Multi-object Graph-based Segmentation with Non-overlapping Surfaces

Patrick M. Jensen, Anders B. Dahl, Vedrana A. Dahl
Department of Applied Mathematics and Computer Science
Technical University of Denmark, Kgs. Lyngby, Denmark
{patmjen,abda,vand}@dtu.dk

Abstract

For 3D images, segmentation via fitting surface meshes to object boundaries provides an efficient way to handle large images and enforce geometric prior knowledge. Furthermore, fitting such meshes with graph cuts has proven to be a versatile and robust framework. However, when segmenting multiple distinct objects in one image, current methods do not allow the natural constraint that objects should not overlap. In this paper, we present an extension to graph cut based methods which can provide a globally optimal segmentation of thousands of objects while guaranteeing no overlap. Our method works by separating objects with planes whose positions are determined as part of the graph cut. To demonstrate the general applicability of our method, we apply it to several 3D microscopy data sets from both biology and materials science. Our results show both quantitative and qualitative improvements.

1. Introduction

3D microscopy includes techniques such as X-ray micro- or nano-CT, light-sheet microscopy, optical coherence tomography, and confocal microscopy. These techniques, widely used in materials and bio-science, often result in large 3D images that are difficult to interpret through visual inspection. It is therefore important to be able to quantify 3D structures, *e.g.* the size of cells, and here segmentation is an essential tool.

Segmenting multiple objects that are densely packed can easily result in an overlap between the detected objects, which is not physically possible. Therefore, it is often desirable to constrain the segmentation such that the detected objects do not intersect. Besides giving a sensible solution, avoiding overlap also regularizes the segmentation in general. Thus, the resulting segmentation of individual objects is more accurate than without this constraint.

In this work, we propose a method for accurately segmenting multiple densely packed objects while preventing overlap. Our approach uses an s-t graph cut to fit surface meshes to object boundaries. Using surface meshes provides a compact representation of the segmentation, allowing us to process large volumes with a large number of objects.

In general, incorporating non-overlap constraints directly into the graph cut formulation is challenging, and existing approaches are based on assumptions that limits their application. Examples are situations where only a few objects are to be detected, where objects have a certain shape, or where user-provided scribbles are available.

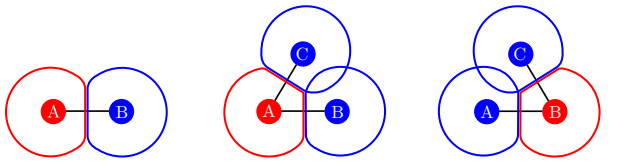
We propose a method to incorporate non-overlap constraints for problems where pairs of interacting objects can be separated by a plane. This is always the case for convex objects [2], but can also be valid for non-convex shapes. As a result, our approach can be applied to a range of problems involving the segmentation of non-overlapping densely packed objects. Our method allows us to simultaneously detect thousands of non-overlapping objects by finding a globally optimal solution to the modeled segmentation problem. We provide an implementation of the method at: https://github.com/patmjen/NOS.

1.1. Approaches for preventing overlap

By definition, an s-t graph cut provides a bipartition of a graph, so binary segmentation is the most straightforward use of the s-t graph cut in image analysis. Segmenting multiple objects with a single graph involves constructing sub-graphs dedicated to each object. This opens a possibility for adding interactions between objects, such as *containment* and *exclusion*, by adding edges between nodes of the sub-graphs. While containment is relatively easy to enforce, exclusion (preventing overlap) is in general challenging.

Several authors have explored the topic of preventing overlap in graph-based multi-object segmentation [1, 7, 12, 20, 25]. The key challenge is that graph cuts can only be used to minimize so-called submodular energy functions. Adding Exclusion typically results in non-submodularity and is therefore not straightforward to model with graphs. A common way to circumvent this is to represent some objects with their complement, which turns exclusion between A and B into containment of A in B^c. This means that non-

overlap may only be enforced between pairs of surfaces. More specifically, we must divide our surfaces into two groups, and separation can only be enforced between two surfaces if they are in different groups. A simple example is illustrated in Figure 1.



(a) Surfaces A and B split into groups {A} and {B}.

(b) Surfaces A, B, and C grouped as {A} and {B, C}.

(c) Surfaces A, B, and C grouped as {A, C} and {B}.

Figure 1: Usual construction of separation constraints for multiple surfaces. For the case in (a) we can easily split two surfaces into groups. For (b) and (c) there will always be a pair of surfaces with no separation constraints.

Pairwise exclusion may suffice for some applications, especially if only a few objects are to be segmented, *e.g.* organ segmentation in medical image analysis. However, for our problem this type of construction is not adequate – we will need to support separation constraints between an arbitrary number of surfaces. Furthermore, previous methods usually require that we identify a region of interaction for pairs of surfaces, and then add edges between corresponding nodes in the graph for each surface [12, 20, 25]. This may require re-meshing of the surfaces or significant changes to the graph, which in turn may be highly non-trivial if several surfaces share a region of interaction.

A way of handling non-submodular energy terms is to use a more general framework, quadratic pseudo-boolean optimization (QPBO). If QPBO provides labeling for all nodes, the solution is guaranteed to be optimal. However, QPBO might leave some graph nodes unlabeled, which is typically problematic when enforcing exclusion in pixel-wise segmentation. Still, unlabeled nodes will not be an issue if relatively few energy terms are non-submodular, and for our approach, QPBO often provides a globally optimal solution.

2. Method

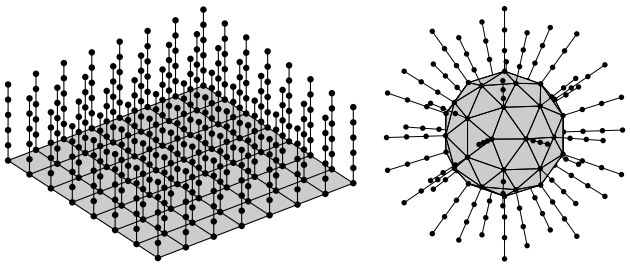
Our method builds upon existing approaches for graph cut segmentation. To make the paper self-contained, we first review the two most relevant algorithms for our work: graph-based surface detection, and quadratic pseudo-boolean optimization. Next, we detail our extension which ensures that the fitted surface meshes are separated by planes. Finally, we cover a few implementational details of our approach. For clarity, we reserve the terms *vertex* and *edge* for meshes and use *node* and *arc* when dealing with graphs.

2.1. Surface fitting with graph cuts

Wu and Chen [24] were the first to suggest using an s-t graph cut, and its polynomial-time solution, to detect surfaces in a 3D image. The approach was initially used for terrain-like and tubular surfaces [15]. For terrain-like surfaces, the solution is found by searching along the columns of the volume. Tubular surfaces are found by searching radially along rays from the tube center. An interpretation of this approach, which can be generalized to any shape, involves a meshed *base-surface*.

In this interpretation, the meshed base-surface is fit to the data by displacing the mesh vertices along the vertex normals. The goal is to place the vertices at the boundary of the object of interest, such that the region inside the mesh gives a segmentation of the object. The optimal displacement of vertices is found using a s-t graph cut.

While base-surfaces for terrain-like and tubular objects may be a regular quad mesh (see Figure 2, left), this may not be suitable for other shapes. For roughly spherical objects, a regular polyhedron may be used [8, 23] (see Figure 2, right). A different strategy involves using a rough initial segmentation as a base-surface [25].



(a) Quad. mesh.

(b) Regular polyhedron.

Figure 2: Examples of different base-surfaces and the positions of the graph nodes computed from the mesh. The nodes belonging to a normal-aligned column are connected by lines.

The underling graph construction is shared by all approaches, and is summarized in the following. With a *base-mesh* as an input, we will construct a graph $G = (\mathcal{V} \cup \{s, t\}, \mathcal{E})$ with nodes $\mathcal{V}$, a source node s, a sink node t, and arcs $\mathcal{E}$.

For a node set, $\mathcal{V}$, we consider each mesh vertex i, with position $\mathbf{v}_i$, and generate a series of candidate positions along the (outward) vertex normal, $\mathbf{n}_i$, as

$$\mathbf{v}_{i,k} = \mathbf{v}_i + k\,\delta_{\text{step}}\,\mathbf{n}_i \quad \text{for} \quad k = 0, 1, ..., n_{\text{step}}. \tag{1}$$

Here, δ_{step} and n_{step} are user defined parameters which define the step length and number of steps, respectively. With each candidate positions $\mathbf{v}_{i,k}$ we associate a graph node $v_{i,k} \in \mathcal{V}$. We will refer to the set of nodes,

$$C_i = \{v_{i,k} \mid k = 0, 1, ..., n_{\text{step}}\}, \tag{2}$$

associated with a mesh vertex i as the *normal-aligned column* of i. Furthermore, we say that two normal aligned columns are neighbors, if their corresponding mesh vertices are connected with an edge.

The arc set $\mathcal{E}$ consists of terminal arcs and internal arcs. Internal arcs impose geometric constraints on the solution, such that vertices have a well-defined position along each normal-aligned column, and that the displacement of two neighboring vertices does not vary more than a pre-set value Δ. The internal arcs have infinite weight and consist of:

Intracolumn arcs $(v_{i,k}, v_{i,k-1})$ between successive nodes of each normal-aligned column C_i. Here $k = 1, 2, ..., n_{\text{step}}$.

Intercolumn arcs $(v_{i,k}, v_{j,l})$ from nodes of each normal-aligned column C_i, to nodes of a neighboring normal-aligned column C_j, where $l = \max\{0, k - \Delta\}$.

These arcs are illustrated in Figure 3. The parameter Δ constrains the displacement of neighbouring vertices relative to the base-mesh. If the base-mesh is chosen to be smooth, then Δ can be thought of as a regularization parameter.

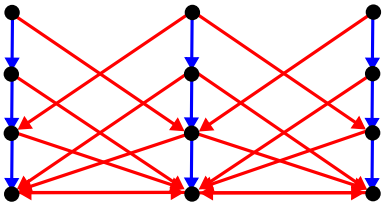


Figure 3: Illustration of intracolumn arcs (blue) and intercolumn arcs (red) for $\Delta = 2$.

The terminal arcs connect graph nodes with either the source s or the sink t. Those arcs have finite weights derived from a cost function which is given by the image values. Several cost functions have been used in the literature, and they typically lead to slightly different weight functions. The most suitable cost function is often problem-specific with popular choices being based on image edges [15] or regions [10]. A region-based cost function is constructed such that it attains negative values where image data supports object, and positive values where image data supports background.

If the node $v_{i,k}$ has a negative weight $-w$ it will be connected to the source with an arc $(s, v_{i,k})$ having a weight w. If the node has a positive weight w it will be connected to the sink with an arc$(v_{i,k}, t)$ having a weight w. However, we always connect the innermost node in each normal aligned column to the source, to guarantee at least one node is included in the object.

It has been shown [15, 24] that the vertex displacements which result in the minimum total cost, while satisfying the constraints set by Δ, can be found by computing a minimum s-t cut in this arc weighted graph. Such a cut splits the nodes into two disjoint subsets; the source set, S, and the sink set, T. Nodes in S are then defined to be inside the object of interest and vice versa for T. Mesh vertices are then moved to the position associated with the outermost node in their normal aligned columns which is in S. The s-t cut is often computed with the efficient algorithm by Boykov and Kolmogorov [3]. Furthermore, several extensions to this basic construction have since been developed [4].

Setting up a segmentation via surface fitting requires some knowledge of the geometry of the segmentation problem, or pre-processing, in order to get an appropriate initialization.

2.2. Quadratic pseudo-boolean optimization

It turns out that the construction from the previous section can be reformulated in a more general framework known as quadratic pseudo-boolean optimization (QPBO). The benefit of this is that QPBO allows us to model constraints that are not possible with the construction in Section 2.1 [9, 14]. Specifically, it enables us to enforce that two graph nodes cannot both be in the source set, which is what allows us to ensure that surfaces do not overlap.

A quadratic pseudo-boolean (QPB) function is a function $f : \{0, 1\}^n \to \mathbb{R}$ which can be written as

$$f(\mathbf{x}) = \theta_{\text{const}} + \sum_{i=1}^{n} \theta_i(x_i) + \sum_{i=1}^{n} \sum_{j>i} \theta_{ij}(x_i, x_j). \quad (3)$$

Computing a minimum s-t cut can then be reformulated as minimizing the following QPB energy function [13]

$$E(\mathbf{x}) = \theta_{\text{const}} + \sum_{v \in \mathcal{V}} \theta_v(x_v) + \sum_{(u,v) \in \mathcal{E}} \theta_{uv}(x_u, x_v), \quad (4)$$

where

$$x_v = \begin{cases} 0, & v \text{ is in the source set } S, \\ 1, & \text{otherwise.} \end{cases} \quad (5)$$

To define the θ_v and θ_{uv} functions, it will be convenient to use the following notation

$$\theta_{v,i} = \theta_v(i) \quad \text{and} \quad \theta_{uv,ij} = \theta_{uv}(i, j). \quad (6)$$

Furthermore, unless we explicitly specify a value for $\theta_{v,i}$ and $\theta_{uv,ij}$ we assume it to be zero. Now, let $w((u, v))$ be the weight of arc (u, v), and add the following unary terms

$$\theta_{v,0} = w((s, v)) \quad \text{and} \quad \theta_{v,1} = w((v, t)), \quad (7)$$

and the following binary terms

$$\theta_{uv,01} = w((u, v)) \quad \text{and} \quad \theta_{uv,10} = w((v, u)). \quad (8)$$

Minimizing $E(\mathbf{x})$ then corresponds to computing the minimum s-t cut on G.

Several papers have explored methods to minimize QPB functions [11, 13, 19]. If (and only if) the function is submodular, meaning that all binary functions satisfy

$$\theta_{uv,00} + \theta_{uv,11} \leq \theta_{uv,01} + \theta_{uv,10}, \quad (9)$$

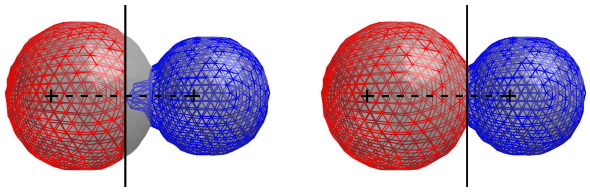
then the minimization can be done with the construction in Section 2.1 [9, 14]. In general, however, the minimization of QPB functions is NP-hard [13]. Thus, many methods focus on computing partial solutions, where each variable, x_i, can be given the value 0, 1, or unknown. The strength is that if 0 or 1 is assigned, then it is guaranteed to be the globally optimal value. The downside is that some variables may remain unassigned.

For our application, we found that even using the basic method of [13] would very often result in fully assigned (globally optimal) solutions. This method works by building a graph, similar to the construction in Section 2.1, with two nodes, a primal and dual, for every binary variable x_i. Non-submodular energy terms can then be modelled by adding edges between primal and dual nodes, and the minimization is done by computing a minimum s-t cut.

Finally, for the few cases where a full assignment was not immediately returned, a small adjustment of the method parameters would then fix it.

2.3. Adding overlap constraints

We now detail our extension, which allows us to enforce that surfaces do not overlap. Our construction proceeds as follows; assume we have two objects, A and B, to segment. Let $\mathbf{c}_\mathrm{A}$ and $\mathbf{c}_\mathrm{B}$ be the center of the base-surface meshes for object A and B, respectively. Now, consider a plane P with normal vector $\mathbf{n}_\mathrm{P} = \mathbf{c}_\mathrm{B} - \mathbf{c}_\mathrm{A}$ placed between A and B. If we constrain each mesh to not cross the plane, we would ensure that they cannot overlap. However, if the plane is not placed optimally the resulting segmentation may be subpar (see Figure 4). Therefore, to make the method more robust, we



(a) Bad plane placement. (b) Good plane placement.

Figure 4: Importance of plane placement for the segmentation of two overlapping balls. The center of each base-mesh is marked with a '+'. In (a) the plane is placed too far to the left. In (b) the plane is placed correctly which has resulted in each object being well segmented.

want the position of the plane to be determined dynamically as part of the segmentation problem.

To achieve this, first build graphs $G_\mathrm{A} = (\mathcal{V}_\mathrm{A} \cup \{s,t\}, \mathcal{E}_\mathrm{A})$ and $G_\mathrm{B} = (\mathcal{V}_\mathrm{B} \cup \{s,t\}, \mathcal{E}_\mathrm{B})$, according to the graph construction from Section 2.1. Then, merge the graphs into $G = (\mathcal{V}, \mathcal{E}) = (\mathcal{V}_\mathrm{A} \cup \mathcal{V}_\mathrm{B} \cup \{s,t\}, \mathcal{E}_\mathrm{A} \cup \mathcal{E}_\mathrm{B})$. Next, generate a number of candidate positions $\mathbf{p}_0, \mathbf{p}_1, ..., \mathbf{p}_n$ where

$$\mathbf{p}_i = \frac{i}{n}\mathbf{c}_\mathrm{B} + \left(1 - \frac{i}{n}\right)\mathbf{c}_\mathrm{A}, \tag{10}$$

and $n = \lfloor \|\mathbf{c}_\mathrm{B} - \mathbf{c}_\mathrm{A}\| / \delta_\mathrm{step} \rfloor$. Recall that δ_step is the step size used for building the normal-aligned columns for each surface graph, cf. Section 2.1. Notice that positions start at object A and then move towards object B. Now, add nodes $u_1, u_2, ..., u_n$ to $\mathcal{V}$ where u_k is associated with $\mathbf{p}_k$. Furthermore, add arcs (u_i, u_{i-1}) for $i = 1, 2, ..., n$ to $\mathcal{E}$ where each arcs has infinite weight. Note that this construction is the same as for the normal-aligned columns, see Figure 5. Thus, we can define the final position of the plane as $\mathbf{p}_i$ where i is the smallest number such that $u_i \in T$.

Next, we add interaction between the surfaces and the planes. Here, it will be more convenient to work with the QPB energy function $E(\mathbf{x})$, cf. Section 2.2. Iterate over all candidate plane positions, $\mathbf{p}_i$, and add energy terms to $E(\mathbf{x})$ according to the following rules:

1. For every normal-aligned column $C \subset \mathcal{V}_\mathrm{A}$, find the first node $v_k \in C$ whose position $\mathbf{v}_k$ satisfies $(\mathbf{v}_k - \mathbf{p}_i)^T \mathbf{n}_P > 0$. Add the term $\theta_{v_k u_i, 01} = \infty$ to $E(\mathbf{x})$ (see Figure 5, left). This will enforce that if $v_k \in S$ then $u_i \in S$. We only need to consider this first node since the construction of the surface graph ensures that if any later node $v_l \in S$ then $v_k \in S$.

2. For all normal-aligned columns $C \subset \mathcal{V}_\mathrm{B}$ find the first node $v_k \in C$ whose position $\mathbf{v}_k$ satisfies $(\mathbf{v}_k - \mathbf{p}_i)^T \mathbf{n}_P < 0$. Add the term $\theta_{v_k u_i, 00} = \infty$ (see Figure 5, right). This term acts like an exclusion arc as it ensures that if either v_k or u_i are in S, then the other cannot. Again, we only consider the first node as $v_k \in T$ will ensure that all later nodes are also in T.

These terms ensure that surface A and B never cross the plane and therefore cannot overlap. The complete construction is visualized in Figure 5. Intuitively, terms from step 1 will cause surface A to 'push' the plane away as it grows, and terms from step 2 will cause the plane to 'push' on surface B. As both surfaces grow to fit the data they will move the plane to a position of equilibrium where no surface can grow without degrading the overall segmentation. Note that the terms from step 2 are not submodular, which means we must use QPBO based methods to minimize the energy.

When dealing with more than two surfaces, we first construct a graph for each surface. Then, all graphs are merged into one graph and the above construction is added for each pair of interacting surfaces. Finally, the method from [13] is used to perform all segmentations simultaneously. If no variables are unassigned, the resulting segmentation is thus guaranteed to be globally optimal while having no overlap.

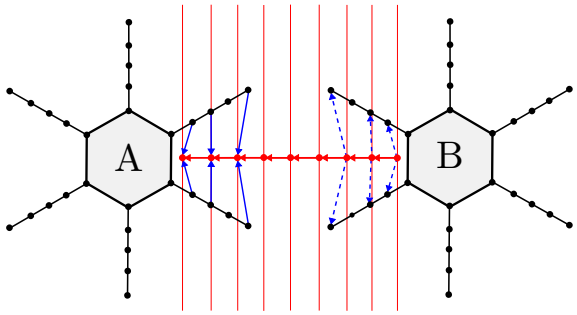


Figure 5: Illustration of the construction used to separate two surfaces with a moving plane. Graphs for two objects A and B are shown, but without intra- and intercolumn arcs. Red nodes indicate candidate plane positions, and each candidate plane is shown as a vertical red line. Red arcs are added to pairs of candidate position nodes pointing left, as candidate points are ordered going from A to B. Interaction terms from step 1 are shown as solid blue arcs, and terms from step 2 are shown as dashed blue arcs.

2.4. Detecting potential overlaps

We now describe a strategy for determining which meshes need to have overlap constraints added. One could add constraints between every pair of surfaces, but this would result in an unnecessarily large graph. As the run-time and resource use of the QPBO method is related to the size of the graph, we want it to be as small as possible.

Often, one chooses a base-surface mesh which approximates a sphere of some radius r centered at a point $\mathbf{c}$. In this case, the node farthest from the center in each normal aligned column will have distance $d = r + \delta_{\text{step}} n_{\text{step}}$ from $\mathbf{c}$. Thus, the fitted mesh will be contained in a bounding sphere of radius d centered at $\mathbf{c}$.

From this, it follows that two meshes (whose base-surface approximates a sphere) can only overlap if their bounding spheres intersect. This happens when $D < d_A + d_B$, where D is the distance between the mesh centers, and d_A, d_B are the radii of the two bounding spheres. Therefore, when adding overlap constraints, we first compute all pairwise center distances, and then only add constraints for meshes whose bounding spheres intersect.

For more general meshes, one could also use distance fields, bounding boxes, or, in the case of a few objects, manual annotation to determine which meshes could overlap.

3. Results and discussion

We now apply the developed method to three data sets with known ground truth segmentations. The first consists of a simulated fluorescence microscopy image of HL60 cell nuclei (see Figure 6a) [22]. As the data is computer-generated, we know the ground truth segmentation for the entire volume. The task is to segment the cells and the main challenge is the low contrast between foreground and background and the small distance between cells. We increased the segmentation difficulty further by adding Poisson and Gaussian noise (std. dev. of 40 with image intensity values between 0 and 255) per the description in [21]. We refer to this data set as *simulated cells*.

The second data set is a 3D confocal microscopy image of C. Elegans embryos (see Figure 6b) [17]. Here, we only have ground truth segmentations for a single xy-slice. Again, the task is to segment the cells and the challenge is the high noise level of the image and poorly defined object boundaries. Furthermore, we again have many closely packed objects. We refer to this data set as *cells*.

The third data set is a 3D X-ray tomographic microscopy image of liquid foam (see Figure 6c) [16, 18] from the TomoBank data repository [6]. As no ground truth is provided, we have manually annotated a single xy-slice to quantitatively evaluate the performance of the methods. The task is to segment individual foam bubbles, which is made difficult since the walls between bubbles are often not visible in the 3D image. We refer to this data set as *foam*.

The sizes of the data sets are shown in Table 1.

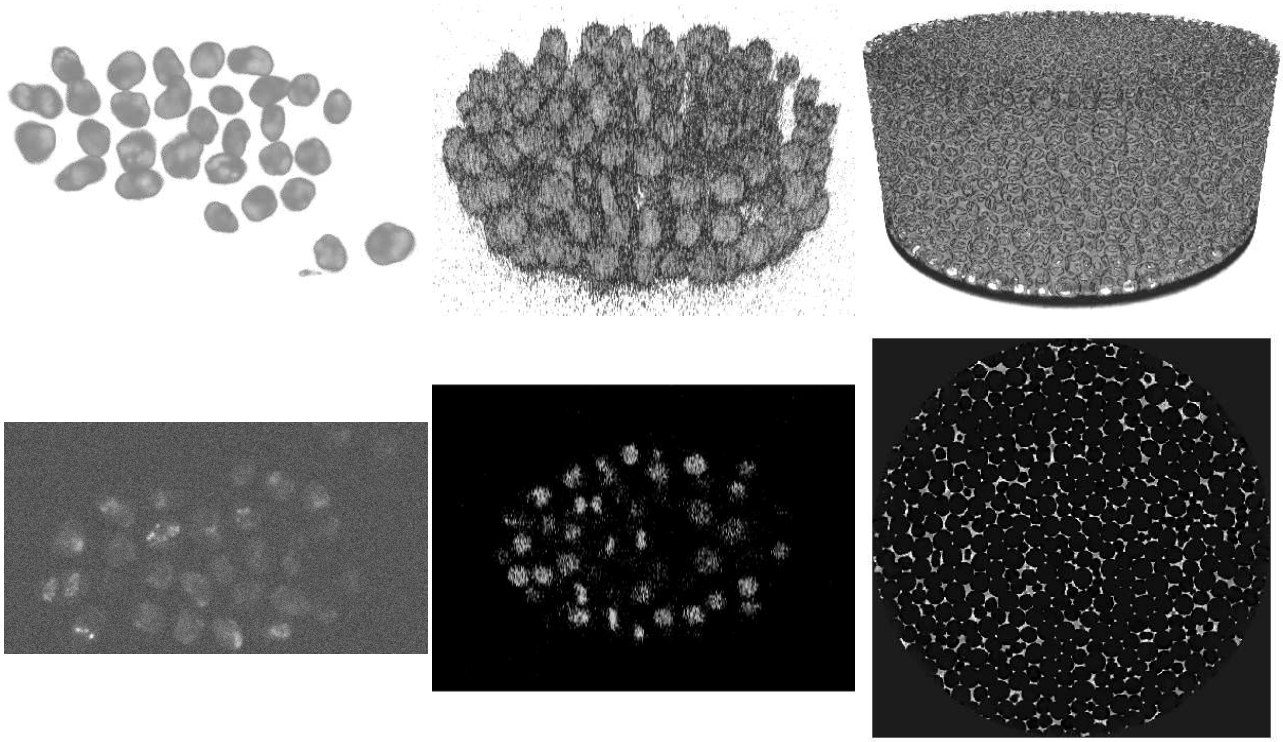
Data set	Size [voxels]	Voxel size [µm]
Sim. cells [22]	$349 \times 639 \times 59$	$0.125 \times 0.125 \times 0.200$
Cells [17]	$512 \times 708 \times 35$	$0.090 \times 0.090 \times 1.000$
Foam [18, 16]	$504 \times 504 \times 230$	$6.000 \times 6.000 \times 6.000$

Table 1: Size of the data sets.

For the segmentation of the images, we place a geodesic 4-frequency subdivided icosahedron at the center of each object. For the simulated cells and cells data set, center positions were manually annotated. For the foam data set, bubble centers were found by first binarizing the image and then finding local maxima of the distance transform. The radius of the base-mesh was chosen as one voxel. When sampling along the normal-aligned columns, steps were scaled to move an equal number of µm along each axis. We use the region based cost function from [10]. The parameters used for segmentation are shown in Table 2. For the cells data set, the in-region and out-region costs were scaled such that their sum over the image region resulted in the same value.

We evaluated the segmentation performance using the Dice similarity coefficient (also known as F1 score), boundary F1 (BF) score [5], and Jaccard score (also known as intersection over union). For the simulated cells data set, a score was computed for each segmented object. For the cells and foam data set, a score was only computed for meshes which intersected the xy-slice with known ground truth. Correspondences between segmentations and ground truth labels were determined a priori. If an intersecting mesh did not have a corresponding label, it was assigned a score of 0.

The segmentation of each data set with and without over-



(a) Sim. cells (b) Cells (c) Foam

Figure 6: Top row: 3D renderings of the data sets. Bottom row: xy-slices of the data sets.

Param.	Sim. cells w/o	Sim. cells w/	Cells w/o	Cells w/	Foam w/o	Foam w/
μ_{in}	105	105	100	100	-0.0002	-0.0002
μ_{out}	90	90	5	5	0.0022	0.0022
σ_{in}	4	4	20	20	0.02	0.02
σ_{out}	4	4	20	20	0.04	0.04
Δ	7	8	9	9	2	2
δ_{step}	0.5	0.5	0.5	0.5	0.5	0.5
n_{step}	90	90	50	50	30	30

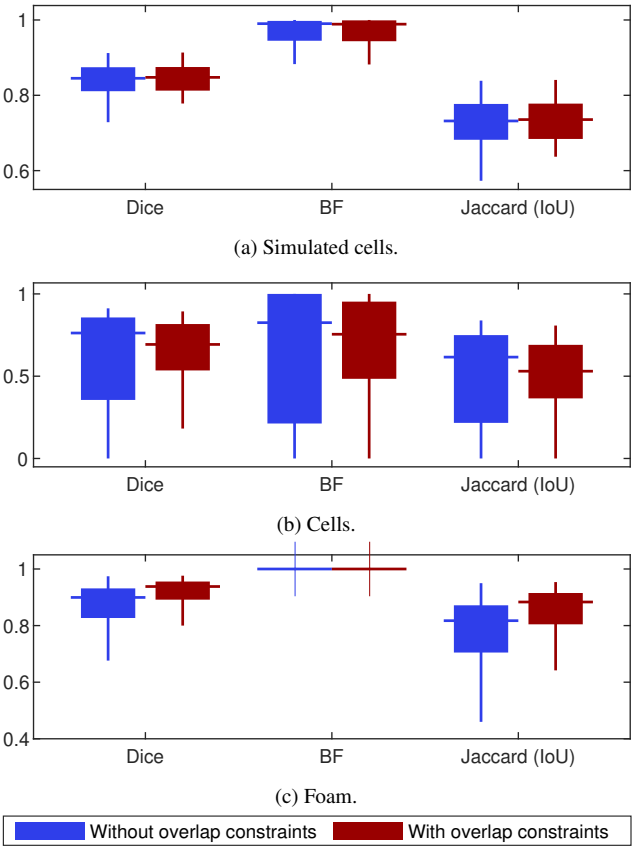
Table 2: Method parameters used for segmentation. See [10] for explanation of the μ and σ parameters.

		Sim. cells w/o	Sim. cells w/	Cells w/o	Cells w/	Foam w/o	Foam w/
Dice	Mean	0.83	**0.84**	0.61	**0.62**	0.81	**0.88**
	Max.	0.91	**0.91**	**0.91**	0.89	0.97	**0.98**
	Min.	0.51	**0.58**	0.00	0.00	0.00	0.00
BF	Mean	0.96	**0.97**	0.66	**0.66**	0.95	**0.98**
	Max.	1.00	1.00	1.00	1.00	1.00	1.00
	Min.	0.77	**0.88**	0.00	0.00	0.00	0.00
Jaccard	Mean	0.72	**0.73**	**0.50**	0.49	0.73	**0.81**
	Max.	0.84	**0.84**	**0.84**	0.81	0.95	**0.95**
	Min.	0.34	**0.41**	0.00	0.00	0.00	0.00

Table 3: Summary of statistics segmentation scores with and without overlap constraints. The best scores for each data set have been marked with bold (the marking is based on additional decimals).

lap constraints are shown in Figure 9. Figure 7 and Table 3 summarize the segmentation scores for each data set. For the segmentations with overlap constraints, we achieved a full assignment for all data sets. Adding overlap constraints generally results in an improved segmentation, especially for the foam data set. For the cells data set, there is a small decrease in the maximum score. This can partly be attributed to the fact that we only measure the segmentation quality in a single xy-slice. Thus, adding the overlap constraints may cause some local degradations of segmentation quality, even if the overall (3D) segmentation may be better. The reason this does not happen for the foam data set is that overlap is a much bigger problem here than for the cells data set. Therefore, fixing it results in a greater overall improvement.

As Figure 7 demonstrates, most of the segmentation improvements come from increasing the quality of the worst segmentations. This is a direct result of removing non-physical segmentation overlap, as shown in Figure 8. When objects are allowed to overlap the segmentation of one object can stray into a neighboring object which increases

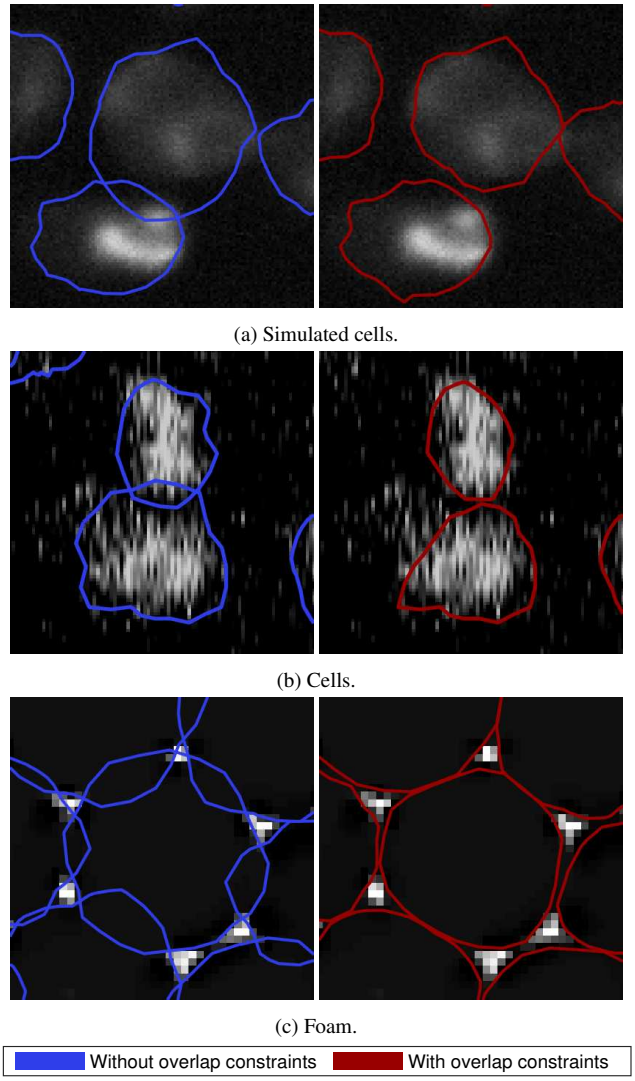


(a) Simulated cells.

(b) Cells.

(c) Foam.

Figure 7: Boxplots of segmentation scores with and without overlap constraints.



(a) Simulated cells.

(b) Cells.

(c) Foam.

Figure 8: Examples of non-physical segmentation overlaps which are corrected by our method. Figures show cross-sections of fitted surface meshes overlaid on xy-slices of the data.

the number of false positives. When this is prevented, the segmentations follow the true object contours more closely. Furthermore, the segmentations now resemble the physical reality more, since distinct object cannot overlap.

Finally, the method was found to be fairly robust to small parameter changes. However, changing Δ, δ_{step}, or δ_{step} could have significant effects on the run time, as they directly affect the size of the graph.

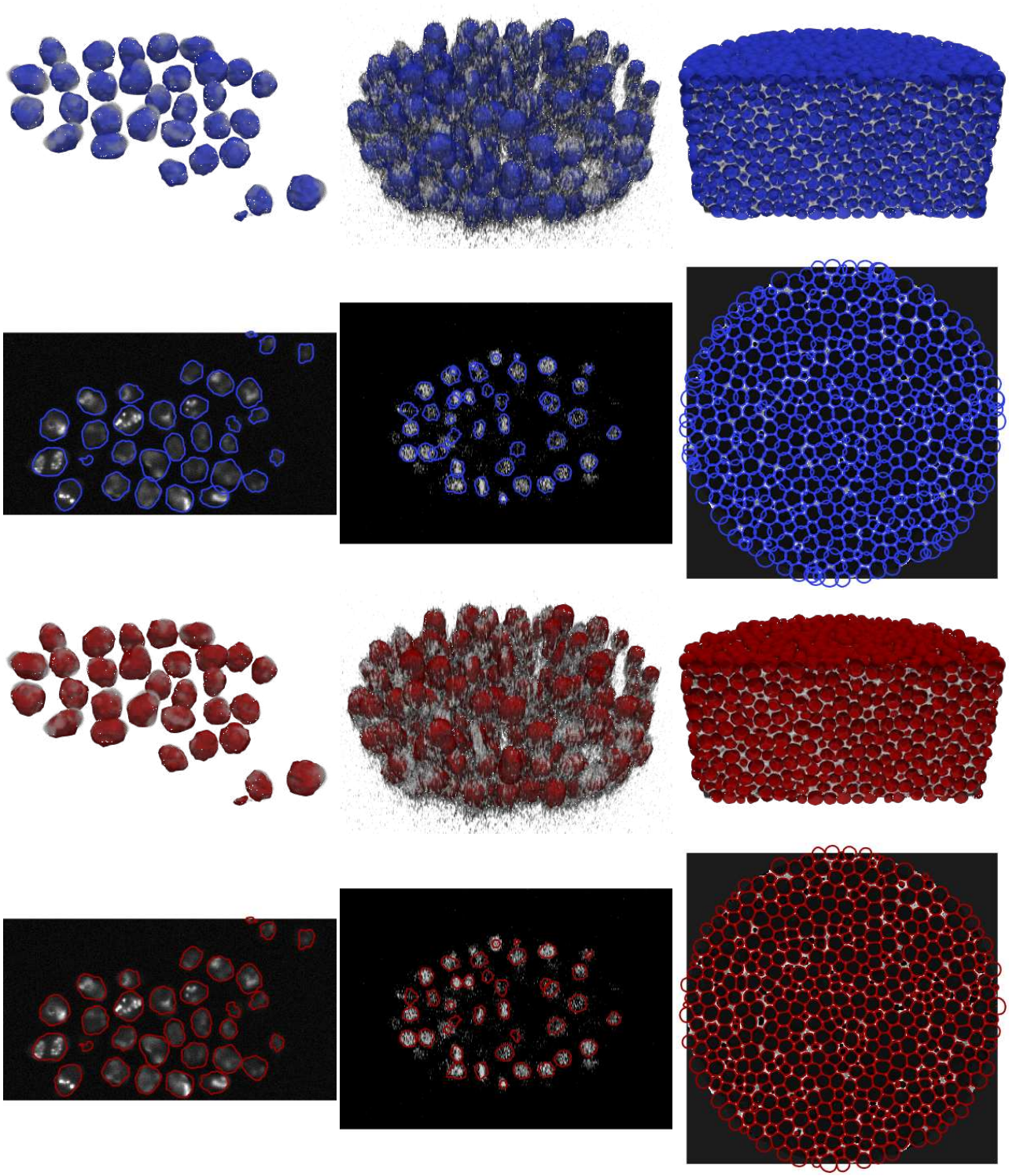
4. Conclusion and further work

In this paper, we have presented an extension for multi-object graph-based segmentation which allows us to enforce that segmented objects may not overlap. The extension worked by separating neighboring objects with planes whose position was determined dynamically. When applied to data sets with known ground truth segmentations, adding overlap constraints resulted in quantitative improvements. Furthermore, overlap prevention also qualitatively improved the segemtnations as they better complied with physical reality.

It is worth noting that our extension is not restricted to the basic version of the graph cut method used in this paper. Indeed, it can be applied to any method in the graph cut family, since it only adds additional constraints to the existing graph structures but does not otherwise modify them.

Furthermore, it is possible to generalize the extension to separate objects with other surfaces than planes, which would increase its usefulness. The current construction could instead use the level sets of any suitable function as the separating surfaces. However, if the separating surfaces have curvature, some amount of overlap will be possible due to the discrete nature of meshes. Developing the mathematical theory for this more general construction will be the subject of future work.

Acknowledgements: This work is partly supported by The Center for Quantification of Imaging Data from MAX IV (QIM) funded by The Capital Region of Denmark



(a) Sim. cells (b) Cells (c) Foam

Figure 9: Segmentation results overlaid on data. Blue surfaces and curves were fitted without overlap constraints, red ones with overlap constraints. Rows 1 and 3 show the fitted surface meshes overlaid on 3D rendering of data. Rows 2 and 4 show cross-sections of surface meshes overlaid on xy-slices of the data.

References

[1] Junjie Bai, Abhay Shah, and Xiaodong Wu. Optimal multi-object segmentation with novel gradient vector flow based shape priors. *Computerized Medical Imaging and Graphics*, 69:96–111, nov 2018. 1

[2] Stephen P. Boyd and Lieven. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004. 1

[3] Yuri Boykov and Vladimir Kolmogorov. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 26(9):1124–1137, 2004. 3

[4] Xinjian Chen and Lingjiao Pan. A Survey of Graph Cuts/Graph Search Based Medical Image Segmentation. *IEEE Reviews in Biomedical Engineering*, 11:112–124, 2018. 3

[5] Gabriela Csurka, Diane Larlus, Florent Perronnin, and France Meylan. What is a good evaluation measure for semantic segmentation?. In *British Machine Vision Conference*, volume 27, page 2013, 2013. 5

[6] Francesco De Carlo, Doğa Gürsoy, Daniel J Ching, K Joost Batenburg, Wolfgang Ludwig, Lucia Mancini, Federica Marone, Rajmund Mokso, Daniël M Pelt, Jan Sijbers, et al. TomoBank: a tomographic data repository for computational X-ray science. *Measurement Science and Technology*, 29(3):034004, 2018. 5

[7] Andrew Delong and Yuri Boykov. Globally Optimal Segmentation of Multi-Region Objects. In *International Conference on Computer Vision*, pages 285–292. IEEE, 2009. 1

[8] Jan Egger, Miriam H.A. Bauer, Daniela Kuhnt, Barbara Carl, Christoph Kappus, Bernd Freisleben, and Christopher Nimsky. Nugget-cut: A segmentation scheme for spherically- and elliptically-shaped 3D objects. In *Joint Pattern Recognition Symposium*, pages 373–382. Springer, 2010. 2

[9] Daniel Freedman and Petros Drineas. Energy minimization via graph cuts: Settling what is possible. In *Computer Vision and Pattern Recognition*, pages 939–946. IEEE, 2005. 3, 4

[10] Mona Haeker, Michael Abràmoff, Milan Sonka, Xiaodong Wu, and Randy Kardon. Incorporation of Regional Information in Optimal 3-D Graph Search with Application for Intraretinal Layer Segmentation of Optical Coherence Tomography Images. In *Information Processing in Medical Imaging*, pages 607–618. Springer, 2007. 3, 5, 6

[11] Fredrik Kahl and Petter Strandmark. Generalized roof duality for pseudo-boolean optimization. In *International Conference on Computer Vision*, pages 255–262. IEEE, 2011. 3

[12] Dagmar Kainmueller, Hans Lamecker, Stefan Zachow, and Hans Christian Hege. Coupling deformable models for multi-object segmentation. In *International Symposium on Biomedical Simulation*, pages 69–78. Springer, 2008. 1, 2

[13] Vladimir Kolmogorov and Carsten Rother. Minimizing non-submodular functions with graph cuts - A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1274–1279, 2007. 3, 4

[14] Vladimir Kolmogorov and Ramin Zabih. What Energy Functions can be Minimized via Graph Cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004. 3, 4

[15] Kang Li, Xiaodong Wu, Danny Z. Chen, and Milan Sonka. Optimal surface segmentation in volumetric images - A graph-theoretic approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):119–134, 2006. 2, 3

[16] Rajmund Mokso, Christian M Schlepütz, Gerd Theidel, Heiner Billich, Elmar Schmid, Tine Celcer, Gordan Mikuljan, Leonardo Sala, Federica Marone, Nick Schlumpf, et al. GigaFRoST: the gigabit fast readout system for tomography. *Journal of synchrotron radiation*, 24(6):1250–1259, 2017. 5

[17] John Isaac Murray, Zhirong Bao, Thomas J Boyle, Max E Boeck, Barbara L Mericle, Thomas J Nicholas, Zhongying Zhao, Matthew J Sandel, and Robert H Waterston. Automated analysis of embryonic gene expression with cellular resolution in c. elegans. *Nature methods*, 5(8):703, 2008. 5

[18] C Raufaste, B Dollet, K Mader, S Santucci, and R Mokso. Three-dimensional foam flow resolved by fast X-ray tomographic microscopy. *Europhysics Letters*, 111(3):38004–38010, 2015. 5

[19] Carsten Rother, Vladimir Kolmogorov, Victor Lempitsky, and Martin Szummer. Optimizing Binary MRFs via Extended Roof Duality. In *Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 3

[20] Qi Song, Xiaodong Wu, Yunlong Liu, Mark Smith, John Buatti, and Milan Sonka. Optimal graph search segmentation using arc-weighted graph for simultaneous surface detection of bladder and prostate. In *Medical Image Computing and Computer-Assisted Intervention*, pages 827–835. Springer, 2009. 1, 2

[21] David Svoboda, Michal Kozubek, and Stanislav Stejskal. Generation of digital phantoms of cell nuclei and simulation of image formation in 3D image cytometry. *Cytometry Part A*, 75A(6):494–509, 2009. 5

[22] David Svoboda and Vladimír Ulman. MitoGen: A framework for generating 3D synthetic time-lapse sequences of cell populations in fluorescence microscopy. *IEEE Transactions on Medical Imaging*, 36(1):310–321, 2016. 5

[23] Yao Wang and Reinhard Beichel. Graph-based segmentation of lymph nodes in CT data. In *International Symposium on Visual Computing*, pages 312–321. Springer, 2010. 2

[24] Xiaodong Wu and Danny Z. Chen. Optimal Net Surface Problems with Applications. In *International Colloquium on Automata, Languages, and Programming*, pages 1029–1042. Springer, 2002. 2, 3

[25] Yin Yin, Xiangmin Zhang, Rachel Williams, Xiaodong Wu, Donald D. Anderson, and Milan Sonka. LOGISMOS-layered optimal graph image segmentation of multiple objects and surfaces: Cartilage segmentation in the knee joint. *IEEE Transactions on Medical Imaging*, 29(12):2023–2037, dec 2010. 1, 2