1</td>
<td>Overview</td>
<td>20</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Data structure</td>
<td>22</td>
</tr>
<tr>
<td>2.3.3</td>
<td>Relaxation</td>
<td>26</td>
</tr>
<tr>
<td>2.3.4</td>
<td>Prolongating coarse level wrap count vector</td>
<td>27</td>
</tr>
<tr>
<td>2.3.5</td>
<td>Complete Multilevel unwrapping cycle scheme</td>
<td>28</td>
</tr>
<tr>
<td>2.4</td>
<td>Computational Complexity</td>
<td>28</td>
</tr>
<tr>
<td>2.5</td>
<td>Experimental results</td>
<td>31</td>
</tr>
<tr>
<td>3</td>
<td>A Multilevel algorithm for the residue cut approach</td>
<td>33</td>
</tr>
<tr>
<td>3.1</td>
<td>An overview of the algorithm</td>
<td>33</td>
</tr>
<tr>
<td>3.2</td>
<td>Data structure</td>
<td>34</td>
</tr>
<tr>
<td>3.3</td>
<td>Coarsening Phase</td>
<td>36</td>
</tr>
<tr>
<td>3.4</td>
<td>Prolongation Phase</td>
<td>41</td>
</tr>
<tr>
<td>3.5</td>
<td>Computational Complexity</td>
<td>43</td>
</tr>
<tr>
<td>3.6</td>
<td>Experimental results</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>Bibliography</td>
<td>51</td>
</tr>
</tbody>
</table>
## Figures

### Figure

1.1 Basic imaging geometry for SAR .......................... 4  
1.2 Example of wrapped and unwrapped phase values ........ 6  
1.3 One-dimensional phase unwrapping example ............. 7  
1.4 Wrapped phase graph example ........................... 10  
1.5 Cycle distance ........................................ 13  
1.6 Dual graph ............................................ 14  

2.1 Sheared planes .......................................... 19  
2.2 Hierarchical Segmentation ............................... 21  
2.3 An example of a segmentation level ..................... 24  
2.4 Multi-level segmentation-based algorithm applied to sheared planes example .......................... 32  
2.5 Multi-level segmentation-based algorithm applied to a SAR phase example .......................... 32  

3.1 Dipole matching example ................................ 35  
3.2 Operations for $c$ vertices ............................... 38  
3.3 Graph hierarchy structure for an example 4-level hierarchy ........ 40  
3.4 Coarse level edges construction ......................... 42  
3.5 Edges and requests prolongation ......................... 44  
3.6 Sine-shaped ramp wrapped phase input .................. 46  
3.7 Multi-level residue cut algorithm applied to sine-shaped ramp example .................................. 47  
3.8 Multi-level residue cut algorithm applied to a synthetic phase example .......................... 49  
3.9 Multi-level residue cut algorithm applied to a SAR phase example .......................... 50
Abstract

In this work we study the problem of two-dimensional phase unwrapping and propose two algorithms for its solution. Two-dimensional phase unwrapping is the problem of deducing unambiguous “phase” from values known only modulo $2\pi$. Many authors agree that the objective of phase unwrapping should be to find a weighted minimum of the number of places where adjacent (discrete) phase values differ by more than $\pi$ (called discontinuities). This NP-hard problem is of considerable practical interest, largely due to its importance in interpreting data acquired with synthetic aperture radar (SAR) interferometry. Consequently, many heuristic algorithms have been proposed. Our first algorithm considers the wrapped phase array as a grey level image and applies the image segmentation problem to this image. Based on the segmentation, we develop an efficient relaxation method for decreasing discontinuities in the phase image. The second algorithm presents an efficient multi-level graph algorithm for the approximate solution of an equivalent problem—minimal residue cut in the dual graph.
## List of Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAR</td>
<td>synthetic aperture radar</td>
</tr>
<tr>
<td>φ</td>
<td>true phase array</td>
</tr>
<tr>
<td>ψ</td>
<td>wrapped phase array</td>
</tr>
<tr>
<td>W</td>
<td>wrapping operator</td>
</tr>
<tr>
<td>$D_{i,j}^x, D_{i,j}^y$</td>
<td>discrete derivatives of the approximated phase array</td>
</tr>
<tr>
<td>$\triangle_{i,j}^x, \triangle_{i,j}^y$</td>
<td>Wrapped derivatives of the wrapped array $\psi$</td>
</tr>
<tr>
<td>$c$</td>
<td>wrap count array</td>
</tr>
<tr>
<td>$A_l^{t+1}$</td>
<td>segmentation aggregation matrix</td>
</tr>
<tr>
<td>$S_l$</td>
<td>number of segments in level $l$ of segmentation hierarchy</td>
</tr>
<tr>
<td>$\hat{\phi}$</td>
<td>approximate phase array solution</td>
</tr>
<tr>
<td>WC</td>
<td>wrap count vector</td>
</tr>
<tr>
<td>$C$</td>
<td>subset of coarse graph vertices</td>
</tr>
<tr>
<td>$F$</td>
<td>subset of fine graph vertices</td>
</tr>
<tr>
<td>$D_l$</td>
<td>length scale of level $l$</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

This chapter introduces the phase unwrapping problem and common approaches for solving it. The main source for this chapter is [7] which studies this problem in detail. Phase unwrapping is a key step in many applications such as magnetic resonance imaging, optical interferometers and synthetic aperture radar interferometry. In all these applications a two-dimensional array of phase values relating to some physical quantity is obtained. Phase values are calculated by some inverse trigonometric function (e.g. arctangent) providing only principal values limited to the interval \([-\pi, \pi]\). The true phase needs to be “unwrapped” before further processing of the physical quantity measured, as described below. We will use the synthetic aperture radar (SAR) interferometry as an example to introduce the phase unwrapping problem. See [1] for an elementary description.

1.1 Synthetic Aperture Radar

SAR is a valuable technique, providing terrain imaging, with applications in mapping, reconnaissance and remote sensing. SAR provides its own illumination of a single frequency microwave radiation; therefore there is no color associated with the raw data. However, it provides two major benefits from its not being
dependent on natural light: the ability to image at all weather conditions, e.g., through clouds and at night. Another benefit is that the signal phase, not only amplitude, is retained in the radar echo. Generally, the echo amplitude supplies physical information of the target, such as electrical properties and moisture content. The terrain height topography is extracted from the phase values.

SAR works by illuminating the Earth with coherent microwave radiation. This means that the transmitted signals share the same wavelength, amplitude and phase. An antenna mounted on an aircraft or spacecraft transmits a radar signal in a side-looking direction towards the earth surface. The signal is reflected from the surface and received a fraction of a second later by the same antenna. The amplitude and phase of this echo is measured and recorded and the data are then used to construct a complex SAR image. The SAR interferometry exploits coherence using two SAR images of the same surface. The second SAR image must be provided from a different receiver position or at a different time. The best known application of SAR interferometry uses two spatially separated radar
antennas, supplying two different look angle images of the same terrain. Figure 1.1 shows such an application: two antennas A1 and A2 simultaneously viewing the same surface. The distance between A1 and the point on the ground being imaged is the slant range $\rho$, while $\rho + \delta\rho$ is the distance between A2 and the same point. The topography of the image can be constructed from the path difference at each pixel point. A pixel path difference is the difference between the distances from the target point to the antennas, $\delta\rho$ in the figure. The path difference is proportional to phase difference. This phase difference is computed by multiplying one image by the complex conjugate of the other image. The result is an interferogram whose phase at each pixel is proportional to the path difference. The measured phase values of the interferogram can only take values between $-\pi$ and $\pi$, that is, the phase is ‘wrapped’. The phase of the interferogram, displayed as an image, contains fringes that trace the topography. An example is shown in Figure 1.2. These images were taken from the web site http://earth.esa.int/workshops/fringe_1996/fornaro/. The discontinuities in the interferogram are caused by the wrapping. In order to compute the absolute phase difference, which is needed for extracting the topography, the $2\pi$ ambiguity inherent in the phase measurements must be solved using techniques of phase unwrapping.

1.2 Phase unwrapping

Phase wrapping is a process whereby a real-valued surface, $\phi(s)$, is wrapped into the interval $[-\pi, \pi]$ by the following operation:

$$W\{\phi(s)\} = \psi(s) = \phi(s) + 2\pi k(s), \quad (1.1)$$

where $k(s)$ is an integer function that forces $-\pi \leq \psi < \pi$. For the two-dimensional case, $s = (x, y)$. We assume that the measurements are sampled on a rectangular equidistant grid, as is typical in practical applications. Accordingly we denote
\[ \psi_{i,j} = \psi(x_i, y_j) \text{ and } \phi_{i,j} = \phi(x_i, y_j), \text{ with} \]
\[ i = 0, 1, 2, \ldots M, \quad j = 0, 1, 2, \ldots N. \]  
\text{(1.2)}

Unwrapping is the process of estimating the actual function, \( \phi \), from the wrapped function, \( \psi \). We say that a candidate reconstruction, \( \phi \), is feasible if \( W(\phi) = \psi \). Constraints must be placed on the choice of \( \phi \), since any two surfaces that differ point-wise by some integer multiple of \( 2\pi \) are feasible solutions. The most common approach is to seek the “smoothest” possible unwrapped estimation. According to this smoothness criterion, unwrapping algorithms try to minimize some norm of the discontinuities in the reconstructed surface. A discontinuity is defined by a pair of neighboring pixels whose difference exceeds \( \pi \) in magnitude. Often, a user defined weighting is associated with each such pairing, to reflect the confidence in the smoothness assumption at the corresponding locations.

For simplicity, consider first the trivial one-dimensional case (\( s = x \)). Let us assume, for example, that the actual phase function is \( \phi(x) = 5\pi x \). We are given only the wrapped phase function \( \psi(x) \) produced by the wrapping operator
Figure 1.3 depicts the true and wrapped phase by solid and dashed lines, respectively. Visually, it is easy to determine that the true phase can be correctly reconstructed by: adding nothing to the phase in the interval $0 \leq x < 0.2$, adding $1 \times 2\pi$ to the phase in the interval $0.2 \leq x < 0.6$ and adding $2 \times 2\pi$ to the phase in the interval $0.6 \leq x < 1$.

This “visual determination” should be replaced with a mathematical procedure. Many phase unwrapping algorithms have been developed during the last twenty years. These may be divided into two general classes: minimum-norm and path-following methods. The question of which of the two approaches is better is yet unsettled, partially because of the constant improvement of algorithms for both approaches. Both algorithms proposed in this work use the path-following approach. Next we describe both approaches giving more details of the path-following approach, as this is the approach we are pursuing.
1.3 Minimum norm methods

The minimum-norm approach is to seek the unwrapped phase that minimizes some norm of the distance between the discrete derivatives of the approximated surface $\phi$, and the wrapped derivatives of the wrapped surface $\psi$. The discrete derivatives of the approximated surface are defined by

$$D^x_{i,j} = \phi_{i+1,j} - \phi_{i,j}, \quad i = 0,1,2,...M-1, j = 0, 1, 2,...N.$$

$$D^y_{i,j} = \phi_{i,j+1} - \phi_{i,j}, \quad i = 0,1,2,...M, j = 0, 1, 2,...N-1.$$

Wrapped derivatives of the wrapped surface $\psi$ are defined by

$$\Delta^x_{i,j} = W\{\psi_{i+1,j} - \psi_{i,j}\}, \quad i = 0,1,2,...M-1, j = 0, 1, 2,...N.$$

$$\Delta^y_{i,j} = W\{\psi_{i,j+1} - \psi_{i,j}\}, \quad i = 0,1,2,...M, j = 0, 1, 2,...N-1.$$

The minimum $L^p$-norm solution $\phi$ minimizes

$$J = \sum_{i=0}^{M-1} \sum_{j=0}^{N} w^x_{i,j}|D^x_{i,j} - \Delta^x_{i,j}|^p$$

$$+ \sum_{i=0}^{M} \sum_{j=0}^{N-1} w^y_{i,j}|D^y_{i,j} - \Delta^y_{i,j}|^p,$$

where $w^x_{i,j}$ is a user-defined weight assigned to the horizontal adjacency between pixels $(i, j)$ and pixels $(i+1, j)$, and $w^y_{i,j}$ is similarly defined for vertical adjacency. Such formulation of the phase unwrapping problem enables us to adopt minimum norm techniques for constructing the solution, as in [5, 8, 11]. Often, minimum $L^p$-norm algorithms are used with $0 < p \leq 2$. The problem generally becomes
more difficult as \( p \) is decreased, but many authors agree that the reconstructions generally improve as \( p \) tends to zero [3, 4, 5, 7, 9, 10].

1.4 Path Following Methods

One major difference between path-following and minimum-norm algorithms is that in the path following approach only feasible solutions are produced. That is, only an integral number of cycles is added to the measurements in the unwrapping process. When considering only feasible solutions, phase unwrapping is essentially the process of assigning an integer multiple \( c_{i,j} \) of \( 2\pi \) to each pixel of the wrapped phase data \( \phi_{i,j} \) ([6]). The unwrapped solution \( \phi_{i,j} \) satisfies the relation

\[
\phi_{i,j} = \psi_{i,j} + 2\pi c_{i,j}
\]

(1.8)

The integer multiples are called wrap counts ([6]), and they are chosen so as to minimize discontinuities in the solution in some sense.

Consider again the case of one dimensional phase unwrapping. Finding wrap counts for an optimal solution in this case can be done by the following procedure: initialize \( c_0 = 0 \), then scan the pixels in order, and set \( c_i \) according to the difference in \( \psi \) between the current and previous pixel:

\[
c_i = \begin{cases} 
  c_{i-1} - 1, & \psi_i - \psi_{i-1} \geq \pi \\
  c_{i-1} + 1, & \psi_i - \psi_{i-1} \leq -\pi \\
  c_{i-1}, & \text{otherwise}
\end{cases}
\]

(1.9)

In this method a pixel wrap count is determined by integrating discontinuities in the wrapped measurements. Note that each pixel’s wrap count is chosen so that, in the constructed solution, the difference between values of adjacent pixels will not exceed \( \pi \)—there are no discontinuities in the constructed solution, therefore it is an optimal solution.
Can this algorithm be extended to the two-dimensional case? Before answering this question, let us use a graph representation of the wrapped phase, with each pixel \((i, j)\) of the wrapped phase designated as a vertex \(v_{i,j}\). Vertex \(v_{i,j}\) is marked with the wrapped measurement value, \(\psi_{i,j}\), normalized by \(2\pi\) for clarity. Each vertex is connected by edges to its four nearest neighbors. These edges are assigned with the user-defined weights. We represent the existence of a discontinuity by a directed edge connecting two neighboring pixels. The direction of the edge is from the lower value pixel to the higher value pixel. We refer to such a graph as the problem’s **primal graph**. An example of a primal graph (with the user-defined edge weights omitted for clarity) is shown in Figure 1.4.

A natural extension of the algorithm of integrating discontinuities to the two-dimensional case is described for the primal graph: The unwrapping of pixel \((i, j)\) is determined according to some reference pixel—for example we use reference pixel \((0, 0)\) (located at the top-left corner of the image)—by choosing some path from vertex \((0, 0)\), to vertex \((i, j)\). \(c_{0,0}\) is arbitrarily initialized to zero. Then the vertices are considered in order following the path. The wrap count associated with each vertex \(v\) in turn is calculated according to the last step in the path, as
in (1.9)—extended to the two-dimensional case and using the edge direction as a discontinuity indicator:

\[
c_v = \begin{cases} 
  c_u - 1, & e \text{ directed from } u \text{ to } v, \\
  c_u + 1, & e \text{ directed from } v \text{ to } u, \\
  c_u, & e \text{ not directed},
\end{cases} \tag{1.10}
\]

where \( u \) precedes \( v \) in the path and \( e \) is the edge connecting \( u \) to \( v \).

Can we use this algorithm to determine a pixel wrap count by integrating discontinuities over any path from the reference pixel? The problem is illustrated in the wrapped phase example of Figure 1.4; different paths produce different results. Integrating the discontinuities from pixel \( \psi_{0,0} \) (top left most vertex) to pixel \( \psi_{2,2} \) can result in two different wrap count values for \( c_{2,2} \). If we use the path which travels two vertical edges downwards and then two horizontal edges to the right, the wrap count \( c_{2,2} \) will be zero, as there are no directed edges in the path. If we use the path which travels first two horizontal edges to the right and then two vertical edges downwards, the wrap count \( c_{2,2} \) will be +1. There is one directed edge in this path, and it is traversed against the edge direction.

### 1.4.1 Residue Cut Tree Algorithms

The concept of many path following algorithms, called residue cut tree algorithms, is to eliminate a set of edges from the graph. The edges are chosen so that the unwrapping problem becomes path independent for the resulting graph. That is, determining the wrap count of any pixel from any reference pixel, yields the same result for all paths connecting the two pixels that contain no eliminated edges. After such a set is eliminated, the solution is constructed by the flood fill algorithm for the remaining graph. The flood fill algorithm implements the path discontinuity integration algorithm, efficiently modified for
the case of a path independent graph. We start from an arbitrary vertex whose wrap count is set to zero. Remaining pixels are considered when they have a neighbor that is already unwrapped. They are then unwrapped according to the rule in (1.10), where \( u \) is an already unwrapped neighboring pixel. This can be achieved, for example, by scanning the connected component pixels by a breadth-first search (BFS) order. After applying the flood fill algorithm, discontinuities may occur only at edges belonging to the eliminated set. Therefore, residue cut algorithms strive not only to find an appropriate set (that leaves the graph path independent), but to find the one with the smallest weight. Minimizing the total weight of the eliminated set of edges essentially minimizes the weighted number of discontinuities in the unwrapped solution, hence the residue cut problem coincides with the \( L^p \) minimum norm problem with \( p \to 0 \).

We need to examine the conditions for a graph to be path independent. Before continuing, we introduce the following definitions relating to the primal graph:

- A cycle **integral** is defined for closed paths (loops) in the graph. The cycle integral is the counterclockwise integration of discontinuities according to rule (1.10).

- A cycle in the graph is called **balanced** if its integral is zero.

- A **face-cycle** is the cycled path around a face of the primal graph. Note that the primal graph is planar so this can be defined.

Obviously, a graph is path independent if and only if all cycles in the graph are balanced. This condition is not very valuable, since it is impractical to test all loops of a graph. We are looking for a condition that can be tested easily. Note that the integral of any loop in the primal graph is equal to the sum of
The integrals of all face-cycles inside the loop. This is illustrated in Figure 1.5. The value marked at each face in the graph is the integral of its face-cycle. The integral of the cycle emphasized by thick edges is zero, and it is equal to the sum of the integrals of the three faces inside it. The graph would be path independent if all the internal edges of the emphasized loop would be removed from the graph. In fact, if we choose any cycle encircling the two unbalanced faces, and eliminate the edges inside this cycle, the resulting graph will be path independent. More generally, an appropriate set of edges can be found by constructing cycles in the primal graph, covering all unbalanced faces, with each cycle being balanced. The eliminated set of edges in this case includes all edges that are internal to any one of the cycles. As mentioned earlier, the aim is to choose a set whose total weight is as small as possible. The residue-cut approach described next is based on the above conclusion. It solves an equivalent problem, introduced in [10] as a tree problem.

**The residue cut approach**  
Residue cut is defined on a dual graph constructed from the primal graph, by designating a vertex per each face of the primal graph. Each dual graph vertex is assigned with the integral of the corre-
sponding face-cycle. For the wrapped phase, resulting values are either +1, -1 or 0. Non-zero results are called residues or charges. Vertices in the dual graph are connected by an edge if the corresponding faces in the primal graph are adjacent. The weights of the edges in the dual graph are given by the weights of the corresponding edges in the primal graph. Figure 1.6 shows the dual graph of the primal graph of Figure 1.5. Note that the total sum of charges in the dual graph is zero. This will be the case whenever a dual graph is constructed from a wrapped phase image, since every discontinuity edge is counted twice—once for each face bordering on it. The equivalent problem is to find a minimum-weight subset of dual graph edges that covers all charged nodes such that each connected component is neutral. That is, the sum of the charges in each connected component needs to be zero. Note the equivalence: there is a 1 to 1 correlation between edges forming a connected component in the dual graph and edges within a loop in the primal graph. Evidently, the graph induced by the optimal set of edges is a forest comprised of one or more neutral trees. Residue cut algorithms aim at constructing neutral trees such that each charged vertex is included in one of the trees, and the sum of weights of edges totaled over all trees is as small as possible. This problem has been shown to be NP-hard ([4]). Residue cut tree algorithms are therefore heuristic or approximate algorithms.

Figure 1.6: The dual graph constructed for the wrapped phase of Figure 1.5.
The “classic” residue cut tree algorithm was proposed by Goldstein et al. ([10]). The algorithm considers in turn each charge that has not yet been connected. The beginning of a tree is formed by connecting it to the nearest charge, regardless of sign. If the tree is not neutral, the next nearest charge is connected. This process continues until the tree becomes neutral. Goldstein’s algorithm is efficient, and it often produces satisfactory results. Its weakness is partly due to the fact that it does not incorporate weighting of edges, resulting in unsatisfactory results when these should be used.

Several algorithms use a dipole matching strategy in which the residue cut solution is constructed by matching pairs of negative and positive charges. In the initial matching algorithm proposed by Huntley ([12]) each unpaired charge is taken in turn and then paired to the nearest unpaired charge of the opposite sign. In [3] a minimum cost matching solution is constructed by use of the Hungarian algorithm. The Hungarian algorithm is often discussed in term of matching N workers to N jobs. The cost of matching worker i to job j is \( w_{ij} \). The algorithm finds a matching that minimizes this cost. The Hungarian algorithm is applied for the residue cut problem by representing positive and negative charges as workers and jobs, respectively. The matching cost \( w_{ij} \) are taken as the distances between the charges. The minimum cost matching algorithm finds a minimum solution for the residue cut problem only in the case that trees are restricted to connect only a pair of dipole charges. In Chapter 3 we demonstrate how such restrictions may produce unsatisfactory results.

In [4] an algorithm is developed which is an improvement to the classic residue cut algorithm. This algorithm constructs the residue cut solution as a single tree that connects all the charges. Restricting the residue cut solution to a single tree increases the total forest weight, but the total weight of discontinuities in the unwrapped phase is not necessarily increased, since not all tree edges nec-
essarily contain discontinuities. The algorithm begins from an arbitrary charge as the beginning of the tree. The nearest charge (using Dijkstra’s algorithm) is then connected to the tree, and the weights of edges on the path leading to it are zeroed. Nearest charges are successively connected to the tree until all charges are connected.

Flynn’s algorithm ([6]) finds the minimum $L_1$-norm over all feasible unwrapped phases. This algorithm does not solve the residue cut problem, it uses a path-following approach by expressly identifying and accommodating discontinuities during the reconstruction process. The algorithm makes successive modifications to the wrapped phase. Each modification partitions the phase image into two connected sets of pixels, and raises the unwrapped phase by $2\pi$ in one of the sets, reducing the weighted discontinuities in the phase.

We present in this work two algorithms for phase unwrapping. The first algorithm, described in Chapter 2, applies image segmentation to the grey level image of the wrapped phase. Image segmentation is an important problem in image processing, thus much effort and resources are invested for developing algorithms for this problem. We suggest how to use image segmentation algorithms in order to solve the phase unwrapping problem. The second algorithm, described in Chapter 3, tackles the residue cut problem. It allows weighted edges, and aims to efficiently solve this problem by a multilevel approach.
Chapter 2

Phase unwrapping using segmentation of the wrapped image

2.1 Motivation

In some two dimensional wrapped phase examples, a proper unwrapping can easily be deduced just by looking at the grey level image of the phase array. Figure 2.1 demonstrates such an example. Figure 2.1(a) shows a three dimensional view of two planar surfaces that have been tilted relative to one another. The wrapped phase is shown in Figure 2.1(b). The grey level images of the true and wrapped phase are shown in Figures 2.1(c) and 2.1(d), respectively. In the grey level image, dark pixels indicate low phase values while bright pixels indicate high phase values. The grey level image of the wrapped phase may be segmented into the regions shown in Figure 2.1(e) marked by $A_1, \ldots, L$. An unwrapped solution can then be constructed by lifting or dropping whole regions by an integer number of cycles so as to locally minimize the discontinuities around each region's boundary. For example, consider region $A$ which has two neighboring regions, $B$ and $G$. Dropping region $A$ by one cycle will cancel all discontinuities along the boundary between regions $A$ and $B$. Note that no lifting or dropping of region $A$ can cancel all the discontinuities along the boundary between $A$ and $G$. Dropping region $A$ by one
cycle is, locally, the best position for region $A$, since most discontinuities along its borders are cancelled. After unwrapping $A$ we can continue the unwrapping process by moving to other regions.

The approach of our first algorithm is to utilize a hierarchical image segmentation algorithms to partition the grey level image of the input phase array into progressively larger regions. We then construct an unwrapping method which determines the locally optimal position of each region. The main advantage of this approach is that we make use of the efforts and resources that have been invested in the image segmentation problem, which is a main step in many image processing applications.

We thank Achi Brandt for suggesting the idea in the base of this algorithm.

### 2.2 Image Segmentation

Image segmentation is the process of partitioning images into constituent parts, called regions or segments, whose pixels have similar properties, e.g., grey level and textural properties. Image segmentation is a key step in many approaches to data compression and image analysis. Natural images contain objects at different scales, sometimes with a recursive structure, that is, can be segmented into smaller objects. Therefore, advanced image segmentation algorithms may provide a hierarchical decomposition of the image into segments of varying sizes. Sharon, Brandt and Basri ([13]) developed a multi-scale segmentation algorithm that provides such a hierarchical segmentation. Our phase unwrapping algorithm uses a segmentation algorithm based on this approach. The resulting hierarchy contains several segmentation levels: in the finest level each pixel is a separate segment. In successively coarser levels, segments are formed by aggregation of several segments of the finer level. An example of such a segmentation hierarchy
Figure 2.1: (a) Three dimensional view of the sheared planes true phase, $\phi$. (b) Three dimensional view of the wrapped phase, $\psi$. (c) Grey level image of the true phase, $\phi$. (d) Grey level image of the wrapped phase $\psi$. (e) Segmentation of the wrapped phase image.
is shown in figure 2.2.

In our tests we employ a segmentation algorithm based on the fast multi-scale image segmentation algorithm from [13]. We represent the hierarchical segmentation obtained by \( M \) binary aggregation matrices: \( A_0^1, A_1^2, \ldots, A_{M-1}^M \). These matrices describe a segmentation hierarchy containing \( M+1 \) levels, with 0 being the finest level index and \( M \) being coarsest level index. In level 0, each segment is a single pixel. Matrix \( A_l^{l+1} \) indicates how the segments of level \( l \) were aggregated to form the segments of level \( l + 1 \). The size of the matrix is \( S_l \times S_{l+1} \) where \( S_l \) and \( S_{l+1} \) are the number of segments at hierarchy levels \( l \) and \( l+1 \), respectively. Each row, \( i \), of the matrix has only one entry, \( j \), set to 1, indicating that segment \( i \) of level \( l \) corresponds to segment \( j \) of level \( l + 1 \). Note that a higher level in the hierarchy includes less segments than the lower one, thus \( S_{l+1} < S_l \).

### 2.3 Multi-level unwrapping algorithm

#### 2.3.1 Overview

The first step in our unwrapping process is to obtain a multi-scale segmentation of the grey level image of the input phase array, using a multilevel segmentation algorithm. Let us denote by \( \hat{\phi} \) the approximate solution constructed by our algorithm. Initially \( \hat{\phi} \) is the given phase array. We then constantly improve the approximate solution, \( \hat{\phi} \), by a relaxation method performed for each level in the segmentation hierarchy. This relaxation method considers each segment of the hierarchy level in turn, and adds an integer multiple of \( 2\pi \) to the phase values of all it’s points. The number of \( 2\pi \) cycles added to each segment is chosen so as to locally minimize the \( L^0 \)-cost of \( \hat{\phi} \), i.e., the weighted number of discontinuities in the approximated phase array.

In order to smooth the image at all scales, relaxations are applied to each
Figure 2.2: An example of hierarchical segmentation applied to the sheared planes wrapped image. (a) shows the finest segmentation level in which each segment is a single pixel. Coarser segmentation levels are shown in (b)...(g). Each segment in a coarser level is an aggregation of one or more finer level segments.
level, using a multilevel unwrapping cycle which consists of two phases. In the first phase we scan the levels of the segmentation hierarchy from fine (where each segment is a single pixel) to coarse, and construct some data structure per each level, used later by the relaxation procedure. The actual relaxation of each level is executed from coarse to fine in the second phase, using the constructed data structures hierarchy.

Our algorithm ensures that any change we make decreases the $L^0$-norm of the approximated phase array. Furthermore, the input phase array of our algorithm is not restricted to wrapped phase values. Thus, our algorithm can be used to improve results of any other unwrapping algorithm. We use our algorithm iteratively, with each iteration including segmentation of the current grey level image of the phase array and a complete multi-scale smoothing cycle. This iteration terminates when no improvement to the $L^0$-norm is obtained.

2.3.2 Data structure

Given $M+1$ segmentation levels, the multilevel unwrapping cycle constructs a hierarchy of directed graphs $\{G_l\}_{l=0}^M$. The segments in level $l$ are the vertices of the graph $G_l$, and each pair of adjacent segments are connected by an edge in $G_l$. Graph edges are assigned with data which allows to determine the locally optimal number of cycles to lift or drop each segment, by analyzing all the edges outgoing from the segment. Each segment/vertex is assigned with its wrap count value—the integer number of $2\pi$ cycles added by the algorithm to all the segment pixels. In the algorithm description we will use a vector denoted $WC_l$, containing an entry per each vertex of level $l$, to store the vertices’ wrap count values.
2.3.2.1 Edge histogram

The data assigned to each edge is a histogram. Generally, histograms are used to describe the partitioning of objects into several bins according to some common property. Each bin is associated with two values: the property value common to all its members, which we call the bin’s tag, and the total weight of its members.

The objects partitioned by an edge histogram in our algorithm are ordered pairs of neighboring pixels in the phase array. The property we associate with each such pair of pixels is an integer value called discrepancy. The discrepancy of an ordered pair of pixels with phase values \((\psi_1, \psi_2)\) is the unique number of wrap count cycles needed to be added to the value of \(\psi_1\) in order to avoid a discontinuity between the two pixels. For example, if \(\psi_1 = \pi/2\) and \(\psi_2 = 4\pi\), then the discrepancy for \((\psi_1, \psi_2)\) is 2 since there is no discontinuity between \(\psi_1 + 2 \times (2\pi)\) and \(\psi_2\).

A histogram assigned to an edge connecting two segments \((s_1, s_2)\) lists pairs of pixels \((\psi_1, \psi_2)\), such that \(\psi_1 \in s_1\) and \(\psi_2 \in s_2\), i.e., pairs forming the border between the two segments. These pixel pairs are partitioned into bins according to their discrepancy. The weight of each such pair of pixels is taken to be the user-defined weight. To illustrate this, consider Figure 2.3. This figure shows an example of some segmentation of a phase array (phase values are normalized by \(2\pi\) for clarity). The pixels are grouped into the four segments, \(s_1, s_2, s_3\) and \(s_4\). Therefore, the graph constructed by our algorithm for this segmentation level will have four vertices \(s_1, s_2, s_3\) and \(s_4\). Segment \(s_1\) has two adjacent segments, \(s_2\) and \(s_3\). Thus, vertex \(s_1\) has two outgoing edges \((s_1, s_2)\) and \((s_1, s_3)\). The histogram of edge \((s_1, s_2)\) includes two ordered pairs of pixel values \((0.75, 0.15)\) and \((0.55, 0.15)\). The discrepancies of these pairs are \(-1\) and \(0\), respectively. Assuming that the
user-defined weights of adjacent pixel pairs are equal, the histogram contains two bins with discrepancy tags $-1$ and 0, and both bins’ weights are 1. The histogram of edge $(s_1, s_3)$ includes three pixel pairs with phase values $(0.55, 1.00), (0.55, 1.25)$ and $(0.65, 1.30)$. The discrepancies of these pairs are 0, 1, 1, respectively. The histogram bins’ jump count tags are 0 and 1, with weights 1 and 2, respectively.

![Figure 2.3: An example of a segmentation level.](image)

### 2.3.2.2 Initializing finest level variables

The finest level graph is constructed by designating a vertex for each pixel, with an edge outgoing from each vertex to its four nearest neighbors. The histogram assigned to each such edge lists only one pair of pixels, $(\psi_1, \psi_2)$, and thus contains only one bin. The bin tag is the wrap count of $(\psi_1, \psi_2)$, and the bin weight is the user defined weight of that pixel pair. The wrap count value of each segment is initialized to zero.

The next coarser level graphs are constructed by recursive coarsening steps.
2.3.2.3 Constructing a coarser level graph

In a coarsening step we construct the graph of level \( l + 1 \) from the data of level \( l \) using the segmentation aggregation matrix \( A_{l+1} \). We designate a vertex in graph \( G_{l+1} \) per each segment of level \( l + 1 \). The wrap count value of each segment is initialized to zero. An edge between two different coarse level vertices \( s_{l+1}^1 \) and \( s_{l+1}^2 \) is constructed by merging histograms of level \( l \) edges outgoing from vertices that were aggregated into \( s_{l+1}^1 \) by the segmentation, and incoming to vertices that were aggregated into \( s_{l+1}^2 \). When merging input histograms \( H_1, H_2, ... H_n \) into \( H \), there is one bin for each discrepancy value appearing in any one of the input histograms, and the weight of each bin in the merged histogram \( H \) is the sum of the weights of the bins associated with the same discrepancy in the input histograms.

Assume, for example, that the segments of Figure 2.3 describe a segmentation of hierarchy level \( l \). Suppose that these segments are aggregated for the next coarser segmentation level to form two segments such that one segment, \( s_{l+1}^1 \), aggregates segments \( s_1 \) and \( s_2 \), and the other segment, \( s_{l+1}^2 \), aggregates segments \( s_3 \) and \( s_4 \). The edge \((s_{l+1}^1, s_{l+1}^2)\) histogram will be constructed by merging the histograms of the \( l \)-level edges \((s_1, s_3)\) (containing two bins, tagged 0,1, and weighted 1,2, respectively), \((s_2, s_3)\) (containing one bin tagged 1 and weighted 2) and \((s_2, s_4)\) (containing one bin tagged 1 and weighted 1). The resulting histogram (containing two bins, tagged 0,1, and weighted 1 and 5, respectively) describes the partitioning by discrepancy of the pairs of pixels forming the border between the coarse segments, as required.
2.3.3 Relaxation

A relaxation procedure is executed for each segmentation level, aimed at smoothing the phase image on that level. Consider a single segment in some level of the segmentation hierarchy. Applying a wrap count change of some integer $i$ to all the pixels of the segment can be visually thought of as lifting (or dropping) $\hat{\phi}$ in the entire segment by $i$ cycles of $2\pi$. In a relaxation sweep we go over all the segments, one by one in any order, and determine (Gauss-Seidel style) the locally $L^0$-norm optimal number of cycles to lift or drop each segment. That is, if we were able to only lift the specific segment by some integer number of cycles, what would be our best choice so as to minimize the weighted number of discontinuities in the phase array. No change (no lifting or dropping) is applied unless the $L^0$-norm cost is reduced. Relaxation sweeps are applied until stabilization, i.e., no improvement in the $L^0$-norm cost is obtained.

When lifting a whole segment, the only pairs of neighboring pixels that affect the $L^0$-norm cost are the pairs on the segment border, such that only one member of the pair belongs to the segment. We refer to these pairs as the segment’s border pairs. For example, in Figure 2.3, the border pairs of $s_1$ are $(0.75, 0.15), (0.55, 0.15), (0.55, 1.00), (0.55, 1.25), (0.65, 1.30)$. Lifting segment $s_1$ may cause or cancel discontinuities only among these pairs. To determine the optimal position for some segment, we construct a histogram separating all the segment’s border pairs into bins according to their discrepancies. The pixels of each pair are ordered so that the first pixel is the one that belongs to the segment. If we change the segment wrap count by any integer $j$, the only border segment pairs that do not add any cost to the $L^0$-norm are the ones assigned with a discrepancy $j$. Obviously the optimal wrap count change for the segment is the discrepancy tag $j'$ whose bin has the maximum weight. In this case we reduce
the $L^0$-norm by $U(j') - U(0)$ where $U(j)$ is the weight associated with the bin of discrepancy $j$ in the histogram. In our algorithm, constructing a border histogram for a segment $s$ is simple, and obtained by merging the histograms assigned to all edges outgoing from $s$.

In the example of Figure 2.3, the result of merging the histograms of all edges outgoing from $s_2$ ($s_2, s_1$, $s_2, s_3$ and $s_2, s_4$) is a histogram with three bins tagged with wrap counts 0 and 1 and weighting 1 and 4, respectively. This is indeed a histogram of all the border pairs of segment $s_2$ and shows that the locally optimal wrap count for $s_2$ is $j = 1$. Choosing this wrap count cancels discontinuities with a total weight of 4 and forms discontinuities with a total weight of 1, thus lowering the $L^0$-norm cost of the phase array by 3.

In the relaxation sweep we analyze each segment in turn and determine its locally optimal wrap count change. For each wrap count change, $j$, chosen for a segment, $s$, we update the data structure to reflect this selection:

- The wrap count value for segment $s$ is increased by $j$.
- The histograms of edges outgoing from $s$ are updated by subtracting $j$ from the discrepancy tag associated with each bin of these histograms.
- The histograms of edges incoming to $s$ are updated by adding $j$ to the discrepancy tag associated with each bin of these histograms.

### 2.3.4 Prolongating coarse level wrap count vector

The recursive call to the coarser $l + 1$ level returns the wrap count vector which was applied for the coarser segments ($WC_{l+1}$). The $l$-level variables should be modified to reflect this coarse level correction. Such flow of information from coarse level to fine level is called prolongation. The coarse level wrap count vector is first prolonged to an $l$-level wrap count vector to indicate for each
$l$-segment the number of wrap count cycles by which it was lifted/dropped in the process of smoothing the coarser levels. This prolongation is obtained by multiplying the aggregation matrix $A_l^{i+1}$ by $WC_{l+1}$. The $l$-level variables are then modified to reflect these coarse level corrections. For each segment $s$ with prolonged wrap count value $j$: $j$ is added to its $l$-level wrap count value; histograms of edges outgoing from $s$ are updated by subtracting $j$ from the discrepancy tag associated with each bin and histograms of edges incoming to $s$ are updated by adding $j$ to the discrepancy tag associated with each bin.

### 2.3.5 Complete Multilevel unwrapping cycle scheme

The algorithm employs several iterations of the multilevel procedure, each consisting of a hierarchical segmentation of the grey level image of the current phase array approximation and a complete multilevel unwrapping cycle. The unwrapping cycle starts by constructing the hierarchy of graphs from the finest level to the coarsest one, and then relaxes the phase array level by level, from the coarsest to the finest level. The multilevel unwrapping cycle routine is implemented recursively as shown next. The input parameters for level $l$ are the graph $G_l$, the index of the coarsest level $M$ and the aggregation matrices obtained from the segmentation algorithm $A_0^1, \ldots, A_{M-1}^M$. The function returns the wrap count vector $WC_l$, indicating the corrections of coarser levels up to level $l$; and the modified graph of level $l$.

### 2.4 Computational Complexity

The segmentation algorithm we used (the fast multi-scale image segmentation algorithm from [13]) run time complexity is $O(N)$, where $N$ denotes the number of pixels in the image. The number of levels in the segmentation hierarchy

---

function \((WC_l, G_l) = MLUCycle(l, G_l, M, A^1_l, \ldots, A^M_l)\),

Initialize wrap count vector to all zeros,

\[ WC_l = 0 \]

\[ \text{if}(l < M) \{ \quad \text{** On every level but the coarsest **} \]

Construct coarse level data (section 2.3.2.3),

\[ G_{l+1} = \text{AggregateSegments}(A^{l+1}_l, G_l) \]

Call MLUCycle recursively,

\[ (WC_{l+1}, G_{l+1}) = MLUCycle(l + 1, G_{l+1}, M, A^1_l, \ldots, A^M_l) \]

Prolongate and update coarse level correction (section 2.3.4),

\[ (WC_l, G_l) = \text{AddCoarseCorrection}(G_l, WC_l, A^{l+1}_l \ast WC_{l+1}) \]

\[ \}

Relax on \( G_l \) until stabilization (section 2.3.3),

\[ (G_l, WC_l) = \text{Relax}(G_l, WC_l) \]

Algorithm 1: Complete multilevel unwrapping cycle scheme
is $O(\log(N))$ and the total number of segments is $O(N)$.

The dominant part of the computational complexity of the first phase in the multilevel unwrapping cycle (construction of the graph hierarchy) is proportional to the total size of edge histograms constructed. The total number of histograms is proportional to the number of segments, but their sizes depend on the variety of discrepancy values listed in the edges histograms. For the input wrapped phase array, discrepancy can only get values of -1, 0, or 1, yielding an $O(N)$ complexity for this phase. Since our algorithm is not restricted to wrapped phase array only, this bound is not true for all cases. Still, the total size of histograms at each level is bounded by $O(N)$, thus the total number of histograms at all levels by $O(N \log(N))$. We do not expect this bound to be reached in most phase array examples, since the variety of discrepancy values found in a phase array is normally small even for unwrapped arrays, thus, $O(N)$ is normally the true computational cost of this phase.

The second phase of the multilevel unwrapping cycle consists of two actions per each level: prolongation of coarse level corrections and several relaxation sweeps. Similarly to the construction of coarser graphs, the computational cost of the prolongation for all the different hierarchy levels is proportional to the total size of edges histograms. Here, the sizes of histograms depend on the variety of discrepancy values found in the reconstructed solution $\phi$, which is constantly changed by the relaxations performed at each level. Again, we expect a bound of $O(N)$ in most examples for this part. The computational cost of each relaxation sweep of some level is proportional to the number segments of that level, but the number of relaxation sweeps needed in order to achieve stabilization depends on the data itself and on the order of scanning the segments. We avoid examples in which the order is not efficient, by reconsidering neighbors of a segments that have been lifted or dropped by the relaxation procedure. Using this strategy we
see, in practice, that the average number of times each segments is considered for relaxation can be normally less than 3. We conclude that the algorithm complexity is normally linear in the number of pixels in the input phase array.

2.5 Experimental results

We demonstrate the performance of our algorithm using two different wrapped phase inputs. All inputs tested contain $64 \times 64$ pixels. The weight function is given by $w(u, v) = |\pi - |\psi(u) - \psi(v)||/\pi$, which imposes a relatively large weight if some multiple of $2\pi$ added to one of its vertices will make the difference across this edge small (hence the solution smooth).

Figure 2.4 shows the result of applying our algorithm to the sheared planes example. The $L^0$ cost of the input phase array whose grey level image is shown in Figure 2.4(a) is 276.1846. Figure 2.4(b) shows the grey level of the unwrapped phase array after applying one multilevel cycle. The cost of this image is 52.1538. The segmentation algorithm partitioned the image into the segments shown in Figure 2.1 (e), but this is not enough to reconstruct the true phase array in a single cycle. Evidently, each segment is at a locally optimal location. Figure 2.4(c) shows the grey level image of the unwrapped phase array after a second cycle. The cost of this image is 28.7538— in this example the true phase is reconstructed after two multilevel cycles.

Figures 2.5(a) and 2.5(c) show wrapped phase values obtained by SAR technology. This phase input is a part of a wrapped phase image found in http://www.cs.sandia.gov/tech_reports/hpc93/hpc12.html. The cost of the wrapped surface is 391.2734. The unwrapping solution produced by our algorithm, after one multilevel cycle, is shown in Figures 2.5(b) and 2.5(d). The cost of this solution is 2.8203.
Figure 2.4:  (a) Grey level image of input phase array. (b) Grey level image of unwrapped phase resulting after one multilevel cycle. (c) Grey level image of unwrapped phase resulting after two multilevel cycle.

Figure 2.5: (a): Three dimensional view of the wrapped phase $\phi$. (b): Three dimensional view of the unwrapped phase constructed by our multilevel algorithm. (c): Grey level image of the wrapped phase. (d): Grey level image of the unwrapped phase constructed by our multilevel algorithm.
Chapter 3

A Multilevel algorithm for the residue cut approach

3.1 An overview of the algorithm

We propose a multilevel algorithm for the residue-cut problem. Articles describing the algorithm appearing in this chapter were submitted to Computing and Visualization in Science—Springer and to the EMG2005 Proceeding. The algorithm constructs a hierarchy of successively coarser graphs by recursively coarsening the dual graph. Each vertex in a coarse-level graph represents several vertices of the next finer level, while each fine vertex is represented by one or more coarse vertices. A coarsening step is a partitioning of the finer-graph vertices into two sets, $C$ and $F$, similarly to classical Algebraic Multigrid (AMG) [2]. Each vertex in the coarse graph is assigned with the charges of all vertices in its neighborhood on the finer graph, and thus represents a patch of vertices of the finer-level graph. Recursively, the vertices at each level thus represent patches of the finest graph.

The hierarchy constructed enables us to recognize charges whose distance from each other is relatively small. The solution forest is constructed by connecting pairs of oppositely signed charges (called dipoles) that are found to be relatively close to each other.
There are two phases in our algorithm. In the **coarsening** phase the hierarchy levels are constructed from fine to coarse. During this phase, edges from different hierarchy levels are collected to the solution, as we connect dipole charges. Next, in the **prolongation** phase the solution edges are prolongated from coarse to fine, towards the finest, target level.

**Dipole matching.** Consider the example of an input dual graph shown in Figure 3.1(a), containing three positive and three negative charges. Connecting these charges in dipole pairs, using a simple nearest neighbor strategy, produces the solution shown in Figure 3.1(b). This strategy was proposed by Huntley in [12]: each unpaired charge is taken in turn and then paired to the nearest unpaired charge of the opposite sign. However, Figure 3.1(c) shows a better solution for connecting these charges in one neutral clump, rather than in pairs. Our algorithm uses a dipole approach—connecting the charges in pairs; in order to overcome the shortcoming of the dipole approach illustrated in Figure 3.1 we apply two concepts to our algorithm.

- We first connect dipole charges which are relatively close to each other.
- Whenever edges are added to the solution, we update the weight of these “used” edges to zero, so that the distances between unpaired residues in our database reflect the true (additional) cost for joining them.

### 3.2 Data structure

We denote by subscript $l$ the level in the hierarchy, with $l = 0$ and $l = M$ denoting the finest and coarsest levels, respectively. For example, $V_l$ indicates the set of vertices in $G_l$—the graph of hierarchy level $l$.

Consider the set of charges $\{c_i\}_{i=0}^{C}$ of the input graph. A charge, $c_i$, is described by the couple $(v_j, p)$, where $j$ is the index of the charged vertex in the
Figure 3.1: (a) Example of a dual graph (with all weights assumed equal). Positive and negative residues reside in the black-filled and grey-filled vertices, respectively. (b) The residue cut solution produced when connecting the charges in dipole pairs using a nearest-neighbor strategy. The cut edges are shown by thick lines. The cost of this solution is 28. (c) A better residue cut solution using only 19 edges.
input graph, and $p \in \{+1, -1\}$ is the charge.

At each level, $l$, we maintain the following variables:

- A graph $G_l(V_l, E_l)$.
- An edge based weight vector $w_l : E_l \rightarrow \mathbb{R}^+$.
- A mapping, $\text{VertexCharges}_l : V_l \rightarrow 2^{\{\text{Charges}\}}$, of each vertex $v$ to a set of charges. These are the charges residing in the finest-level patch that $v \in V_l$ represents. For each such mapping we assign $\text{ChargeDist}_l(v, c)$, which is the cost for connecting the charge $c$ to the vertex $v$.
- $D_l$: the length scale of level $l$. This is actually the (approximate) radius of the patches; that is, each vertex represents charges that are within a distance $D_l$.

### 3.3 Coarsening Phase

The coarsening phase involves several iterative coarsening steps, each constructing a next-coarser level graph. We construct each coarser level graph by doubling the patch radius. Doing so, we can represent the graph by a subset $(C)$ of vertices which is significantly smaller than the set of fine-grid vertices.

The heart of a coarsening step is the partitioning of $V_l$ into two sets, $C$ and $F$. We do this by marking each vertex as belonging to either $C$ or $F$. Initially all vertices are unmarked. We then choose an unmarked vertex $v$ for $C$ and perform the following main tasks:

- Expanding $v$’s patch for level $l + 1$: At first $v$’s $l$-level patch includes charges with distance less than $D_l$ to $v$. We expand this $l$-patch to $v$’s $l + 1$ patch—to include charges that are within a distance of $D_{l+1} = 2D_l$. 
• Diluting the set $C$ by removing vertices whose $l$ level patch is completely covered by $v$’s expanded patch.

• Dipole matching: We search for a pair of oppositely signed charges in $v$’s expanded patch. If such a pair is found, we add edges connecting its charges to the solution and remove both charges from the graph.

Recall that whenever edges are added to the solution they are contracted by setting their weight to zero. Thereby, some vertices become closer to $v$ and thus the patch of $v$ needs to be updated to possibly include more charges that came to be $2D_l$ distant from $v$. This is illustrated in Figure 3.2. Hence, the above three tasks are performed iteratively until reaching a homogenously charged patch (with only positive or only negative charges), where no dipole matching is possible.

Before we show the details of the iterative procedure performed for each $v$ chosen to be included in $C$, we discuss how we obtain from the variables of graph level $l$ the expanded $l + 1$ level patch of $v$. We expand the patch of vertex $v$ by testing a charge $c$ found in a patch of another vertex $u \in V_l$. If the distance in $G_l$ between $v$ and $u$ and the distance from $u$ to the charge (denoted $\text{ChargeDist}_l(u, c)$) sums up to less than $2D_l$, then the charge should be added to $v$’s patch for level $l + 1$ with its distance to $v$ estimated as the sum of the distances. Obviously it is enough to consider charges only of vertices that are within a distance $2D_l$ from $v$. We define the neighborhood of a vertex, as the set of vertices that are within a distance $2D_l$. Below are the details of the iterative method performed for each vertex chosen to be included in $C$:

Initially we consider vertices found in $v$’s neighborhood.

Patch expansion: We need to create the patch of $v$ for level $l + 1$ to include charges with a distance less than $D_{l+1} = 2D_l$ from $v$. We do this by
Figure 3.2: An example illustrating the steps performed for each vertex chosen to be included in $C$. This example shows a part of the fine level graph, containing a vertex $v_0$ chosen to be included in $C$. (a) Assume initially that only charges $+1$, $+2$, $+3$ and $-1$ are within a distance $2D_l$ from $v_0$ and are thus included in $v_0$ expanded patch. (b) Charges $+1$ and $-1$ are matched. The edges connecting them (thick red lines) are added to the solution. (c) After the edges added to the solution are contracted, vertex $v_{11}$ becomes closer to $v_0$ and thus its charge $-2$ now needs to be included in $v_0$’s expanded patch.
examining all the charges in the patches of the neighborhood vertices: $v$’s $l+1$ level patch inherits the charges whose distance from $v$ is less than $2D_l$. Our algorithm allows patch overlap, thus a single charge may be mapped to several neighboring vertices—if this occurs, $v$ inherits the charge from the neighboring vertex that minimizes the distance of the charge to $v$.

**Vertex dilution:** From the neighboring vertices, we mark as being in $F$ each vertex, $u$, whose distance from $v$ is less than $D_l$. Since, on level $l$, the patch of $u$ includes charges with distance less than $D_l$ from $u$, and since the $l+1$-level patch constructed around $v$ is expanded to contain charges with distance less than $2D_l$ from $v$, we will have the entire patch of $u$ covered by $v$’s $l+1$-level expanded patch.

**Dipole matching:** If the expanded patch contains both positive and negative charges, we couple a dipole consisting of the positive charge closest to $v$ and the negative charge closest to $v$. This coupling is done by adding to the solution the edges connecting $v$ to the neighborhood vertices from which the dipole charges were inherited. We also add to the solution a request (that will be processed during prolongation), to connect each of the dipole charges to the vertex it was inherited from. Lastly we remove from the graph the connected dipole charges.

**Contracting used edges:** The weight of the edges added by the dipole matching step is set to zero. As a result, $v$’s neighborhood is changed and may contain more vertices or shorter distances to neighboring vertices. We construct the updated neighborhood and re-execute steps patch expansion, vertex dilution and dipole matching. This looping continues until the expanded $v$ patch is homogenously charged.
In a coarsening step from finer level $l$ to coarser level $l + 1$, some edges of the finer level graph may be added to the solution, and their weight is zeroed. Therefore, each hierarchy level contains two graphs—the initial graph denoted by $G_l$, and a modified graph denoted $G'_l$, which differs from $G_l$ only by edge weights, as some non-zero edges of $G_l$ may be zeroed in $G'_l$. Figure 3.3 describes the structure of the hierarchy. Each hierarchy level contains two graphs, $G_l$ and $G'_l$. The coarser level graph, $G_{l+1}$ is actually a coarse representation of the modified weight graph $G'_l$. That is, it represents charges of distance $D_{l+1}$ from each coarse vertex, according to the modified-weight graph.

**Deriving the remaining variables.** After the coarse vertex set, $C$, has been selected, determining the coarse variables is simple. The vertices $V_{l+1}$ are the vertices in $C$. The mapping $VertexCharges_{l+1}$, from $V_{l+1}$ to the charges in each coarse vertex patch and their distances is created during the process of choosing the set $C$, in the patch expansion stage. The length scale, $D_{l+1}$ is twice $D_l$. As for the edges in $G_{l+1}$: we wish to add an edge $(v_1, v_2)$ to $E_{l+1}$ if their patches are adjacent. There are three different scenarios for two patches to be adjacent.
These are illustrated in Figure 3.4. The weight of the edge is the minimum weight over all paths connecting coarse vertices that obey one of the three cases.

Recall that whenever charges are coupled they are removed from the graph. Recursion continues until reaching a graph with no charges.

Choosing the set $C$. When choosing the set of vertices $C$ in the coarsening process, we do not aim for it to be minimal. Actually, we need overlapping of patches, as we would like any two charges that are within a distance $D_{l+1}$ to belong to some patch. To achieve this overlap, we choose the order in which we loop over vertices when determining $C$ such that the next vertex selected is the one closest to some vertex that has already been chosen to belong to $C$.

### 3.4 Prolongation Phase

In the prolongation phase edges and requests are prolongated, scanning the hierarchy from coarse to fine. A single construction step is performed by inspecting two consecutive levels in the hierarchy. We will refer to construction step $l$, as that involving levels $l+1$ and $l$. Thus, the solution forest construction starts from construction step $M - 1$, and continues down to construction step 0. The edges created in construction step $l$ reside in $E_l$. The graphs considered in construction step $l$ are $G'_l$ and $G_{l+1}$. Edges and requests referring to $G'_l$ are then trivially prolonged to the same edges and requests in $G_l$. The tasks performed at construction step $l$ are described below and are also illustrated in Figure 3.5.

**Edge prolongation.** Each edge, $e_{l+1}$, from the previous construction step resides in $E_{l+1}$ and needs to be prolonged to $E'_l$. Recall that $e_{l+1}$ was determined by the minimum weight path in $G'_l$ connecting the two endpoint vertices. The edge $e_{l+1}$ is prolonged by reconstructing the edges forming this path.
Figure 3.4: Illustrating the three cases that vertices \( v, u \in C \) will be connected by an edge in the coarse graph. Each of the three figures shows a schematic coarsening scenario. The finer level graphs are shown with the partitioning into \( C \) and \( F \) marked by circular and cross vertices, respectively. Each \( C \) vertex patch is illustrated by the closed curve surrounding it. The coarse vertices will be connected in the coarse graph if (a) An edge in the fine graph connects both vertices or (b) there is an edge connecting \( u \) to a fine vertex that is within the patch of \( v \), or (c) there is an edge connecting two fine vertices such that one fine vertex belongs to the patch of \( v \) and the other fine vertex belongs to the patch \( u \).
Request prolongation. Construction step $l$ needs to prolongate requests of the form $(v, c)$ where $v \in V_{l+1}$ and $c$ is a charge. These are requests to connect the charge $c$ to the vertex $v$. Vertex $v$ is also in $V_l$ (coarser level vertices are a subset of the finer level vertices). Prolongation involves reconstructing the expanded patch in level $l$ around $v$ to find the vertex $u \in V_l$ from which this charge is inherited. We then prolongate the request to the request $(u, c)$, and to the edges of the path connecting $u$ to $v$.

3.5 Computational Complexity

The computational cost of the coarsening of level $l$ is dominated by the cost of constructing the neighborhoods of the representative vertices $C \subset V_l$. Recall that a neighborhood of a vertex $v$ includes all vertices that are within a distance $2D_l$. Thus, the neighborhood of $v$ requires scanning paths outgoing from $v$ with a limited number of edges. Therefore, the computational cost of a coarsening step is $O(Poly(DEG_l) \cdot |V_l|)$, where $DEG_l$ is some typical degree of vertices on level $l$ and $Poly$ is a low-order polynomial. We assume that the number of charges is small relative to the number of vertices—therefore the additional cost of edge contractions is insignificant. The computational cost of the prolongation steps is similar since its main operation is the reconstruction of the vertex neighborhoods.

The computational cost for the complete algorithm is $O(Poly(DEG) \cdot |V_0| \cdot c)$. Here, $DEG$ is a typical degree of vertices in all hierarchy levels. The degree of vertices in graph level $l = 0$ is four, but it may change at coarser levels—depending on the distribution of the user-defined weights. $c$ is the ratio between the total number of vertices in all hierarchy levels and the number of vertices in level 0. This value depends on the number of coarsening steps, and the coarsening ratio. In practice, we see values of $c < 3$ for cases we have tested. We conclude that
Figure 3.5: Illustration of edges and request prolongation from level $l+1$ to $l$. 
(a) A part of $G_{l+1}$. Assume the edge $(v_2, v_5)$ and the request to connect charge $+1$ to vertex $v_5$ need to be prolonged. (b) The circular dashed lines indicate the reconstructed neighborhoods around both vertices. The coarse level edge is prolonged to the finer level edges of the path connecting the coarse vertices (edges $(v_2, v_4)$ and $(v_4, v_5)$). The request is prolonged to a finer level request to connect $+1$ to $v_7$ (which is the fine vertex from which $v_5$ inherited this charge) and to the edges connecting $v_7$ to $v_5$. 
the algorithm complexity is normally linear in the number of pixels in the input phase image.

3.6 Experimental results

This section demonstrates the performance of our algorithm applied to three different wrapped phase inputs. All inputs tested contain $64 \times 64$ pixels. The weight function is given by $w(u, v) = |\pi - |\psi(u) - \psi(v)||/\pi$, which imposes a relatively large weight if some multiple of $2\pi$ added to one of its vertices will make the difference across this edge small (hence the solution smooth). The initial length scale ($D_0$) is taken to be the median of the level zero edge weights. Our results are compared with Huntley’s “matching” algorithm ([12]) which connects oppositely signed charges by a nearest neighbor strategy.

The wrapped phase values in Figure 3.6, were synthesized using a known true phase surface of a sine-shaped ramp. Figure 3.6(a) shows the true phase values $\phi$. The $L^0$ cost, i.e., the weighted number of discontinuities, of the true surface is 41.32181. Figure 3.6(b) shows the wrapped surface, $\psi$, whose cost is 136.4637. Figures 3.6(c) and 3.6(d) show both surfaces as grey level images.

Figure 3.7(a) shows the charges of the dual graph and our solution to the residue-cut problem. It is actually composed of a single tree—the four branches are in fact connected through the “external” face vertex of the dual graph. The total cost of this tree is 44.54375. There were 5 coarsening steps involved in the construction of the tree solution. Figure 3.7(c) shows the grey level output image, obtained by integrating discontinuities of the wrapped phase along paths that exclude tree edges. The cost of this image is 43.54375. Note that the tree cost bounds the integrated image cost from above. Evidently, the tree is not locally optimal, as many paths are impossible due to the coarsening. Therefore we always
Figure 3.6: (a): Three dimensional view of the true phase $\phi$. (b): Three dimensional view of the wrapped phase $\psi$. (c): Grey level image of the true phase $\phi$. (d): Grey level image of the wrapped phase $\phi$. 
Figure 3.7: (a): Location of charges in the dual graph—positive and negative charges are shown by diamonds and circles, respectively. Dark lines represent the edges of the tree solution produced by the multilevel algorithm. (b): Tree solution produced by Huntley’s “matching” algorithm. (c): Grey level image of the unwrapped phase after integrating discontinuities of the wrapped phase along paths that exclude all tree edges. (d): Unwrapped image after local smoothing post processing. (e): Three dimensional view of the smoothed solution. (f): Three dimensional view of phase unwrapped with the “matching” algorithm solution.
apply several cheap post-processing relaxation sweeps where each pixel value in 
turn is incremented by $2\pi k$, $k \in \mathbb{Z}$, with $k$ chosen so as to locally minimize the 
cost (Gauss-Seidel style)—this is actually a cheap version of the segmentation 
based unwrapping algorithm containing only one pixel-per-segment scale level. 
Figures 3.7(d) and 3.7(e) show the resulting solution as a grey level image and 
three dimensional surface, respectively. Evidently, in this problem we obtain a 
perfect reconstruction. Figure 3.7(b) shows the tree solution constructed when 
using Huntley’s “matching” algorithm. Figure 3.7(f) shows the three dimensional 
surface obtained by integrating discontinuities according to this tree. The cost of 
this solution is 54.3288.

Figure 3.8 shows another synthetic example. Here we demonstrate how our 
algorithm overcomes the shortcoming of the dipole approach illustrated by Figure 
3.1. The $L^0$ cost of the true surface (Figures 3.8(a) and 3.8(c)) is 26.97693. The 
cost of the wrapped surface (Figures 3.8(b) and 3.8(d)) is 432.1654. The location of 
the dual graph charges in this example is similar to the charge locations in Figure 
3.1. Our algorithm connects the charges in one neutral clump, and Figure 3.8(e) 
shows a grey level image, focused on the center of the phase solution constructed 
by our algorithm, after applying the post processing smoothing method. The cost 
of this solution is 26.97693. Evidently, the phase has been reconstructed exactly. 
Huntley’s matching algorithm connects the charges in pairs, and Figure 3.8(f) 
shows a grey level image, focused on the center of the constructed phase solution. 
The cost of this solution is 77.76276.

Figures 3.9(a) and 3.9(c) show wrapped phase values obtained by SAR tech-

ology. The cost of the wrapped surface is 391.2734. The unwrapping solution 
produced by our algorithm is shown in Figures 3.9(b) and 3.9(d). The cost of this 
solution is 2.8203.
Figure 3.8: (a): Three dimensional view of the true phase $\phi$. (b): Three dimensional view of the wrapped phase $\psi$. (c): Grey level image of the true phase. (d): Grey level image of the wrapped phase. (e): Zoom in on the grey level image of the unwrapped phase constructed by our multilevel algorithm. (f): Zoom in on the grey level image of the unwrapped phase constructed by the matching algorithm.
Figure 3.9: (a): Three dimensional view of the wrapped phase $\phi$. (b): Three dimensional view of the unwrapped phase constructed by our multilevel algorithm. (c): Grey level image of the wrapped phase. (d): Grey level image of the unwrapped phase constructed by our multilevel algorithm.
Bibliography


שיהזור רב-שקבי של פאות בד מימד

עדיהט שלם
שחזרו רב-שכבתי של פאות בד מימד

היבחר על מחקר

ليس מילה חולק שלHDRיושת לקבילת החיזור
מגיסטר למדעים במדעי המחשב

עדית שלם

הוגש לסנט הטכניון — מרכז תכונולגיה לישראל
אדר תשס"ז, חיפה, מרץ 2007
ברצוני לאותת את הנוחות של הפרופסור יברון על שיתוף הפעולה המודגש בין נשותاقת ומייניק. ולא הוזזתי כי בחרתי את חיבור זה בצפיפות של חיבורי ועל כך שקורא ותיק ותיו. כך הוריתי על נכונותי לשמור על הבנות בכל עת שהזדקקתי ל. כמו כן אני מודה למשפחתי ועל , תודה לבעלי על התמיכה הנפשית. שקראה את החיבור וחלק עמי רעיונות לגביו, התענינותו במחקר. מחקרי הכתיבתי בכתיבתי ול yan. עד ההערכה לכתיבתי, שקראתי את הכתיבתי בכל ערי רעיונות לגביו. זוהי לבלי על התמיכת הנסחרת, וול. על כן שדעת אנשי ליום המなくて.

אני מודה לטכניון על העזרה הכספית שהובלה בשתייתן.
I
1.1
1.2
1.3
1.4
1.4.1
2
2.1
2.2
2.3
2.3.1
2.3.2
2.3.3
2.3.4
2.3.5
2.4
2.5
3
3.1
3.2
3.3
3.4
3.5
3.6
45
51
רשימת איורים

1.1 גאומטריה בסיסית SAR
1.2 דוגמא Lehrer פאזה מקוצצים ומושפעים
1.3 דוגמה Lehrer פאזה מקוצצים ומושפעים בחך-فرق
1.4 דוגמה Lehrer פאזה מקוצצים
1.5 מרחוק משגזר
1.6 גרף דו-ילני

2.1 משטחים משופעים
2.2 סגמנטציה היררכי
2.3 דוגמה Lehrer סגמנטציה בודדת
2.4 יישום Lehrer המבוסס סגמנטציה לדגרות המ若您ים המשמשים
2.5 יישום Lehrer המבוסס סגמנטציה לדגרות כלים

3.1 חיבור מטענים בזוגות מנוגדים
3.2 פעולות עבור צומת \( c \)
3.3 מבנה היררכי גורח לדגרות של היררכי ארבע-שכבות
3.4 בניית קשתות הודמה הנבנית
3.5 יעיוני קשתות הבקרה
3.6 כל קשת מקוצצים של סוללה בצורת סיבוב
3.7 יישום Lehrer חיבור מטענים רכש מכברת לדגרות סיבוב
3.8 יישום Lehrer חיבור מטענים רכש מכברת לדגרות כלים
3.9 Lehrer יישום Lehrer חיבור מטענים רכש מכברת לדגרות כלים
תקציר המחקרים

הקדמה

ה죄 בעיות ומציגים שני אלגוריתמים לפתרונה. בדו מימדית, שיחזור הפאזה

החוקרים רבים מסכימים כי המטרה היא לשחזר הפאזה הנובעת של ערכי שבר של ערכים סמוכים בעלי הפרש של יותר ממחצית מחזור על מספרם הממוצע של גל되었י היינו נתונים כדגימות על שריפ דו מימדית אחיד. ערכים הנתוניםينااختبار פאזה של מקוצצים, השם של מקוצצים השフト

ופאזה המקוצצים בשטח את \( \psi \), נאמר כי שיחזור \( \phi \) הוא ישים

אם קיים

מערך דו-מימדי של ערכים שלמים \( k(i,j) \), המקיים

\( \phi(i'j) = \psi(i'j) + 2\pi k(i'j) \), כאשר

\( k(i'j) \) היא פונקציה בעלת ערכים

שלמים. במקרהים מערכיים זה יש להגביל את האפשרויות לבחירת \( \phi \) מכיוון שכל שני מערכים שהפרשי

הוא כפולה שלמה של \( 2\pi \) בכל פיקסל ביניהם הערכים

הגישה הרווחת היא למצוא \( \psi \) הם פתרונות ישימים

אלגוריתמי שיחזור פאזה מנסים למזער נורמה, על מנת ליישם גישה זו. Był 후"ש חלק" את השחזור המставка שינו

ה攻略 לשחזור פאזה

נורמת מינימום בשיטות מזעור הנורמה מחפשים את הפתרון הממזער לירות

ה酾תן של המקוצצים השフト

נורמת מינימום \( L^p \) \( L^p \) מוגדרת על ידי מזעור הביטוי

\[ \sum_{i=0,..M-1} \sum_{j=0,..N-1} w_{ij} | D_{ij}^x - \Delta_{ij}^x |^p + \sum_{i=0,..M} \sum_{j=0,..N-1} w_{ij} | D_{ij}^y - \Delta_{ij}^y |^p \]

של המרחק בין הנגזרות הדיסקרטיות \( \phi \) לבל יבצעו פענוח סינתזה של השונות \( (\pi, \pi) \) על ידי פעולת

modulo \( 2\pi \).}

шуים

שלם פונקציה בעלת ערכים

שלמים. במקרהים מערכיים זה יש להגביל את האפשרויות לבחירת \( \phi \) מכיוון שכל שני מערכים שהפרשי

הוא כפולה שלמה של \( 2\pi \) בכל פיקסל ביניהם הערכים

הגישה הרווחת היא למצוא \( \psi \) הם פתרונות ישימים

אלגוריתמי שיחזור פאזה

נורמת מינימום בשיטות מזעור הנורמה מחפשים את הפתרון הממזער

לrious

של המרחק בין הנגזרות הדיסקרטיות \( \phi \) לבל יבצעו פענוח סינתזה של השונות \( (\pi, \pi) \) על ידי פעולת

$w_{i,j}^x$, $w_{i+1,j}^x$, $D_{i,j}^x$, $D_{i,j}^y$ זהות המשקלים הנתונים עבור השכנויות המאוזנות בין זוג פיקסלי ($i,j$) ו$(-i+1,j)$, $w_{i,j}^y$ הם באופן דומה משקלי שכנויות מאונכות. $\Delta_{i,j}^x$ ו$\Delta_{i,j}^y$ הם הנגזרות הדיסקרטיות המאוזנות ומאונכות ב俸יח. $\psi(x)$ היא משטח המתכתتحرك והיה כל שינויי בטין בכפייה, $p_0$ הוא ריציפה נמוכה יותר ככל שמשתמשים בערכי הערכית המרוכש עם ערך נמוך יותר, אלא שהבעיה קשה יותר לפתרון עבור ערכים אלו של $p_0$.

שיטות עוקבות מסלול

השליה של התכונה המאוזנת של משטח המורכב של פיקסלים היא פתרון ישים למשטח המשוחזר על ידי_di וריציפה נמוכה למשטחitary מש合わון המשטח הנתון. אם $p_0$ הוא ריציפה נמוכה למשטח משוחזרו, אז,$w_{i,j}^x$ הם משקלי השקית המאוזנת או מאונכות, המשכילים על משטח המשוחזר, $\psi(x)$ הוא משטח המשוחזרו, $w_{i,j}^y$ הם משקלי השקית המאוזנת או מאונכות המשכילים על משטח המשוחזרו, $\Delta_{i,j}^x$ ו$\Delta_{i,j}^y$ הם הנגזרות הדיסקרטיות המאוזנות ומאונכות המשכילים על משטח המשוחזרו, $\psi(x)$ הוא משטח המשוחזרו, $p_0$ הוא ריציפה נמוכה יותר לכל משטח משוחזרו ב俸יח.

שיטות עוקבות מסלול

השליה של התכונה המאוזנת של משטח המורכב של פיקסלים היא פתרון ישים למשטח המשוחזר על ידי $p_0$ וריציפה נמוכה למשטחריי מש合わון המשטח הנתון. אם $p_0$ הוא ריציפה נמוכה למשטח המשוחזרו, אז,$w_{i,j}^x$ הם משקלי השקית המאוזנת או מאונכות, המשכילים על משטח המשוחזרו, $\psi(x)$ הוא משטח המשוחזרו, $w_{i,j}^y$ הם משקלי השקית המאוזנת או מאונכות המשכילים על משטח המשוחזרו, $\Delta_{i,j}^x$ ו$\Delta_{i,j}^y$ הם הנגזרות הדיסקרטיות המאוזנות ומאונכות המשכילים על משטח המשוחזרו, $\psi(x)$ הוא משטח המשוחזרו, $p_0$ הוא ריציפה נמוכה יותר לכל משטח המשוחזרו ב俸יח.

שיטות עוקבות מסלול

השליה של התכונה המאוזנת של משטח המורכב של פיקסלים היא פתרון ישים למשטח המשוחזר על ידי $p_0$ וריציפה נמוכה למשטחריי מש合わון המשטח הנתון. אם $p_0$ הוא ריציפה נמוכה למשטח המשוחזרו, אז,$w_{i,j}^x$ הם משקלי השקית המאוזנת או מאונכות, המשכילים על משטח המשוחזרו, $\psi(x)$ הוא משטח המשוחזרו, $w_{i,j}^y$ הם משקלי השקית המאוזנת או מאונכות המשכילים על משטח המשוחזרו, $\Delta_{i,j}^x$ ו$\Delta_{i,j}^y$ הם הנגזרות הדיסקרטיות המאוזנות ומאונכות המשכילים על משטח המשוחזרו, $\psi(x)$ הוא משטח המשוחזרו, $p_0$ הוא ריציפה נמוכה יותר לכל משטח המשוחזרו ב俸יח.

שיטות עוקבות מסלול

השליה של התכונה המאוזנת של משטח המורכב של פיקסלים היא פתרון ישים למשטח המשוחזר על ידי $p_0$ וריציפה נמוכה למשטחריי מש合わון המשטח הנתון. אם $p_0$ הוא ריציפה נמוכה למשטח המשוחזרו, אז,$w_{i,j}^x$ הם משקלי השקית המאוזנת או מאונכות, המשכילים על משטח המשוחזרו, $\psi(x)$ הוא משטח המשוחזרו, $w_{i,j}^y$ הם משקלי השקית המאוזנת או מאונכות המשכילים על משטח המשוחזרו, $\Delta_{i,j}^x$ ו$\Delta_{i,j}^y$ הם הנגזרות הדיסקרטיות המאוזנות ומאונכות המשכילים על משטח המשוחזרו, $\psi(x)$ הוא משטח המשוחזרו, $p_0$ הוא ריציפה נמוכה יותר לכל משטח המשוחזרו ב俸יח.

שיטות עוקבות מסלול

השליה של התכונה המאוזנת של משטח המורכב של פיקסלים היא פתרון ישים למשטח המשוחזר על ידי $p_0$ וריציפה נמוכה Lמשטחריי מש合わון המשטח הנתון. אם $p_0$ הוא ריציפה נמוכה Lמשטח המשוחזרו, אז,$w_{i,j}^x$ הם משקלי השקית המאוזנת או מאונכות, המשכילים על משטח המשוחזרו, $\psi(x)$ הוא משטח המשוחזרו, $w_{i,j}^y$ הם משקלי השקית המאוזנת או מאונכות המשכילים על משטח המשוחזרו, $\Delta_{i,j}^x$ ו$\Delta_{i,j}^y$ הם הנגזרות הדיסקרטיות המאוזנות ומאונכות המשכילים על משטח המשוחזרו, $\psi(x)$ הוא משטח המשוחזרו, $p_0$ הוא ריציפה Nמוכה יותר לכל משטח המשוחזרו ב俸יח.

שיטות עוקבות מסלול

השליה של התכונה המאוזנת של משטח המורכב של פיקסלים היא פתרון ישים Lמשטחריי מש合わון המשטח הנתון. אם $p_0$ הוא ריציפה Nמוכה Lמשטח המשוחזרו, אז,$w_{i,j}^x$ הם משקלי השקית המאוזנת או מאונכות,משכילים על משטח המשוחזרו, $\psi(x)$ הוא משטח המשוחזרו, $w_{i,j}^y$ הם משקלי השקית המאוזנת או מאונכות המשכילים על משטח המשוחזרו, $\Delta_{i,j}^x$ ו$\Delta_{i,j}^y$ הם הנגזרות הדיסקרטיות המאוזנות ומאונכות המשכילים על משטח המשוחזרו, $\psi(x)$ הוא משטח המשוחזרו, $p_0$ הוא ריציפה Nמוכה יותר לכל משטח המשוחזרו ב俸יח.

שיטות עוקבות מסלול

השליה של התכונה המאוזנת של משטח המורכב של פיקסלים היא פתרון ישים Lמשטחריי מש合わון המשטח הנתון. אם $p_0$ הוא ריציפה Nמוכה Lמשטח המשוחזרו, אז,$w_{i,j}^x$ הם משקלי השקית המאוזנת או מאונכות,משכילים על משטח המשוחזרו, $\psi(x)$ הוא משטח המשוחזרו, $w_{i,j}^y$ הם משקלי השקית המאוזנת או מאונכות המשכילים על משטח המשוחזרו, $\Delta_{i,j}^x$ ו$\Delta_{i,j}^y$ הם הנגזרות הדיסקרטיות המאוזנות ומאונכות המשכילים על משטח המשוחזרו, $\psi(x)$ הוא משטח המשוחזרו, $p_0$ הוא ריציפה Nמוכה יותר لكل משטח המשוחזרו ב俸יח.

שיטות עוקבות מסלול

השליה של התכונה המאוזנת של משטח המורכב של פיקסלים היא פתרון ישים Lמשטחריי מש合わון המשטח הנתון. אם $p_0$ הוא ריציפה Nמוכה Lמשטח המשוחזרו, אז,$w_{i,j}^x$ הם משקלי השקית המאוזנת או מאונכות,משכילים על משטח המשוחזרו, $\psi(x)$ הוא משטח המשוחזרו, $w_{i,j}^y$ הם משקלי השקית המאוזנת או מאונכות המשכילים על משטח המשוחזרו, $\Delta_{i,j}^x$ ו$\Delta_{i,j}^y$ הם הנגזרות הדיסקרטיות המאוזנות ומאונכות המשכילים על משטח המשוחזרו, $\psi(x)$ הוא משטח המשוחahrו, $p_0$ הוא ריציפה Nמוכה יותר לכל משטח המשוחזרו ב俸יח.

שיטות עוקבות מסלול

השליה של התכונה המאוזנת של משטח המורכב של פיקסלים היא פתרון ישים Lמשטחריי מש合わון המשטח הנתון. אם $p_0$ הוא ריציפה Nמוכה Lמשטח המשוחזרו, אז,$w_{i,j}^x$ הם משקלי השקית המאוזנת או מאונכות,משכילים על משטח המשוחזרו, $\psi(x)$ הוא משטח המשוחזרו, $w_{i,j}^y$ הם משקלי השקית המאוזנת או מאונכות המשכילים על משטח המשוחזרו, $\Delta_{i,j}^x$ ו$\Delta_{i,j}^y$ הם הנגזרות הדיסקרטיות המאוזנות ומאונכות המשכילים על משטח המשוחזרו, $\psi(x)$ הוא משטח המשוחזרו, $p_0$ הוא ריציפה Nמוכה יותר لكل משטח המשוחזרו ב俸יח.

שיטות עוקבות מסלול

השליה של התכונה המאוזנת של משטח המורכב של פיקסלים היא פתרון ישים Lמשטחריי מש合わון המשטח הנתון. אם $p_0$ הוא ריציפה Nמוכה Lמשטח המשוחזרו, אז,$w_{i,j}^x$ הם משקלי השקית המאוזנת או מאונכות,משכילים על משטח המשוחזרו, $\psi(x)$ הוא משטח המשוחזרו, $w_{i,j}^y$ הם משקלי השקית המאозנת או מאונכות המשכילים על משטח המשוחזרו, $\Delta_{i,j}^x$ ו$\Delta_{i,j}^y$ הם הנגזרות הדיסקרטיות המאוזנות ומאונכות המשכילים על משטח המשוחזרו, $\psi(x)$ הוא משטח המשוחזרו, $p_0$ הוא ריציפה Nמוכה יותר לכל משטח המשוחזרו ב俸יח.
אלגוריתם בסיסי סגמנטציה

 monuments of sets אזורים, האזורים הנקראים, למספר אזורים נחלקת התמונה, האזורים הם מוטטים
בהם המרוכבים מאורכים כנראה קיים. אלגוריתם סגמנטציה איזור, וכלelier הולך וشروק. ברד
תמונה לוספת מבודד שוטה. שחרור, בחר飲み פיתוח האזורים סגמנטציה adapted - עלון וממסים
אינו אלגוריתם שיווק התמונה, התמונה של האזורים סגמנטציה מוכנ לרצועה של סגמנטציה, הברגה
akening בויר של פייקולוה או סגמנטציה בודד. חל הרצועה ובעון, חל הרצועה של סגמנטציה.
שבוניי הרצועה עד gyro של דה thơ המגמה

ה궐ח התוכן: באלגוריתם השיווק, האלגוריתם סגמנטציה הסתכל לרצועה של רמות את האפור של_street
הycastle קלאס סגמנטציה הירכית לרצועה. והלא עמוס מבעד יפר�� סגמנטציה של תזות
הלוחית או התוכנית בכל הרמה. התוכנית המרובעת פיון גורם לרצועה סגמנטציה, מברך יחד
多样性 של הרצועה בברווחית התוכן של המרור, מũוק הרמה של מריים, מũוק במברך סגמנטציה של רמה
של התוכן בברווחית התוכן של המרור, מũוק המרור, מũוק בברווחית התוכן של המרור, מũוק

ה럴ורית קריסיסית, פאזה ברה סגמנטציה של רמה של להירצועה

כל הרמה ברה סגמנטציה של רמה של להירצועה
c.פることは שהписанים המושרו. כמך של האלגוריתם של
יאני מובילים תריצי קריסיסית. לכל האלגוריתם צורת לשמש ליישר תצוגה של האלגוריתם
שוחר אחר 발표יה, ראו דריציס מברך איריזואית של האלגוריתם, מחשב את האלגוריתם מברך סגמנטציה של התוכן של המרור, מברך סגמנטציה של רמה של להירצועה

ה럴ורית קריסיסית, פאזה ברה סגמנטציה של רמה של להירצועה

ה럴ורית קריסיסית, פאזה ברה סגמנטציה של רמה של להירצועה
c.פ הדברים שהписанים המושרו. כמך של האלגוריתם של
יאני מובילים תריצי קריסיסית. לכל האלגוריתם צורת לשמש ליישר תצוגה של האלגוריתם
שוחר אחר 발표יה, ראו דריציס מברך איריזואית של האלגוריתם, מחשב את האלגוריתם מברך סגמנטציה של התוכן של המרור, מברך סגמנטציה של רמה של להירצועה

ה럴ורית קריסיסית, פאזה ברה סגמנטציה של רמה של להירצועה

ה럴ורית קריסיסית, פאזה ברה סגמנטציה של רמה של להירצועה
c.פ tendência של המרור, מברך סגמנטציה של רמה של להירצועה

The algorithm performs two stages. The first stage is the bottom-up clustering of nodes within each layer in the graph, and the second stage is the top-down clustering of nodes within each layer in the graph. Each node represents an area of the graph, and each area is merged recursively to represent a more detailed area of the graph. The algorithm is based on the principle that nodes that are close to each other in the hierarchy are merged to form larger clusters.

The algorithm is composed of two stages: the bottom-up stage, which merges clusters of nodes within each layer in the graph, and the top-down stage, which merges clusters of nodes within each layer in the graph. Each node represents an area of the graph, and each area is merged recursively to represent a more detailed area of the graph. The algorithm is based on the principle that nodes that are close to each other in the hierarchy are merged to form larger clusters.

The results of the algorithm are presented in the final version of the algorithm, which is based on the principle that nodes that are close to each other in the hierarchy are merged to form larger clusters.

The result of the algorithm is presented in the final version of the algorithm, which is based on the principle that nodes that are close to each other in the hierarchy are merged to form larger clusters.