Minimal Surfaces Extend Shortest Path Segmentation Methods to 3D

Leo Grady, *Member*, *IEEE*

Abstract—Shortest paths have been used to segment object boundaries with both continuous and discrete image models. Although these techniques are well defined in 2D, the character of the path as an object boundary is not preserved in 3D. An object boundary in three dimensions is a 2D surface. However, many different extensions of the shortest path techniques to 3D have been previously proposed in which the 3D object is segmented via a collection of shortest paths rather than a minimal surface, leading to a solution which bears an uncertain relationship to the true minimal surface. Specifically, there is no guarantee that a minimal path between points on two closed contours will lie on the minimal surface joining these contours. We observe that an elegant solution to the computation of a minimal surface on a cellular complex (e.g., a 3D lattice) was given by Sullivan [47]. Sullivan showed that the discrete minimal surface connecting one or more closed contours may be found efficiently by solving a Minimum-cost Circulation Network Flow (MCNF) problem. In this work, we detail why a minimal surface properly extends a shortest path (in the context of a boundary) to three dimensions, present Sullivan's solution to this minimal surface problem via an MCNF calculation, and demonstrate the use of these minimal surfaces on the segmentation of image data.

Index Terms—3D image segmentation, minimal surfaces, shortest paths, Dijkstra's algorithm, boundary operator, total unimodularity, linear programming, minimum-cost circulation network flow.

1 Introduction

Shortest path algorithms on weighted graphs have found many applications in computer vision, including segmentation [34], [16], centerline-finding [7], video summarization [38], robot navigation [11], perceptual grouping [13], solving PDEs [51], and optical flow [48]. Since computer vision techniques have been increasingly applied to 3D data in the context of video sequences or medical acquisitions, researchers have looked for 3D extensions of many conventional 2D techniques. Several of the proposed extensions of the shortest path techniques to 3D have employed a network of paths that are used to define the surface of the 3D object. Unfortunately, there is no guarantee that these shortest paths will lie on the minimal surface or that a dense enough sampling of paths will approach the true minimal surface, even in the limit. Instead of employing a network of paths in 3D, we observe that the minimal surface may be solved for directly using the elegant solution provided by Sullivan [47] in the context of discrete differential geometry.

Shortest paths are used as object boundaries in several 2D image segmentation algorithms, most notably in the popular intelligent scissors/live wire algorithm [34], [16]. The intelligent scissors algorithm treats the image as a graph that is weighted to reflect intensity changes and inputs two points from a user along an object boundary. These points are then used to define the endpoints for a shortest path computation. The shortest path is then viewed as a piece of the object boundary that may be extended by the placement of additional points. In 2D intelligent scissors, shortest paths provide minimal boundaries of the segmented object.

A sustained interest in intelligent scissors/live wire has prompted several researchers to pursue a 3D extension of this segmentation technique. However, it was noted as recently as 2006 by Armstrong et al. [5] that "There is no straightforward extension of Live Wire to surfaces." Although various different extensions to intelligent scissors have been proposed, the common theme is that intelligent scissors is fundamentally a path-based technique, and therefore, 3D extensions have focused on the reconstruction of 3D surfaces from networks of paths.

Falcão and Udupa [15] separate the 3D image into slabs for which the object is assumed to have constant genus. A user then employs the shortest path segmentation on several cross sections (with some constraints) which are used for surface reconstruction. This technique was extended [23] to include a more sophisticated choice of point ordering within the cross sections for connection via shortest paths. Schenk et al. [41], [42] employ intelligent scissors to obtain a series of closed contours on key slices, between which a surface is fit using shape-based interpolation. Salah and Bartz [40] use intelligent scissors to find a 2D segmentation in one slice and then use a sophisticated propagation of the control points to subsequent slices, upon which 2D intelligent scissors is subsequently run. Knapp et al. [26] employ orthogonal cross sections to reconstruct the surface, which may have a nontrivial topology. These cross sections are obtained via shortest paths. König and Hesser [29] find the shortest path connecting three clicked points, producing a closed surface patch. This patch is then filled in using a

• *The author is with Siemens Corporate Research, Department of Imaging and Visualization, 755 College Rd., East Princeton, NJ 08540. E-mail: leo.grady@siemens.com.*

Manuscript received 6 Oct. 2007; revised 11 July 2008; accepted 18 Nov. 2008; published online 2 Dec. 2008.
Recommended for acceptance by R. Zabih.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2007-10-0675.
Digital Object Identifier no. 10.1109/TPAMI.2008.289.

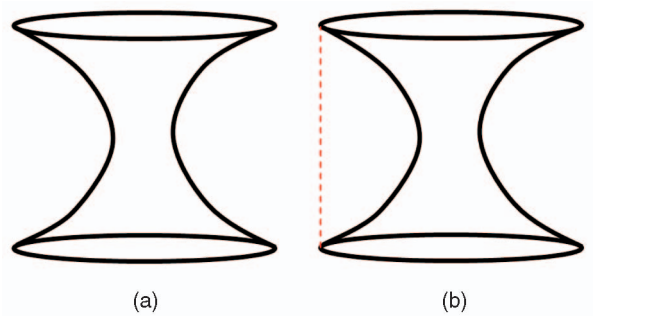

Fig. 1. An example for which shortest paths joining two contours will not lie on the minimal surface connecting the contours. (a) The catenoid is the classical solution to the minimal surface problem joining two closed contours at different elevations. (b) A shortest path (red) joining two points on the closed contours does not lie on the surface of the catenoid.

network of shortest paths and this process of producing and filling surface patches is repeated until a closed surface is obtained. Finally, although not a direct extension of intelligent scissors to 3D, Li et al. [31] propose a method that will produce a minimal surface, provided that the minimal surface satisfies some constraints. These constraints require that the surface is simply connected and possesses the property termed by Li et al. [31] as "terrain-like" (a ray cast from the central axis/point of the object intersects the surface only once). We note that such an object is also known as "star shaped." Their primary objective was to provide a framework in which a minimal surface could be obtained that also satisfied secondary objectives, such as smoothness or an inner/outer relationship of two surfaces.

One may also view the use of shortest paths in the continuum as a continuous counterpart of the intelligent scissors/live wire segmentation paradigm [14]. In contrast to discrete shortest paths, the continuous shortest paths are computed with the fast marching method [44]. This continuous shortest path segmentation approach has also been extended to 3D segmentation by Ardon et al. [3], [4]. In the first work [3], a series of shortest paths were used to join a point (or closed contour) with a closed contour, which were then interpolated to form a surface. The second work [4] finds the surface via a level-set problem constrained such that the network of shortest paths lies on the surface. As with discrete methods, these approaches construct a surface from a network of shortest paths, rather than extending the shortest path problem of 2D to the minimal surface problem of 3D.

The approaches reviewed above operate under the assumption that the 3D surface upon which the shortest paths are found will be minimal. However, it is possible that shortest paths joining two closed contours will not lie on the minimal surface connecting these two contours. For example, it is well known that, in continuous euclidean space, the minimal surface connecting two closed contours at different elevations is a catenoid. However, the shortest paths joining any two points on these contours will not lie on the surface of the catenoid. Fig. 1 illustrates this situation. This counterexample demonstrates that the technique of using shortest paths to find minimal surfaces can never be guaranteed to produce the correct result, even by using paths of arbitrary density. Therefore, instead of adopting the use of shortest paths to find the minimal surface, we observe that the minimal surface may be solved for directly using Sullivan's method. Finally, in addition to the problems mentioned above, we note that another problem with using shortest paths to define a surface in 3D is that 1D paths are of the wrong dimension type to bound 3D objects (i.e., application of the boundary operator to a 3D object produces a 2D surface).

1.1 Characterization of Minimal Path/Surface Problems

Minimal surfaces have been studied extensively outside of computer vision in the fields of geometric measure theory and variational calculus [33]. However, these fields focus on finding minimal surfaces in a space that is continuous and euclidean, rather than on the discrete, weighted lattices that arise naturally in computer vision. In order to avoid confusion with this body of existing literature, we will employ the term **minimum-weight surfaces** to refer to minimal surfaces defined on the space of discrete, weighted lattices. Although lattices are the most relevant structure for computer vision, the techniques presented here also apply to more general discrete structures.

The shortest path problem with nonnegative edge weights requires specification of additional constraints in order to avoid the solution of a null path. Additional constraints are typically included in one of two forms.

1. The shortest path is required to enclose one specified region of space while excluding a second specified region (i.e., separating the two regions).
2. The shortest path is required to have a specified boundary (i.e., endpoints).

These methods for specifying constraints for the minimal path problem will be referred to as **Type I** and **Type II** constraints, respectively. Type I constraints give rise to a source separation problem that is solved efficiently on discrete spaces by Ford and Fulkerson's max-flow/min-cut algorithm [37]. Type II constraints give rise to a source connection problem that is solved efficiently on discrete spaces by Dijkstra's algorithm [37]. Note that if negative edge weights were permitted (e.g., derived from a probabilistic formulation of the segmentation problem), then no constraints of the above types would be necessary to avoid the null solution. However, finding the global minimum of the path/cut problem with negative edge weights on an arbitrary graph is, in general, difficult (see [28] for a survey of this problem in the context of computer vision). It should be noted that the minimum-cut problem with negative weights may be solved exactly in polynomial time on planar graphs (2D) [22], [1], [46]. Finally, we recognize that the value of targeting energies for which a global optimum exists is a point of debate within the community. In this work, we simply note that there has been a sustained interest in finding a solution for the 3D extension of the short path problem (minimal surfaces) with nonnegative weights. Since a global minimum does exist for this problem and the previous heuristics reviewed above are not guaranteed to find it, we simply demonstrate that this problem may be solved using Sullivan's method.

Type I and Type II constraints are also necessary in the specification of minimum-weight surfaces in order to avoid

the null solution (given nonnegative edge weights). The minimum-weight surface problem with Type I constraints is equivalent to solving the max-flow/min-cut problem in 3D, which has known solutions in both continuous [2] and discrete spaces [37]. The problem of how to assign graph weights to best approximate the continuous solution to a Type I problem was studied in [10]. Type II constraints require that the minimum-weight surface has a prescribed boundary. Surface boundaries are always given by closed contours, an example of which is the wire rim giving the boundary of a soap bubble. The minimum-weight surface problem with Type II constraints has been studied in continuous space as Plateau's problem (e.g., [33]), but less attention has been paid to finding minimal surfaces with Type II constraints in discrete space. There are only two works that we are aware of that address the problem of solving for minimal surfaces with Type II constraints on a discrete space (although it should be noted that the primary goal of both of these works is the approximation of the continuous solution). The first of these papers is the work of Kirsanov and Gortler, who considered the limited case in which it is possible to translate Type II constraints into Type I constraints [25]. Unfortunately, for most cases of interest to computer vision, their formulation is not applicable since the prescribed boundary must lie on the borders of the volume. Sullivan [47] addresses the problem of approximating continuous minimal surface solutions with discrete cell complexes and recognized that the problem could be viewed as an instance of Minimum-cost Circulation Network Flow. Although Sullivan's interest was in the approximation of continuous minimal surfaces, we demonstrate the utility of Sullivan's technique for computing minimum-weight surfaces in the context of image segmentation. These computational details will be discussed more thoroughly in Section 2.3.

Since Type II constraints govern shortest path problems on 2D weighted graphs, our focus is on the solution of the Type II constrained minimum-weight surface problem on 3D weighted graphs. Although it may be argued that object surfaces are in some sense continuous, the standard reconstruction of image data as a discrete lattice has led many researchers to adopt discrete algorithms to analyze the image data. Sullivan and Kirsanov/Gortler both treated the case of continuous minimal surfaces with Type II constraints by approximation via a cell complex. However, given image data arranged in a 3D lattice and the adoption of surface weights from the image content, it is unclear how to formulate or solve the weighted continuous minimal surface problem in this context. Consequently, by adopting the discrete formulation on a 6-connected lattice, for which formulation and solution may be done efficiently, one could view the resulting minimal surfaces as approaching the continuous minimal surfaces in the limit of small grid size and the presence of an ℓ_1 norm.

Neither Type I nor Type II constraints have priority over the other, i.e., different applications call for different constraint types. As evidence for this position, we note the enduring interest in both (2D) graph cuts [9] and intelligent scissors [34], [16], despite the fact that graph cuts apply Type I constraints and intelligent scissors applies Type II constraints to the shortest path problem. Additionally, our goal is not to argue that minimum-weight surfaces are the best tool for 3D segmentation, taking the position that graph cuts have already established interest in this problem (albeit with Type I constraints). In fact, computer vision using minimal surfaces with Type I constraints has been heavily studied and applied [12], [39], [9]. Our intention is that the method for solving Type II minimal surface problems presented in this work will permit new applications of minimal surfaces to computer vision beyond segmentation. Additionally, previous attempts to extend intelligent scissors to 3D indicate a clear interest in the resolution of this problem. However, these previous treatments have not solved for the minimum-weight surface directly.

This paper is outlined as follows: In Section 2, we show how the extension of the shortest path problem to 3D leads to the minimum-weight surface problem. Sullivan's method for solving this minimum-weight surface problem is then presented. In Section 3, we demonstrate the application of this algorithm to synthetic 3D segmentation problems of various character, and then, apply the algorithm to real 3D data. Finally, Section 4 provides concluding remarks and suggests directions for further research.

A conference version of this work previously appeared in [21].

2 Method

In this section, we first outline a framework for viewing graph-based algorithms that produce boundaries in an image. This framework is based on the notion of a primal and dual lattice. Using this framework, we review the shortest path problem and present the minimum-weight surface problem. We conclude this section by giving an exposition of Sullivan's method [47] for solving the minimum-weight surface problem with Type II boundary conditions on a cellular complex (e.g., a 3D lattice) by reducing the computation to finding a Minimum-cost Circulation Network Flow [20].

2.1 Duality

The notion of **duality** has played a role in graph theory (and combinatorial topology) since the time of Poincaré, in which a dual graph was defined from a planar graph by replacing each facet (cycle) with a dual node and connecting two nodes if their respective facets shared an edge. In this example, the **primal** graph is defined as the initial planar graph and the **dual** graph is defined as the result of the duality operation. However, this duality operation is more general in the context of algebraic topology [30] and, in fact, depends on the dimensionality of the ambient space in which the graph is embedded. In fact, a clear understanding of duality has recently come to the forefront of numerical computing (see [32] for an excellent treatment). In a general context, the standard node/face duality may be thought of as the 2-dual, in the sense that nodes are isomorphic to 2D simplices (i.e., facets). For example, one could just as easily define a 1-dual of a graph by replacing each edge with a node and connecting nodes based upon whether or not their respective edges coterminated at a node (this 1-dual is sometimes called the *line graph*). In general, a traditional, d-dimensional dual is possible when each $(d-1)$-dimensional simplex is shared by exactly two d-dimensional

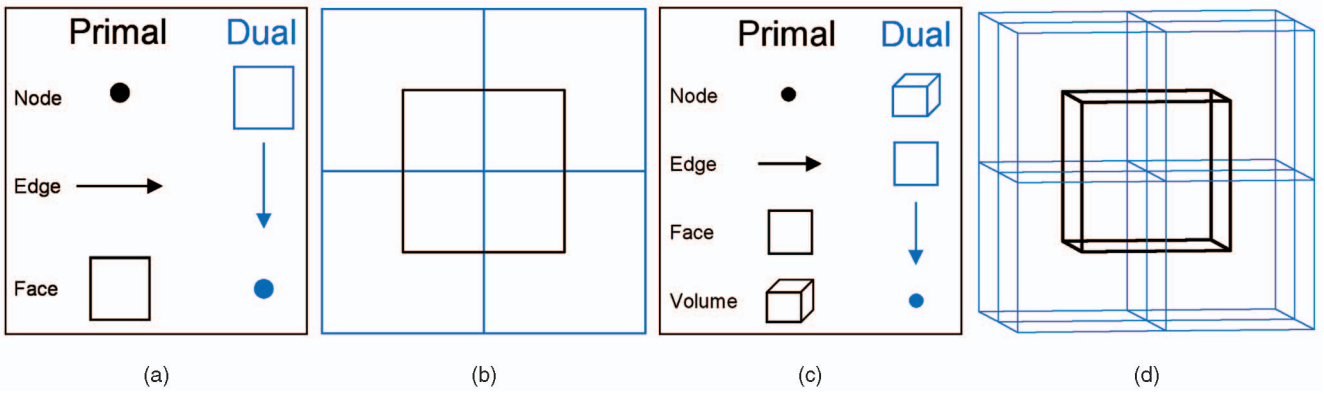

Fig. 2. Duality in two and three dimensions. (a) Primal and dual structures for a 2D, 4-connected lattice. (b) The primal (black) and dual (blue/gray) 2D lattices. (c) Primal and dual structures for a 3D, 6-connected lattice. (d) The primal (black) and dual (blue/gray) 3D lattices. Identifying image pixels with nodes in the primal lattice, *edges* of the dual lattice define a separating boundary of the pixels in 2D. In contrast, *facets* of the dual lattice are required to "box in" the voxels of the primal lattice in 3D. In 2D, the dual edges correspond to the "cracks" or "bels" between pixels often mentioned in the context of intelligent scissors/live wire [16].

simplices. Fig. 2 offers a picture of the relationship between the primal and dual complexes. In general, given a k-simplex in p dimensions, its dual will be a $(p-k)$ simplex [32]. Fig. 2 illustrates simplices and their duals for two and three dimensions.

The duality operation naturally produces an "outside facet" (representing the edge/facet that a path/surface would need to include in order to enclose pixels/voxels on the border of the data, see Fig. 2), but this outside facet may be given several interpretations. For example, one might assign the outside facet to a single node in the dual graph. Instead of this construction, we have chosen to subdivide the outside facet in the dual into cells for two reasons: 1) A lattice is easier to work with computationally and 2) the addition of weighted extra edges/facets on the outside allows for a higher cost to be assigned to a longer path/surface enclosing the border pixels/voxels.

Duality offers a convenient taxonomy of graph-based segmentation algorithms. There are two basic types of graph-based segmentation algorithm: implicit boundary algorithms and explicit boundary algorithms. Implicit boundary algorithms, such as graph cuts [9] or normalized cuts [45], label each node (pixel) as foreground/background and the boundary between them is implied by the labeling. In contrast, explicit boundary algorithms, such as intelligent scissors [34], [16], identify the boundary explicitly and a foreground/background labeling is given implicitly. The notion of duality provides a convenient way of viewing these segmentation algorithms in which one may treat the image data as existing at the nodes of the primal lattice. In this framework, implicit boundary algorithms operate on the primal lattice, while explicit boundary algorithms operate on the dual lattice. Therefore, the components of the dual lattice are used to "box-in" the pixels in the primal lattice (e.g., edges/paths in 2D, facets in 3D). However, as illustrated in Fig. 2, the corresponding dual lattice changes with dimension, while the primal lattice remains constant, prompting the need for greater modification of explicit boundary algorithms than implicit boundary algorithms when seeking extension to higher dimension.

Specifically, the popular graph cuts algorithm of [9] provides (Type I) fixed conditions at the nodes of the *primal* lattice and seeks a minimal cut (dual to a closed contour) between the source and sink nodes. In contrast, the intelligent scissors approach of [34], [16] fixes points along the boundary (Type II) in the *dual* lattice (sometimes, referred to as the "cracks" or "bels" between the pixels [16]) and seeks the minimal boundary (path) that includes these endpoints. When considering higher dimensional images, graph cuts extend naturally, because edge cuts are always dual to the $(p-1)$-surfaces that define the boundary of a p-dimensional set of voxels. However, the explicit boundary approach given by intelligent scissors must be redeveloped for each dimension. Specifically, since the shortest path algorithm used in the 2D case is inappropriate to find a bounding surface in 3D, a minimum-weight surface must be used.

2.2 Preliminaries

Before beginning the exposition, we fix our notation. For our present purposes, the primal and dual complexes will be 3D, 6-connected lattices. Define a 3D lattice **complex** [30] as consisting of a set $P = (V, E, F, C)$ with **vertices (nodes)** $v \in V$, **edges** $e \in E \subseteq V \times V$, **facets** $f \in F \subseteq E \times E \times E \times E$, and **cubes (volumes)** $c \in C \subseteq F \times F \times F \times F \times F \times F$ (since we will be dealing exclusively with 6-connected lattices). Let $n = |V|$ and $m = |E|$, where $|\cdot|$ denotes cardinality. Nodes, edges, facets, and cubes will all be indexed by single subscripts. A **weighting** assigns a value to each edge called a **weight**. The weight of an edge e_i is denoted by w_i and considered in this work to be nonnegative.

To each edge, we may assign an ordering of its constituent vertices, i.e., we associate with edge e_{ij} the pair $\{v_i, v_j\}$. Likewise, for each facet, we may associate an ordering of the vertices obtained by traversing its constituent edges in a closed cycle, i.e., $\{v_i, v_j, v_k, v_h\}$ in which $e_{ij}, e_{jk}, e_{kh}, e_{ih} \in E$. If the node ordering associated with an edge is in the same order in which the nodes are traversed on a facet, then we consider the facet and edge to have a *coherent* orientation [24]. Note that the notion of orientation and coherency used here for the degenerate simplices that comprise facets of a 6-connected lattice exactly matches the

standard parity definitions for nondegenerate simplices using the combining technique presented by Tonti [49].

We wish to stress that, although, in this work, we are treating the standard 6-connected lattice found in computer vision, the problem considered here (finding minimum-weight surfaces with Type II boundary conditions) may be solved using Sullivan's method for any orientable cell complex with trivial homology. Additionally, results are given in Appendix B that give more general conditions under which the minimum-weight surface problem can be solved using linear programming (e.g., when the complex is not cellular). A **cell complex** is defined by stating that for each dimension $k \leq d$, for maximum dimension d, there is a set C_k of k-dimensional cells (homeomorphic to balls), such that all the cells are disjoint and the boundary of any k-cell is the union of a finite number of lower dimensional balls. If each $(d-1)$-dimensional facet bordering a d-dimensional cell is part of the boundary of exactly two d-dimensional cells, then the minimum-weight hypersurface may be calculated, since such a complex is dual (isomorphic) to a graph in which d-dimensional cells are mapped to nodes and $(d-1)$-dimensional facets are mapped to edges. Consequently, the corresponding d-$(d-1)$-incidence matrix is totally unimodular.

Graph-based segmentation algorithms typically focus on partitioning a weighted graph, with weights given on the primal edges via a function of the image intensity, e.g.,

$$w_i = \exp\left(-\beta(I_j - I_k)^2\right) \quad \text{for } \{v_j, v_k\} \in e_i, \tag{1}$$

where I_j indicates the image (volume) intensity at voxel v_j. Note that several other functions [8] or features (e.g., color, texture response) have also been used to set edge (facet) weights.

2.2.1 Shortest Paths

The minimum-path problem may be viewed as the solution to the optimization problem

$$\min_y Q(y) = \sum_i w_i y_i, \tag{2}$$

where y_i represents an indicator vector on the set of (dual) edges, with w_i representing the weight of the corresponding primal edge, $y_i = 1$ indicating that edge e_i belongs to the path, and $y_i = 0$ indicating that edge e_i does not belong to the path.

In the absence of constraints, the solution of (2) yields the vector $y_i = 0\ \forall e_i \in E$, since all weights are nonnegative. Type I constraints may be introduced by specifying disjoint node subsets that must appear in separate connected components if the edges in the computed path are removed from the complex. Type II constraints may be introduced by specifying endpoints for the path. An algebraic formulation of Type II constraints is given by

$$Ay = p, \tag{3}$$

where p is a vector of all zeros except for a $p_s = 1$ and $p_t = -1$, for endpoints $\{v_s, v_t\}$. The matrix A is the node-edge incidence matrix

$$A_{v_k, e_{ij}} = \begin{cases} +1 & \text{if } k = i, \\ -1 & \text{if } k = j, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

The node incidence matrix in (3) plays the role of the *boundary operator* [30]. In this case, the boundary operator inputs an edge path (indicated by y) and returns the nodal boundary of that path (fixed by p). In general, the boundary operator inputs the indicator function of a complex and outputs an indicator function of its boundary. Therefore, use of the boundary operator in (3) allows us to fix the path boundary and succinctly expresses the Type II constraints for the minimal path problem. We note that this formulation of the minimal path problem is not new. For example, Papadimitriou and Steiglitz [37] establish the minimal path problem as the optimization of (2) with respect to (3) and proceed to derive Dijkstra's algorithm as a particular optimization of these equations.

2.2.2 Minimum-Weight Surfaces

In order to increase the dimensionality of the shortest path formulation, we now pass from minimal paths to minimum-weight surfaces. Fortunately, the dimensionality of the minimal path problem may be increased simply by using the dimension-appropriate incidence matrix (boundary operator) and boundary vector p. This dimension-increased shortest path problem, therefore, asks the question: *Given the boundary of a 2D surface (i.e., a closed contour or series of closed contours), find the minimum-weight 2D surface with the prescribed boundary*. As anticipated in Section 1, this is the minimum-weight surface problem with Type II conditions.

In this dimension-increased problem, the incidence matrix (boundary operator) in question is the edge-facet incidence matrix defined as

$$B_{e,f} = \begin{cases} +1, & \text{if the edge borders the facet with coherent orientation,} \\ -1, & \text{if the edge borders the facet without coherent orientation,} \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Note that it is essential in the following development to include each facet twice in B with opposite orientation. Such a device is also present when solving the shortest path problem, since any path could be traversed in either direction. Instead of the lower dimension boundary vector p, we can now employ the vector r as a signed binary indicator vector of a closed contour with an associated ordering of vertices obtained via a traversal along the edges comprising the contour. Given a contour represented by an ordering of vertices $(a, b, c, \ldots, a)$ such that each neighboring pair of vertices is contained in the edge set, the contour may be represented with the vector

$$r_i = \begin{cases} +1, & \text{if the vertices comprising edge } e_i \text{ are contained in the contour with coherent orientation,} \\ -1, & \text{if the vertices comprising edge } e_i \text{ are contained in the contour without coherent orientation,} \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

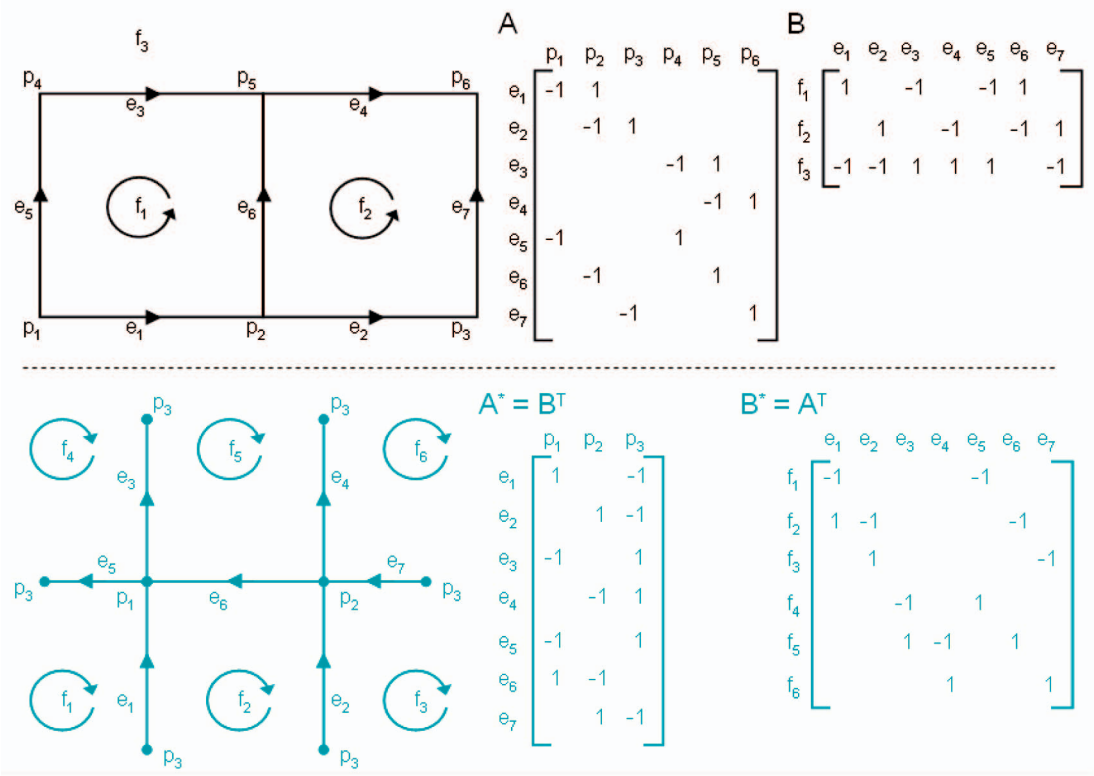

Fig. 3. Example of a small 2D lattice consisting of nodes, edges, and facets with various orientations, and the corresponding node-edge incidence matrix and edge-facet incidence matrix. The complex representing the 2-dual of the primal complex is also provided, illustrating that the incidence matrices are transposes of each other.

Therefore, the minimum-weight surface problem is

$$\begin{aligned} &\min_z Q(z) = \sum_i w_i z_i, \\ &\text{subject to} \quad Bz = r, \end{aligned} \tag{7}$$

where z is a nonnegative vector indicating whether or not a facet (in the dual complex) is present in the minimum-weight surface (and with what multiplicity) and w_i is meant to indicate the weights of a facet. Since the facets in the dual lattice correspond to edges in the primal lattice (where the image data are located), (1) may be used to produce the set of facet weights.

The minimum-weight surface problem described in (7) was extensively treated by Sullivan, who showed that a fast algorithm exists for its solution. In the next section, we present Sullivan's method for efficiently solving this problem. In Appendices A and B, we address in more detail when a solution to (7) exists and when it may be solved using generic linear programming (e.g., if the complex is not cellular, as required by Sullivan's method).

2.3 Minimum-Cost Circulation Network Flow

In order to arrive at a fast MCNF algorithm for solving the minimum-weight surface problem, Sullivan transforms the original problem in (7) into a second problem. Specifically, Sullivan observed that if the RHS of (7) is generated by *some* surface that does not self-intersect, represented by the vector z_0, then

$$r = Bz_0. \tag{8}$$

Therefore, the constraint in (7) may be rewritten as

$$Bz = Bz_0, \tag{9}$$

which reveals that

$$B(z - z_0) = 0, \tag{10}$$

$$Cx = z - z_0, \tag{11}$$

for some matrix C representing the null-space of B. The entire optimization problem of (7) may then be recast in terms of this new variable x as

$$\begin{aligned} &\min_x w^T(Cx + z_0), \\ &\text{s.t.} - Cx = z_0 - z \leq z_0, \\ &x \geq 0. \end{aligned} \tag{12}$$

The first inequality is true because z is nonnegative and the second inequality may be asserted since $Ck = 0$ for any constant vector k, when C represents an edge-node incidence matrix (see below). In other words, by assuming that there exists *an* integer solution to (7), the initial integer programming problem of (7) can be transformed into a second integer programming problem (12).

A boundaryless set of facets (i.e., enclosing a volume) represented by an indicator vector c_0 would necessarily take a zero if the boundary operator were applied, i.e., $Bc_0 = 0$. A basis set of such c_i vectors would then generate the columns of the null-space C. In the same manner as (17), C maps volumes to facets and is well known to be the **volume-facet** incidence matrix [32]. In the case of the 6-connected lattice in

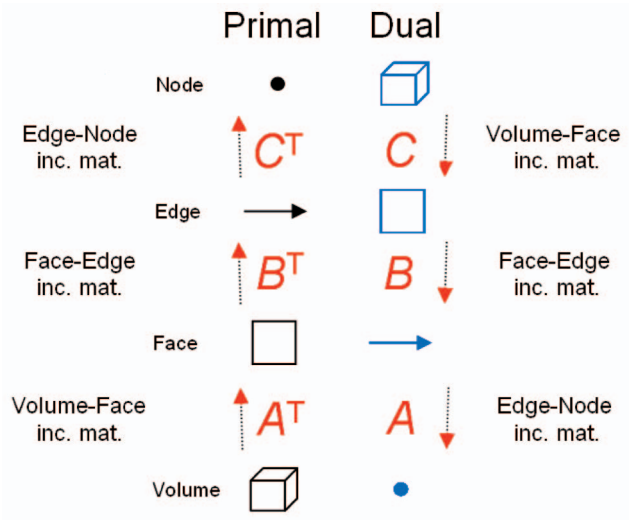

Fig. 4. Comparison of incidence matrices in the primal and dual 3D lattice. The incidence matrix mapping a d-dimensional complex to a $(d-1)$-dimensional complex in the dual lattice equals the transpose of the incidence matrix mapping the $(3-(d-1))$-dimensional complex to a $(3-d)$-dimensional complex in the primal lattice (e.g., the complex in Fig. 3). For this reason, it may be seen that the volume-facet incidence matrix of the dual lattice (in which we are working) equals the edge-node incidence matrix in the primal lattice. Since any edge-node incidence matrix is totally unimodular and total unimodularity is preserved under the transpose operation, the facet-edge incidence matrix in our dual lattice is also totally unimodular.

a 3D image, each volume is identified directly with a voxel. If the volumes are such that each facet is shared by exactly two volumes, incident with opposite orientation, then C can be seen to be the *edge-node* incidence matrix of another complex, specifically the *primal* complex, where each volume is identified with a node and each edge is identified with a facet (shared by two volumes in the manner than an edge is shared by two nodes). Note that the overrepresentation of each facet by two facets with opposite orientation generates more "volumes" in the null-space (enclosed by the facet with both orientations) which may be interpreted as extra nodes that are connected between each neighboring voxel. When each facet in the complex is shared by exactly two volumes as in the case of a 6-connected 3D lattice, then C is an edge-node incidence matrix and is, therefore, necessarily, totally unimodular [35]. Consequently, for our purposes, (12) is guaranteed to produce an integer solution (which also implies that LP is guaranteed to produce an integer solution when applied to the original integer programming problem of (7), see Appendix B). The identification of incidence matrices in the primal and dual lattices is illustrated in Fig. 4. Note that if the domain is more complicated than a 6-connected lattice (e.g., representing a 3-torus), the matrix C may not correspond to the node-edge incidence matrix of some primal graph. However, as mentioned above, we only treat the case of an orientable cell complex with trivial homology (specifically, the 6-connected lattice).

Although the total unimodularity of the constraint matrix C and integrality of z_0 guarantee that x is integer, there is no guarantee that x will be binary. For example, addition of any constant to x will still satisfy the constraints and give the same value for the objective function. However, even nontrivial cases may occur in which the solution x is nonbinary. For example, it would be possible to construct the weighting and contours just right such that the final solution would consist of two nested "bubbles" in which case the inner "bubble" could take nonbinary integer values. In practice, however, we follow Sullivan by solving the dual problem, and thus, never actually compute the values for x.

Although the problems defined by (12) may be solved with a generic LP solver, Sullivan went a step further to show that it is possible to apply a specialty solver to (12) that yields an even faster solution [47]. To see how Sullivan used a Minimum-cost Circulation Network Flow (MCNF) algorithm to solve (12), we begin by forming the dual LP problem to (12) in terms of variable f:

$$\begin{aligned} &\max_{f} \; -z_0^T f, \\ &\text{s.t. } C^T f \leq C^T w, \\ &\quad f \geq 0. \end{aligned} \tag{13}$$

We now decompose the solution f into the sum of two vectors $f = \tilde{f} - f'$ such that

$$C^T \tilde{f} \leq C^T w, \tag{14}$$

$$C^T f' = 0. \tag{15}$$

However, $\tilde{f} = w$ is the only value satisfying (14). This statement may be shown by recalling that C^T is the node-edge incidence matrix of the primal complex, which implies that $1^T C^T v = 0$ [6]. Consequently, if $\tilde{f}$ satisfies $C^T \tilde{f} = C^T w$, then there is no v satisfying $C^T(\tilde{f} + v) < C^T w$ since it would imply that there exists a v satisfying $C^T v < 0$.

Since $\tilde{f}$ is a constant, we may treat f' as the variable to be optimized over. Rewriting (13) as an optimization over f' gives us

$$\begin{aligned} &\max_{f'} \; z_0^T f', \\ &\text{s.t. } \; C^T f' = 0, \\ &\quad f' \leq \tilde{f} = w. \end{aligned} \tag{16}$$

This optimization problem asks us to find the maximum divergence-free flow (on the edges of the primal graph) that passes through the initial surface z_0 with capacities given by the graph weights. As (16) represents the dual to our original problem, it is the saturated set of primal edges (dual facets) that comprise the desired minimum-surface solution. It was recognized by Sullivan that the optimization described by (16) is the MCNF problem [47], which may be solved using a variety of existing algorithms [20]. The work of Kolmogorov [27] provides a fast method for applying the primal-dual MCNF algorithm of Ford and Fulkerson [17], [18] by eliminating the need for Dijkstra computations. The timings reported here are based on the application of Kolmogorov's method (and code available from his Website).

The MCNF approach represents the best-known algorithm for solving an LP problem of the form (12) and, consequently, for computing the 3D minimum-weight surface on real data, given one or more surface boundaries. We note that the MCNF approach to solving (12) represents a primal-dual algorithm to the linear programming problem, and consequently, is likely to be the most efficient method. To

date, we are not aware of a corresponding primal-dual algorithm for the original linear programming problem of (7).

The use of (12) and, consequently, an MCNF procedure depends on our ability to form a node-edge incidence matrix of a primal graph that spans the null-space of the facet-edge incidence matrix of the original (dual) graph. In computer vision, when employing a 6-connected lattice, an MCNF procedure is available. However, for general problems, such a procedure may not be available since the dual graph is unknown (or the basis of the null-space for B is not totally unimodular). In Appendix B, we treat the more general case by characterizing the complexes (represented by a B matrix) that allow the minimal surface to be found by applying linear programming to (7).

2.4 Finding an Initial Surface

The MCNF approach presented above presupposes that it is possible to find *some* solution z_0 that satisfies (8). Given the application paradigm of joining the 2D segmentations obtained via an outside algorithm (e.g., 2D intelligent scissors), we may assume that the initial closed contours are nonintersecting and exist on one or more axis aligned slices in the 3D data. Consequently, the initial z_0 can be computed by including all facets in the connected components of the slices in the interior of the provided contours. Although this method generally produces a disjoint z_0, it should be noted that z_0 is allowed to consist of multiple sets of facets, so long as the constraint (8) is satisfied.

In a more general application of this minimum-weight surface technique, our input contours may not be axis aligned with the data. For example, our 2D contours could be generated via an automatic technique that does not respect the axis alignment. In these more general situations where it is not easy to find an initial contour, there are two possibilities. If the initial contours are compact and roughly planar (even if that plane does not align with the data axes), then the more general LP method defined by solving (7) (and detailed in Appendix B) could be used with unity weighting to quickly find *some* initial z_0 which could then be employed by the MCNF algorithm. However, if the input 2D contours are more complicated (e.g., nonplanar), then the LP method could still be applied directly to the weighted complex to produce the optimal solution, albeit more slowly than the MCNF algorithm.

2.5 Algorithm Summary

In the previous sections, we have shown that the natural extension of the shortest path problem to higher dimension leads to a solution of the minimum-weight surface problem. This problem may be solved efficiently using Sullivan's method, which ultimately requires the solution to an MCNF problem. As a segmentation algorithm, we input a closed contour (or series of closed contours) and return a minimum-weight surface. Since closed contours are the output of standard (2D) intelligent scissors, the outputs of the 2D intelligent scissors provide inputs for a 3D intelligent scissors. Alternately, any other 2D segmentation algorithm could also be used to produce the inputs to the minimum-weight surface problem.

We may summarize the entire segmentation algorithm as follows:

1. Obtain an oriented closed contour on one or more slices through an outside algorithm (e.g., 2D intelligent scissors). This contour is represented in (7) by vector r.
2. Define facet weights from the image content using (1). Note that "outside" facets must be assigned to an arbitrary value—we have employed $w = 0.5$.
3. Identify *any* surface z_0 that has the desired boundary (see Section 2.4).
4. Find the minimum-cost circulation through z_0 (corresponding to solving (16)), which may be solved using a variety of existing algorithms [20].
5. The set of saturated edges in the primal graph (corresponding to facets in the dual) comprises the minimum-weight surface having the boundary given by the closed contours obtained in Step 1.

3 Results

In the previous sections, we presented the generalization of the (2D) shortest path problem with Type II constraints to the analogous (3D) minimum-weight surface problem with Type II constraints. Consequently, a natural extension of intelligent scissors to 3D has been provided. In this section, our goal is to verify the correctness of the algorithm on synthetic data, and then, to demonstrate its application to the segmentation of 3D data.

3.1 Correctness

We begin with three examples to demonstrate correctness. First, we use the algorithm to segment a black sphere (in a white background), given an initial contour around one parallel. Second, we segment the same sphere using an input consisting of contours around two parallels (i.e., a contour given on two slices). Finally, we segment a "lunchbox" shape given a medial contour. This experiment shows that the algorithm correctly handles changes in object genus without the special handling employed by the path-based methods reviewed above.

Fig. 5 shows the results of these three experiments, verifying the correctness of the algorithm. In contrast to the shortest path problem in which two points are necessary to define a path, Fig. 5a shows that a single closed contour is sufficient to define the boundary of a surface.

As a reference implementation, we employed both a generic LP solver (implemented in the COIN library [19]) to solve the LP problem defined by (7) and the fast MCNF solver from Sullivan's method. From a speed standpoint, applying the generic LP solver to solving the minimum-weight surface problem on a $64 \times 64 \times 64$ lattice representing Fig. 5a required 819 seconds when run on an IBM ThinkPad T42 laptop with a 1.70 GHz processor and 512 MB of RAM. The same solution required 0.55 seconds to obtain with the MCNF solver on the same machine. The LP solver implemented in the COIN library was general purpose and did not take advantage of the significant structure inherent in the facet-edge incidence matrix of a lattice. However, the discrepancy in speed between the generic LP solver and the MCNF solver is so great that one must assume that the MCNF solver would outperform even dedicated LP code for (7).

Another similarity with the shortest path problem is that the minimum-weight surface may not be unique. For

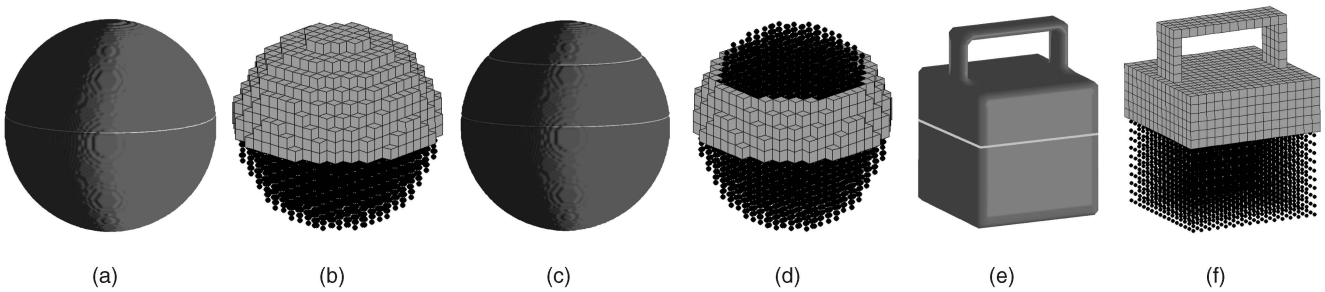

Fig. 5. Synthetic examples to illustrate correctness. Renderings of the original object (with the input contours) are shown, along with the algorithm outputs. The input volumes all had black voxels indicating voxels belonging to the object and white voxels indicating background. The white stripe in each of the rendered views shows the input contour(s). In the solution visualizations, black dots are plotted at the center of the black (object) voxels and facets are shown to indicate the computed surface. (a) and (b) A sphere with an input contour along a parallel. Note that, unlike 2D intelligent scissors, a single boundary input (contour) is sufficient to define a solution. (c) and (d) A sphere with input contours at two parallels of different heights. (e) and (f) A lunchbox shape with a handle on the top and a medial contour input. The algorithm will correctly find minimum-weight surfaces with topological changes.

example, a closed contour located precisely at the equator of the sphere in Fig. 5a could result in a solution indicating either the upper or the lower hemisphere. This situation would be analogous in the shortest path problem to the multiple solutions possible when given the input of two opposite points on a circle. Fig. 6 illustrates this issue.

We stress that by using a combinatorial formulation of the minimal surface problem on a 6-connected 3D lattice, the continuous, euclidean minimal surface will not necessarily be obtained. Accordingly, the continuous euclidean minimal surface and the minimum-weight surface may not agree. For example, the solution to the continuous minimal surface is a catenoid when given a boundary of two, identical, closed contours at different heights. In contrast, the minimum-weight surface is a cylinder when the underlying complex is a 6-connected graph. This contrast between the discrete and continuous domains is analogous to the fact that the shortest path on a four-connected lattice will not necessarily be the same as a straight line in the plane.

3.2 Real Data

In this section, we illustrate the application of minimum-weight surfaces to the segmentation of real data. Minimum-weight surfaces with Type I boundary conditions have previously been applied to image segmentation in the context of graph cuts [9]. Therefore, we simply illustrate the use of minimum-weight surfaces with Type II boundary conditions in the context of image segmentation (i.e., as a 3D extension of intelligent scissors). Two of the 3D data sets used in these experiments were SPECT cardiac data and CT cardiac data. A third example is given by a CT scan showing the branching of the aorta near the iliac bifurcation. These data are used to illustrate that the algorithm described here has no difficulty joining more than two closed contours, even when the underlying object splits into multiple sections. Note that the resulting surface, like a pair of pants, satisfies the desired constraints—the only surface boundaries are at the locations specified by closed contours. Type II boundary conditions (closed contours) were generated using conventional intelligent scissors on two slices of each data set, and we computed the minimum-weight surface that had these contours as a boundary. Fig. 7 shows the result of these experiments.

4 Conclusion

Previous attempts to extend shortest path segmentation algorithms to 3D have all focused on using a network of paths drawn between closed contours (or contours and points) to produce the surface of a 3D object. However,

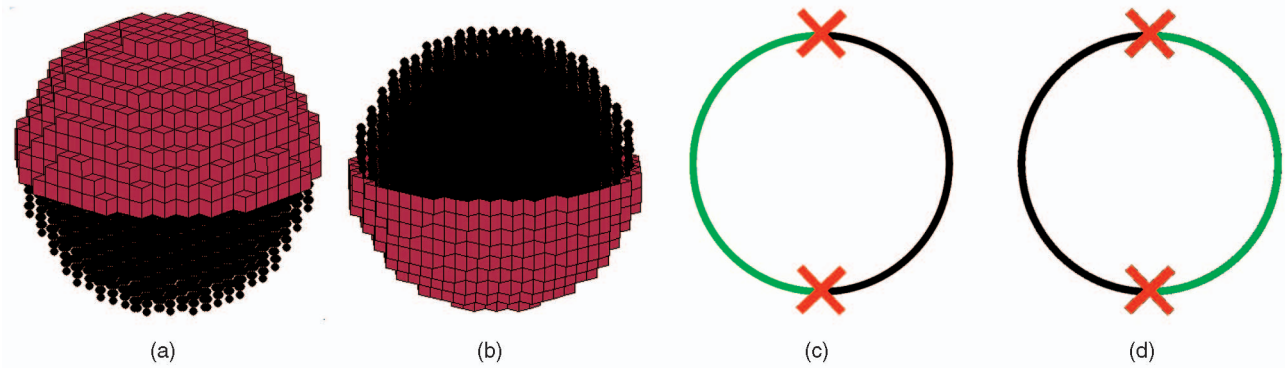

Fig. 6. The minimum-weight surface given a boundary is not necessarily unique. For example, if the surface boundary is given as a closed contour at the equator of a sphere, then either (a) the upper or (b) the lower hemisphere is a valid minimum solution. This same lack of uniqueness may also appear in the shortest path problem. Analogously, if two endpoints were placed at antipodal points of a circle, the shortest path may be returned as either the left (c) or right (d) path around the circumference of the circle.

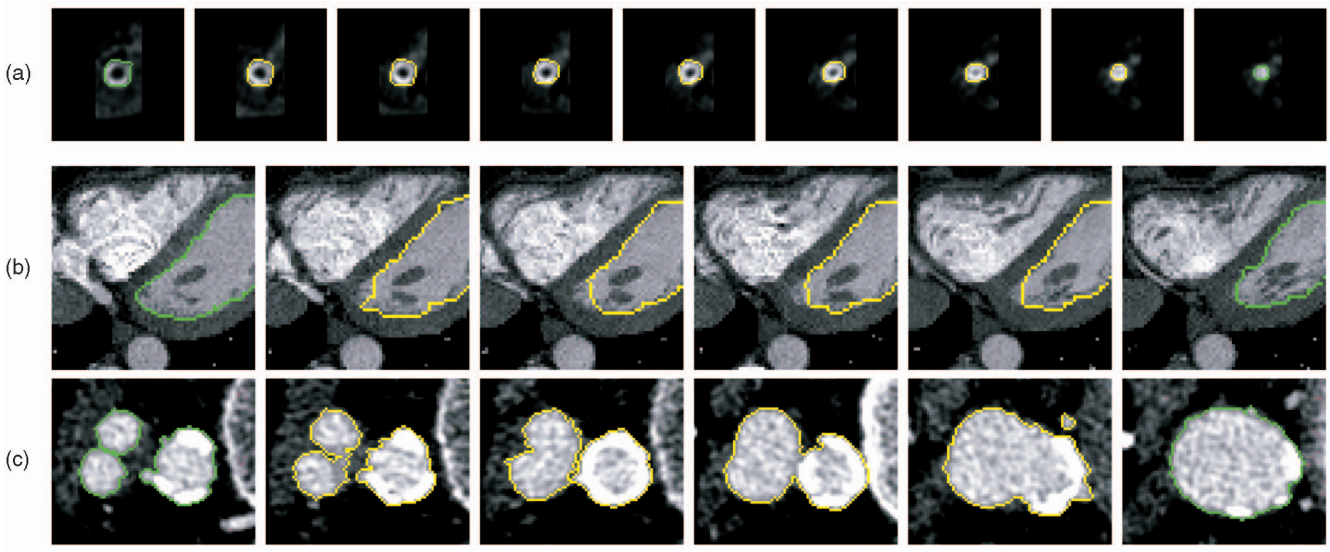

Fig. 7. Application of the algorithm to segmentation of 3D medical data. In each study, the green/gray contours were placed on the left and right slices and the intermediate yellow/white contours represent the minimum-weight surface between these contours. This figure demonstrates that the algorithm behaves as expected for a minimal surface approach to 3D image segmentation. (a) SPECT cardiac data. (b) CT cardiac data. (c) CT aorta near iliac branch—notice that multiple closed contours may be placed and a single surface may be found that splits accordingly to accommodate the prescribed boundaries. Note that all slices between the closed contours are not displayed.

shortest paths joining two closed contours are not guaranteed to lie on the minimal surface joining the two contours, implying that no density of shortest paths will ever be guaranteed to produce the minimal surface. In this work, we demonstrated that the natural extension of shortest path techniques to 3D segmentation is to directly solve for the minimal surface. By directly solving for the minimal surface, we make no assumptions on object genus, permit the use of a single closed contour, split and merge the surface as needed, may employ arbitrary (nonnegative) weighting functions, and are guaranteed to find the exact minimum surface. Using Sullivan's method, the reduction of the minimal surface problem to an MCNF problem permits a very efficient solution.

In contrast to the max-flow/min-cut approach to finding minimum-weight surfaces by defining Type I boundary conditions, we detail how to find minimum-weight surfaces through the specification of Type II boundary conditions. Both Type I and Type II boundary conditions find continuing application in 2D for the computation of shortest paths. We hope that this presentation of how intelligent scissors/live wire can be extended to higher dimension will permit the popularization of this ubiquitous segmentation method in a 3D context.

Appendix A

Feasibility of a Solution to the Minimum-Weight Surface Problem

In this appendix, we address the question: Will any r representing a closed contour have a solution in (7)? The answer turns out to depend on the underlying complex—some complexes will always have a solution, while others may not. Fortunately, for use in computer vision, all complexes likely to be of interest (e.g., the 6-connected lattice) will always have a feasible solution. Note that the dependence of feasibility on the underlying complex is analogous to the shortest path case (3), in which there will be no feasible solutions if the underlying graph is disconnected and the path endpoints are placed in different components.

We begin by formally stating that the boundary of a boundary is zero in terms of the incidence matrices:

$$AB = 0. \tag{17}$$

Since r is the signed indicator vector of a closed contour, we note that

$$Ar = 0. \tag{18}$$

We can now make the following statement regarding the feasibility of finding a solution to (7) given a closed contour, represented by an r that satisfies (18).

Proposition 1. *If the edge-facet incidence matrix has $m - n + 1$ independent columns and a nonzero vector r satisfies (18), then (7) is guaranteed to have a solution.*

Proof. The node-edge incidence matrix is known to have a right null-space of rank $m - n + 1$ [30]. Since (17) holds and the edge-facet incidence matrix has $m - n + 1$ independent columns, then the edge-facet incidence matrix spans the right null-space. In other words, if

$$Ar = 0, \tag{19}$$

then r may be expressed as a linear combination (with constants c) of the columns of the edge-facet incidence matrix

$$r = Bc, \tag{20}$$

giving the proposition. □

In a more general context, the question of feasibility hinges on whether or not the contour represented by r encloses a "hole" in the complex. For example, if the

underlying complex were the triangulated surface of a torus, then an r representing a contour encircling the handle would not have a feasible solution. This situation is analogous to the minimal path problem in which it will not be possible to find a minimal path joining two points placed in separate components of a disconnected graph. In the context of image processing on a 6-connected lattice, the homology group is trivial (i.e., any r may be expressed as a linear combination of B), and therefore, any r representing a closed contour will have a feasible solution z.

Appendix B

Solution via Linear Programming—The General Case

In this section, we are concerned with knowing when the integer programming problem defined by (7) may be solved using a generic linear programming solver. It was shown by Sullivan [47] that linear programming could be used to solve (7) if the underlying complex is cellular and there exists a feasible solution. We now proceed to discuss a characterization of when the minimal surface integer programming problem defined by (7) can be solved with generic LP when the underlying complex is not cellular.

If the constraint of an integer programming problem (i.e., (7)) is given as an *inequality*, then the total unimodularity (t.u.) property is both necessary and sufficient to guarantee that a solution obtained via linear programming is integer for an arbitrary feasible integer right-hand side [37]. In contrast, if the constraint is formulated as an *equality* (as in our case), then a t.u. constraint matrix is simply sufficient, but not necessary. It has long been known that edge-facet incidence matrices are not, in general, totally unimodular [43], [36]. Recall that a matrix is totally unimodular when the determinant of all submatrices takes one of the values $\{-1, 0, 1\}$.

In the conference publication on this topic [21], the question of total unimodularity of the constraint matrix was not properly handled. The issues are: 1) Total unimodularity of the constraint matrix is mischaracterized as both sufficient *and* necessary for an equality constraint to guarantee an integral solution. In fact, total unimodularity is simply sufficient, but not necessary to guarantee an integral solution in the presence of a constraint of this form [35]. 2) The proofs of total unimodularity for the lattice do not prove this property since they are predicated on the (false) premise that total unimodularity is preserved via elementary matrix operations. The elementary matrix operations preserve unimodularity, but not total unimodularity. However, the two operations introduced do preserve orientability (since the torsion coefficients, as defined in [30], [52], are preserved under these operations) and therefore show that the lattice is orientable. Therefore, the connection between orientability and total unimodularity is that a t.u. incidence matrix necessarily represents an orientable complex (since the invariant factors for a t.u. matrix are all unit valued), but the converse statement is not necessarily true. 3) The edge-facet incidence matrix of the lattice is not, in general, totally unimodular. However, despite the aforementioned troubles with this topic in [21], the primary conclusion remains correct, for reasons that will be explained in this section. Namely, on the lattice, solving (7) is guaranteed to produce an integer solution when using generic linear programming.

Truemper [50] settled the issue of when an equality constraint in the same form as (7) was both necessary and sufficient to guarantee that a solution is integer in the presence of an integral right-hand side by introducing the concept of **unimodular** (u.) matrices. However, in Section 2.3, it was shown that one can employ the additional information that *an* integer solution exists (i.e., (8)). This additional information provides extra power in analyzing when an equality constraint in the same form as (7) will be guaranteed to give an integer solution. We develop this idea further by introducing the notion of a **preunimodular matrix** (p.u. matrix).

Let us call matrix A preunimodular if

$$\begin{aligned} &\min_x \ w^T x, \\ &\text{s.t.} \quad Ax = Ax_0, \\ &\qquad x \geq 0 \end{aligned} \tag{21}$$

has integer solution x for all integer x_0. Note that A, B, and C in this appendix are not intended to correspond to the incidence matrices used in the body of this paper.

Let A be of size $m \times n, \operatorname{rank}(A) = r$.

Theorem 1. *The following conditions are equivalent:*

1. *A is preunimodular.*
2. *For all bases B with $A = [BC]$, matrix $D = \{B^{-1}\}C$ is integer.*
3. *For all bases B with $A = [BC]$, matrix $D = \{B^{-1}\}C$ is t.u.*
4. *For at least one base B with $A = [BC]$, matrix $D = \{B^{-1}\}C$ is t.u.*
5. *There exists t.u. matrix U such that $\operatorname{Ker} A = \operatorname{Im} U$.*
6. *A can be converted to a t.u. matrix by elementary row operations (namely, adding a (possibly fractional) multiple of one row to another, adding/removing zero row).*

Let B be some base, with $A = [BC]$. In the following, write $x = [y; z]$, where y is a vector of size r corresponding to columns of B and z is a vector of size $(n - r)$ corresponding to columns of C. Although B is not generally square, we write $D = \{B^{-1}\}C$ to indicate the space spanned by D in the expression $BD = C$. Note that C is $m \times (n - r)$, B is $m \times r$, and D is $r \times (n - r)$.

Equation $Ax = Ax_0$ is equivalent to

$$By + BDz = By_0 + BDz_0 \tag{22}$$

or

$$y + Dz = y_0 + Dz_0. \tag{23}$$

Thus, the LP problem can be rewritten as

$$\begin{aligned} &\min_x w^T x = w^T [y; z], \\ &y + Dz = y_0 + Dz_0, \\ &y \geq 0, \\ &z \geq 0. \end{aligned} \tag{24}$$

Proof. *1 ⇔ 2*: For a fixed $x_0 = [y_0; z_0]$, all extreme solutions of the LP are given by $\{y = y_0 + Dz_0, z = 0\}$ for all possible choices of base B. Thus, 1) is equivalent to the statement: $y_0 + Dz_0$ is integer for all bases and for all integer vectors $[y_0; z_0]$, or Dz_0 is integer for all bases and for all integer vectors z_0. The latter is easily shown to be equivalent to 2.

1 ⇔ 4: Let B be an arbitrary base of A. If either 1 or 4 holds, then the matrix D is integer (as shown above). Therefore, we may assume that D is integer. Denote $D' = [I\ D]$.

Condition 1 takes the form of (24), which is equivalent to the statement that

$$\begin{aligned} \min_x w^T x &= w^T [y; z], \\ D'[y; z] &= b, \\ y &\geq 0, \\ z &\geq 0 \end{aligned} \tag{25}$$

has integer solution $[y; z]$ for all integer b, which is equivalent to stating that D is t.u.

Note that in the sequence above, B was taken to be an arbitrary base of A. This fact shows that $1 \equiv 3$.

5 ⇔ 1: In the same manner as (12), we may transform (21) to the form

$$\begin{aligned} &\min_q w^T (Uq + x_0), \\ &\text{s.t.} - Uq = x_0 - x \leq x_0, \end{aligned} \tag{26}$$

which may take integer values iff U is t.u. [35].

4 ⇔ 5: $Ax = 0$ is equivalent to $By + BDz = 0$, or $y + Dz = 0$. Thus, $\mathrm{Ker}A$ is spanned by vectors $[-De_i; e_i]$, where e_i $(i = 1, \ldots, n - r)$ is the ith unit vector. Therefore, $\mathrm{Ker}A = \mathrm{Im}\, U$, where U is the $n \times (n - r)$ matrix of these vectors (i.e., the ith column of U equals $[-De_i; e_i]$). U can be written as $[-D; I]$, and thus, is t.u.

1 ⇔ 6: Without loss of generality, we can assume that A has the form $[I\ D]$ (since any matrix can be converted to such a form by elementary row operations and swapping columns, and such operations preserve the p.u. property). Using the fact that $1 \equiv 4$, we obtain that A is p.u. $\equiv D$ is t.u. $\equiv A$ is t.u. □

Theorem 2. *Suppose A, U are matrices such that $\mathrm{Ker}A = \mathrm{Im}\, U$. Then, A is p.u. if and only if U^T is p.u.*

Proof. If: Suppose A is p.u. Then, by 5, there exists t.u. matrix $\hat{U}$ such that $\mathrm{Ker}A = \mathrm{Im}\hat{U}$. Therefore, $\mathrm{Im}U = \mathrm{Im}\hat{U}$ implies that the row space of U^T and $\hat{U}^T$ is the same, which implies that U^T can be converted to $\hat{U}^T$ by elementary row operations, finally, implying that U^T is p.u. (by 6).

Only if: Suppose U^T is p.u. Then, by 6, U^T can be converted to a t.u. matrix $\hat{U}^T$ using elementary row operations. There holds $\mathrm{Im}\hat{U} = \mathrm{Im}U = \mathrm{Ker}A$. Therefore, by 5, matrix A is p.u. □

Additional properties of a p.u. matrix derived from the above are:

1. Any matrix with full column rank (i.e., $r = n$) is p.u.
2. If A is p.u., then $[A\ A]$ is p.u.
3. If A is p.u., then $[A; I]$ is p.u., since $[A; I]$ has full rank.
4. If A is p.u., the removal of any row j produces a new matrix A_j, that is, p.u., since $\mathrm{Ker}A_j = \mathrm{Ker}A$.

Lemma 1. *If A is p.u. and the rank of A is preserved by removing columns or adding rows to form A^*, then A^* is p.u.*

Proof. Part 1: If A is p.u. and the removal of column i preserves the rank, then A_i (the matrix A without column i) is p.u. $A = [BC]$, $D = \{B^{-1}\}C$ is t.u. Since removal of a_i preserves rank, a basis is preserved such that $A_i = [B\ C_i]$ and $D_i = \{B^{-1}\}C_i$ is t.u. if D is t.u., since removal of any row/column preserves the t.u. property.

Part 2: If A is p.u. and rows are added that preserve the rank, then the new rows $B = v^T A$ for some v. $\mathrm{Ker}A = \mathrm{Ker}[A; B]$. □

In the context of the above discussion, the facet-edge incidence matrix of the 6-connected lattice is p.u. (by condition 5 of Theorem 1), and therefore, (by definition) the integer programming problem described by (7) is guaranteed to produce an integer solution when linear programming is applied.

The characterization of p.u. matrices given above demonstrates that it is sometimes possible to use linear programming to solve for minimum-weight surfaces on more general (noncellular) complexes for which it would not be possible to employ Sullivan's method.

Finally, we note that p.u.ness is a property of the *constraint* matrix, defined in the minimum-weight surface problem by the facet-edge incidence matrix. Therefore, it is the structure of the complex itself that determines whether or not the minimum-weight surface problem is solvable with LP, rather than the weights associated with the facets (provided that the weights are nonnegative). This point is important in the context of 3D image processing—regardless of the image content (leading to weights), the minimum-weight surface problem can always be solved with LP on the 6-connected lattice.

Acknowledgments

The author would like to thank Marie-Pierre Jolly for first proposing work on a proper extension of Intelligent Scissors to 3D and Christopher Alvino for the suggestion of the catenoid as a counterexample to the minimal paths approach to constructing a surface. He would especially like to thank Vladimir Kolmogorov for alerting him to the problem with the conference version of this work and for all of his assistance with Section 2.3 and Appendix B. Useful insights on this topic were also provided by Yuri Boykov and Ali Kemal Sinop. Finally, he would like to thank the reviewers and editor for their detailed suggestions which significantly improved the exposition.

References

[1] K. Aoshima and M. Iri, "Comments on F. Hadlock's Paper: Finding a Maximum Cut of a Planar Graph in Polynomial Time," *SIAM J. Computing*, vol. 6, pp. 86-87, 1977.

[2] B. Appleton and H. Talbot, "Globally Optimal Surfaces by Continuous Maximal Flows," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 1, pp. 106-118, Jan. 2006.

[3] R. Ardon and L.D. Cohen, "Fast Constrained Surface Extraction by Minimal Paths," *Int'l J. Computer Vision,* vol. 69, no. 1, pp. 127-136, Aug. 2006.

[4] R. Ardon, L.D. Cohen, and A. Yezzi, "A New Implicit Method for Surface Segmentation by Minimal Paths: Applications in 3D Medical Images," *Proc. Int'l Workshop Energy Minimization Methods in Computer Vision and Pattern Recognition,* A. Rangarajan, ed., pp. 520-535, 2005.

[5] C.J. Armstrong, W.A. Barrett, and B. Price, "Live Surface," *Proc. Volume Graphics '06,* vol. 22, pp. 661-670, Sept. 2006.

[6] N. Biggs, *Algebraic Graph Theory.* Cambridge Univ. Press, 1974.

[7] I. Bitter, A.E. Kaufman, and M. Sato, "Penalized-Distance Volumetric Skeleton Algorithm," *IEEE Trans. Visualization and Computer Graphics,* vol. 7, no. 3, pp. 195-206, July-Sept. 2001.

[8] M.J. Black, G. Sapiro, D.H. Marimont, and D. Heeger, "Robust Anisotropic Diffusion," *IEEE Trans. Image Processing,* vol. 7, no. 3, pp. 421-432, Mar. 1998.

[9] Y. Boykov and M.-P. Jolly, "Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images," *Proc. Int'l Conf. Computer Vision,* pp. 105-112, 2001.

[10] Y. Boykov and V. Kolmogorov, "Computing Geodesics and Minimal Surfaces via Graph Cuts," *Proc. Int'l Conf. Computer Vision,* vol. 1, Oct. 2003.

[11] A.J. Briggs, C. Detweiler, D. Scharstein, M. College, and A. Vandenberg-Rodes, "Expected Shortest Paths for Landmark-Based Robot Navigation," *Int'l J. Robotics Research,* vol. 23, nos. 7/8, pp. 717-728, 2004.

[12] C. Buehler, S.J. Gortler, M.F. Cohen, and L. McMillan, "Minimal Surfaces for Stereo," *Proc. Seventh European Conf. Computer Vision,* vol. III, pp. 885-899, May 2002.

[13] L. Cohen and T. Deschamps, "Grouping Connected Components Using Minimal Path Techniques. Application to Reconstruction of Vessels in 2D and 3D Images," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* vol. 2, pp 102-109, 2001.

[14] L.D. Cohen and R. Kimmel, "Global Minimum for Active Contour Models: A Minimal Path Approach," *Int'l J. Computer Vision,* vol. 24, no. 1, pp. 57-78, 1997.

[15] A.X. Falcão and J.K. Udupa, "A 3D Generalization of User-Steered Live-Wire Segmentation," *Medical Image Analysis,* vol. 4, pp. 389-402, 2000.

[16] A.X. Falcão, J.K. Udupa, S. Samarasekera, S. Sharma, B.H. Elliot, and R. de A. Lotufo, "User-Steered Image Segmentation Paradigms: Live Wire and Live Lane," *Graphical Models and Image Processing,* vol. 60, no. 4, pp. 233-260, 1998.

[17] L.R. Ford and D.R. Fulkerson, "A Primal-Dual Algorithm for the Capacitated Hitchcock Problem," *Naval Research Logistics Quarterly,* vol. 4, pp. 47-54, 1957.

[18] L.R. Ford and D.R. Fulkerson, *Flows in Networks.* Princeton Univ. Press, 1962.

[19] J. Forrest, D. de la Nuez, and R. Lougee-Heimer, *CLP User Guide.* IBM Research, 2004.

[20] A.V. Goldberg, E. Tardos, and R.E. Tarjan, "Network Flow Algorithms," *Paths, Flows and VLSI-Design,* B. Korte, L. Lovasz, H. Proemel, and A. Schrijver, eds., pp. 101-164, Springer-Verlag, 1990.

[21] L. Grady, "Computing Exact Discrete Minimal Surfaces: Extending and Solving the Shortest Path Problem in 3D with Application to Segmentation," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 69-78, June 2006.

[22] F. Hadlock, "Finding a Maximum Cut of a Planar Graph in Polynomial Time," *SIAM J. Computing,* vol. 4, no. 3, pp. 221-225, 1975.

[23] G. Hamarneh, J. Yang, C. McIntosh, and M. Langille, "3D Live-Wire-Based Semi-Automatic Segmentation of Medical Images," *Proc. SPIE Medical Imaging '05: Image Processing,* pp. 1597-1603, 2005.

[24] P.J. Hilton and S. Wylie, *Homology Theory.* Cambridge Univ. Press, 1960.

[25] D. Kirsanov, "Minimal Discrete Curves and Surfaces," PhD thesis, Harvard Univ., 2004.

[26] M. Knapp, A. Kanitsar, and M.E. Gröller, "Semi-Automatic Topology Independent Contour-Based $2\frac{1}{2}$D Segmentation Using Live-Wire," *J. WSCG,* vol. 12, no. 2, pp. 229-236, 2004.

[27] V. Kolmogorov, "Primal-Dual Algorithm for Convex Markov Random Fields," Technical Report MSR-TR-2005-117, Microsoft, Sept. 2005.

[28] V. Kolmogorov and C. Rother, "Minimizing Nonsubmodular Functions with Graph Cuts—A Review," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 29, no. 7, pp. 1274-1279, July 2007.

[29] S. König and J. Hesser, "3D Live-Wires on Pre-Segmented Volume Data," *Proc. SPIE Medical Imaging '05: Image Processing,* pp. 1674-1679, 2005.

[30] S. Lefschetz, *Algebraic Topology,* vol. 27. Am. Math. Soc. Colloquium Publications, 1942.

[31] K. Li, X. Wu, D.Z. Chen, and M. Sonka, "Optimal Surface Segmentation in Volumetric Images—A Graph-Theoretic Approach," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 1, pp. 119-134, Jan. 2006.

[32] C. Mattiussi, "The Finite Volume, Finite Element and Finite Difference Methods as Numerical Methods for Physical Field Problems," *Advances in Imaging and Electron Physics,* pp. 1-146, Academic Press, Inc., Apr. 2000.

[33] F. Morgan, *Geometric Measure Theory,* third ed. Academic Press, 2000.

[34] E. Mortensen and W. Barrett, "Interactive Segmentation with Intelligent Scissors," *Graphical Models in Image Processing,* vol. 60, no. 5, pp. 349-384, 1998.

[35] G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization.* John Wiley & Sons, 1999.

[36] S. Okada, "On Mesh and Node Determinants," *Proc. IRE,* vol. 43, p. 1527, 1955.

[37] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization.* Dover, 1998.

[38] S.V. Porter, M. Mirmehdi, and B.T. Thomas, "A Shortest Path Representation for Video Summarisation," *Proc. 12th Int'l Conf. Image Analysis and Processing,* pp. 460-465, Sept. 2003.

[39] S. Roy and I. Cox, "A Maximum-Flow Formulation of the n-Camera Stereo Correspondence Problem," *Proc. Int'l Conf. Computer Vision,* pp. 492-499, 1998.

[40] Z. Salah and J.O.D. Bartz, "Live-Wire Revisited," *Proc. Workshop Bildverarbeitung in der Medizin,* pp. 158-162, 2005.

[41] A. Schenk, G. Prause, and H.-O. Peitgen, "Efficient Semiautomatic Segmentation of 3D Objects in Medical Images," *Proc. Int'l Conf. Medical Image Computing and Computer-Assisted Intervention,* pp. 186-195, 2000.

[42] A. Schenk, G. Prause, and H.-O. Peitgen, "Local Cost Computation for Efficient Segmentation of 3D Objects with Live Wire," *Proc. SPIE Medical Imaging,* M. Sonka and K.M. Hanson, eds., pp. 1357-1364, 2001.

[43] S. Seshu, "The Mesh Counterpart of Shekel's Theorem," *Proc. IRE,* vol. 43, p. 342, 1955.

[44] J.A. Sethian, "A Fast Marching Level Set Method for Monotonically Advancing Fronts," *Proc. Nat'l Academy of Sciences USA,* vol. 93, no. 4, pp. 1591-1595, 1996.

[45] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, pp. 888-905, Aug. 2000.

[46] W.-K. Shih, S. Wu, and Y.S. Kuo, "Unifying Maximum Cut and Minimum Cut of a Planar Graph," *IEEE Trans. Computers,* vol. 39, no. 5, pp. 694-697, May 1990.

[47] J.M. Sullivan, "A Crystalline Approximation Theorem for Hypersurfaces," PhD thesis, Princeton Univ., Oct. 1990.

[48] C. Sun, "Fast Optical Flow Using 3D Shortest Path Techniques," *Image and Vision Computing,* vol. 20, nos. 13/14, pp. 981-991, Dec. 2002.

[49] E. Tonti, "On the Geometrical Structure of Electromagnetism," *Gravitation, Electromagnetism and Geometrical Structures,* G. Ferraese, ed., pp. 281-308, Pitagora, 1996.

[50] K. Truemper, "Algebraic Characterizations of Unimodular Matrices," *SIAM J. Applied Math.,* vol. 35, no. 2, pp. 328-332, Sept. 1978.

[51] J.N. Tsitsiklis, "Efficient Algorithms for Globally Optimal Trajectories," *IEEE Trans. Automatic Control,* vol. 40, no. 9, pp. 1528-1538, Sept. 1995.

[52] A.J. Zomorodian, *Topology for Computing.* Cambridge Univ. Press, 2005.

Leo Grady received the BSc degree in electrical engineering from the University of Vermont in 1999 and the PhD degree from the Cognitive and Neural Systems Department at Boston University, in 2003. Since Autumn 2003, he has been working as a senior research scientist at Siemens Corporate Research, Princeton, New Jersey, in the Imaging and Visualization Department. His research focuses on image segmentation, data clustering, learning and filtering using techniques from graph theory, combinatorial topology, and PDEs. Other interests include pattern/object recognition, applied mathematics, nonuniform data processing, image registration, cellular automata, machine learning, robotics, and emergent phenomena. He is a member of the IEEE and the IEEE Computer Society.