Graph Algorithmic Techniques for Biomedical Image Segmentation

Mona K. Garvin and Xiaodong Wu

Abstract After presenting an introduction to a more traditional graph-based 2D segmentation technique, this chapter presents an in-depth overview of two state-of-the-artgraph-based methods for segmenting three-dimensional structures in medical images: graph cuts and the Layered Optimal Graph Image Segmentation of Multiple Objects and Surfaces (LOGISMOS) approach. In each case, an overview of the underlying optimization problem is presented first (i.e., the formulation of an energy/cost function and the specified constraints), followed by the graph-based representation of the optimization problem which enables the globally optimal solution to be found in polynomial time. In particular, in the 2D case, a 2D boundary segmentation optimization problem is transformed into that of finding a minimum-cost path in a graph. In the graph-cuts approach, a 3D object/background labeling problem is transformed into that of finding a minimum *s*–*t* cut in a graph, and in the LOGISMOS approach, a single or multiple 3D surface segmentation problem is first transformed into that of finding a minimum-cost closure in a graph (which is further transformed into finding a minimum *s*–*t* cut in a graph). For each approach, example applications and extensions are also presented.

M.K. Garvin (✉)
Iowa City VA Health Care System, Iowa City, IA 52246, USA

The University of Iowa, Iowa City, IA 52242, USA
e-mail: mona-garvin@uiowa.edu

X. Wu
The University of Iowa, Iowa City, IA 52242, USA
e-mail: xiaodong-wu@uiowa.edu

P.K. Saha et al. (eds.), *Advanced Computational Approaches to Biomedical Engineering*,
DOI 10.1007/978-3-642-41539-5_1, © Springer-Verlag Berlin Heidelberg 2014

1 Introduction

Segmentation problems in medical imaging often require the precise identification of multiple boundaries/regions in volumetric images. Graph-based segmentation approaches are becoming increasingly popular for the automated delineation of medical structures/objects of interest within three-dimensional images, in part, due to the ability of many of the more recent approaches (such as graph cuts [1–6] and the Layered Optimal Graph Image Segmentation of Multiple Objects and Surfaces (LOGISMOS) approach [7–11]) to efficiently produce globally optimal three-dimensional segmentations in a single pass (and correspondingly not get stuck in local optima). In addition, graph-based approaches such as LOGISMOS enable the simultaneous optimal detection of multiple surfaces in volumetric images, which is important in many medical image segmentation applications.

In this chapter, after presenting an introduction to a more traditional 2D segmentation technique in Sect. 2 (which is still very useful in its own right for 2D boundary delineation problems), we present an in-depth overview of two state-of-the-art-graph-based methods for segmenting three-dimensional structures in medical images: graph cuts (Sect. 3) and the LOGISMOS approach (Sect. 4). In each case, an overview of the underlying optimization problem is presented first (i.e., the formulation of an energy/cost function and the specified constraints), followed by the graph-based representation of the optimization problem which enables the globally optimal solution to be found in polynomial time. In particular, in the 2D case, a 2D boundary segmentation optimization problem is transformed into that of finding a minimum-cost path in a graph. In the graph-cuts approach, a 3D object/background labeling problem is transformed into that of finding a minimum s–t cut in a graph, and in the LOGISMOS approach, a single or multiple 3D surface segmentation problem is first transformed into that of finding a minimum-cost closure in a graph (which is further transformed into finding a minimum s–t cut in a graph). For each approach, we also present example applications and extensions.

2 2D Boundary Delineation as a Minimum-Cost Path Problem

2.1 The 2D Single Boundary Optimization Problem

For our first example graph-based segmentation approach, we will describe how one can formulate a single 2D boundary segmentation problem as a minimum-cost path problem. While many of the high-level concepts discussed in this section (such as the formulation of an optimization problem with constraints) and the transformation to a well-known graph problem are similar to that discussed in later sections, the graph-based path problem representation in the 2D case is often much more intuitive

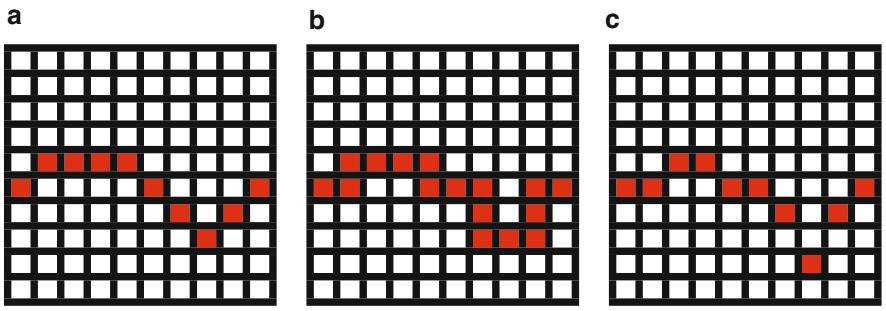

Fig. 1 Three candidate boundaries (in *red*) for the minimum-cost-path-based approach shown on a small image grid. While all three examples seem to contain reasonable boundaries to separate the top and bottom of the image, only (**a**) would be a feasible boundary using a smoothness constraint, Δ, of 1. (**b**) Is not feasible because some of the columns have more than one boundary point defined. (**c**) Is not feasible because the change in boundary position from column 6 to column 7 and from column 7 to column 8 has a magnitude of 2 and the smoothness constraint requires a maximum change of 1 unit. However, (**c**) would be feasible if Δ were defined to be 2

to understand. For simplicity, we will only consider the case in which we desire to find an optimal boundary that is defined from the leftmost column of an image to the rightmost column of an image. In particular, we assume that only one boundary point should exist from each column of the image so that the desired boundary can be written by the function $f(x)$ mapping each column (x-position) to a desired y-value within the image. We also define a smoothness constraint that requires that the boundary's position between neighboring columns does not change more than a given constant: $|f(x+1)-f(x)| \leq \Delta$ for $0 \leq x \leq N_x-2$, where N_x is the number of columns of the image. Figure 1 provides an example of a feasible boundary and two non-feasible boundaries in a small image when $\Delta = 1$. While such feasibility constraints may initially seem quite limiting, it is important to remember that many common transforms, such as a Cartesian-to-polar transform, may be used to enable the desired boundaries to satisfy these constraints. For example, use of polar coordinates can make the boundaries of roughly circular objects, such as the boundaries of the heart, the boundaries of vessel walls, and the boundaries of the optic disc/cup, satisfy such constraints (Figs. 2b, c and 3).

From the original image $I(x, y)$, we also assume that we can define a cost image, $c(x, y)$, reflecting each pixel's unlikeliness of belonging to the boundary. The cost $E(f)$ of a feasible boundary $f(x)$ can correspondingly be defined as follows:

$$E(f) = \sum_{x=0}^{N_x-1} c(x, f(x)). \tag{1}$$

Thus, the optimization problem we desire to solve is to find the minimum-cost (according to Eq. (1)) boundary $f(x)$ subject to the feasibility constraints.

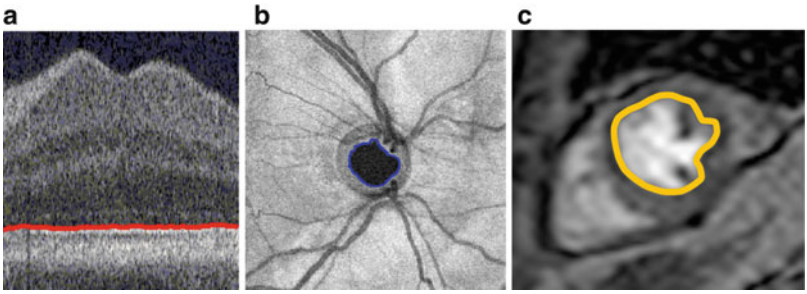

Fig. 2 Example applications for the 2D single boundary optimization problem. (**a**) Single retinal boundary (in *red*) on spectral-domain optical coherence tomography slice (no transformation necessary). (**b**) Optic cup boundary (in *blue*) on spectral-domain optical coherence tomography projection image (Cartesian-to-polar transform necessary). (**c**) Left ventricular boundary (in *yellow*) on magnetic resonance imaging slice (Cartesian-to-polar transform necessary)

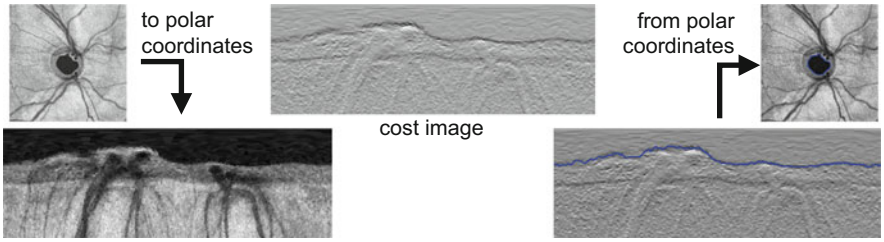

Fig. 3 Example use of Cartesian-to-polar transform to enable the desired boundary to satisfy the feasible constraints required for the 2D single boundary optimization problem (e.g., that one boundary point exists in each column of the image and that the boundary is sufficiently smooth). The boundary is found from a cost image defined in polar coordinates using a minimum-cost path approach, with the final boundary being obtained with a polar-to-Cartesian transform

2.2 *Graph Representation of the 2D Single Boundary Optimization Problem*

The 2D single boundary optimization problem as defined in Sect. 2.1 can be readily transformed into a minimum-cost path problem within a node-weighted graph (Fig. 4). Each pixel in the cost image, $c(x, y)$, becomes a node with weight given by the intensity value. Because of this direct correspondence between pixels and nodes, we can label each node with its corresponding (x, y) position. Directed edges are added for each "feasible" transition from a boundary point from one column to the next column. In particular, given a smoothness constraint Δ, for each node (x, y), a directed edge is added to nodes $(x+1, y-\Delta), (x+1, y-\Delta+1), \ldots, (x+1, y+\Delta)$ (as long as these destination nodes exist). If a destination node does not exist (as on the boundaries), no directed edge is added. Finding the minimum-cost path in such

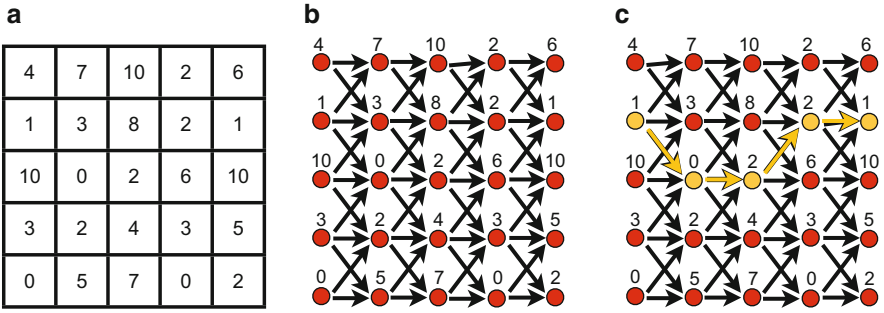

Fig. 4 Example construction of graph for finding minimum-cost boundary according to Eq. (1). (**a**) Small example cost image $c(x, y)$. (**b**) Node-weighted graph representation when the smoothness constraint, Δ, is 1. Edges reflect allowed transitions between boundary points. The minimum-cost path from any node in the *leftmost column* to any node in the *rightmost column* will correspond to the minimum-cost 2D boundary in the image. (**c**) Highlighted (in *yellow*) minimum-cost path in graph

a node-weighted graph directly corresponds to finding the minimum-cost boundary in the original image.

While a number of approaches exist for finding minimum-cost paths in general graphs, the simple structure of this graph (including its acyclic nature) makes a dynamic programming approach a good choice for finding the minimum-cost path. In fact, in implementing this approach, one does not even need to explicitly define the nodes/edges and can simply use an implicit definition of the underlying graph (e.g., given a node position, you can use a simple formula to determine the neighboring nodes). Figure 5 illustrates the basic concepts of a dynamic programming approach. When $\Delta = 1$, you can define the following recursive definition for the cost of the minimum-cost path $p(x, y)$ from any node in the first column ($x = 0$) to node at position (x, y):

$$\begin{cases} c(x,y) & x{=}0(\text{basecase}) \\ \min[p(x-1,y), p(x-1,y+1)] + c(x,y) & x > 0; y = 0 \\ \min[p(x-1,y-1), p(x-1,y), p(x-1,y+1)] + c(x,y) & x > 0; 0 < y < N_y{-}1 \\ \min[p(x-1,y-1), p(x-1,y)] + c(x,y) & x > 0; y{=}N_y - 1 \end{cases} \tag{2}$$

Here, N_y is the number of rows in the image. A similar recursive definition can be defined for other values of Δ. Note that the nodes/edges are defined "implicitly" by directly writing the recursive formula for the path costs.

Because of this recursive definition, the path costs can be efficiently found by computing the path-cost values $p(x, y)$ one column at a time from $x = 0$ to $x = N_x - 1$. Conceptually, in computing each path-cost value, one also keeps track of the predecessor "node" that provided the minimum (although this is not

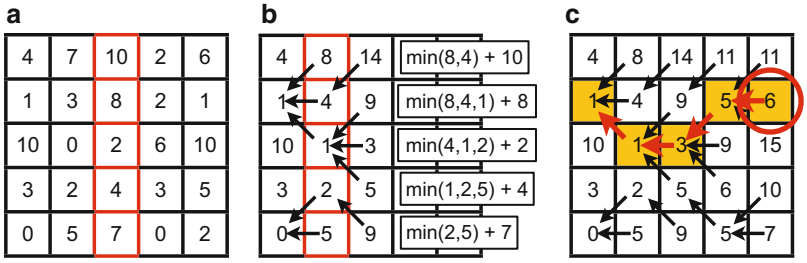

Fig. 5 Use of dynamic programming to find minimum-cost path corresponding to the optimal boundary when $\Delta = 1$. (**a**) Cost image, $c(x, y)$ with highlighted column 2 (note that the leftmost column is column 0). (**b**) Computation of minimum-cost path costs for all of the nodes in column 3. For each node, the minimum-cost path value is computed by adding the cost image value to the path-cost value of the predecessor [for a non-boundary case and $\Delta = 1$, the predecessors of node (x, y) are nodes $(x - 1, y - 1)$, $(x - 1, y)$ and $(x - 1, y + 1)$] with the minimum path-cost value. *Arrows* are shown to indicate each selected predecessor node. (**c**) Highlighted minimum-cost path. The minimum-cost path is found by selecting the node in the last column of $p(x, y)$ with the smallest value and then backtracking (i.e., following the *predecessor arrows*) until reaching a node in the first column

absolutely necessary as determining this node can be done in constant time by simply re-examining all of the values). Finding the actually minimum-cost path involves simply finding the smallest value in the last column and then backtracking through the predecessor nodes until reaching a node in the first column.

If the underlying boundary is roughly circular such that a Cartesian-to-polar transform was used in creating the cost image (meaning that boundaries define one radial value per angle), it is also useful to require that the found boundary be "closed" (i.e., a smoothness constraint exists from the last column to the first column). With a dynamic programming approach, one option for enforcing this is to separately find the minimum-cost path from each of the starting nodes in the first column to the set of nodes in the last column that would obey the constraint by setting the node weights of all other nodes in the first column and the weight of all non-feasible nodes in the last column to ∞. The overall minimum-cost path can then be found as the smallest out of these N_y minimum-cost paths.

3 3D Object Segmentation Using Graph Cuts

3.1 The Graph-Cut Optimization Problem (A Labeling Problem)

In this section, we describe the graph-cut approach of Boykov et al. [1–6] for segmenting a 3D object within a 3D image. While the 2D single boundary segmentation problem discussed in Sect. 2 and the single and multiple surface

segmentation problems to be discussed in Sect. 4 are inherently formulated to focus on the detection of an object boundary or set of object boundaries (with the number of boundaries specified a priori), the optimization problem associated with the graph cut approach is inherently formulated as a labeling problem (i.e., determining the set of pixels/voxels that should be labeled as object). One example corresponding difference is that it is possible that the "object" to be segmented will be disconnected using the standard formulation of the graph-cut approach, whereas the feasibility constraints prevent this in the surface-based methods. (In some applications, preventing such disconnections is desirable, and in other applications, it is not desirable.)

More specifically, in the graph-cut approach, if $\mathscr{P}$ is the set of voxels in the image, we assume that each voxel $p \in \mathscr{P}$ has a cost $R_p(\text{"obj"})$ and $R_p(\text{"bkg"})$ associated with being labeled as an object voxel or background voxel, respectively. We furthermore assume that each neighboring pair of voxels $\{p, q\} \in \mathscr{N}$, where $\mathscr{N}$ is the set of pairs of neighboring voxels, has a cost $B_{p,q}$ associated with the pair of voxels having different labels. Our goal is to find the optimal labeling $A = (A_1, A_2, \ldots, A_{|\mathscr{P}|})$ to assign each voxel p to a label A_p where A_p is "bkg" or "obj" to minimize the following "energy" or "cost" function:

$$E(A) = \lambda \cdot R(A) + B(A) \,, \tag{3}$$

where $R(A)$ is the regional term given by

$$R(A) = \sum_{p \in \mathscr{P}} R_p(A_p) \,, \tag{4}$$

and $B(A)$ is the boundary term given by

$$B(A) = \sum_{\{p,q\} \in \mathscr{N}} B_{p,q} \cdot \delta_{A_p \neq A_q} \,, \tag{5}$$

where $\delta_{A_p \neq A_q}$ is 0 if the label of A_p is equal to that of A_q and 1 otherwise. λ provides the relative weighting between the regional cost terms and the boundary cost terms.

Figure 6 provides example regional (i.e., object/background) and boundary costs associated with a small toy image and Fig. 7 provides two example labelings and their associated costs based on Eq. (3).

3.2 *Direct Representation of the Graph-Cut Optimization Problem as a Minimum s–t Cut Problem*

The graph-cut optimization problem discussed in Sect. 3.1 can be readily formulated as a minimum *s–t* cut problem in an edge-weighted graph [2, 6]. In an *s–t* cut

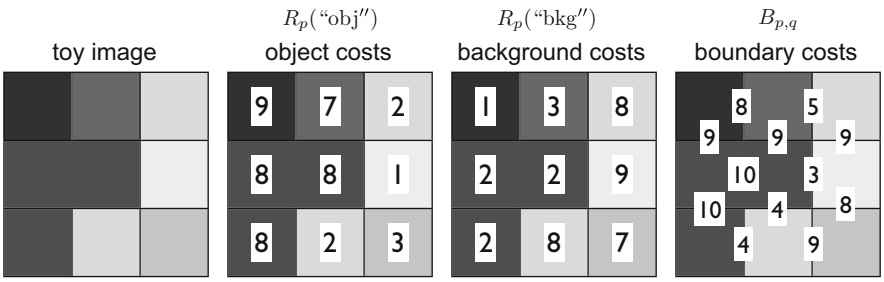

Fig. 6 Example toy image and object ($R_p(\text{“obj''})$), background ($R_p(\text{“bkg''})$), and boundary costs ($B_{p,q}$) for use in a graph-cuts approach

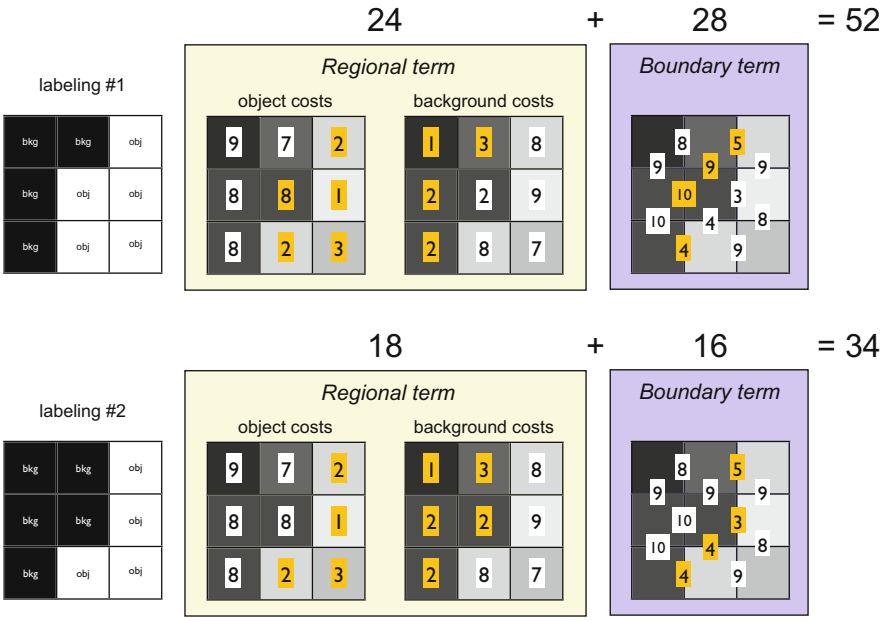

Fig. 7 Two example labelings and associated total costs from toy example from Fig. 6. For simplicity, the weighting λ in Eq. (3) between region and boundary costs is set to 1

formulation, you have an edge-weighted graph with two terminal nodes: s and t. A cut of (S, T) of the graph is a partition of the nodes V into two disjoint sets: S and T ($T = V - S$) with the requirement that $s \in S$ and $t \in T$. The cost of a cut is defined as the summation of the weights of the edges from nodes in S to nodes in T (intuitively the edges that are “cut” in the separation) [12]. The goal in a minimum s–t cut problem is to find the cut with the minimum cost. Once the problem is formulated as a minimum s–t cut problem, many algorithms exist for finding the globally optimal solution in low-order polynomial time [4].

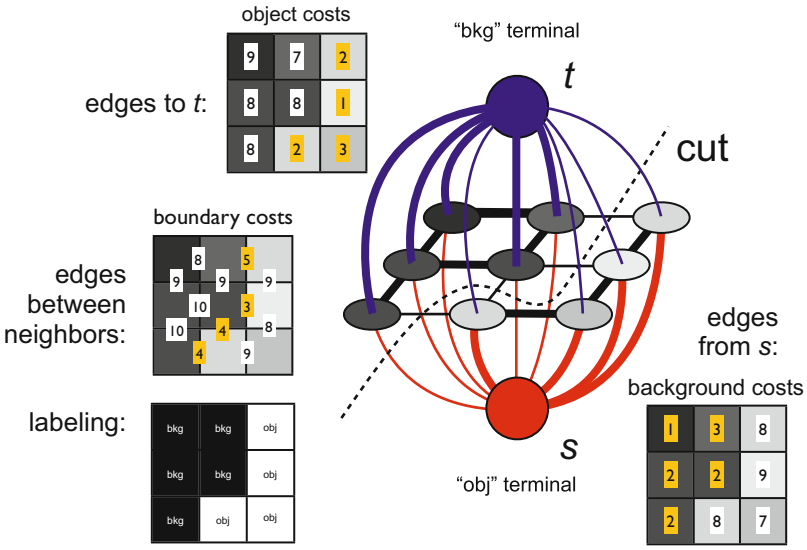

Fig. 8 Example graph representation and minimum cut from regional and boundary costs shown in Figs. 6 and 7. Graph example adapted from examples presented by Boykov et al. [2, 6]

In transforming the graph-cut optimization problem into a minimum s–t cut problem, first, a node p is added for every voxel in the image and edges are added between each pair of nodes corresponding to neighborhood voxels with weight $B_{p,q}$ (these edges between neighbors are often referred to as "n-links" as their weights correspond to the neighbor discontinuity costs). Next, two extra "terminal" nodes, s (the "obj" terminal) and t (the "bkg" terminal), are added to the graph and for each voxel in the image (corresponding to node p in the graph), two edges are added (often called "t-links" for "terminal"): $\{s, p\}$ and $\{p, t\}$. The weight of each edge between "obj" terminal s and node p, $\{s, p\}$, is given by the background cost value for the associated voxel (times a constant): $\lambda \cdot R_p(\text{"bkg}'')$. The weight of each edge between node p and "bkg" terminal t, $\{p, t\}$, is given by the object cost value for the associated voxel (times a constant): $\lambda \cdot R_p(\text{"obj}'')$. When determining the minimum s–t cut of this constructed graph, the nodes remaining connected to the "obj" terminal s correspond to the voxels belonging to the segmented object (i.e., the voxels that should be given a label "obj"). Figure 8 illustrates the graph and associated minimum cut associated with the toy example shown in Figs. 6 and 7. The associated labeling (segmentation) associated with the cut is also shown. The edges cut include the edges from the nodes in the final object set (with a label "obj") to the background terminal t (with each p in this set contributing $\lambda \cdot R_p(\text{"obj}'')$ to the cut cost), the edges from object terminal s to the nodes in the final background set (with label "bkg", with each p in this set contributing $\lambda \cdot R_p(\text{"bkg}'')$ to the cut cost) and each of the edges between a node in the object set and a node in the background set (with each such pair of edges $\{p, q\}$ contributing $B_{p,q}$ to the cut cost). Thus,

with the above graph construction, the value of the minimum-cut corresponds to the value of the minimal labeling according to Eq. (3).

Note that the above formulation assumes an undirected edge-weighted graph, but many *s–t* cut algorithms are inherently designed for directed edge-weighted graphs. However, the conversion to the corresponding directed graph is straightforward. In particular, each edge $\{s, p\}$ between terminal s and node p becomes directed with s as the tail and p as the head. Similarly, each edge $\{p, t\}$ between node p and terminal t becomes directed with p as the tail and t as the head. Each edge $\{p, q\}$ between neighbors p and q becomes two edges (each with the same weight): one from node p to node q and one from node q to node p.

When using graph cuts approaches, it is also common to specify a set of "seed" object and background voxels (i.e., voxels you wish to constrain as being labeled as object and background, respectively). Let $\mathscr{O}$ be the set of object seeds and $\mathscr{B}$ be the set of background seeds. You can enforce this constraint in the minimum *s–t* cut graph representation by modifying the edge weight from s to every node in $\mathscr{O}$ to have a weight of infinity (intuitively, ensuring the edges from the object terminal to those nodes that correspond to seed object voxels will not be cut) and modifying the edge weight of every node in $\mathscr{B}$ to t to have a weight of infinity (intuitively, ensuring the edges from nodes that correspond to seed background voxels and the background terminal will not be cut). It is also common to modify the weight of edges $\{s, p\}$ where $p \in \mathscr{B}$ and $\{p, t\}$ where $p \in \mathscr{O}$ to have a weight of zero, but this is not theoretically necessary (assuming the original weights were also not infinity).

3.3 Example Applications and Extensions

3.3.1 Multi-Label (Multi-Region) Extensions

A multi-label extension of the original graph cuts energy function in Eq. (3) follows:

$$E(A) = \lambda \cdot \sum_{p \in \mathscr{P}} R_p(A_p) + \sum_{\{p,q\} \in \mathscr{N}} B_{p,q}(A_p, A_q) \cdot \delta_{A_p \neq A_q} \,, \tag{6}$$

where $A \in \{\text{obj}_1, \text{obj}_2, \ldots, \text{obj}_N\}^{|\mathscr{P}|}$ is a multi-object image labeling. However, unfortunately, solving this particular multi-label problem is NP-hard in the general case [3, 6]. Thus, approximation algorithms, such as the $\alpha\beta$-swap algorithm and the α-expansion algorithm, have been proposed for efficiently obtaining a close-to-optimal solution [3]. At a high level, both the $\alpha\beta$-swap algorithm and the α-expansion algorithm are examples of a local search approach (Fig. 9), where, after assigning an arbitrary initial labeling as the current candidate solution, a best "neighboring" solution is generated and saved as the current candidate solution if it has a lower cost based on Eq. (6). The generation of "neighboring" candidate

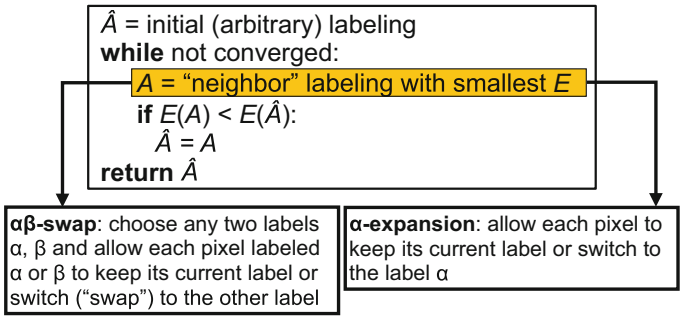

Fig. 9 The $\alpha\beta$-swap algorithm and the α-expansion algorithm are two local search approaches for efficiently determining a close-to-optimal solution to the multi-label graph cuts problem with the energy function of Eq. (6) [3]. In both cases, determination of the best "neighboring" solution can be obtained by solving a single graph cuts problem

solutions (and saving them as the current solution if they have a lower cost) is repeated until convergence. The key clever idea behind both approaches is in the efficient generation of the best "neighboring" solution. In the $\alpha\beta$-swap algorithm, a neighboring candidate solution is one in which, after an arbitrary selection of any two labels α and β, every pixel with label α or β maintains its current label or switches ("swaps") to the other label (Fig. 10). In the α-expansion algorithm, a neighboring candidate solution is one in which after arbitrary selection of label α every pixel maintains its current label or switches to the label α (Fig. 11). With either definition of a neighboring solution, determination of the best neighboring solution can be obtained by solving a single graph cut problem as only a binary labeling choice needs to be made at each voxel location (i.e., α versus β for the $\alpha\beta$-subset of voxels in the $\alpha\beta$-swap algorithm and α versus "original label" for all voxels in the α-expansion algorithm). In practice, the α-expansion algorithm is often more effective than the $\alpha\beta$-swap algorithm, but care must be taken to ensure that submodularity requirements of the cost function are satisfied [3, 13].

While approximation algorithms discussed above may work well for solving the multi-label optimization problem for certain application domains, by recognizing special cases (e.g., through additional geometric constraints) of the multi-label/multi-region labeling problem, certain multi-label/multi-region problems can be actually solved optimally. (Of course, the binary graph cuts formulation using the energy term of Eq. (3) is an obvious example of a special case.) For example, through the addition of specific geometric interaction constraints (in part, motivated from the multi-surface graph-based approach discussed in Sect. 4) such as the fact that "region C contains region A," "region D contains region B," and "region C excludes region D," Delong and Boykov [13] have proposed a formulation of the multi-region graph cuts approach for which an optimal solution can be found with a single graph cut.

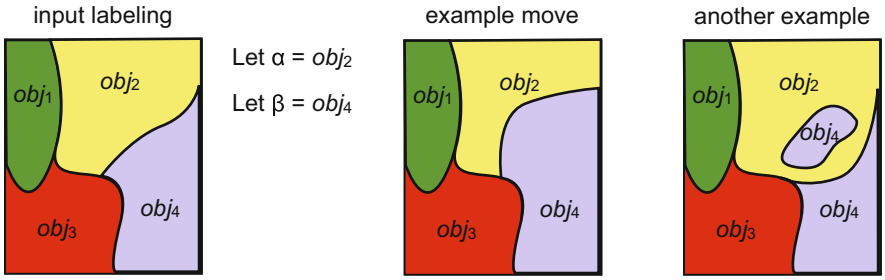

Fig. 10 Two example $\alpha\beta$-swap moves from initial input labeling on the left. In this example, α and β were arbitrarily selected as obj_2 and obj_4, respectively. Valid neighboring solutions are thus labelings where any pixels originally labeled as obj_2 keep their original label or change to label obj_4 and any pixels originally labeled as obj_4 keep their original label or change to label obj_2. (Pixels with labels other than obj_2 and obj_4 must keep their original label.) The binary nature of these moves means that the best move can be determined with a single graph cut

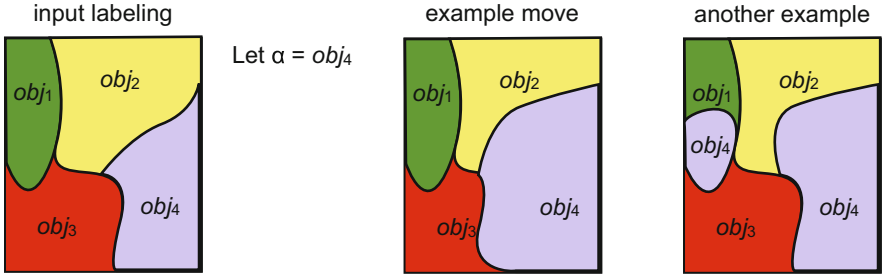

Fig. 11 Two example α-expansion moves from initial input labeling on the left. In this example, α was arbitrarily selected as obj_4. Valid neighboring solutions are thus labelings where any pixels keeps its original label or changes to label obj_4 (thus "expanding" the number of pixels labeled as obj_4). The binary nature of these moves means that the best move can be determined with a single graph cut

3.3.2 Co-segmentation of Tumors in PET–CT Images

Positron emission tomography–computed tomography (PET–CT) has revolutionized modern cancer imaging. The integrated PET–CT, by adding precise anatomic localization to functional imaging, currently provides the most accurate information available on tumor extent and distribution for various common cancers. It has been increasingly playing an indispensable role in cancer treatment planning, therapy response assessment, and tumor staging. This section presents an optimal co-segmentation method for tumor delineation from PET–CT scans with the extension of the graph-cut method [14].

Modeling PET–CT Co-segmentation. The basic idea is to formulate the simultaneous segmentation of tumor from PET–CT scans as minimizing the Markov Random Field (MRF) energy function, which consists of a data fidelity term and

a boundary smoothness term. The data fidelity term measures how well voxels fit into the object or background model. The boundary term penalizes the discontinuity between the object and background. More precisely, the MRF energy functions commonly used for segmentation on the individual PET and CT images (denoted by $\mathscr{I}_{\text{PET}}$ and $\mathscr{I}_{\text{CT}}$, respectively) are as follows:

$$\mathscr{E}_{\text{PET}}(f^P) = \sum_{p \in \mathscr{I}_{\text{PET}}} \mathscr{D}_p^P(f_p^P) + \lambda_1 \sum_{(p,q) \in \mathscr{N}_P} \mathscr{V}_{pq}^P(f_p^P, f_q^P) \tag{7}$$

$$\mathscr{E}_{\text{CT}}(f^C) = \sum_{p \in \mathscr{I}_{\text{CT}}} \mathscr{D}_p^C(f_p^C) + \lambda_2 \sum_{(p,q) \in \mathscr{N}_C} \mathscr{V}_{pq}^C(f_p^C, f_q^C) \tag{8}$$

$\mathscr{D}_p^P(f_p^P)$ is the individual penalty for assigning a voxel p to "object" or "background" in the PET $\mathscr{I}_{\text{PET}}$; $\mathscr{V}_{pq}^P(f_p^P, f_q^P)$ is the penalty for assigning labels f_p^P and f_q^P to two neighboring voxels p and q according to the neighborhood setting $\mathscr{N}_P$. $\mathscr{D}_p^C(f_p^C)$ and $\mathscr{V}_{pq}^C(f_p^C, f_q^C)$ have the same meaning for $\mathscr{I}_{\text{CT}}$. Most close voxels are expected to have the same label ("object" or "background"). Thus, one may expect no penalty if neighboring voxels have the same label and a penalty w_{pq} otherwise. We can then apply the graph cut method in Sect. 3.2 to segment the target tumor from PET and CT, respectively, by minimizing the energy functions $\mathscr{E}_{\text{PET}}(f^P)$ and $\mathscr{E}_{\text{CT}}(f^C)$, which, however, does not utilize the information from the other modality.

To co-segment the tumor from both PET and CT scans, an additional PET–CT context term $\mathscr{E}_{\text{PET–CT}}$ is introduced to the energy function, which penalizes the segmentation difference between two image datasets. Without loss of generality (WLOG), assume that the PET and CT images are well registered. Let (p, p') denote a pair of corresponding voxels in $\mathscr{I}_{\text{PET}}$ and $\mathscr{I}_{\text{CT}}$. The label difference is penalized with $\delta_{pp'}(f_p^P, f_{p'}^C)$ for p and p', with

$$\delta_{pp'}(f_p^P, f_{p'}^C) = \begin{cases} g(p, p') & \text{if } f_p^P \neq f_{p'}^C \\ 0 & \text{if } f_p^P = f_{p'}^C \end{cases}, \tag{9}$$

where $g(p, p') > 0$ is employed to penalize the disagreement between labels of corresponding voxels p and p'. The PET–CT context term then takes the form:

$$\mathscr{E}_{\text{PET}--\text{CT}}(f^P, f^C) = \sum_{(p,p') \text{ with } p \in \mathscr{I}_{\text{PET}}, p' \in \mathscr{I}_{\text{CT}}} \delta_{pp'}(f_p^P, f_{p'}^C) \tag{10}$$

Note that the PET–CT context constraint is soft with $\delta_{pp'}(f_p^C, f_{p'}^P) < +\infty$. It is not assumed that the tumor contour in CT is identical to that in PET. Hence, the uncertainties of imaging and registration are accommodated in this model. The energy function of the co-segmentation is then defined as follows:

$$\mathscr{E}_{\text{cs}}(f^P, f^C) = \mathscr{E}_{\text{PET}}(f^P) + \mathscr{E}_{\text{CT}}(f^C) + \mathscr{E}_{\text{PET–CT}}(f^P, f^C) \tag{11}$$

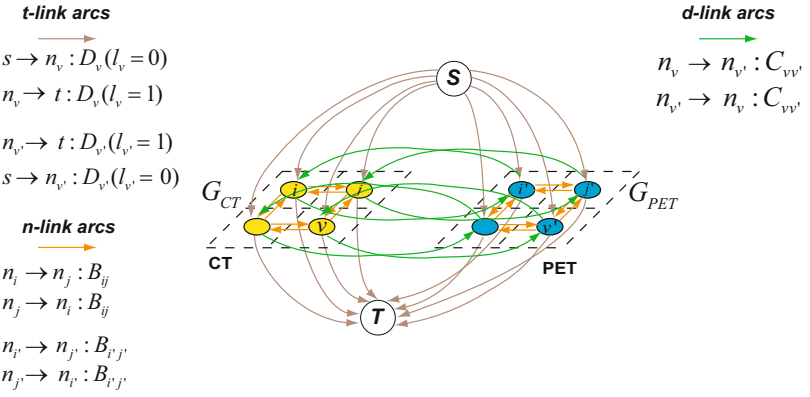

Fig. 12 Graph construction of G with two subgraphs G_{CT} and G_{PET} for the co-segmentation of PET–CT images [14]. Three types of arcs are introduced. The *orange arcs* (n-links) encode the boundary terms; the *brown arcs* (t-links) encode data terms; and the *green arcs* (d-links) enforce the PET–CT context term in the co-segmentation energy function

Most previous co-segmentation algorithms in computer vision [15–18] work on a pair of images with similar (or nearly identical) foregrounds and unrelated backgrounds, and explicitly make use of histogram matching, which makes the models computationally intractable. Batra et al. recently applied the graph cut method for co-segmentation [19], in which a common appearance model is assumed across all the segmented images. However, the PET and CT images may not have such a common appearance model. The co-segmentation scheme here provides a more flexible PET–CT context term $\mathscr{E}_{\text{PET–CT}}$ to make use of the dual modalities information.

Optimization. The co-segmentation problem is solved by extending the graph cut method in Sect. 3.2 with a computation of a minimum-cost s–t cut in a transformed graph G, which admits a globally optimal solution in low-order polynomial time.

The graph G consists of two node-disjoint subgraphs G_{PET} and G_{CT}, each being used for the search of the tumor in the PET $\mathscr{I}_{\text{PET}}$ and the CT $\mathscr{I}_{\text{CT}}$. The construction of each G_{PET} and G_{CT} follows the graph cut method in Sect. 3.2 to encode the energy terms $\mathscr{E}_{\text{PET}}$ and $\mathscr{E}_{\text{CT}}$, respectively. To enforce the PET–CT context term $\mathscr{E}_{\text{PET–CT}}$, additional inter-subgraph arcs are introduced between G_{PET} and G_{CT}. For every pair of corresponding voxels (p, p') with $p \in \mathscr{I}_{\text{CT}}$ and $p' \in \mathscr{I}_{\text{PET}}$, two directed arcs are added between the corresponding nodes of the two subgraphs in opposite directions. The weight of each arc is assigned to penalize the labeling difference of the two voxels p and p' (i.e., $\delta_{pp'}(f_p^P, f_{p'}^C)$ with $f_p^P \neq f_{p'}^C$). Figure 12 illustrates the construction of the graph G, with green arcs encoding the PET–CT context term $\mathscr{E}_{\text{PET–CT}}$.

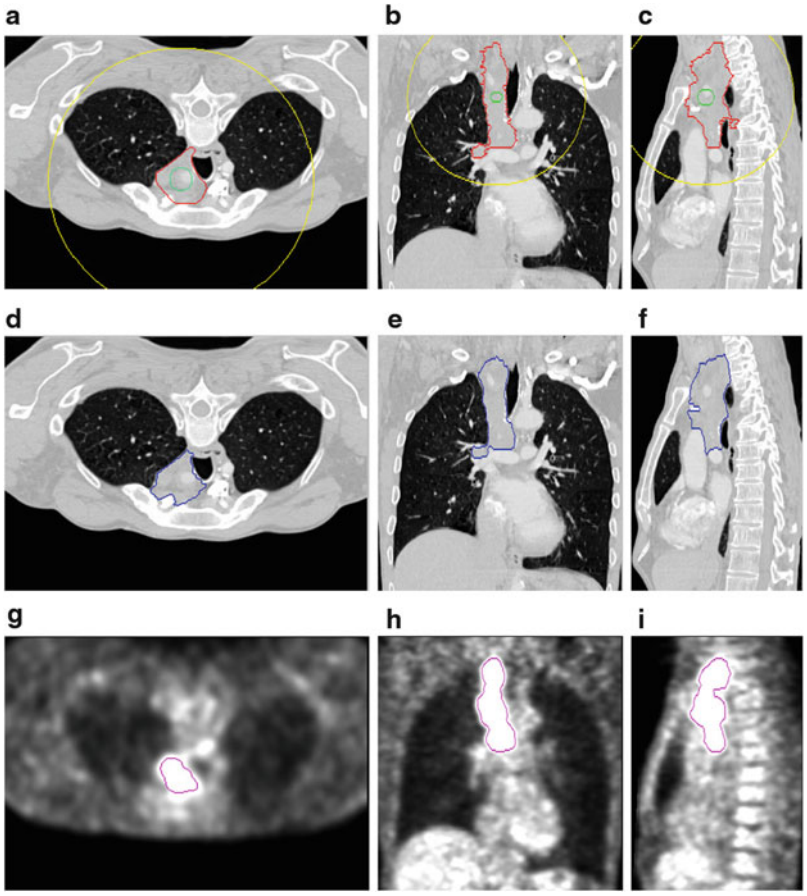

Fig. 13 Typical tumor segmentation in the transverse (*left*), coronal (*middle*), and sagittal (*right*) views [14]. (**a**–**c**) 2D slices of a 3D CT image with the reference standard (*red*) and outlines of spherical initialization (*green* and *yellow*). (**d**–**f**) Proposed co-segmentation results in the CT image. (**g**–**i**) Co-segmentation results in the PET image

As shown in [14], the minimum-cost s–t cut $\mathscr{C}^* = (A^*, \overline{A}^*)$ in G defines an optimal delineation of tumor in both PET and CT images. The target tumor volume on the CT image includes those voxels whose corresponding nodes in G_{CT} belong to the source set A^*. Similarly, the segmented tumor volume on the PET image is given by those voxels whose associated nodes in G_{PET} belong to the source set A^*.

Typical lung tumor segmentation in three (transverse, coronal and sagittal) views are shown in Fig. 13. The co-segmentation method demonstrated accurate PET and CT segmentation results [14].

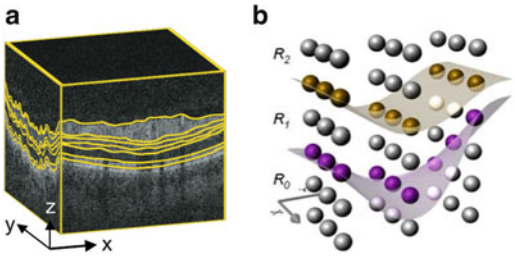

Fig. 14 Optimal surface detection. (**a**) An example 3D retinal optical coherence tomography (OCT) image with surfaces of interest specified. (**b**) The schematic surfaces and the orientation

4 Optimal Layered Graph Search for Image Segmentation

This section presents the theory of the optimal layered graph search method[7–11], also sometimes called the LOGISMOS approach, which is applicable to the multi-surface segmentation of terrain-like surfaces (represented by orthogonal n-D graphs), tubular objects (cyclic regular and irregular graphs), objects with complex shapes, and multiple partially interacting objects. WLOG, we focus on using terrain-like surfaces and 4-neighborhoods to discuss the optimal layered graph search method. However, the simpler principles used for this illustration are directly applicable to arbitrarily irregular surfaces.

4.1 Optimal Surface Detection Problems

Let $\mathscr{I}$ be a given 3D volumetric image with size of $n = X \times Y \times Z$. For each (x, y) pair, $0 \leq x < X$ and $0 \leq y < Y$, the voxels with different z-coordinates, that is, the voxel subset $\{\mathscr{I}(x, y, z) \mid 0 \leq z < Z\}$, form a *column* parallel to the **z**-axis, denoted by $\mathrm{Col}(x, y)$. Two columns are *neighboring* if their (x, y)-coordinates satisfy some neighborhood conditions. For example, under the 4-neighboring setting, the column $\mathrm{Col}(x, y)$ is neighboring to $\mathrm{Col}(x', y')$ if $|x - x'| + |y - y'| = 1$. Henceforth, a model of the 4-neighboring setting is used; this simple model can be easily extended to other neighboring settings. Each of the target *terrain-like surfaces* contains one and only one voxel in each column of $\mathscr{I}$ (Fig. 14). The feasibility of the target surfaces is governed by the surface smoothness and separation constraints. The surface *smoothness constraint* is specified by two *smoothness parameters*, $\Delta_{\mathbf{x}}$ and $\Delta_{\mathbf{y}}$, which define the maximum allowed change in the **z**-coordinate of a surface along each unit distance change in the **x** and **y** dimensions, respectively. If $\mathscr{I}(x, y, z')$ and $\mathscr{I}(x + 1, y, z'')$ (resp., $\mathscr{I}(x, y + 1, z'')$) are two (neighboring) voxels on a feasible surface, then $|z' - z''| \leq \Delta_{\mathbf{x}}$ (resp., $|z' - z''| \leq \Delta_{\mathbf{y}}$) (Fig. 15a). In multiple surface detection, for each pair of the target surfaces S and S', we use two parameters, $\delta^l \geq 0$ and $\delta^u \geq 0$, to represent the surface *separation constraint*, which defines the relative positioning and the distance range of the two surfaces. That is, if $\mathscr{I}(x, y, z) \in S$ and $\mathscr{I}(x, y, z') \in S'$, we have $\delta^l \leq z' - z \leq \delta^u$ for every (x, y)

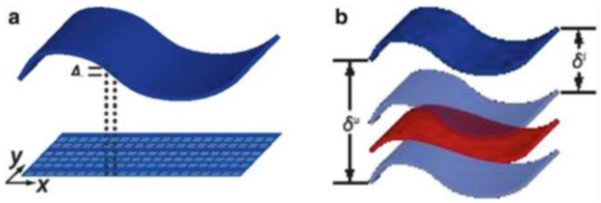

Fig. 15 The surface feasibility. (**a**) The surface smoothness constraint. (**b**) The surface separation constraint

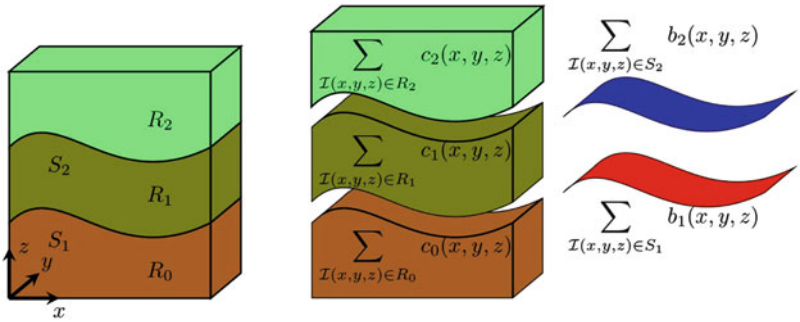

Fig. 16 Example schematic cost of two surfaces for the optimal surface detection problem. The two surfaces divide the volume into three regions

pair (Fig. 15b). A set of λ surfaces $\mathscr{S} = \{S_1, S_2, \ldots, S_\lambda\}$ are considered *feasible* if each individual surface in the set satisfies the given surface-specific smoothness constraints and if each pair of surfaces satisfies the surface separation constraints.

Each voxel $\mathscr{I}(x, y, z)$ may be considered as having an *edge-based* real valued cost $b_i(x, y, z)$ (called an *on-surface cost*) associated with it for each target surface S_i. As shown in Fig. 14b, the λ surfaces form $\lambda + 1$ regions $\{R_0, R_1, \ldots, R_\lambda\}$. For each region R_i $(i = 0, 1, \ldots, \lambda)$, every voxel $\mathscr{I}(x, y, z)$ is assigned a real-valued *region-based cost* $c_i(x, y, z)$ (called an *in-region cost*). The edge-based cost of each voxel in $\mathscr{I}$ is inversely related to the likelihood that it may appear on a desired surface, while the region-based costs $c_i(\cdot)$ $(i = 0, 1, \ldots, \lambda)$ measure the inverse likelihood of a given voxel preserving the expected regional properties (e.g., homogeneity) of the partition $\{R_0, R_1, \ldots, R_\lambda\}$. Then, the total energy $\mathscr{E}(\mathscr{S})$ induced by the λ surfaces in $\mathscr{S}$ is defined as (Fig. 16)

$$\mathscr{E}(\mathscr{S}) = \sum_{i=1}^{\lambda} b_i(S_i) + \sum_{i=0}^{\lambda} c_i(R_i) = \sum_{i=1}^{\lambda} \sum_{\mathscr{I}(x,y,z)\in S_i} b_i(x, y, z) + \sum_{i=0}^{\lambda} \sum_{\mathscr{I}(x,y,z)\in R_i} c_i(x, y, z). \tag{12}$$

The *optimal surface detection* (*OSD*) problem on medical images is: Given a 3D image $\mathscr{I}$ and an integer $\lambda > 0$, find a feasible set of λ surfaces $\mathscr{S} = \{S_1, S_2, \ldots, S_\lambda\}$, such that the total cost $\mathscr{E}(\mathscr{S})$ is minimized.

The OSD problem (for which $\lambda = 1$) was first proposed for the cardiac border detection by Thedens et al. [20, 21]. In the past decades, variations of the OSD problem have been studied extensively in theoretical computer science, computer vision, and operations research. The most related problems include the metric labeling problem in theoretical computer science or so-called MRF optimization problem in computer vision, which introduces an additional pairwise penalty energy in the objective function. The metric labeling problem captures a broad range of classification problems where the quality of a labeling depends on the pairwise relations between the underlying set of objects, such as image restoration [3,22], image segmentation [23–26], visual correspondence [27,28], and deformable registration [29]. After introduced by Kleinberg and Tardos [30], it has been studied extensively in theoretical computer science [30–34]. The best known approximation algorithm for the problem is an $O(\log L)$ (L is the number of labels) [30, 33] and has no $\Omega(\sqrt{\log L})$ approximation unless NP has quasi-polynomial time algorithms [32]. Due to the application nature of the problem, researchers in image processing and computer vision have also developed a variety of good heuristics that use classical combinatorial optimization techniques, such as network flow and local search (e.g., [22–24, 27, 35]), for solving some special cases of the metric labeling problem. This section focuses on polynomial-time solutions to the OSD problem.

4.2 Overview of the OSD Algorithms

The basic idea for solving the OSD problem is to formulate it as computing a minimum-cost closed set in a directed graph with arbitrary node weights. A *closed set* $\mathscr{C}$ in a directed graph with real-valued node weights is a subset of graph nodes such that there is no arc from a node in $\mathscr{C}$ to a node in the complement of $\mathscr{C}$. The cost of a closed set $\mathscr{C}$ is simply the total sum of the weights of all nodes in $\mathscr{C}$. It is well known that a minimum-cost closed set can be computed in low-order polynomial time by solving a minimum s–t cut problem [25, 36]. The solution to the OSD problem is built upon novel observations that capture the *self-closure structures* of the OSD problem, which relate the target problem to the minimum-cost closed set problem. The OSD algorithm uses the following three major steps:

- *Graph construction.* Construct a node-weighted directed graph $G = (V, E)$, which consists of λ node-disjoint subgraphs $G_i = (V_i, E_i)$. Each subgraph G_i is used for identifying the ith surface in $\mathscr{I}$. The construction hinges on the self-closure structures of the OSD problem. The surface smoothness and separation constraints are enforced with graph arcs; while the optimality of the solution is encoded with node weights.

- *Computing a minimum-cost closed set.* Compute a minimum-cost nonempty closed set $\mathscr{C}^*$ in G, which is solvable by computing a minimum s–t cut in an arc-weighted directed graph transformed from G.
- *Surface recovery.* The set of λ optimal surfaces is recovered from the minimum-cost closed set $\mathscr{C}^*$ with each surface being specified by the envelop of $\mathscr{C}^* \cap V_i$.

4.3 Optimal Single Surface Detection

This section presents the algorithm for solving the optimal single surface detection (OSSD) problem (in which $\lambda = 1$).

To facilitate the discussion of transforming the OSD to a minimum-cost closed set problem, we introduce the concept of bottom-most neighbors and the intra-layer self-closure structure of the OSSD problem. For a voxel $\mathscr{I}(x, y, z)$ and each neighboring column $\mathrm{Col}(x', y')$ of $\mathrm{Col}(x, y)$, the *bottom-most neighbor* of $\mathscr{I}(x, y, z)$ on $\mathrm{Col}(x', y')$ is the voxel on $\mathrm{Col}(x', y')$ with the smallest **z**-coordinate that can appear together with $\mathscr{I}(x, y, z)$ on the same feasible surface in $\mathscr{I}$ (Fig. 17a). Note that the bottom-most neighbor of $\mathscr{I}(x, y, z)$ on $\mathrm{Col}(x + 1, y)$ (resp., $\mathrm{Col}(x, y + 1)$) is the voxel $\mathscr{I}(x+1, y, \max\{z-\Delta_{\mathbf{x}}, 0\})$ (resp, $\mathscr{I}(x, y+1, \max\{z-\Delta_{\mathbf{y}}, 0\})$). For a feasible surface S, denote by $S(x, y)$ the z-coordinate of the voxel $\mathscr{I}(x, y, z)$ on the surface S. We say that a voxel $\mathscr{I}(x, y, z)$ is *below* a surface S if $S(x, y) > z$, and denote by $BL(S)$ all the voxels of $\mathscr{I}$ that are on or below S. A key observation is that *for any feasible surface S in $\mathscr{I}$, the bottom-most neighbors of every voxel in $BL(S)$ are also contained in $BL(S)$* (Fig. 17b). This intra-layer self-closure property relates our target problem to the minimum closed set problem. In our approach, instead of finding an optimal surface S^* directly, we seek in $\mathscr{I}$ a voxel set $BL(S^*)$, which uniquely defines the surface S^*.

4.3.1 Graph Construction

The construction of the directed node-weighted graph $G = (V, E)$ hinges on the intra-layer self-closure structure of the OSSD problem. Every node $v(x, y, z) \in V$ represents one and only one voxel $\mathscr{I}(x, y, z) \in \mathscr{I}$. G can thus be viewed as a geometric graph defined on a 3D grid. Arcs are added to G to make sure that each closed set includes all the nodes associated with the corresponding surface voxels plus all those below the surface. This is done by adding two types of arcs: *intracolumn arcs* and *intercolumn arcs*. The intracolumn arcs ensure that all nodes below a given node (within one column) are also included in the closed set. The intercolumn arcs ensure that the smoothness constraints are satisfied. As an example in Fig. 17c, we consider the added arcs for one node v not involved in boundary conditions to avoid cluttering the exposition of our key ideas. It will be associated with two intracolumn arcs: one directed towards the node below it in the column and one from the node above it (red arcs in Fig. 17c). Two intercolumn arcs will

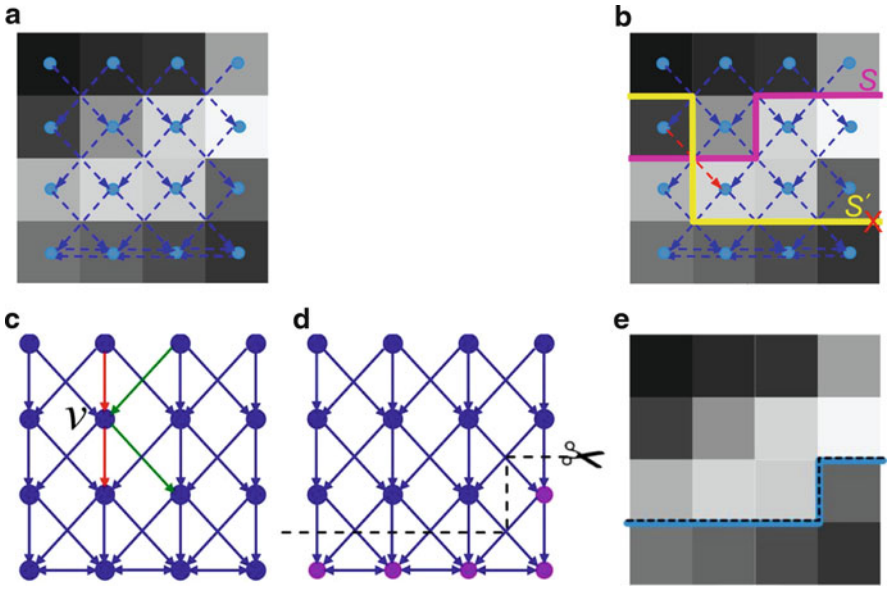

Fig. 17 Modeling single surface detection. An example 2D slice from a 3D image is used. The surface smoothness parameter $\Delta_{\mathbf{x}} = 1$. (**a**) *Dashed line arrows* point to lowest-neighbors of a voxel. (**b**) Illustrating the intra-layer self-closure structure. The surface S is a feasible one while S' is not. The *red line arrow* indicates the violation of the smoothness constraint for S'. (**c**) The constructed graph enforces the surface geometry. The minimum-cost closed set in (**d**) consisting of all *purple nodes* defines the optimal surface in (**e**)

also exist for each neighboring column in the **x**-direction (**y**-direction): one directed to the bottom-most neighbor of v on the neighboring column and one from the node on the neighboring column whose bottom-most neighbor on the column of v is node v (green arcs in Fig. 17c).

To finish the construction of the graph G, we need to encode the energy function Eq. (12) by assigning appropriate weights to the nodes in G. For each node $v(x, y, z) \in V$ corresponding the voxel $\mathscr{I}(x, y, z)$, its weight $w(x, y, z)$ is defined as follows:

$$w(x,y,z) = \begin{cases} b(x,y,z) + [c_0(x,y,z) - c_1(x,y,z)] & \text{if} z = 0, \\ [b(x,y,z) - b(x,y,z-1)] + [c_0(x,y,z) - c_1(x,y,z)] & \text{if} z > 0. \end{cases} \tag{13}$$

The following theorem is the key for solving the OSSD problem as computing a minimum-cost nonempty closed set in G.

Theorem 1 ([7,8]). *(1) Any finite nonempty closed set $\mathscr{C}$ in G specifies a feasible surface S in $\mathscr{I}$ whose total cost $\mathscr{E}(S)$ differs from the cost of $\mathscr{C}$ by a constant. (2) Any feasible surface S in $\mathscr{I}$ corresponds to a nonempty closed set $\mathscr{C}$ in G whose cost differs from $\mathscr{E}(S)$ by a constant.*

4.3.2 Computing a Minimum-Cost Nonempty Closed Set

It is well known that a minimum-cost closed set in a directed node-weighted graph can be computed by solving a minimum s–t cut problem [25, 36]. However, if there is no "negative" cost closed set, the minimum-cost closed set is empty, which offers little useful information for recovering the target surface. In fact, we need to find a minimum-cost "nonempty" closed set.

Based on the construction of G, the bottom-most nodes of all columns form a closed set $\mathscr{C}_0$, that is, $\mathscr{C}_0 = \{v(x, y, 0) \mid 0 \leq x < X, 0 \leq y < Y\}$. Note that $\mathscr{C}_0$ is a subset of any nonempty closed set in G. If the minimum-cost closed set in G is empty, it indicates that the cost of every nonempty closed set in G is non-negative. We do the following transformation to reduce the cost of each closed set by a constant and to make sure that at least one closed set whose cost is negative after the transformation. Let C be the total cost of nodes in $\mathscr{C}_0$; If $C \geq 0$, choose arbitrarily a node $v(x', y', 0) \in \mathscr{C}_0$ and assign a new weight $w(x', y', 0) - C - 1$ to $v(x', y', 0)$. After this transformation, the total cost of the closed set $\mathscr{C}_0$ is negative. Since $\mathscr{C}_0$ is a subset of any nonempty closed set in G, the cost of any nonempty closed set is reduced by $(C + 1)$. Hence, the minimum-cost nonempty closed set in G is not changed after the transformation and it is not empty. Thus, we can simply apply the algorithm in [25, 36] to find the minimum-cost closed set $\mathscr{C}^*$ in G after performing the transformation. $\mathscr{C}^*$ is then a minimum-cost nonempty closed set in G before the transformation.

4.3.3 Surface Recovery

From a minimum-cost nonempty closed set $\mathscr{C}^*$, an optimal surface S^* in $\mathscr{I}$ can be defined as follows. For each (x, y)-pair, let $\mathscr{C}(x, y) \subset \mathscr{C}$ be the set of nodes on the column $\mathrm{Col}(x, y)$ of G. Due to the construction of G, we have $\mathscr{C}_0 \subset \mathscr{C}^*$. Thus, $\mathscr{C}(x, y)$ is not empty. Let $z_{x,y}$ denote the largest z-coordinate of the nodes in $\mathscr{C}(x, y)$. Then, the optimal surface S^* is defined as $S(x, y) = z_{x,y}$ for every (x, y)-pair. Figure 17e shows an example of an optimal surface defined by a minimum-cost closed set in Fig. 17d.

4.4 Optimal Multiple Surface Detection

This section presents the algorithm for solving the Optimal Multiple Surface Detection (OMSD) problem, i.e., simultaneous detection of $\lambda > 1$ interrelated surfaces in a 3D image $\mathscr{I}$ such that the total cost of the λ surfaces is minimized.

In simultaneously detecting multiple distinct but interrelated surfaces, the optimality is not only determined by the inherent costs and smoothness properties of the individual surfaces, but also confined by their interrelations (i.e., the surface separate constraints). Obviously, computing each of the λ surfaces individually using our

algorithm in Sect. 4.3 does not work well. The solution thus obtained might even be infeasible. In addition, the integration of the region-based costs makes the problem more complicated.

To efficiently solve the OMSD problem, the intrinsic common characteristics of the smoothness and the separation constraints are explored to extend the OSSD technique to multiple surface detection. Assume that $\mathscr{S} = \{S_1, S_2, \ldots, S_\lambda\}$ is a feasible set of λ surfaces in $\mathscr{I}$ with S_{i+1} being "on top" of S_i. For each pair of the sought surfaces, S_i and S_{i+1}, two parameters $\delta_i^l \geq 0$ and $\delta_i^u \geq 0$ are used to specify the surface separation constraint (note that the separation constraint can be defined between any pair of the surfaces in $\mathscr{S}$). First, consider each individual surface $S_i \in \mathscr{S}$. Recall that $BL(S_i)$ denotes the subset of all voxels of $\mathscr{I}$ that are on or below S_i. As in Sect. 4.3, we observe that each $BL(S_i)$ also has the intra-layer self-closure structure. However, the OMSD problem is more involved since the λ surfaces in $\mathscr{S}$ are inter-related. The following observation reveals the common essential structure between the smoothness and the separation constraints, leading to further study the closure structure *between* the $BL(S_i)$ and $BL(S_{i+1})$. One may view the 3D image $\mathscr{I}$ as a set of X 2D slices embedded in the **yz**-plane. Thus, a feasible surface S of $\mathscr{I}$ is participated into X **z**-monotone curves with each in a 2D slice. Observe that each feasible **z**-monotone curve is subjective to the smoothness constraint in the corresponding slice, and any pair of adjacent **z**-monotone curves expresses to meet the analogical separation constraints. This observation suggests that the surface separation constraint in a d-D image may be viewed as the surface smoothness constraint in the $(d+1)$-D image consisting of the stack of a sequence of λ d-D images. Hence, we intend to map the detection of λ optimal surfaces in d-D to the problem of finding a single optimal surface in $(d+1)$-D. To distinguish the self-closure structures, define below the *upstream* and *downstream* voxels of any voxel $\mathscr{I}(x, y, z)$ in $\mathscr{I}$ for the given surface separation parameters δ_i^l and δ_i^u: if $\mathscr{I}(x, y, z) \in S_i$, then the ith upstream (resp., downstream) voxel of $\mathscr{I}(x, y, z)$ is the voxel on column $\mathrm{Col}(x, y)$ with the smallest **z**-coordinate that can be on S_{i+1} (resp., S_{i-1}). Together with the intra-layer self-closure structures, the following *inter-layer self-closure* structure is the key for solving the OMSD problem: *Given any set $\mathscr{S}$ of λ feasible surfaces in $\mathscr{I}$, the ith upstream (resp., downstream) voxel of each voxel in $BL(S_i)$ is in $BL(S_{i+1})$ (resp., $BL(S_{i-1})$), for every $1 \leq i < \lambda$ (resp., $1 < i \leq \lambda$).* Both the intra-layer and inter-layer self-closure structures together bridge the OMSD problem and the minimum closed set problem.

4.4.1 Graph Construction

The graph $G = (V, E)$ constructed for solving the OMSD problem consists of λ node-disjoint subgraphs $\{G_i = (V_i, E_i) : i = 1, 2, \ldots, \lambda\}$. Each G_i is constructed as in Sect. 4.3 to reflect the intra-layer self-closure structure of each surface and is used for searching the ith surface S_i. Every node $v_i(x, y, z) \in V_i$ represents one and only one voxel $\mathscr{I}(x, y, z)$. The intracolumn and intercolumn arcs are added in E_i to reflect the intra-layer self-closure structure of surface S_i, which enforce the

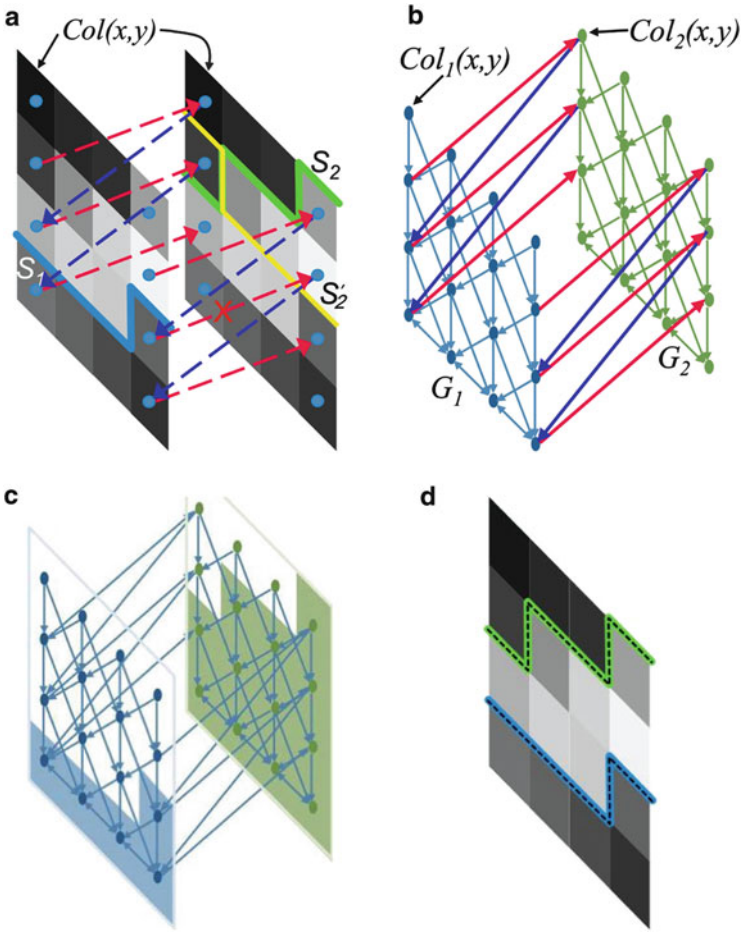

Fig. 18 Modeling multiple surface detection. An example 2D slice from a 3D image is used. Two surfaces are sought and the distance in between ranges from $\delta^l = 1$ to $\delta^u = 2$. The surface smoothness parameter $\Delta_{\mathbf{x}} = 1$. (**a**) Illustrating the inter-layer self-closure structure. For the visualization purpose, the slice is duplicated and the sought two surfaces are visualized in the two separated slices. The *red dashed line arrows* point to the upstream voxels and the *blue dashed line arrows* point to the downstream voxels. S_1 and S_2 are two feasible surfaces, but S_1 and S_2' are not. The *red line arrow* with a mark "X" indicates the violation of the surface separation constraint. (**b**) The constructed graph to enforce the surface separation constraints (arcs are only shown for the first and the last columns). The envelopes of the minimum-cost closed set in (**c**) consisting of all shaded nodes defines the optimal surfaces in (**d**)

monotonicity and the smoothness constraint of each sought surface. The separation constraints between any two surfaces S_i and S_j are enforced in G by a set of arcs E_s, connecting the corresponding subgraphs G_i and G_j, in a way to reflect the inter-layer self-closure structure. Suppose for the two sought surfaces S_i and S_j, the prior knowledge puts S_i below S_j (Fig. 18a in which $i = 1$ and $j = 2$).

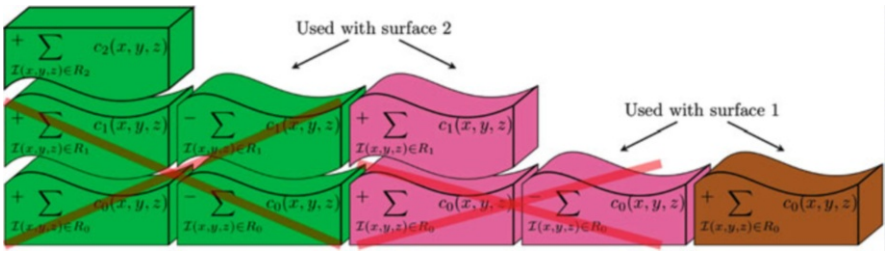

Fig. 19 Schematic showing how the assignment of node weights encodes the total in-region cost of the objective function $\mathscr{E}(\mathscr{S})$

Assume that the distance between S_i and S_j ranges from $\delta_{ij}^l = 1$ to $\delta_{ij}^u = 2$. For convenience, denote by $\mathrm{Col}_i(x, y)$ ($\mathrm{Col}_j(x, y)$) the column of nodes in G_i (G_j) corresponding to the column $\mathrm{Col}(x, y)$ of voxels in $\mathscr{I}$. The separation constraints between S_i and S_j are incorporated into G in the following way. For each node $v_i(x, y, z)$ on $\mathrm{Col}_i(x, y)$ with $z < Z - \delta_{ij}^l$, as illustrated in Fig. 18b, an arc is put in E_s from $v_i(x, y, z)$ to $v_i(x, y, z + \delta_{ij}^l)$ in G_i (red arcs in Fig. 18b). Note that $\mathscr{I}(x, y, z + \delta_{ij}^l)$ is the upstream voxel of $\mathscr{I}(x, y, z)$ (red dashed line arrows in Fig. 18a). On the other hand, each node $v_j(x, y, z)$ on $\mathrm{Col}_j(x, y)$ with $z \geq \delta_{ij}^l$ has an arc in E_s to $v_i(x, y, z^o) \in \mathrm{Col}_i(x, y)$ with $z^o = \max\{0, z - \delta_{ij}^u\}$ (blue arcs in Fig. 18b). $\mathscr{I}(x, y, z^o)$ is the downstream voxel of $\mathscr{I}(x, y, z)$ (blue dashed line arrows in Fig. 18a). The node set of G is the union of the node sets of the λ subgraphs (i.e., $V = \cup_{i=1}^{\lambda} V_i$), and the arc set of G is the union of the arc sets of all the subgraphs plus E_s (i.e., $E = \cup_{i=1}^{\lambda} E_i \cup E_s$).

To encode the objective function $\mathscr{E}(\mathscr{S})$ in the graph model, we assign the node weight for each node $v_i(x, y, z)$ as follows:

$$w_i(x, y, z) = \begin{cases} b_i(x, y, z) + [c_{i-1}(x, y, z) - c_i(x, y, z)] & \text{if } z = 0, \\ [b_i(x, y, z) - b_i(x, y, z-1)] + [c_{i-1}(x, y, z) - c_i(x, y, z)] & \text{for } z > 0. \end{cases} \tag{14}$$

Figure 19 illustrates the intuition behind the node weight assignment to encode the objective function, in which two surfaces are simultaneously detected. Readers are referred to [8] for the formal proof.

4.4.2 Computing a Minimum-Cost Nonempty Closed Set

Note that our goal is to compute a minimum-cost nonempty closed set in G, which can be used to define an optimal set of λ surfaces in $\mathscr{I}$. However, the constructed graph G so far does not yet work for this purpose. In the graph construction above for G_i and G_j, the node $v_i(x, y, z)$ with $z \geq Z - \delta_{ij}^l$ has no arc to any node on $\mathrm{Col}_j(x, y)$, and there is no arc from the node $v_j(x, y, z)$ with $z < \delta_{ij}^l$ to any node on $\mathrm{Col}_i(x, y)$. Those nodes are called *deficient nodes*. The voxels corresponding to

the deficient nodes cannot be on any corresponding feasible surfaces. We need to remove those deficient nodes from G. Otherwise, if a closed set $\mathscr{C}$ in G includes a deficient node as the topmost node on a column that is in $\mathscr{C}$, which is possible, then it does not define a set of λ feasible surfaces in $\mathscr{I}$. However, the removal of deficient nodes may in turn cause other nodes in G to become deficient (i.e., the voxels associated with those nodes cannot be on any corresponding feasible surfaces). We can apply the deficient node pruning scheme in [8] to remove all the deficient nodes as well as those recursively caused by their removal, resulting in a graph G' without any deficient nodes. Note that in the resulting graph G' if any column $\mathrm{Col}_i(x, y) = \emptyset$, then there is no feasible solution to the OMSD problem. We thus assume in the rest of this section that the OMSD problem admits feasible solutions. For each (x, y)-pair and every $i = 1, 2, \ldots, \lambda$, let $z_i^{\mathrm{bot}}(x, y)$ and $z_i^{\mathrm{top}}(x, y)$ be the smallest and largest z-coordinates of the nodes on the column $\mathrm{Col}_i(x, y)$ of G', respectively. The deficient node pruning process may remove some useful arcs whose ending nodes need to be changed. For any node $v_i(x, y, z)$ with $z < z_i^{\mathrm{bot}}(x, y)$ in G, if it has an incoming arc, the ending node of the arc needs to be changed to the node $v_i(x, y, z_i^{\mathrm{bot}}(x, y))$ in G'.

Denote by Z_0 the set of the bottom-most nodes (i.e., whose z-coordinate is $z < z_i^{\mathrm{bot}}(x, y)$) of every column $\mathrm{Col}_i(x, y)$ in G'. Z_0 in fact is a closed set in G', and for any nonempty closed set $\mathscr{C}'$ in G', Z_0 is a subset of $\mathscr{C}'$ [8]. Thus, the same transformation as that for the OSSD problem can be applied to compute a minimum-cost nonempty closed set $\mathscr{C}'^*$ in G'.

4.4.3 Surface Recovery

From the minimum-cost nonempty closed set $\mathscr{C}'^*$ in G', we define an optimal set $\mathscr{S}^*$ of λ surfaces, $\mathscr{S}^* = \{S_1^*, S_2^*, \ldots, S_\lambda^*\}$, as follows. Let $\mathscr{C}^*$ denote the corresponding node set of $\mathscr{C}'^*$ in G. Recall that we search for each surface S_i^* in the subgraph $G_i = (V_i, E_i)$. For each $i = 1, 2, \ldots, \lambda$, let $\mathscr{C}_i = \mathscr{C} \cap V_i$. Considering any (x, y)-pair, denote by $\mathscr{C}_i(x, y)$ the set of nodes of $\mathscr{C}_i$ that are on $\mathrm{Col}_i(x, y)$ of G_i. Then, the voxel $\mathscr{I}(x, y, z)$ corresponding to the "top" node in $\mathscr{C}_i(x, y)$ (i.e., the node whose z-coordinate is the largest among those in $\mathscr{C}_i(x, y)$) is on the ith optimal surface S_i^*. We thus define an optimal set $\mathscr{S}^*$ of λ surfaces from the minimum-cost nonempty closed set $\mathscr{C}'^*$.

Theorem 2 ([8]). *The OMSD problem is solvable by computing a minimum-cost closed set in a derived graph.*

4.5 Optimal Surface Detection with Shape and Context Priors

Up to this point, in the OSD framework, only node weights are employed in the graph to represent the desired segmentation properties, and the desired surface

smoothness is hard-wired as connectedness of neighboring columns. This representation limits the ability to incorporate a broader variety of a prior knowledge. The connectedness of one voxel to the voxels of its neighboring columns is basically of equal importance in the OSD framework discussed above, which prevents us from fully utilizing image edge information as well as from taking full advantage of shape priors. To make full use of prior information, an arc-weighted graph representation has been proposed, which utilizes the weights of both graph nodes and arcs to incorporate a wider spectrum of constraints [37]. Two additional pairwise terms are added into the energy function, which encode the shape and the context prior information using a set of convex functions. For optimization, the new pairwise terms are enforced by adding specific weighted arcs in the graph. A globally optimal solution is then computed by solving a single maximum flow problem in the graph, which corresponds to optimal surfaces.

4.5.1 Incorporation of Shape and Context Priors

To simplify the notation, we use $p(x, y)$ or simply p to denote a column $\mathrm{Col}(x, y)$ in $\mathscr{I}$. For surface S_i, let $S_i(p)$ denote the z-coordinate of the voxel of the column p on S_i.

The shape prior of a surface is incorporated by enforcing the surface height changes between pairs of neighboring columns. More specifically, for any pair of neighboring columns p and q, the shape change of surface S_i between p and q can be measured by $h^i_{pq} = S_i(p) - S_i(q)$. Assume that $\overline{h}^i_{pq}$ represents the learned shape change model. The deviation of the shape changes from the learned model (that is, $|h^i_{pq} - \overline{h}^i_{pq}|$) is penalized by a convex function $f_s(h^i_{pq} - \overline{h}^i_{pq})$. In addition, we can enforce the possible ranges of the shape change deviations with $|h^i_{pq} - \overline{h}^i_{pq}| \leq \Delta^i_{pq}$ (namely, a *hard shape constraint*), where Δ^i_{pq} is a given shape change deviation parameter between columns p and q for surface S_i.

For a set of surfaces, the context prior is enforced by penalizing the surface distance changes between two adjacent surfaces. Let S_i and S_j be two adjacent surfaces in the set of λ target surfaces, and without loss of generality, assume that S_i is on the top of S_j. The surface distance between S_i and S_j on column p is defined as $d^{ij}_p = S_i(p) - S_j(p)$. Denote by $\overline{d}^{ij}_p$ the learned prior surface distance model. As for the shape prior constraint, the deviation of the surface distances from the learned model (i.e., $d^{ij}_p - \overline{d}^{ij}_p$) is penalized by a convex function $f_c(d^{ij}_p - \overline{d}^{ij}_p)$. We may also enforce the possible range of the surface distance deviation on each column p, for instance, $|d^{ij}_p - \overline{d}^{ij}_p| \leq \delta^{ij}_p$, which is called a *hard context constraint*.

Two additional terms are added into the energy function $\mathscr{E}(\mathscr{S})$ (see Eq. (16)) to incorporate the shape and context priors, with

$$\mathscr{E}(\mathscr{S}) = \sum_{i=1}^{\lambda} \sum_{\mathscr{I}(x,y,z)\in S_i} b_i(x,y,z) + \sum_{i=0}^{\lambda} \sum_{\mathscr{I}(x,y,z)\in R_i} c_i(x,y,z) + \underbrace{\sum_{i=1}^{\lambda} \sum_{(p,q)\in\mathscr{N}_s} f_s(h^i_{pq} - \overline{h}^i_{pq})}_{\text{shape prior penalty term}} + \underbrace{\sum_{p} \sum_{(i,j)\in\mathscr{N}_c} f_c(d^{ij}_p - \overline{d}^{ij}_p)}_{\text{context prior penalty term}}, \quad (15)$$

where $\mathscr{N}_s$ denotes the pairwise neighboring relation of all columns in the image $\mathscr{I}$ and $\mathscr{N}_c$ specifies a set of interacting surface pairs for which the surface context constraint needs to be enforced. The problem is to find an optimal set $\mathscr{S}$ of λ surfaces, such that (1) each surfaces satisfies the hard shape constraint; (2) each pair of surfaces satisfies the hard context constraint; and (3) the energy $\mathscr{E}(\mathscr{S})$ in Eq. (15) is minimized. We call this problem an OMSD with both shape and context priors (OMSD-P).

The basic idea for solving the OMSD-P problem is to transform it to computing a so-called *minimum-cost s-excess set* in a directed graph. Instead of forcing no arc leaving the sought node set as in the minimum-cost closed set problem, the minimum s-excess problem [7, 25, 38] charges a penalty onto each arc leaving the set (i.e., the tail of the arc is in the set while the head is not). More precisely, given a directed graph $G' = (V', E')$, each node $v' \in V'$ having an arbitrary weight $w'(v')$ and each edge $e' = (u', v') \in E'$ having a non-negative cost $c'(u', v')$, the minimum-cost s-excess problem seeks a node subset $\mathscr{X}' \subseteq V'$ such that the cost $\gamma(\mathscr{X}')$ of $\mathscr{X}'$, with $\gamma(\mathscr{X}') = \sum_{v'\in\mathscr{X}'} w'(v') + \sum_{u'\in\mathscr{X}', v'\in V'-\mathscr{X}'} c'(u', v')$, is minimized. Our goal is to construct a both arc- and node-weighted graph so that: (1) the feasibility of the sought surfaces is enforced with the graph structure; (2) the total on-surface and in-region cost of the surfaces is encoded as the total node-weight of the s-excess set; and (3) the total shape-prior penalty and the context-prior penalty is encoded as the total cost of the cut induced by the s-excess set. The minimum-cost s-excess problem can be solved by using a maximum flow algorithm [25].

4.5.2 Arc-Weighted Graph Construction

We construct a graph $G = (V, E)$ consisting of λ node-disjoint subgraphs $\{G_i = (V_i, E_i) \mid i = 1, 2, \ldots, \lambda\}$ to transform the OMSD-P problem into computing a minimum-cost s-excess set in G. Note that the hard shape and context constraints in the OMSD-P problem are essentially the same as the surface smoothness and separation constraints in the OMSD problem. Each subgraph G_i is constructed as in Sect. 4.4 for the OMSD problem. The hard shape constraint is enforced between any two neighboring columns in G_i; and the hard context constraint is encoded between the corresponding columns in G_i and G_j for any two interacting surfaces S_i and S_j. To translate to the minimum-cost s-excess problem, we assign a $+\infty$ weight to

each arc added so far to the graph G. We next put additional arcs in G to incorporate both the shape prior penalty term and the context prior penalty term.

Let $p(x, y)$ and $q(x', y')$ denote any two neighboring columns in $\mathscr{I}$. For the shape prior penalty term, we want to "distribute" the shape prior penalty $f_s(h^i_{pq} - \overline{h}^i_{pq})$ to arcs between the two adjacent columns $\mathrm{Col}_i(p)$ and $\mathrm{Col}_i(q)$ in G_i. Two intertwined questions need to be answered: (1) how to put arcs between the two columns; and (2) how to assign a non-negative cost to each arc (negative arc weights make the minimum-cost s-excess problem computationally intractable)? Fortunately, the convexity of $f_s(\cdot)$ can be used to resolve the problems. Define the (discrete equivalent of) second derivative of $f_s(\cdot)$ as

$$[f_s(\kappa)]'' = [f_s(\kappa + 1) - f_s(\kappa)] - [f_s(\kappa) - f_s(\kappa - 1)].$$

The first derivative of $f_s(\cdot)$, $[f_s(\kappa)]' = f_s(\kappa + 1) - f_s(\kappa)$, will be used to guide the introduction of new arcs between $\mathrm{Col}_i(p)$ and $\mathrm{Col}_i(q)$. Consider each $\kappa = h^i_{pq} - \overline{h}^i_{pq}$ with $-\Delta^i_{pq} < \kappa < \Delta^i_{pq}$. We distinguish the following three cases for all possible z ($0 \le z < Z$). Note that we do not show the boundary conditions to avoid cluttering the exposition of the key ideas.

- If $[f_s(\kappa)]' \ge 0$, an arc with a weight of $[f_s(\kappa)]''$ is added from $v_i(x, y, z)$ to $v_i(x', y', z - \overline{h}^i_{pq} - \kappa)$.
- If $[f_s(\kappa)]' \le 0$, an arc with a weight of $[f_s(\kappa)]''$ is added from $v_i(x', y', z)$ to $v_i(x, y, z + \overline{h}^i_{pq} + \kappa)$.
- Assume that $f_s(\kappa)$ has its minimum at κ_0. We put in an arc from $v_i(x, y, z)$ to $v_i(x', y', z - \overline{h}^i_{pq} - \kappa_0)$ whose weight is $[f_s(\kappa_0^+)]'' := f_s(\kappa_0 + 1) - f_s(\kappa_0)$. In addition, an arc with a weight of $[f_s(\kappa_0^-)]'' := f_s(\kappa_0 - 1) - f_s(\kappa_0)$ is introduced from $v_i(x', y', z)$ to $v_i(x, y, z + \overline{h}^i_{pq} + \kappa_0)$.

Figure 20 shows an example graph construction for incorporating the shape prior penalty term.

The context prior penalty term is enforced in a similar way by introducing weighted arcs between corresponding subgraphs. Suppose S_i and S_j are two interacting surfaces. For each column $p(x, y)$ in $\mathscr{I}$, by making use of the (discrete equivalent of) second derivative of $f_c(\cdot)$, the context prior penalty $f_c(d^{ij}_p - \overline{d}^{ij}_p)$ is distributed to the arcs between the corresponding columns $\mathrm{Col}_i(p)$ and $\mathrm{Col}_j(p)$ in the subgraphs G_i and G_j, respectively. An example construction is shown in Fig. 21.

4.5.3 Computing a Minimum-Cost s-Excess Set

From the construction of the graph G and the argument for the OMSD problem in Sect. 4.4, to show that a minimum-cost s-excess set can be used to define an optimal set of λ surfaces for the OMSD-P problem, we need to demonstrate that: (1) the total weight of the arcs cut by a feasible surface S_i between two neighboring columns

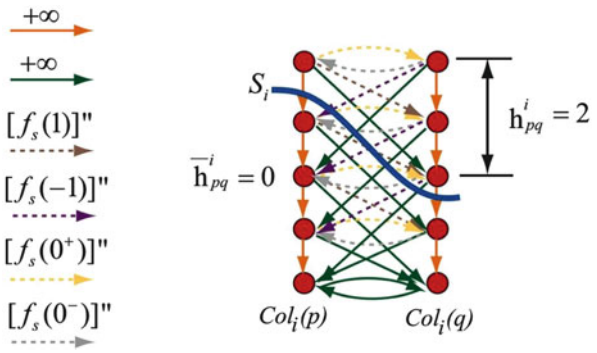

Fig. 20 Arc-weighted graph construction for the incorporation of the shape prior penalty on surface S_i between neighboring columns p and q [37]. The intra-column arcs are shown in *orange* with $+\infty$ weight. The hard shape constraint $|h^i_{pq} - \overline{h}^i_{pq}| \leq \Delta^i_{pq}$ is enforced by *green arcs*. Here we suppose $\overline{h}^i_{pq} = 0$ and $\Delta^i_{pq} = 2$. The shape prior penalty is incorporated by arcs with *dashed lines* (*brown*, *purple*, *yellow*, and *gray*). Here we assume that $[f_s(0)]' = 0$ and $f_s(0) = 0$. The target surface S_i cuts arcs with weight $[f_s(1)]''$ (*brown*) and $[f_s(0^+)]'' = f_s(1) - f_s(0)$ (*yellow*). The total weight is equal to $f_s(2)$

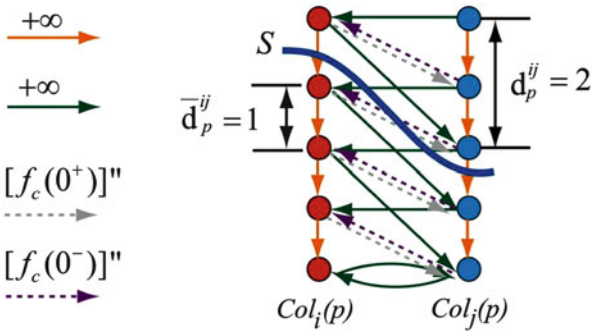

Fig. 21 Arc-weighted graph construction for the incorporation of the context prior constraints between subgraphs G_i (*red*) and G_j (*blue*) on column p [37]. The hard context constraint $|d^{ij}_p - \overline{d}^{ij}_p| \leq \delta^{ij}_p$ is incorporated by *green arcs*. Here $\overline{d}^{ij}_p = 1$, $\delta^{ij}_p = 1$. The context-prior penalty is enforced by *gray* and *purple arcs*. Assume that $[f_c(0)]' = 0$ and $f_c(0) = 0$. The pseudo-surface S (connecting S_i and S_j) cuts arcs with weight $[f_c(0^+)]''$ (*gray*). The total weight is equal to $f_c(1)$

$\mathrm{Col}_i(p)$ and $\mathrm{Col}_i(q)$ equals the shape prior penalty $f_s(h^i_{pq} - \overline{h}^i_{pq})$; and (2) the total weight of the arcs cut by the pseudo-surface of two interacting surfaces S_i and S_j (connecting the node $v_i(p) \in \mathrm{Col}_i(p)$ on the surface S_i and the corresponding node $v_j(p) \in \mathrm{Col}_j(p)$ on the surface S_j for all the columns p) equals the context prior penalty $f_c(d^i_{pq} - \overline{d}^i_{pq})$. The reader is referred to [7] for the detailed proof.

A minimum s-excess set $\mathscr{X}^*$ in G can then be computed by using a minimum s–t cut algorithm [7,25]. If the minimum s-excess set in G thus obtained is empty, we

can first apply on G the same transformation as in Sect. 4.4 to compute a minimum nonempty s-excess set in G.

4.5.4 Surface Recovery

The minimum s-excess set $\mathscr{X}^*$ can be used to specify an optimal set of λ surfaces as in Sect. 4.4.

4.6 Layered Graph Search for Segmentation of Objects with Complex Shapes

This section generalizes the OSD framework for solving the problem of detecting boundary surfaces for objects with complex topologies. Several key issues need to be resolved for the success of this generalization: (i) how to obtain relevant information about the target object boundaries; (ii) how to capture such information by a graph; and (iii) how to search the graph for the optimal surfaces of the target objects. The following five steps constitute the general strategy to address these three key issues.

Step 1: Pre-segmentation. Image segmentation frequently starts with the localization of the object of interest (to distinguish the object from other objects in the image). Given an input 3D/4D image, a pre-segmentation obtains an approximation to the (unknown) surfaces for the target object boundaries. Such approximate surface detection methods are generally available, e.g., using parametric deformable models (cf. [39–42]), geometric deformable models (cf. [43–45]), and other similar approaches. The pre-segmentation gives useful information about the topological structures of the target objects.

Step 2: Mesh Generation. From the resulting approximate surfaces, a mesh is constructed. The mesh is used to specify the structure of a graph G_{B}, called *base graph*. G_{B} defines the neighboring relations among voxels on the sought (optimal) surfaces. Trivial surface geometries (e.g., terrain-like, tubular, or spherical surfaces) may not need a pre-segmentation and allow a direct definition of a mesh.

Step 3: Image Resampling. For each voxel v on the sought surfaces, a vector of voxels is created that is expected to contain v. This is done by resampling the input image along a ray intersecting every vertex u of the mesh (one ray per mesh vertex). These voxel vectors produced by the resampling form a new image. A big challenge for image resampling is how to avoid column interference (see Fig. 22). A few methods based on medial axes [46, 47] and the electric lines of force (ELF) [11] have been developed for resolving this problem. Steps 1–3 were developed for solving the issue (i) from the (i)–(iii) list presented at the beginning of this section.

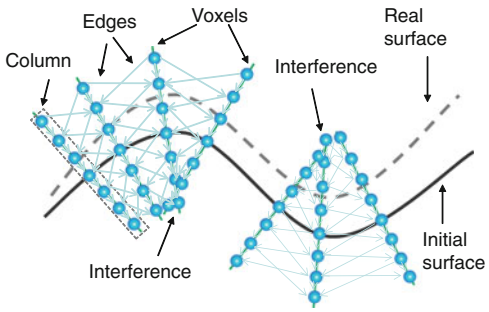

Fig. 22 Illustrating image resampling with column interferences [46]. Interferences caused by inappropriate column lengths introduce disordered structures in the constructed graph

Step 4: Graph Construction. A weighted directed graph G is built using the vectors of voxels (columns) in the resampled image. Each column corresponds to a list of nodes in G. G is a *geometric* graph since it is naturally embedded in a d-D space ($d \geq 3$). The neighboring relations among voxels on the sought surfaces are represented by the adjacency relations among the columns of G, as specified by the edges in the base graph G_B. Each column contains exactly one voxel on the sought surface. The edges of G enforce constraints on the sought surfaces, such as smoothness and inter-surface separation. The node costs of G can encode edge-based and region-based cost functions. This step solves the issue (ii).

Step 5: Graph Search. The OSD scheme ensures that the sought optimal surfaces correspond to a structure of interest in the weighted directed graph G (as proven in [7–9]). The sought optimal surfaces are obtained by searching for a minimum-cost closed set in G. This step solves the issue (iii).

4.7 Example Applications and Extensions

4.7.1 Segmentation of Retinal Layers in Optical Coherence Tomography Volumes

Ophthalmology has recently witnessed a transformation with the relatively recent (since 2007) commercial availability of volumetric spectral-domain optical coherence tomography (SD-OCT) images of the eye. The retinal layers are one important structure within SD-OCT images of the back of the eye as these layers often change in the presence of blinding diseases such as glaucoma, diabetic retinopathy, and age-related macular degeneration. For example, the retinal nerve fiber layer and ganglion cell layer are known to thin in glaucoma. The need for multiple layered surfaces within SD-OCT volumes makes the graph search approach discussed in Sect. 4.4 an excellent choice for their segmentation [10]. For example, Fig. 23 illustrates the use of the graph search approach (with both on-surface and in-region cost terms) for the simultaneous segmentation of seven major surfaces within an SD-OCT slice of the macular region. While the segmentation result of Fig. 23 was obtained through the

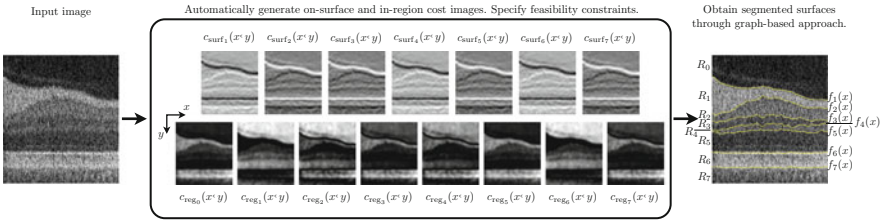

Fig. 23 Example simultaneous segmentation of seven layered surfaces from SD-OCT slice using the multisurface graph-theoretic approach [7, 9, 10]. Given the input image, seven on-surface cost images and eight in-region cost images are first automatically generated. Given these cost images, along with feasibility constraints, the graph-based approach finds the optimal set of surfaces. While 2D images are shown for illustrative purposes, in practice the cost functions and generated and layer segmentations are obtained in 3D

simultaneous segmentation of all seven surfaces using the graph search approach, in practice, the surface segmentation often occurs in groups with the "easier" bounding surfaces being simultaneously segmented first (e.g., surfaces f_1, f_6, and f_7 from Fig. 23) followed by the simultaneous segmentation of the more difficult interior surfaces [10,48]. In addition, a multi-resolution approach is often used for efficiency purposes [49, 50] with up to 11 surfaces commonly being segmented, depending on the ophthalmic application. The cost functions for use in the segmentation of the retinal layers have been both designed primarily "by hand" [10, 51] (e.g., specifying an edge-based filter for bright-to-dark and dark-to-bright on-surface edge-based terms) in possible combination with learning the parameters for specifying the relative importance between in-region and on-surface cost terms as in [10] and/or using a machine-learning approach for a more complete automated design on the cost function terms [52].

In specifying the feasibility constraints for use for the simultaneous segmentation of the retinal layers, instead of specifying global smoothness and surface-interaction constraints (as described in the early versions of the graph search approach), one can specify varying constraints such that each location (i.e., each pair of neighboring columns for smoothness constants and each column for surface-interaction constraints) has its own (potentially different) values for the constraint [10]. The particular values of these location-specific constraints can be learned from a training set [10] to enable, for example, greater changes in surface position to be allowed near the expected location of the fovea of the macula and the cup of the optic nerve head. Soft smoothness constraints can also be imposed through the addition of arc-weighted edges [37] as discussed in Sect. 4.5.

4.7.2 Segmentation of Prostate and Bladder in Computed Tomography Scans

This section shows how to apply the OSD framework for the segmentation of prostate and bladder in CT images [53–55]. The segmentation of pelvic structure is

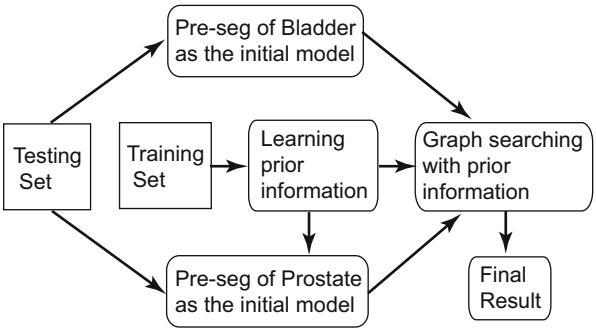

Fig. 24 Workflow for simultaneous segmentation of the bladder and the prostate

particularly difficulty. It involves soft tissues that present a large variability in shape and size. Those soft tissues also have similar intensity and have mutual influence in position and shape. To overcome these difficulties, multiple constraints (i.e., the shape prior and the context prior) are incorporated into the segmentation process using the method in Sect. 4.6. The workflow is shown in Fig. 24.

Initial Model. First, a pre-segmentation step is performed to construct an initial model for each target object, which contains the basic topological information. A 3D geodesic active contour method [56] is conducted for pre-segmentation of the bladder. Three user-defined points are required as an initial input. The prostate shows a much better coherency in shape than the bladder. Hence the mean shape of the prostate is computed from the training set of manual contours. Then an approximate bounding box of interest for the prostate is interactively defined and the obtained mean shape is roughly fitted into the never-before seen CT images using rigid transformations as the initial model of the prostate. Note that the model only serves to provide basic topological structural information; thus accurate segmentation is not required at this stage. Overlapping between the models of the bladder and the prostate is also allowed, which can be resolved in the graph optimization step. From the pre-segmentation results, two triangulated meshes $M_1(V_1, E_1)$ and $M_2(V_2, E_2)$ are constructed respectively using an isosurfacing algorithm (Fig. 25), where V_i $(i \in 1, 2)$ denotes the vertex set of M_i and E_i denotes the edge set of M_i.

Graph Construction. The weighted graph $G_i(N, A)$ is built from the triangulated mesh M_i as follows. For each vertex $v \in V_i$, a column of K nodes $n(v, k)$ is created in G_i, denoted by $p(v)$ (Fig. 25b). The positions of nodes reflect the positions of corresponding voxels in the image domain. The length of the column is set according to the required search range. The number of nodes K on each column is determined by the required resolution. The direction of the column is set as the triangle normal. The nodes on the same column are connected by the intra-column arc from $n(v, k)$ to $n(v, k-1)$ with an infinity weight, as described in Sect. 4.3.

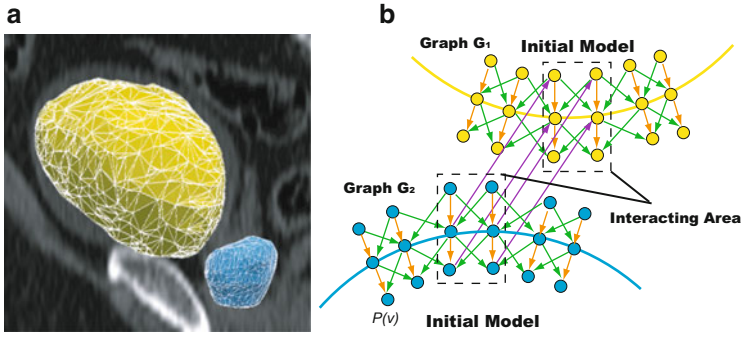

Fig. 25 (**a**) Triangulated meshes for the bladder (*yellow*) and the prostate (*blue*) based on initial models. (**b**) Corresponding graph construction. An example 2D slice is presented. $p(v)$ represents the column with respect to the vertex v on the mesh. *Dots* represent nodes $n_i \in G_i$. Two subgraphs G_1 and G_2 are constructed for the segmentation of the bladder and the prostate, respectively. Note that in the interacting region (*dashed box*), for each column $p(v_1) \in G_1$, there exists a corresponding column $p(v_2) \in G_2$ with the same position. The inter-surface arcs (*purple*) between corresponding columns enforce the surface context constraints in the interacting region

The intra-column arcs make sure that the surface containing exactly one node in each column.

Each column also has a set of neighbors, i.e., if $(v, u) \in E_i$, then $p(v)$ and $p(u)$ are neighboring columns. The shape prior penalty serves to keep the topology of the original shape model. Specifically, for any pair of neighboring columns p and q, the shape change is defined by $h^i_{pq} = S_i(p) - S_i(q)$. Note that the graph is constructed based on the mesh of the initial shape model. Thus the original shape prior $\overline{h}^i_{pq}$ is equal to 0. The shape prior penalty of surface S_i is set as $f_s(h^i_{pq})$, where $f_s(\cdot)$ is a convex function penalizing the shape changes of S_i on p and q. For bladder and prostate segmentation, the penalty function is set with the form: $f_s(h^i_{pq}) = \beta \cdot (h^i_{pq})^2$, where β is a parameter learned from the training data. The shape prior is enforced by introducing additional arcs as in Sect. 4.5.

To avoid the overlapping of two target surfaces, a "partially interacting area" is defined according to the distance between two meshes, which indicates that the two target surfaces may mutually interact with each other at that area. To model the context relation, for each column $p(v_1) \in G_1$ in the partially interacting area, there exists a corresponding column $p(v_2) \in G_2$ with the same position in $\mathscr{I}$, and the target surfaces S_1 and S_2 both cut those columns, as shown in Fig. 25b. For implementation, a one-to-one correspondence between two surface meshes needs to be computed on the partially interacting region. We project the pre-segmented prostate surface mesh on the interacting region to the mesh of the pre-segmented bladder boundary surface. Then we use the projected mesh patch to replace the original bladder surface mesh on the interacting region. Thus, a one-to-one mesh correspondence on the interacting region is established since the two new meshes on that area have exactly the same topological structure. In this way, two graphs share

the same nodes' positions in the partially interacting area. The non-overlapping constraint is enforced in the area as the hard context prior constraint by adding arcs between corresponding columns using the approach proposed in Sect. 4.5. No soft context prior penalties are introduced in our energy.

The optimal set $\mathscr{S}$ of two surfaces corresponding to the bladder and the prostate can then be found by minimizing the following energy through the constructed graph:

$$\mathscr{E}(\mathscr{S}) = \sum_{i=1}^{2} \mathscr{E}_{\text{boundary}}(S_i) + \sum_{i=0}^{2} \mathscr{E}_{\text{region}}(R_i) + \sum_{i=1}^{2} \mathscr{E}_{\text{shape}}(S_i) \tag{16}$$

The boundary energy term serves as an external force, which drives the mesh towards the best fit to the image data. The shape energy term functions as an internal force, which keeps the shape of the original model and restricts the flexibility of the mesh. To incorporate the learned regional information, an additional region energy term is added in our energy function. Specifically, two surfaces for the bladder and the prostate naturally divide the volume into three regions denoted by R_0, R_1, and R_2, which corresponds to the region enclosed by the bladder surface S_1, one between S_1 and the prostate surface S_2 at the partially interacting area, and the region enclosed by S_2, respectively. Our region energy term $E_{\text{region}}(R_i)$ reflects the region property of all voxels inside R_i. An example cost function design for this application can be referred to [55].

The illustrative results in three views are displayed in Fig. 26a–d. The 3D representation is shown in Fig. 26e. From all views, the proposed algorithm produces a very good delineation of both the bladder and the prostate in the 3D space. The shape prior constraints succeed to keep the original topological structure of the target organs. No overlapping of the bladder and the prostate is found due to the enforcement of the context constraints.

4.7.3 Robust Delineation of Pulmonary Tumors Using Surface-Region Context

This section presents a segmentation method that integrates the graph cut method with the OSD method for segmenting the target objects of an arbitrary shape mutually interacting with terrain-like surfaces [57]. This scenario widely exists in the medical imaging field. For instance, lung tumors may be in touch with lung parenchyma or close to the diaphragm (Fig. 27a), and fluid and fluid-associated abnormalities may appear in between the retinal layers (Fig. 27b). The delineation of such a target object could be very challenging due to the similar intensity profiles of the surrounding tissues. However, in that setting, the boundary surfaces of the adjacent structures of the target object can serve as valuable prior information to help separate the target object from those structures. It is expected to be promising to simultaneously segment those boundary surfaces together with the target object.

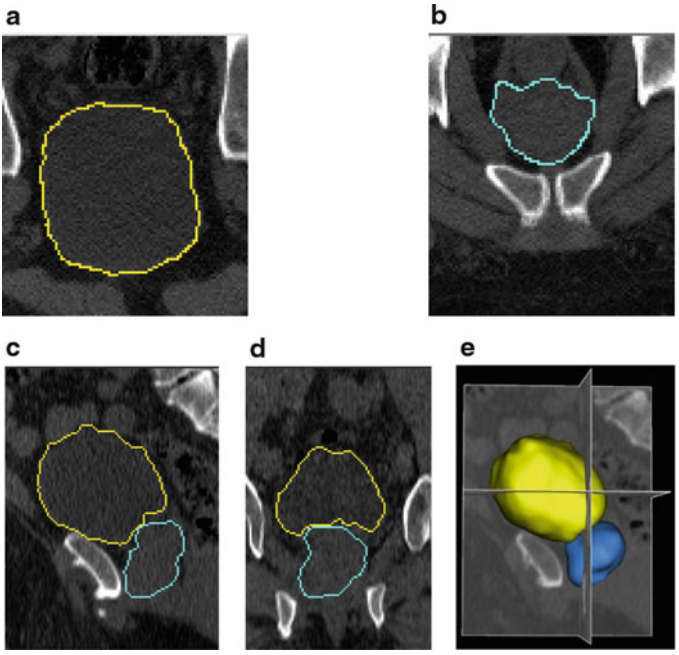

Fig. 26 Typical slices of the simultaneous segmentation result of the bladder (*yellow*) and the prostate (*blue*) in 3D CT images. (**a** and **b**) Transverse view. (**c**) Sagittal view. (**d**) Coronal view. (**e**) 3D representation of the segmentation result

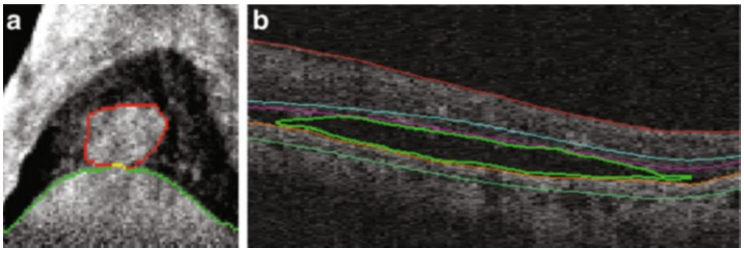

Fig. 27 Examples of surface and region interactions [57]. (**a**) Example slices of a lung tumor (*red*) in megavoltage cone-beam CT. The tumor is attached/adjacent to the diaphragm (*green*). (**b**) Example slices of a fluid-filled region (*green*) in retinal optical coherence tomography (OCT). The fluid region is surrounded by intraretinal layers

The computational feasibility is achieved by integrating the graph cut method in Sect. 3 and the OSD method in Sect. 4. The integration of those two methods into one single optimization process, yet still admitting globally optimal solutions, will certainly become a more powerful tool for medical image analysis.

We illustrate the method using lung tumor segmentation on Mega-Voltage Cone Beam CT (MVCBCT) images. WLOG, assume that the CBCT image $\mathscr{I}$ is oriented

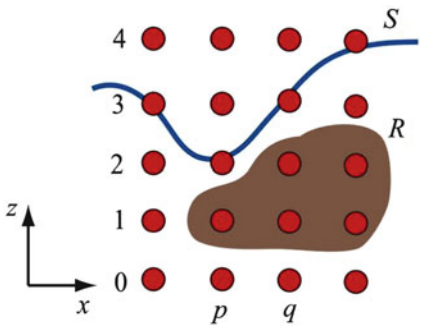

Fig. 28 A sketched example of a 2D slice from a 3D image [57]. The terrain-like surface S is shown in *blue* and the tumor region R of arbitrary shape is shown in *brown*. Col(p) and Col(q) indicate two neighboring columns in the image, and $S(p) = 2$ and $S(q) = 3$

such that the target surface S intersects each column Col(p) of $p(x, y)$ in $\mathscr{I}$, and the target tumor is below the surface (Fig. 28). The same principles used for this illustration are directly applicable to multiple pairs of surfaces and regions with interactions between those two types of targets. The method can also handle the case when the tumor region is above the target surface [57].

The neighborhood setting of those columns is specified by $\mathscr{N}_s$. Let $S(p)$ denote the height (z-coordinate) of the surface S on the column Col(p). The OSD method is adopted to compute the surface S. Each voxel $v(x, y, z) \in \mathscr{I}$ is associated with an on-surface cost $b(x, y, z)$, which is inversely related to the likelihood that the desired surface S indeed contains the voxel. We can also incorporate the shape prior of the target surface S as in Sect. 4.5. The optimal surface S can be computed by minimizing the following energy function:

$$\mathscr{E}_s(S) = \sum_{v(x,y,z)\in S} b(x, y, z) + \sum_{(p,q)\in\mathscr{N}_s} h_{pq}(S(p) - S(q)), \tag{17}$$

where $h_{pq}(\cdot)$ is a convex function.

The graph cut method in Sect. 3 is employed to segment the target tumor region R. Let f_v denote the label of each voxel $v \in \mathscr{I}$: $f_v = 1$ means that v belongs to the target tumor region R and $f_v = 0$ means that v is in the background. The MRF energy $\mathscr{E}_c$ for segmenting the tumor region R can be expressed as follows:

$$\mathscr{E}_c(f) = \sum_{v\in\mathscr{I}} \mathscr{D}_v(f_v) + \sum_{(v_i,v_j)\in\mathscr{N}_c} \mathscr{V}_{i,j}(f_{v_i}, f_{v_j}), \tag{18}$$

where $\mathscr{N}_c$ defines the neighboring setting between voxels, and $\mathscr{D}_v(\cdot)$ and $\mathscr{V}_{i,j}(\cdot, \cdots)$ are the data term and the boundary term in the graph cut method (Sect. 3), respectively.

The geometric interaction relation between the boundary surface S and the target tumor region R is to enforce that R is at least $d > 0$ voxels "below" S. Denote by γ_v the penalty of a tumor voxel $v \in R$ that violates the interaction constraint. We introduce a surface-region interaction penalty term $\mathscr{E}_{\text{surf–tumor}}$, which is the total

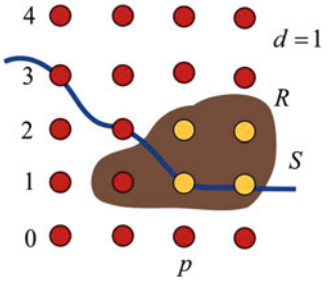

Fig. 29 The surface-region interaction constraint in which the region R tends to be positioned "lower" than the terrain-like surface S [57]. For voxels $v \in R$ and $S(p) - z(v) < d$ with $v \in \text{Col}(p)$ (*yellow voxels*), a penalty γ_v is enforced; d is set as 1

penalty of those tumor voxels that violates the interaction constraint (Fig. 29). More specifically, let $z(v)$ denote the height (z-coordinate) of voxel $v \in \mathscr{I}$, and v is on the column $\text{Col}(p)$ of $p(x, y)$ (that is, the x- and y-coordinates of v is x and y, respectively). Then, if $v \in R$ and $S(p) - z(v) < d$, a penalty γ_v is enforced. Hence,

$$\mathscr{E}_{\text{surf-tumor}}(S, f) = \sum_{v \in \mathscr{I}} \sum_{\substack{v \in \text{Col}(p) \\ S(p) - z(v) < d}} \gamma_v \cdot f_v. \tag{19}$$

To enforce the boundary surface prior, we co-segment the tumor region as well as the boundary surface by minimizing the energy function $\mathscr{E}(S, f)$, with

$$\mathscr{E}(S, f) = \mathscr{E}_s(S) + \mathscr{E}_c(f) + \mathscr{E}_{\text{surf-tumor}}(S, f). \tag{20}$$

To optimize the energy function $\mathscr{E}(S, f)$, two subgraphs, $G_c = (N_c, A_c)$ and $G_s = (N_s, A_s)$, are constructed to encode the terms $\mathscr{E}_c(f)$ and $\mathscr{E}_s(S)$, respectively, using the approaches in Sects. 3 and 4.5. For each voxel $v(x, y, z) \in \mathscr{I}$, denote by $n_c(x, y, z)$ ($n_s(x, y, z)$) the node in G_c (G_s) corresponding to v. The surface-region penalty term $\mathscr{E}_{\text{surf-tumor}}$ is incorporated by adding directed arcs between the corresponding nodes of the two subgraphs with a weight of the penalty for violating the interaction constraint. More specifically, for each voxel $v(x, y, z)$, a directed arc from $n_c(x, y, z)$ to $n_s(x, y, z+d)$, whose weight is γ_v, is added between the two subgraphs G_c and G_s (Fig. 30). Note that we do not consider the boundary conditions here to avoid cluttering the exposition of the key ideas.

Once the graph is constructed, a globally optimal solution can be found by solving a single maximum flow problem, which minimizes the total energy $\mathscr{E}(S, f)$ in a low-order polynomial time [57].

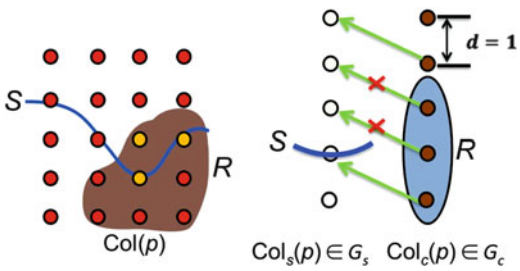

Fig. 30 The introduced arcs in the constructed graph to enforce the surface-region interaction constraint. The node column $\mathrm{Col}_s(p)$ ($\mathrm{Col}_c(p)$) in the subgraph G_s (G_c) corresponds to the column $\mathrm{Col}(p)$ of voxels in the input image. The *green arcs* introduced to enforce the interaction constraint. The two arcs with a *red cross* indicate the penalties for the present surface violating the interaction constraint

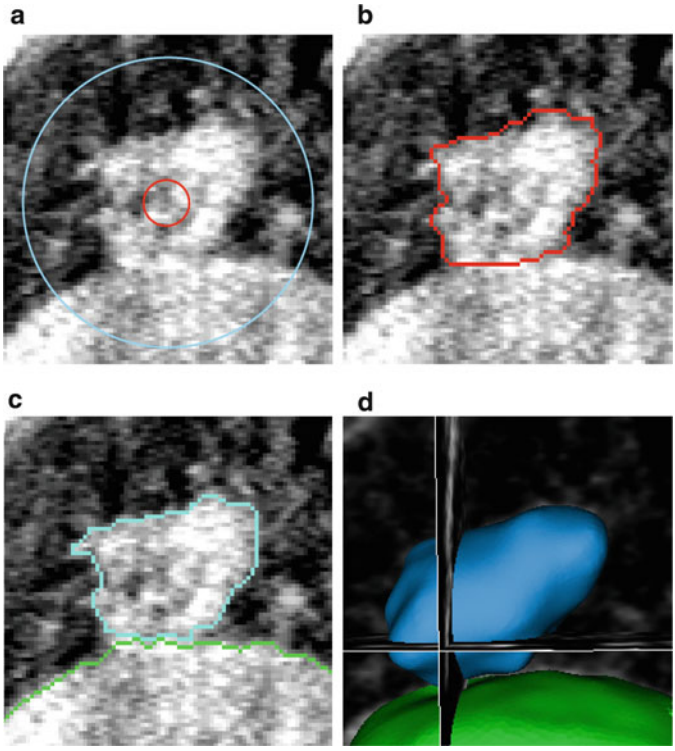

Fig. 31 A typical tumor segmentation result [57]. (**a**) One 2D slice of 3D MVCBCT image with outlines of spherical initialization. (**b**) Manual segmentation of the lung tumor—independent standard. (**c**) Simultaneous region-and-surface segmentation of the diaphragm (*green*) and the lung tumor (*blue*) using the reported approach showing excellent segmentation performance—the Dice similarity coefficient (DSC) is 0.878. (**d**) The 3D representation of the diaphragm (*green*) and the tumor (*blue*)

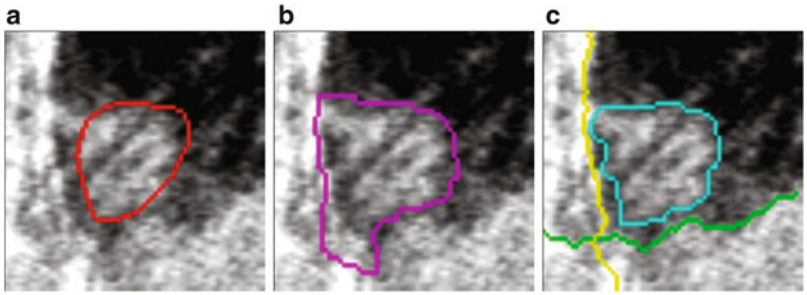

Fig. 32 Performance comparison on a difficult case. (**a**) Independent standard obtained by manual segmentation and shown in one 2D slice of the 3D volume. (**b**) Tumor segmentation failure resulting from the conventional graph cut method—DSC = 0.70. (**c**) Tumor segmentation obtained using the method that simultaneously segments the tumor (*blue*) and the lung boundary surface (shown in *yellow* and *green* in two orthogonal directions)—DSC = 0.84

Figure 31 shows a typical tumor segmentation result on a pulmonary MVCBCT dataset. Figure 32 demonstrates a segmentation result on a difficult case, in which the tumor is closely adjacent to the lung boundary surface from two directions.

References

1. Boykov, Y., Jolly, M.-P.: Interactive organ segmentation using graph cuts. In: Medical Image Computing and Computer-Assisted Intervention (MICCAI) 2000. Lecture Notes in Computer Science, vol. 1935, pp. 276–286. Springer, Berlin (2000)
2. Boykov, Y., Jolly, M.-P.: Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In: IEEE International Conference on Computer Vision (ICCV) 2001, pp. 105–112. IEEE, New York (2001)
3. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **23**(11), 1222–1239 (2001)
4. Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Trans. Pattern Anal. Mach. Intell. **26**(9), 1124–1137 (2004)
5. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts? IEEE Trans. Pattern Anal. Mach. Intell. **26**(2), 147–159 (2004)
6. Boykov, Y., Funka-Lea, G.: Graph cuts and efficient N-D image segmentation. Int. J. Comput. Vis. **70**(2), 109–131 (2006)
7. Wu, X., Chen, D.Z.: Optimal net surface problems with applications. In: Widmayer, P., Eidenbenz, S., Triguero, F., Morales, R., Conejo, R., Hennessy, M. (eds.) Automata, Languages and Programming. Lecture Notes in Computer Science, vol. 2380, pp. 775–775. Springer, Berlin (2002)
8. Wu, X., Chen, D.Z., Li, K., Sonk, M.: The layered net surface problems in discrete geometry and medical image segmentation. Int. J. Comput. Geom. Appl. **17**(3), 261–296 (2007)
9. Li, K., Wu, X., Chen, D.Z., Sonka, M.: Optimal surface segmentation in volumetric images—a graph-theoretic approach. IEEE Trans. Pattern Anal. Mach. Intell. **28**(1), 119–134 (2006)

10. Garvin, M.K., Abràmoff, M.D., Wu, X., Russell, S.R., Burns, T.L., Sonka, M.: Automated 3-D intraretinal layer segmentation of macular spectral-domain optical coherence tomography images. IEEE Trans. Med. Imaging **28**(9), 1436–1447 (2009)
11. Yin, Y., Zhang, X., Williams, R., Wu, X., Anderson, D.D., Sonka, M.: LOGISMOS—layered optimal graph image segmentation of multiple objects and surfaces: cartilage segmentation in the knee joint. IEEE Trans. Med. Imaging **29**(12), 2023–2037 (2010)
12. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd edn. MIT, Cambridge (2001)
13. Delong, A., Boykov, Y.: Globally optimal segmentation of multi-region objects. In: International Conference on Computer Vision (ICCV), Tokyo, pp. 285–292 (October 2009)
14. Song, Q., Bai, J., Han, D., Bhatia, S., Sun, W., Rockey, W., Bayouth, J., Buatti, J., Wu, X.: Optimal co-segmentation of tumor in PET-CT images with context information. IEEE Trans. Med. Imaging **32**(9), 1685–1697 (2013)
15. Mu, Y., Zhou, B.: Co-segmentation of image pairs with quadratic global constraint in MRFs. In: Proceedings of the 8th Asian Conference on Computer Vision, Part II, Tokyo, pp. 837–846 (2007)
16. Mukherjee, L., Singh, V., Dyer, C.R.: Half-integrality based algorithms for cosegmentation of images. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Miami, pp. 2028–2035 (2009)
17. Hochbaum, D.S., Singh, V.: An efficient algorithm for co-segmentation. In: Proceedings of the International Conference on Computer Vision, Kyoto, pp. 269–276 (2009)
18. Rother, C., Minka, T., Bake, A., Kolmogorov, V.: Cosegmentation of image pairs by histogram matching—incorporating a global constraint into MRFs. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, New York, pp. 993–1000 (2006)
19. Batra, D., Kowdle, A., Parikh, D., Luo, J., Chen, T.: Interactively co-segmentating topically related images with intelligent scribble guidance. Int. J. Comput. Vis. **93**(3), 273–292 (2011)
20. Thedens, D.R., Skorton, D.J., Fleagle, S.R.: A three-dimensional graph searching technique for cardiac border detection in sequential images and its application to magnetic resonance image data. In: Proceedings of Computers in Cardiology, Chicago, pp. 57–60 (1990)
21. Thedens, D.R., Skorton, D.J., Fleagle, S.R.: Methods of graph searching for border detection in image sequences with applications to cardiac magnetic resonance imaging. IEEE Trans. Med. Imaging **14**(1), 42–55 (1995)
22. Boykov, Y., Veksler, O., Zabih, R.: Markov Random Fields with efficient approximations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Santa Barbara, pp. 648–655 (1998)
23. Ishikawa, H., Geiger, D.: Segmentation by grouping junctions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Santa Barbara, pp. 125–131 (1998)
24. Greig, D., Porteous, B., Seheult, A.: Exact maximum a posteriori estimation for binary image. J. R. Stat. Soc. Ser. B **51**, 271–279 (1989)
25. Hochbaum, D.S.: An efficient algorithm for image segmentation, Markov random fields and related problems. J. ACM **48**(4), 686–701 (2001)
26. Glocker, B., Komodakis, N., Paragios, N., Glaser, C., Tziritas, G., Navab, N.: Primal/dual linear programming and statistical atlases for cartilage segmentation. In: Ayache, N., Ourselin, S., Maeder, A. (eds.) Medical Image Computing and Computer-Assisted Intervention (MICCAI 2007). Lecture Notes in Computer Science, vol. 4792, pp. 536–543. Springer, Berlin (2007)
27. Roy, S., Cox, I.: A maximum-flow formulation of the n-camera stereo correspondence problem. In: Proceedings of International Conference on Computer Vision (ICCV), Bombay, pp. 492–499 (1998)
28. Kolmogorov, V., Zabih, R.: Computing visual correspondence with occlusions using graph cuts. In: Proceedings of International Conference on Computer Vision (ICCV), Vancouver, pp. 508–515, July 2001

29. Glocker, B., Komodakis, N., Paragios, N., Tziritas, G., Navab, N.: Inter and intra-modal deformable registration: continuous deformations meet efficient optimal linear programming. In: Proceedings of the 20th International Conference on Information Processing in Medical Imaging (IPMI). Lecture Notes in Computer Science, vol. 4584, pp. 408–420. Springer, Berlin (2006)
30. Kleinberg, J., Tardos, E.: Approximation algorithms for classification problems with pairwise relationships: metric labeling and Markov random fields. In: Proceedings of the 40th IEEE Symposium on Foundations of Computer Science, New York City, pp. 14–23 (1999)
31. Archer, A., Fakcharoenphol, J., Harrelson, C., Krauthgamer, R., Talvar, K., Tardos, E.: Approximate classification via earthmover metrics. In: Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, pp. 1079–1089 (2004)
32. Chuzhoy, J., Naor, S.: The hardness of metric labeling. SIAM J. Comput. **36**(5), 1376–1386 (2007)
33. Chekuri, C., Khanna, A., Naor, J., Zosin, L.: A linear programming formulation and approximation algorithms for the metric labeling problem. SIAM J. Discrete Math. **18**(3), 608–625 (2005)
34. Gupta, A., Tardos, E.: Constant factor approximation algorithms for a class of classification problem. In: Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC), Portland, pp. 652–658 (2000)
35. Dubes, R., Jain, A.: Random field models in image analysis. J. Appl. Stat. **16**, 131–164 (1989)
36. Picard, J.C.: Maximal closure of a graph and applications to combinatorial problems. Manag. Sci. **22**, 1268–1272 (1976)
37. Song, Q., Bai, J., Garvin, M.K., Sonka, M., Buatti, J., Wu, X.: Optimal multiple surface segmentation with shape and context priors. IEEE Trans. Med. Imaging **32**(2), 376–386 (2013)
38. Ahuja, R.K., Hochbaum, D., Orlin, J.B.: A cut based algorithm for the convex dual of the minimum cost network flow problem. Algorithmica **39**, 189–208 (2004)
39. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: active contour models. Int. J. Comput. Vis. **1**(4), 321–331 (1987)
40. Amini, A.A., Weymouth, T.E., Jain, R.C.: Using dynamic programming for solving variational problems in vision. IEEE Trans. Pattern Anal. Mach. Intell. **12**(9), 855–867 (1990)
41. Amini, A.A., Weymouth, T.E., Jain, R.C.: On active contour models and balloons. CVGIP: Image Underst. **53**(2), 211–218 (1991)
42. Amini, A.A., Weymouth, T.E., Jain, R.C.: A dynamic finite element surface model for segmentation and tracking in multidimensional medical images with application to cardiac 4D image analysis. Comput. Med. Imaging Graph. **19**(1), 69–83 (1995)
43. Caselles, V., Catte, F., Coll, T., Dibos, F.: A geometric model for active contours. Numerische Mathematik **66**, 1–31 (1993)
44. Malladi, R., Sethian, J.A., Vemuri, C.: Shape modeling with front propagation: a level set approach. IEEE Trans. Pattern Anal. Mach. Intell. **17**(2), 158–175 (1995)
45. Osher, S., Paragios, N.: Geometric Level Set Methods in Imaging, Vision, and Graphics. Springer, New York (2003)
46. Liu, X., Chen, D., Tawhai, M., Wu, X., Hoffman, E., Sonka, M.: Optimal graph search based segmentation of airway tree double surfaces across bifurcations. IEEE Trans. Med. Imaging **32**(3), 493–510 (2013)
47. Liu, X., Chen, D.Z., Wu, X., Sonka, M.: Optimal graph-based segmentation of 3D pulmonary airway and vascular trees across bifurcations. In: First International Workshop on Pulmonary Image Analysis, at the 11th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), New York City, pp. 103–111 (September 2008)
48. Garvin, M.K., Abràmoff, M.D., Kardon, R., Russell, S.R., Wu, X., Sonka, M.: Intraretinal layer segmentation of macular optical coherence tomography images using optimal 3-D graph search. IEEE Trans. Med. Imaging **27**(10), 1495–1505 (2008)
49. Lee, K., Abràmoff, M.D., Niemeijer, M., Garvin, M.K., Sonka, M.: 3-D segmentation of retinal blood vessels in spectral-domain OCT volumes of the optic nerve head. In: Proceedings of SPIE Medical Imaging 2010: Biomedical Applications in Molecular, Structural, and Functional Imaging, vol. 7626, San Diego, p. 76260V-8. SPIE (2010)

50. Quellec, G., Lee, K., Dolejsi, M., Garvin, M.K., Abràmoff, M.D., Sonka, M.: Three-dimensional analysis of retinal layer texture: identification of fluid-filled regions in SD-OCT of the macula. IEEE Trans. Med. Imaging **29**(6), 1321–1330 (2010)
51. Lee, K., Niemeijer, M., Garvin, M.K., Kwon, Y.H., Sonka, M., Abràmoff, M.D.: Segmentation of the optic disc in 3-D OCT scans of the optic nerve head. IEEE Trans. Med. Imaging **29**(1), 159–168 (2010)
52. Antony, B.J., Abràmoff, M.D., Lee, K., Sonkova, P., Gupta, P., Kwon, Y., Niemeijer, M., Hu, Z., Garvin, M.K.: Automated 3D segmentation of intraretinal layers from optic nerve head optical coherence tomography images. In: Proceedings of SPIE Medical Imaging 2010: Biomedical Applications in Molecular, Structural, and Functional Imaging, vol. 7626, San Diego, p. 76260U-12. SPIE (2010)
53. Song, Q., Wu, X., Liu, Y., Smith, M., Buatti, J., Sonka, M.: Optimal graph search segmentation using arc-weighted graph for simultaneous surface detection of bladder and prostate. In: Yang, G.-Z., Hawkes, D., Rueckert, D., Noble, A., Taylor, C. (eds.) Medical Image Computing and Computer-Assisted Intervention—MICCAI 2009. Lecture Notes in Computer Science, vol. 5762, pp. 827–835. Springer, Berlin (2009)
54. Song, Q., Wu, X., Liu, Y., Sonka, M., Garvin, M.K.: Simultaneous searching of globally optimal interacting surfaces with shape priors. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, pp. 2879–2886 (June 2010)
55. Song, Q., Liu, Y., Liu, Y., Saha, P., Sonka, M., Wu, X.: Graph search with appearance and shape information for 3-D prostate and bladder segmentation. In: Jiang, T., Navab, N., Pluim, J., Viergever, M. (eds.) Medical Image Computing and Computer-Assisted Intervention—MICCAI 2010. Lecture Notes in Computer Science, vol. 6363, pp. 172–180. Springer, Berlin (2010)
56. Caselles, V., Kimmel, R., Sapiro, G.: Geodesic active contours. Int. J. Comput. Vis. **22**, 61–97 (1997)
57. Song, Q., Chen, M., Bai, J., Sonka, M., Wu, X.: Surface-Region context in optimal multi-object Graph-Based segmentation: robust delineation of pulmonary tumors. In: Szekely, G., Hahn, H. (eds.) Information Processing in Medical Imaging. Lecture Notes in Computer Science, vol. 6801, pp. 61–72. Springer, Berlin (2011)