Two-dimensional phase unwrapping with minimum weighted discontinuity

Thomas J. Flynn

Sandia National Laboratories, P.O. Box 5800, Mail Stop 0576, Albuquerque, New Mexico 87185-0576

Received October 24, 1996; revised manuscript received March 20, 1997; accepted March 21, 1997

Given an interferometric phase image of a surface profile, the task of two-dimensional phase unwrapping is to reconstruct the profile by adding multiples of 2π to the image. Discontinuities in the unwrapped phase must be restricted to areas of noise and true discontinuity in the profile. Such areas can often be identified by their low quality. This suggests that the unwrapped phase should be chosen to minimize a weighted sum of discontinuity magnitudes. An algorithm is presented that computes such an unwrapped phase from any initial guess. The elementary operation of the algorithm is to partition the image into two connected regions, then raise the unwrapped phase by 2π in one of the regions, reducing the weighted sum; this is done repeatedly until no suitable partitions exist. The operations are found by creating paths that follow discontinuity curves and extending them to form complete partitions. The algorithm terminates when no path can be extended. The behavior of the algorithm and the benefits of weighting are illustrated with an example. © 1997 Optical Society of America [S0740-3232(97)01110-1]

1. INTRODUCTION

A. Phase Unwrapping

The problem of two-dimensional phase unwrapping arises whenever a two-dimensional profile is measured with an interferometric system. Applications include measurements of terrain height by interferometric synthetic-aperture radar (IFSAR)[1] and measurements of deformation of mechanical parts by laser interferometry.[2] After geometric correction the system produces a periodic measurement of the profile:

$$\phi(x, y) = \left[\frac{2\pi h(x, y)}{H} + \nu(x, y)\right] \bmod 2\pi, \tag{1}$$

where x and y are the position coordinates, $h(x, y)$ is the profile, H is the period, $\nu(x, y)$ is the measurement error, and $\phi(x, y)$ is the measurement expressed as a phase angle. In practice, the measurement is sampled on a finite grid, here assumed rectangular:

$$\phi_{mn} \equiv \phi(x_m, y_n), \tag{2}$$

where $x_m = x_0 + m\Delta x$ for $m = 0, 1, ..., M - 1$ and $y_n = y_0 + n\Delta y$ for $y = 0, 1, ..., N - 1$. The array of measurements is called the phase image, and its elements are called pixels. The task of a phase-unwrapping algorithm is to reconstruct the profile by adding multiples of the period:

$$\hat{h}_{mn} = H\hat{\phi}_{mn}/2\pi, \tag{3}$$

where the unwrapped phase is found by adding a multiple of 2π to each pixel, that is,

$$\hat{\phi}_{mn} = \phi_{mn} + 2\pi c_{mn}. \tag{4}$$

The integer array $\mathbf{c}$, called the wrap-count array, is calculated from the phase image $\boldsymbol{\phi}$, possibly with the aid of auxiliary information.

B. Minimum-Discontinuity Criterion

The goal of phase unwrapping is to recover the true profile, allowing steep slopes only where they exist in the profile. A characteristic failure mode is the appearance of large phase differences, known as discontinuities, in incorrect locations. A common unwrapping strategy is to regard all discontinuities as flaws and attempt to eliminate them. This is reasonable if the profile is almost entirely smooth. We define a discontinuity, or jump, in the unwrapped phase as a pair of neighboring pixels whose difference is greater than π in magnitude. The jump can be removed by adding a multiple of 2π to the phase difference. This multiple is called the jump count for this pixel pair; its magnitude measures the steepness of the discontinuity. The vertical (first index) jump counts obey the formula

$$v_{mn} = \left\lfloor \frac{\hat{\phi}_{mn} - \hat{\phi}_{m-1,n} + \pi}{2\pi} \right\rfloor \quad \text{for } (m, n) \in \mathscr{V}, \tag{5}$$

where $\lfloor x \rfloor$ is the largest integer less than or equal to x (the "floor" function) and $\mathscr{V} = \{1, 2, ..., M - 1\} \times \{0, 1, ..., N - 1\}$. Thus v_{mn} is zero if $\hat{\phi}_{mn}$ and $\hat{\phi}_{m-1,n}$ differ by less than π. The horizontal (second index) jump counts are

$$z_{mn} = \left\lfloor \frac{\hat{\phi}_{mn} - \hat{\phi}_{m,n-1} + \pi}{2\pi} \right\rfloor \quad \text{for } (m, n) \in \mathscr{Z}, \tag{6}$$

where $\mathscr{Z} = \{0, 1, ..., M - 1\} \times \{1, 2, ..., N - 1\}$. Applying Eq. (4), we can express the jump counts as functions of the wrap counts and the phase image:

$$v_{mn} = c_{mn} - c_{m-1,n} + \left\lfloor \frac{\phi_{mn} - \phi_{m-1,n} + \pi}{2\pi} \right\rfloor, \tag{7}$$

$$z_{mn} = c_{mn} - c_{m,n-1} + \left\lfloor \frac{\phi_{mn} - \phi_{m,n-1} + \pi}{2\pi} \right\rfloor. \tag{8}$$

0740-3232/97/1002692-10$10.00

Given the jump-count arrays and the wrap count at a single pixel, the entire wrap-count array and unwrapped-phase image can be computed by accumulation of jump counts. For instance, given $\mathbf{v}$, $\mathbf{z}$, and c_{00}, the rest of $\mathbf{c}$ can be found by first setting

$$c_{m0} = c_{m-1,0} + v_{m0} - \left\lfloor \frac{\phi_{mn} - \phi_{m-1,n} + \pi}{2\pi} \right\rfloor \quad (9)$$

successively for $m = 1, \ldots, M - 1$, then setting

$$c_{mn} = c_{m,n-1} + z_{mn} - \left\lfloor \frac{\phi_{mn} - \phi_{m,n-1} + \pi}{2\pi} \right\rfloor \quad (10)$$

for $(m, n) \in \mathcal{Z}$. These formulas are found by solving Eqs. (7) and (8) for c_{mn}.

The sum of the magnitudes of the jump counts,

$$E_0(\mathbf{c}; \boldsymbol{\phi}) = \sum_{(m,n)\in\mathcal{T}} |v_{mn}| + \sum_{(m,n)\in\mathcal{Z}} |z_{mn}|, \quad (11)$$

is a measure of the overall severity of the jumps. The notation emphasizes that the sum is determined by the data and the wrap-count array through Eqs. (7) and (8). The unwrapped phase is called a minimum-discontinuity (MD) solution if it achieves the minimum of $E_0(\mathbf{c}; \boldsymbol{\phi})$ over all choices of $\mathbf{c}$. There may be more than one MD solution for a given phase image.

The MD criterion can be derived from the minimum L^p-norm criterion of Ghiglia and Romero.[3,4] Their algorithm finds the unwrapped phase whose spatial differences match the unwrapped differences in the data as closely as possible in L^p norm; that is, $\hat{\boldsymbol{\phi}}$ is chosen to minimize

$$J = \sum_{(m,n)\in\mathcal{T}} |\hat{\phi}_{mn} - \hat{\phi}_{m-1,n} - W(\phi_{mn} - \phi_{m-1,n})|^p + \sum_{(m,n)\in\mathcal{Z}} |\hat{\phi}_{mn} - \hat{\phi}_{m,n-1} - W(\phi_{mn} - \phi_{m,n-1})|^p, \quad (12)$$

where W is the minimum-magnitude phase-wrapping operator:

$$W(\phi) = \phi - 2\pi \left\lfloor \frac{\phi + \pi}{2\pi} \right\rfloor. \quad (13)$$

The minimum-norm solution is typically not congruent mod 2π with the phase data. If only congruent solutions are allowed, then Eq. (4) must hold, and the L^p norm can be written as

$$J = \sum_{(m,n)\in\mathcal{T}} |\phi_{mn} - \phi_{m-1,n} + 2\pi(c_{mn} - c_{m-1,n}) - W(\phi_{mn} - \phi_{m-1,n})|^p + \sum_{(m,n)\in\mathcal{Z}} |\phi_{mn} - \phi_{m,n-1} + 2\pi(c_{mn} - c_{m,n-1}) - W(\phi_{mn} - \phi_{m,n-1})|^p, \quad (14)$$

from which, by using the definition of W and Eqs. (7) and (8),

$$J = 2\pi \left(\sum_{(m,n)\in\mathcal{T}} |v_{mn}|^p + \sum_{(m,n)\in\mathcal{Z}} |z_{mn}|^p \right). \quad (15)$$

So a MD unwrapped phase achieves the minimum L^1 norm over all unwrapped phases congruent with the data.

C. Discontinuities and Residues

A phase image of a smooth profile contains curves of unit-magnitude jumps, called fringes. Each fringe partitions the image into two connected parts; it can be removed by incrementing the pixels in one part by 2π. Applying this operation to all fringes produces a smooth unwrapped phase. The algorithm of Lin *et al.*[5] implements this method by using edge detection techniques from image processing to locate the fringes.

In actual phase images, noise and true discontinuities of the profile (such as layover in IFSAR or object boundaries in laser interferometry) can cause fringes to merge, break them into segments, or create isolated segments. The places where fringes merge or segments end are called residues by Goldstein *et al.*[6] and discontinuity sources by Huntley.[7] A positive (negative) residue is a loop of four pixels in the phase image, around which the sum of jump counts, taken clockwise, is $+1$ (-1). Changes in the wrap-count array do not change the sum of jump counts around any loop and so do not create or remove residues. A curve of discontinuities starting at any residue must terminate at a residue of opposite polarity or at the boundary of the image. So if residues are present, it is not possible to eliminate jumps entirely; at best, their number can be reduced by a suitable choice of wrap counts.

Many algorithms seek to connect the residues with short jump curves. The algorithms of Refs. 6 and 7 find pairings of nearby positive and negative residues, drawing branch cuts that connect each pair. Discontinuities appear across the branch cuts. Recent algorithms[8,9] modify an initial set of pairings to reduce the sum of the cut lengths; in Ref. 8 the exact minimum of the sum is attained. Now the contribution of a cut to the discontinuity sum is equal to its length, except in cases in which the jumps across overlapping cuts cancel each other. Such overlaps can be removed by rearranging the cuts, reducing the cut length sum. Thus the MD sum and the minimum-cut-length sum are equal and are achieved by the same unwrapped-phase solution(s). The algorithm of Ref. 8 finds a MD unwrapped phase.

D. Weighting

The criterion of MD is not always appropriate for phase data that contain true profile discontinuities or regions of high noise. For example, IFSAR data from mountainous terrain may contain curves of layover, across which the profile is discontinuous. Shadowing and low radar backscatter create noisy regions with many residues. Where regions of noise or layover are close to each other or close to the boundary of the image, the MD unwrapped phase may contain errors: The gap separating the regions, which should be unwrapped smoothly, may be crossed by jump curves that eliminate jumps within the noisy region.

To mitigate this problem, one can identify regions of low-quality (highly discontinuous) data and modify the

criterion to allow jumps in these regions with little or no penalty. Low-quality phase data are often recognizable by their large and rapidly varying spatial differences. Alternatively, the physical measurements used to find the phase data can indicate the quality. In IFSAR the local average used in calculating the phase also produces a correlation coefficient that indicates the reliability of the phase measurement.[10] Low correlation is caused by large variations of the interferometric phase within the averaging window, as a result of noise or a discontinuity within the window.

To guide jumps into low-quality regions, the appropriate criterion is a weighted sum of jump-count magnitudes:

$$E(\mathbf{c};\ \boldsymbol{\phi}) = \sum_{(m,n)\in\mathscr{V}} w^{v}_{mn}|v_{mn}| + \sum_{(m,n)\in\mathscr{Z}} w^{z}_{mn}|z_{mn}|, \tag{16}$$

where $\mathbf{w}^v$ and $\mathbf{w}^z$ are arrays of nonnegative integer weights derived from the quality information. An unwrapped phase is a minimum-weighted-discontinuity (MWD) solution if it achieves the minimum of $E(\mathbf{c};\ \boldsymbol{\phi})$ over all choices of $\mathbf{c}$. The weights must be chosen, if possible, so that the image is not divided into islands separated by curves of zero weight; if this happens, the unwrapped phase will contain arbitrary offsets of $2n\pi$ between islands. (If portions of the scene are in fact isolated from each other by low-quality pixels, it may not be possible to unwrap the scene as a unit. Phase unwrapping relies on the presence of slowly changing, accurately measured phases connecting all parts of the scene.)

It is difficult to incorporate weighting into the algorithm of Ref. 8, because the best branch cut connecting each possible residue pair must be found. The algorithm presented here does not use explicit residue pairing but reduces the weighted discontinuity by modifying the unwrapped phase directly.

E. Outline of Paper

This paper presents an algorithm that computes an unwrapped phase with MWD sum. The algorithm makes successive modifications to an initial guess. Each modification, or elementary operation (EO), partitions the image into two connected sets of pixels and raises the unwrapped phase by 2π in one of the sets, reducing the sum. An EO is represented by a loop that marks the partition, indicating the locations of the jump-count changes caused by the operation. These matters are explained in detail in Section 2. It is shown in Appendix A that if the unwrapped phase can be improved at all, then it allows an EO. Thus the algorithm finds a MWD solution by repeatedly performing EO's, terminating when no EO is possible.

The algorithm is described in Section 3. It works by creating and extending paths of jump-count changes, similar to the loops that represent EO's but not closed. When a path is extended to form a complete loop, it defines an EO. The algorithm then performs the operation, removes the loop from the collection of paths, and resumes path extension. It terminates when none of the paths can be extended. It is shown in Appendix B that the algorithm finds a MWD solution after a finite number of path extensions. Section 4 presents an example data set and unwrap results, illustrating the benefits of discontinuity minimization, errors in the MD solution, and the use of weighting to guide discontinuities. Section 5 draws conclusions and makes suggestions for further investigation.

2. ELEMENTARY OPERATIONS

A. Definition

Given an initial unwrapped phase, our goal is to optimize it (that is, to minimize its weighted discontinuity sum, or error) by changing the wrap-count array. Our strategy is to test a restricted set of simple changes, make an improving change, repeat the test, make another improvement, and so on until no improving simple change is possible. The intent is that this iteration, with the use of a suitable set of simple changes, finds an optimal unwrapped phase with much less effort than an exhaustive search.

A simple change is made by partitioning the image into two connected sets of pixels and increasing the unwrapped phase by 2π in one of the sets. (A set of pixels is connected if any two pixels in the set can be joined by a sequence of horizontal or vertical neighbors that belong to the set.) The jump counts along the partition are changed by ± 1, thus changing the error according to Eq. (16). A simple change is called an elementary operation (EO) if it reduces the error. The fringe-removal operations of Ref. 5 are EO's, but the set of pixels changed in an EO need not be enclosed in a perfect fringe. The operation needs only to remove more discontinuities than it adds, accounting for weighting.

Figures 1 and 2 illustrate an EO. For clarity, weighting is omitted. The numbers in Fig. 1 are the initial unwrapped-phase pixels, expressed in cycles: The frac-

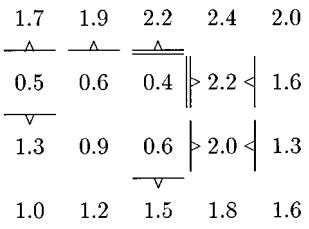

Fig. 1. Initial unwrapped phase with discontinuities marked. The numbers are the unwrapped phase pixels in cycles. The line segments indicate the location and the magnitude of the discontinuities. The direction marker on each segment points toward the higher unwrapped phase. The discontinuity sum is 11.

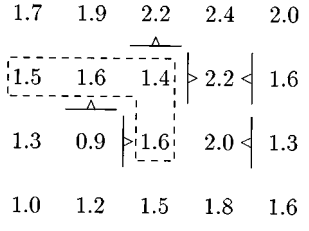

Fig. 2. Result of performing an elementary operation (EO) on the unwrapped phase of Fig. 1. The pixels within the dashed contour have been increased by 1 cycle. The discontinuity markers reflect the changed phases. The discontinuity sum has been reduced to 6.

tional part of each number is the phase datum in cycles, and the integer part is the wrap count. The line segments with direction markers indicate the location and the polarity of discontinuities. Each marker points toward the larger unwrapped phase, and the number of segments separating a pixel pair is the magnitude of the jump count for that pair.

In Fig. 2 the dashed contour marks the partition. The unwrapped phases within it have been increased by 1 cycle, and the discontinuity markers have been changed accordingly. The sum of jump-count magnitudes has decreased from 11 to 6, so the change is an EO.

An initial unwrapped phase may allow many different sequences of EO's. The strategy of optimization by repeated EO's requires that any such sequence terminate with an optimal unwrapped phase. This is implied by the following statements: (1) Any sequence of EO's reaches a final unwrapped phase for which no EO is possible; and (2) if no EO is possible, then the unwrapped phase is optimal. The first statement is true because the error is always nonnegative and is reduced by at least 1 in each EO (because the weights and the jump counts are integers); so it must reach a final value after a finite number of operations, after which no EO is possible. The second statement is proven in Appendix A. So the strategy is guaranteed to find an optimal unwrapped phase; the problem is to implement it. Efficient methods of representing EO's and computing their effects on the error are described in the next subsection. The optimization algorithm itself is described in Section 3.

B. Representation by Loops

The algorithm uses techniques from graph theory to represent and generate elementary operations. Any EO can be specified by a directed closed path, or loop, in an array of nodes interleaved with the phase pixels. A loop is composed of directed steps between neighboring nodes, called edges. The location of the loop specifies the partition; its direction determines which side of the partition is incremented. Figure 3 depicts this representation for the operation of Fig. 2. The nodes are marked with small squares, and the arrows mark the edges of the loop. By convention, because the loop runs counterclockwise, the enclosed pixels are increased by 2π. (For a clockwise loop, the pixels outside the loop are increased.) The node array surrounds the pixel array on all sides, so that a partition can be represented by a loop even if both sets adjoin the boundary of the image.

Each edge goes between a pair of pixels, indicating a change in the jump count for that pair that is due to the EO: A leftward (rightward) edge indicates an increment (decrement) of a vertical jump count; a downward (upward) edge indicates an increment (decrement) of a horizontal jump count. The change of jump count causes a change in the error according to Eq. (16). The value of an edge is defined as the negative of its associated error change. The expression $\delta V(m, n; m', n')$ denotes the value of the edge from the node at coordinates (m, n) to the node at (m', n'). It is defined only for neighboring pairs of nodes. The node at the upper left-hand corner has coordinates $(0, 0)$. An edge from (m, n) to $(m, n + 1)$ causes a decrement to v_{mn}, so its value is

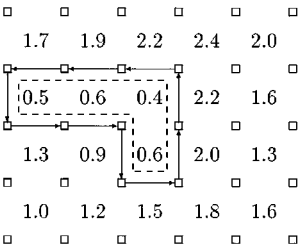

Fig. 3. EO of Fig. 2, represented as a loop. The node array, marked with squares, is superimposed on the initial unwrapped phase of Fig. 1. The arrows mark the edges of the loop. Each edge indicates a change of ± 1 in its associated jump count.

$$\delta V(m, n; m, n + 1) = -w^v_{mn}(|v_{mn} - 1| - |v_{mn}|) = w^v_{mn}\,\mathrm{sgn}(v_{mn} - 1) \quad \text{for } (m, n) \in \mathscr{V}, \tag{17}$$

where $\mathrm{sgn}(x)$ is 1 if $x \geq 0$ or -1 otherwise. An edge from $(m, n + 1)$ to (m, n), which causes an increment to v_{mn}, has value

$$\delta V(m, n + 1; m, n) = -w^v_{mn}(|v_{mn} + 1| - |v_{mn}|) = -w^v_{mn}\,\mathrm{sgn}(v_{mn}) \quad \text{for } (m, n) \in \mathscr{V}. \tag{18}$$

For vertical edges the formulas are

$$\delta V(m, n; m + 1, n) = -w^z_{mn}\,\mathrm{sgn}(z_{mn}) \quad \text{for } (m, n) \in \mathscr{Z}, \tag{19}$$

$$\delta V(m + 1, n; m, n) = w^z_{mn}\,\mathrm{sgn}(z_{mn} - 1) \quad \text{for } (m, n) \in \mathscr{Z}. \tag{20}$$

An edge between two boundary nodes (first index 0 or M, or second index 0 or N) has value zero, because it does not cause a jump-count change.

The value of an EO is the sum of the values of the edges in its loop; it must be positive. The value of a nonclosed path through the node array is also, by definition, the sum of its edge values: If $((m_0, n_0), (m_1, n_1), \ldots, (m_L, n_L))$ is a path starting at node (m_0, n_0), its value is

$$\sum_{k=0}^{L-1} \delta V(m_k, n_k; m_{k+1}, n_{k+1}). \tag{21}$$

The optimization algorithm uses nonclosed paths as partial results in the process of finding EO's.

3. OPTIMIZATION ALGORITHM

A. Path-Extension Search

The task of the optimization algorithm is to find and perform EO's repeatedly, as long as they are possible, and to recognize when no more operations are possible. Recall that an EO is specified by a positive-value loop in the node array. To locate loops efficiently, it makes sense to find nonclosed paths with positive value and extend them to form loops.

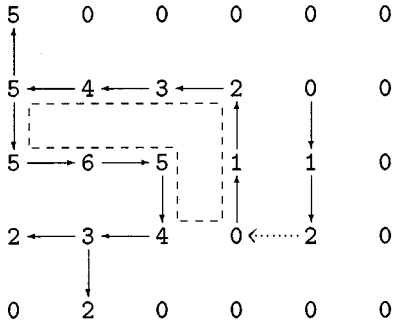

Fig. 4. Node array with a possible collection of paths for the initial unwrapped phase of Fig. 1. The nodes are marked with their values. The outline of the EO is shown for reference. The dotted arrow indicates a type 1 revision.

The algorithm uses this strategy of path extension. It forms a collection of paths through the node array, adding one edge at a time. A path is extended to a new node only if the extended path's value is positive and the node is not already reached by a higher-value path. When an extension forms a loop, the loop represents an EO; the operation is applied to the unwrapped phase, and the loop is removed from the collection of paths. The creation and the extension of paths are then resumed. The algorithm terminates when it cannot create or extend any path.

To save computational effort, the collection of paths is structured. Paths are allowed to divide, but not to merge, thus forming a collection of trees in the node array. Every node is isolated (not in any path), is the root of a tree (paths go out of it but not into it), or is a branch node, reached from the root of its tree by a unique path. The subtree of a branch node contains the node itself, the nodes that are its successors in some path, and the edges between these nodes. The value of the path from a root node to a branch node (m, n), also called the value of the branch node, is denoted $V(m, n)$. Root and isolated nodes have value zero. Paths with zero or negative value are not allowed.

Figure 4 depicts the node array with a possible collection of paths for the initial unwrapped phase in Fig. 1. The pixels are not shown, but the EO of Fig. 2 is marked. Each path can be traced by following the sequence of edges from its root node to its final node. Each node is marked with its value (the value of the path to it). The values are computed by using Eqs. (17–20) and expression (21) with the jump counts of Fig. 1. Each edge has value 1 if it reduces the magnitude of a jump count, -1 if it increases it, or 0 if it connects two boundary nodes. A path can contain some negative-value edges, provided that its total value is positive. The dotted arrow is an edge that is not currently in any path but can be added (as described in the next subsection).

B. Rules for Path Extension

The basic computational step of the algorithm is to revise the set of paths by adding an edge. The edge from (m, n) to (m', n'), if not already present, can be added if the following quantity is positive:

$$\Delta V = V(m, n) + \delta V(m, n; m', n') - V(m', n'). \quad (22)$$

This is the change in the value of (m', n') that is due to the new edge. If it is negative or zero, then the new path to (m', n') would have negative or zero value or would fail to improve upon the existing path. Therefore the edge is not added. If ΔV is positive, the edge is added and the collection of paths is revised to maintain its tree structure. There are three types of revision:

1. *Edge addition.* If (m', n') is a root or isolated node and the path to (m, n), if any, does not start at (m', n'), then the algorithm adds the edge and leaves existing edges unchanged. If both nodes are isolated, this starts a new tree. If (m', n') is a root, the values of all nodes in its subtree are increased by ΔV.
2. *Edge replacement.* If (m', n') is a branch node and the path to (m, n), if any, does not contain (m', n'), then the algorithm removes the existing edge to (m', n') and adds the new edge. The paths through (m', n') are changed to include the new edge, and the values in its subtree are increased by ΔV.
3. *Loop completion.* If the path to (m, n) contains (m', n'), then the new edge completes a loop. The algorithm applies the corresponding EO to the unwrapped phase. The EO changes the jump counts in the loop, thereby changing the values of the loop edges and of the paths that contain them. The values of some paths may become negative or zero, making them invalid. To account for this, all paths containing loop edges are removed and the values of the resulting isolated nodes are set to zero.

Figures 5–7 illustrate a sequence of revisions to the path collection of Fig. 4. The dotted edge of Fig. 4 is

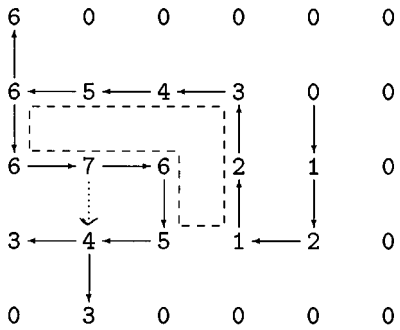

Fig. 5. Collection of paths after the type 1 revision. The new edge has been added, and the values have been changed accordingly. The dotted arrow indicates a type 2 revision.

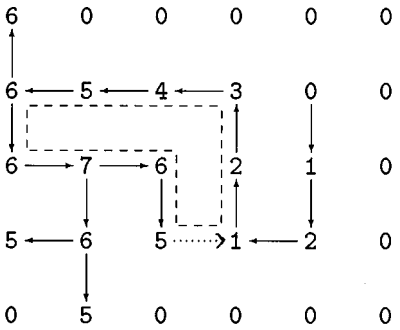

Fig. 6. Collection of paths after the type 2 revision. The new edge has been added, and the existing edge to its destination has been removed. The dotted arrow indicates a type 3 revision.

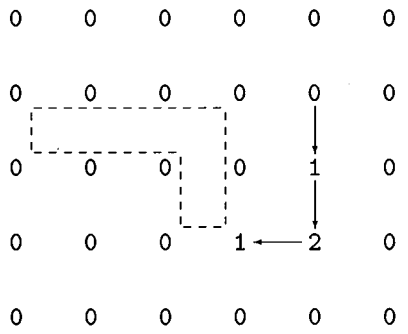

Fig. 7. Collection of paths after the type 3 revision. The loop formed by the new edge, and all paths containing loop edges, have been removed. The existing path leading to the loop has been retained.

added in Fig. 5, causing a type 1 revision. Since the new edge goes to a root node, the path on the right side of the array is prepended to all the paths on the left. The values of these paths are updated accordingly (increased by 1). The dotted edge of Fig. 5 is added in Fig. 6, causing a type 2 revision. The algorithm removes the existing edge that has the same destination node as that of the new edge. Again, the paths that are changed in this revision have their values updated (increased by 2). The dotted edge of Fig. 6 is added in Fig. 7, causing a type 3 revision. The new edge completes the loop corresponding to the EO of Fig. 2. The loop is removed, along with all paths containing loop edges. The path leading up to the loop is retained.

The algorithm alternates between extending paths and increasing values (in type 1 and 2 revisions) and removing paths, setting values to zero, and performing EO's (in type 3 revisions). In this manner it computes a sequence of EO's. If the collection of paths is such that ΔV is negative or zero for all node pairs, then no revisions are possible and the algorithm terminates. It is shown in Appendix B that this can happen only if no EO is possible and that the algorithm makes only a finite number of revisions before finding an EO (if possible) or terminating (if not). These facts, combined with the optimality of an arbitrary EO sequence, imply that the algorithm finds an optimal solution from any initial guess.

C. Implementation and Computational Cost

The MWD phase-unwrapping algorithm goes as follows. Compute the jump-count arrays **v** and **z** from the initial unwrapped phase by using Eqs. (7) and (8). Initialize the collection of paths to be empty, and initialize all values to zero. Repeatedly scan the set of node pairs, using Eq. (22) to test for new edges. As each new edge is found, revise the paths and the values as described in the previous subsection. After each loop completion, modify the jump-count arrays to reflect the EO. When a scan over all node pairs finds no new edges, compute the final unwrapped phase from the jump-count arrays, by using Eqs. (9), (10), and (4), and exit.

We have not found a formula or a model for the computational cost of the MWD algorithm, but a few general remarks on the subject can be made. Computational effort is dominated by actions on subtrees: After any edge addition or replacement in which the new edge connects to an existing tree, the values in a subtree are updated; after every loop completion, the edges in a subtree are removed, and the values are set to zero. The cost of such an action is proportional to the number of nodes in the subtree. Long curves of jumps, especially in high-weight regions, produce large trees because they allow the formation of long paths. A node near a jump curve may have its value updated many times and may be added to and removed from many different trees. Thus the most complicated phase-unwrapping problems, with many jumps to be changed, are also the most expensive.

The cost is affected by the manner in which node pairs are scanned. If many jumps can be removed quickly, the algorithm will form shorter paths and the remaining improvements will require less effort. Our implementation divides the scan into four subscans, which search for rightward, downward, leftward, and upward edges. This allows loops to be traced rapidly in many cases; for instance, a rectangular loop of any size is traced in at most two scans.

Although the MWD algorithm can find an optimal solution directly from the phase data, its cost is reduced by first using a simple preliminary unwrapping algorithm to remove fringes. In the example of Section 4, the preliminary algorithm starts at the center of the image and expands the unwrapped region preferentially through smooth areas. At each step a pixel adjoining the unwrapped region is added to it; the pixel is chosen so that the phase difference connecting it to the unwrapped region is as small as possible. This algorithm removes fringes easily but can produce jumps where neighboring pixels are unwrapped by different routes.

4. EXAMPLE

We demonstrate the performance of our algorithms on an IFSAR interferogram, calculated from a European Remote Sensing Satellite-1 (ERS-1) image pair taken at Fort

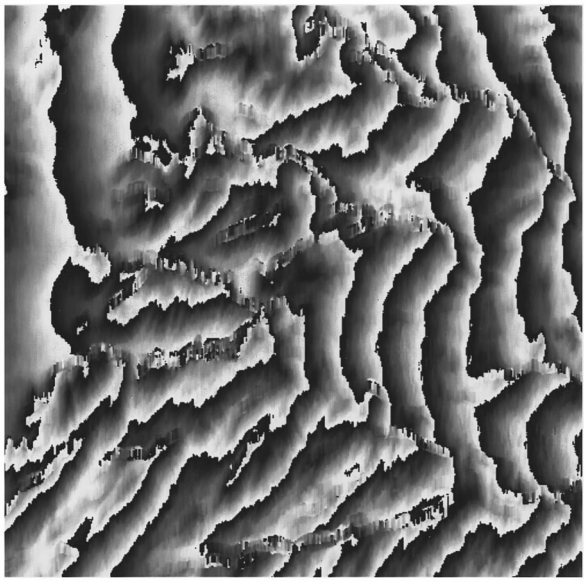

Fig. 8. Phase image computed from an ERS-1 interferometric image pair taken at Fort Irwin, California.

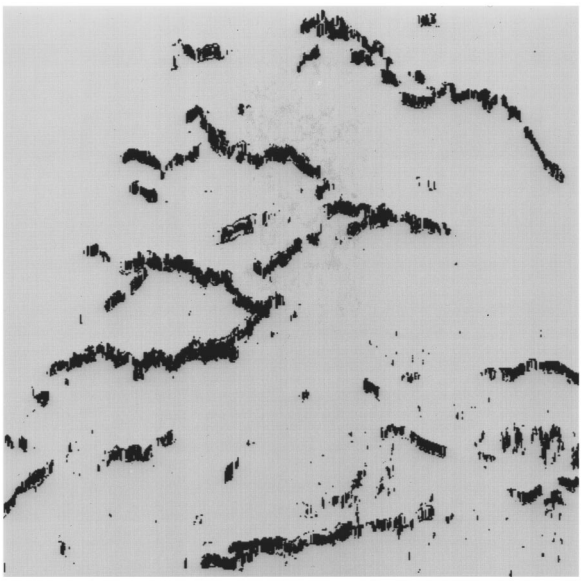

Fig. 9. Thresholded correlation map computed from the ERS-1 image pair. Dark pixels indicate where the correlation is less than 38%.

Fig. 10. Unwrapped phase computed from Fig. 8 by using the preliminary algorithm.

Irwin, California. The dimensions of the interferogram are 500 × 500 pixels. The phase is presented as a gray-scale image in Fig. 8. There are numerous regions of layover in which fringes are broken or merged. A thresholded version of the correlation map is presented in Fig. 9; it is dark where the correlation coefficient is below 38% and light elsewhere. The regions of layover have low correlation. There are places, in the upper right-hand corner and at the bottom, where a layover region comes close to the boundary of the image but does not reach it. The correct unwrapped phase must avoid connecting the layover region to the boundary with jumps.

The result of running the preliminary algorithm on the interferogram is presented as a gray-scale image in Fig. 10. The unwrapped phase has been inverted, so that high-elevation features are light colored, and a horizontal tilt has been removed. The phase is unwrapped smoothly in most of the image, but there are prominent jump curves running between layover regions.

The unwrapped phase found by the unweighted MD algorithm, with use of the preliminary solution as the initial guess, is presented in Fig. 11. The long jump curves have been removed, but short curves still connect the boundary to the nearby layover regions. The true discontinuities that are due to layover create lines of jumps in the unwrapped phase, giving it a somewhat artificial appearance. To reconstruct a profile with true discontinuities correctly, the placement of jumps must be guided.

Fig. 11. Unweighted minimum-discontinuity unwrapped phase computed by using the preliminary solution as an initial guess.

Fig. 12. Minimum-weighted-discontinuity unwrapped phase computed by using the preliminary solution as an initial guess. Jumps between two high-correlation pixels have weight 128; other jumps have weight 1.

The unwrapped phase found by the MWD algorithm, with use of the preliminary solution as the initial guess, in presented in Fig. 12. An edge has weight 128 if both its pixels are above the correlation threshold (light pixels in Fig. 9); otherwise its weight is 1. The nonzero weight in low-quality regions forces the algorithm to find a plausible unwrap solution in these regions. The jumps within the layover regions follow the layover curves more closely than in the unweighted MD solution. The erroneous jumps to the boundary have been eliminated. Thus the MWD solution is the most reasonable of the unwrapped phases in this example.

5. CONCLUSIONS

The goal of two-dimensional phase unwrapping is to recover an interferometrically measured profile as accurately as possible, avoiding errors that are due to noise and misplaced jumps. A reasonable method is to minimize the weighted sum of discontinuities, where the weights are chosen to allow discontinuities in their proper locations and penalize them elsewhere. The weighted minimum-norm algorithms of Refs. 3 and 4 minimize a criterion based on weighted discontinuity, but the solution is typically not congruent with the phase data. Algorithms based on residue matching[6–9] confine jumps to explicit branch cuts connecting the residues. These algorithms produce congruent solutions but cannot easily be generalized to include weighting. In this paper we have developed an algorithm that finds a congruent solution achieving the exact minimum of the weighted-discontinuity sum. It works by locating regions in which a change of wrap count removes jumps, generalizing the fringe-removal method of Ref. 5. The computational cost is made tolerable by a carefully structured search.

The MWD solution depends on the weights, which must themselves be computed from the data according to some rule. If too many of the weights are high, the algorithm may make poor residue pairings because low-weight paths connecting the residues are not available; if too many weights are low, the jumps may not match the true discontinuities. The threshold in our example was chosen for good results; a fully automatic implementation of the MWD algorithm must include a method of computing the weights. In other words, with the algorithm available to compute the optimal solution with respect to any criterion in the weighted-discontinuity class, the remaining problem is to choose the best criterion in that class.

APPENDIX A: OPTIMALITY OF ELEMENTARY OPERATIONS

We use two lemmas to prove that the unwrapped phase is optimal if no EO is possible. First, we show that if the unwrapped phase can be improved at all, then it can be improved by adding 1 to a set of wrap counts.

Lemma 1. Suppose that the phase data $\boldsymbol{\phi}$ and the wrap-count arrays $\mathbf{c}$ and $\bar{\mathbf{c}}$ obey

$$E(\bar{\mathbf{c}};\ \boldsymbol{\phi}) < E(\mathbf{c};\ \boldsymbol{\phi}).$$

Then there exists an array $\delta\mathbf{c}$, whose elements are all 0 or 1, such that

$$E(\mathbf{c} + \delta\mathbf{c};\ \boldsymbol{\phi}) < E(\mathbf{c};\ \boldsymbol{\phi}).$$

Proof: Let $\Delta\mathbf{c} \equiv \bar{\mathbf{c}} - \mathbf{c}$ be the known improving change to the wrap count. Since E is not changed by the addition of a constant to all wrap counts, we can assume that $\Delta c_{mn} \geqslant 0$ for all (m, n). Define $K = \max_{m,n} \Delta c_{mn}$. Now consider the sequence of partially changed wrap counts

$$c^k_{mn} = c_{mn} + \min(k,\ \Delta c_{mn}). \tag{A1}$$

In particular, $\mathbf{c}^0 = \mathbf{c}$ and $\mathbf{c}^K = \bar{\mathbf{c}}$. Each array $\mathbf{c}^k$ determines jump-count arrays $\mathbf{v}^k$ and $\mathbf{z}^k$ according to Eqs. (7) and (8). Define the wrap-count increments

$$\delta\mathbf{c}^k \equiv \mathbf{c}^k - \mathbf{c}^{k-1}. \tag{A2}$$

By construction, $\delta\mathbf{c}^k$ contains only zeros and ones; so we can prove the lemma by showing that $E(\mathbf{c} + \delta\mathbf{c}^k;\ \boldsymbol{\phi}) < E(\mathbf{c};\ \boldsymbol{\phi})$ for some k. The improvement that is due to $\Delta\mathbf{c}$ is the sum of the effects of successive increments:

$$\begin{aligned}\Delta E &\equiv E(\bar{\mathbf{c}};\ \boldsymbol{\phi}) - E(\mathbf{c};\ \boldsymbol{\phi}) \\ &= E(\mathbf{c}^K;\ \boldsymbol{\phi}) - E(\mathbf{c}^0;\ \boldsymbol{\phi}) \\ &= \sum_{k=1}^{K} [E(\mathbf{c}^k;\ \boldsymbol{\phi}) - E(\mathbf{c}^{k-1};\ \boldsymbol{\phi})].\end{aligned} \tag{A3}$$

Since $\Delta E < 0$ by hypothesis, there is some k for which $E(\mathbf{c}^k;\ \boldsymbol{\phi}) - E(\mathbf{c}^{k-1};\ \boldsymbol{\phi}) < 0$. The effect of applying $\delta\mathbf{c}^k$ to the original wrap count is

$$\begin{aligned}\delta E^k &\equiv E(\mathbf{c} + \delta\mathbf{c}^k;\ \boldsymbol{\phi}) - E(\mathbf{c};\ \boldsymbol{\phi}) \\ &= E(\mathbf{c}^0 + \mathbf{c}^k - \mathbf{c}^{k-1};\ \boldsymbol{\phi}) - E(\mathbf{c}^0;\ \boldsymbol{\phi}).\end{aligned} \tag{A4}$$

The lemma is proved if we can show that

$$\delta E^k \leqslant E(\mathbf{c}^k;\ \boldsymbol{\phi}) - E(\mathbf{c}^{k-1};\ \boldsymbol{\phi}) \tag{A5}$$

or equivalently

$$\begin{aligned}0 &\leqslant E(\mathbf{c}^k;\ \boldsymbol{\phi}) - E(\mathbf{c}^{k-1};\ \boldsymbol{\phi}) \\ &\quad - E(\mathbf{c}^0 + \mathbf{c}^k - \mathbf{c}^{k-1};\ \boldsymbol{\phi}) + E(\mathbf{c}^0;\ \boldsymbol{\phi}) \\ &= S^k_v + S^k_z,\end{aligned} \tag{A6}$$

where, from Eq. (16),

$$\begin{aligned}S^k_v = \sum_{(m,n)\in\mathscr{V}} w^v_{mn}(&|v^k_{mn}| - |v^{k-1}_{mn}| + |v^0_{mn}| \\ &- |v^0_{mn} + v^k_{mn} - v^{k-1}_{mn}|),\end{aligned} \tag{A7}$$

$$\begin{aligned}S^k_z = \sum_{(m,n)\in\mathscr{Z}} w^z_{mn}(&|z^k_{mn}| - |z^{k-1}_{mn}| + |z^0_{mn}| \\ &- |z^0_{mn} + z^k_{mn} - z^{k-1}_{mn}|).\end{aligned} \tag{A8}$$

We prove inequality (A6) by showing that each term of the sums in Eqs. (A7) and (A8) is nonnegative. The construction of the increments guarantees that $(v^0_{mn}, v^1_{mn}, \ldots, v^K_{mn})$ is a monotone sequence: If $\Delta c_{mn} > \Delta c_{m-1,n}$, then v^k_{mn} is zero for $k = 0, 1, \ldots, \Delta c_{m-1,n}$, then increases until $k = \Delta c_{mn}$, then remains constant. Similarly, if $\Delta c_{mn} < \Delta c_{m-1,n}$, the sequence is monotone

decreasing. The same reasoning proves that $(z_{mn}^0, z_{mn}^1, \ldots, z_{mn}^K)$ is monotone. So it suffices to show that

$$|a - b + c| \leq |a| - |b| + |c|$$

$$\text{for any } a, b, c \text{ such that } a \geq b \geq c \text{ or } a \leq b \leq c. \tag{A9}$$

This can be seen as follows. On account of monotonicity, $b = \gamma a + (1 - \gamma)c$ for some $\gamma \in [0,1]$. Hence $a - b + c = (1 - \gamma)a + \gamma c$, and by the convexity of the absolute value function,

$$|b| = |\gamma a + (1 - \gamma)c| \leq \gamma|a| + (1 - \gamma)|c|. \tag{A10}$$

Combining these formulas and invoking convexity again, we obtain

$$\begin{aligned} |a - b + c| &= |(1 - \gamma)a + \gamma c| \\ &\leq (1 - \gamma)|a| + \gamma|c| \\ &= |a| + |c| - [\gamma|a| + (1 - \gamma)|c|] \\ &\leq |a| - |b| + |c|. \end{aligned} \tag{A11}$$

This proves inequality (A9), from which the lemma follows.

The array $\delta\mathbf{c}$ divides the image into a set of zeros and a set of ones, but these sets need not be connected. We must prove that there is a partition into two connected sets of pixels such that the unwrapped phase is improved by incrementing one of the sets.

Lemma 2. Suppose that the phase data $\boldsymbol{\phi}$ and the wrap-count arrays $\mathbf{c}$ and $\delta\mathbf{c}$ are such that

$$E(\mathbf{c} + \delta\mathbf{c}; \boldsymbol{\phi}) < E(\mathbf{c}; \boldsymbol{\phi})$$

and that all elements of $\delta\mathbf{c}$ are 0 or 1. Then there exists an array $\epsilon\mathbf{c}$, in which all elements are 0 or 1, with the set of zeros and the set of ones both connected, such that

$$E(\mathbf{c} + \epsilon\mathbf{c}; \boldsymbol{\phi}) < E(\mathbf{c}; \boldsymbol{\phi}).$$

Proof. The boundary of the set of 1 elements in the $\delta\mathbf{c}$ array can be decomposed into a disjoint union of partitions, each of which divides the image into two connected sets. Associated with each partition is the operation that increments all pixels on its 1 side, thereby changing the jump counts on the partition and leaving all other jump counts unchanged. Because the partitions do not overlap, the change in the error that is due to adding $\delta\mathbf{c}$ to the wrap-count array is the sum of the changes that are due to the operations. But adding $\delta\mathbf{c}$ reduces the error, so there must be an operation that reduces the error. Its array of wrap-count changes is the $\epsilon\mathbf{c}$ of the lemma.

The two lemmas imply that if any improving change to an unwrapped phase is possible, then an EO is possible.

APPENDIX B: OPTIMALITY OF PATH EXTENSION

We found in Appendix A that the unwrapped phase is optimal if no EO's are possible. Hence the path-extension algorithm finds an optimal solution if it has these properties: (1) If no EO is possible, the algorithm terminates in finite time (after a finite number of path revisions); and (2) if any EO's are possible, it finds one of them in finite time. Property (1) is true because if no EO is possible, then only path revisions of type 1 or 2 can happen, so the algorithm cannot reduce the value of any node and must increase the value of at least one node in each revision. This can be done only finitely many times, because all node values are upper bounded by the sum of all positive edge values. To prove property (2), we start by showing that if the search terminates, then no EO is possible. At termination the value array satisfies the inequality

$$V(m, n) + \delta V(m, n; m', n') \leq V(m', n') \tag{B1}$$

for all pairs of neighboring nodes (m, n) and (m', n'); if it did not, an edge could be added. Consider a loop of L nodes with coordinate sequence $((m_0, n_0), (m_1, n_1), \ldots, (m_L, n_L))$, where $m_L = m_0$ and $n_L = n_0$. Because each node in the loop is a neighbor of its successor, it follows from inequality (B1) that

$$V(m_i, n_i) + \delta V(m_i, n_i; m_{i+1}, n_{i+1}) \leq V(m_{i+1}, n_{i+1}) \tag{B2}$$

for $i = 0, \ldots, L - 1$. Summing both sides of this inequality from $i = 0$ to $L - 1$, we have

$$\sum_{i=0}^{L-1} V(m_i, n_i) + \sum_{i=0}^{L-1} \delta V(m_i, n_i; m_{i+1}, n_{i+1}) \leq \sum_{i=1}^{L} V(m_i, n_i). \tag{B3}$$

Because nodes 0 and L are the same, the sums over V cancel, and so

$$\sum_{i=0}^{L-1} \delta V(m_i, n_i; m_{i+1}, n_{i+1}) \leq 0. \tag{B4}$$

This means precisely that the loop does not represent an EO. Since the loop is chosen arbitrarily and any EO is represented by a loop, this shows that when the algorithm terminates, no EO is possible. It remains to be shown that if any EO's are possible, one of them is found in finite time. By the argument in the proof of property (1), the algorithm adds only a finite number of edges before forming a loop; since it cannot terminate, the next edge must form a loop, thus finding an EO. This proves property (2), establishing optimality of the algorithm.

ACKNOWLEDGMENTS

Mark Pritt reviewed the entire manuscript and suggested changes to the figures. Dennis Ghiglia pointed out the connection between this paper's optimality criterion and the minimum L^p-norm criterion. The preliminary unwrapping algorithm is based on a manuscript by Michael Roth. Daniel Wahl, Terry Calloway, and Jack Jakowatz contributed valuable insights. The anonymous referees pointed out many improvements and clarifications to the presentation. The ERS-1 interferogram was prepared by Dick Shead.

This work was supported by the U.S. Department of Energy under contract DE-AC04-94AL85000. Sandia National Laboratories is a multiprogram laboratory oper-

ated by Sandia Corporation, a Lockheed Martin company, for the U.S. Department of Energy.

REFERENCES

1. H. A. Zebker and R. M. Goldstein, "Topographic mapping from interferometric synthetic-aperture radar observations," J. Geophys. Res. **91**, 4993–4999 (1986).
2. C. M. Vest, *Holographic Interferometry* (Wiley, New York, 1979).
3. D. C. Ghiglia and L. A. Romero, "Robust two-dimensional weighted and unweighted phase unwrapping that uses fast transforms and iterative methods," J. Opt. Soc. Am. A **11**, 107–117 (1994).
4. D. C. Ghiglia and L. A. Romero, "Minimum L^p-norm two-dimensional phase unwrapping," J. Opt. Soc. Am. A **13**, 1999–2013 (1996).
5. Q. Lin, J. F. Vesecky, and H. A. Zebker, "Phase unwrapping through fringe-line detection in synthetic aperture radar interferometry," Appl. Opt. **33**, 201–208 (1994).
6. R. M. Goldstein, H. A. Zebker, and C. L. Werner, "Satellite radar interferometry: two-dimensional phase unwrapping," Radio Sci. **23**, 713–720 (1988).
7. J. M. Huntley, "Noise-immune phase unwrapping algorithm," Appl. Opt. **28**, 3268–3270 (1989).
8. J. R. Buckland, J. M. Huntley, and S. R. E. Turner, "Unwrapping noisy phase maps by use of a minimum-cost-matching algorithm," Appl. Opt. **34**, 5100–5108 (1995).
9. J. A. Quiroga, A. González-Cano, and E. Bernabeu, "Stable-marriages algorithm for preprocessing phase maps with discontinuity sources," Appl. Opt. **34**, 5029–5038 (1995).
10. H. A. Zebker and J. Villasenor, "Decorrelation in interferometric radar echoes," IEEE Trans. Geosci. Remote Sensing **30**, 950–959 (1992).