Topology Cuts: A Novel Min-Cut/Max-Flow Algorithm for Topology Preserving Segmentation in N-D Images

Yun Zeng
Stony Brook University
Stony Brook, USA, NY 11790
yzeng@cs.sunysb.edu

Dimitris Samaras
Stony Brook University
Stony Brook, USA, NY 11970
samaras@cs.sunysb.edu

Wei Chen
Zhejiang University
Hangzhou, P.R. China, 310027
chenwei@cad.zju.edu.cn

Qunsheng Peng
Zhejiang University
Hangzhou, P.R. China, 310027
peng@cad.zju.edu.cn

Abstract

Topology is an important prior in many image segmentation tasks. In this paper, we design and implement a novel graph-based min-cut/max-flow algorithm that incorporates topology priors as global constraints. We show that optimization of the energy function we consider here is NP-hard. However, our algorithm is guaranteed to find an approximate solution that conforms to the initialization, which is a desirable property in many applications since the global optimum solution does not consider any initialization information. The key innovation of our algorithm is the organization of the search for maximum flow in a way that allows consideration of topology constraints. In order to achieve this, we introduce a label attribute for each node to explicitly handle the topology constraints, and we use a distance map to keep track of those nodes that are closest to the boundary. We employ the bucket priority queue data structure that records nodes of equal distance and we efficiently extract the node with minimal distance value. Our methodology of embedding distance functions in a graph-based algorithm is general and can also account for other geometric priors. Experimental results show that our algorithm can efficiently handle segmentation cases that are challenging for graph cuts algorithms. Furthermore, our algorithm is a natural choice for problems with rich topology priors such as object tracking.

1. Introduction

Many computer vision problems such as segmentation, stereo and image restoration, can be formulated as a minimization of an energy function [22]. These energy functions are naturally divided into two groups: continuous and discrete. For the problem of image segmentation, the level sets method [21] is a representative model in the continuous community, while the graph-based Markov Random Fields (MRFs) [13] is a very popular model in the discrete group. One very efficient algorithm for solving a subclass of the MRF energy function is the graph cuts algorithm [8]. In recent years, there has been a number of works showing the close relationship between level sets and graph cuts ([4, 6, 16], *etc*.), and how shape priors can be incorporated into the graph cuts framework ([12, 18]). In this work, we show how the idea of topology preserving segmentation from the level sets literature [14] can be transposed to the graph-based algorithms. We propose the first min-cut/max-flow algorithm that is designed to explicitly incorporate topology as a global constraint in the segmentation. We call our new algorithm Topology Cuts, in analogy to the popular Graph Cuts algorithm [8].

Topology as a prior is available in many applications. For example, the anatomy of human tissues provides important topological constraints that ensure the correctness in biomedical image segmentation.Existing techniques that enforce topology constraints into the graph cuts algorithm, do so by simply tuning the parameters of the energy function [3, 7]. This scheme usually requires intense user interactions and is not applicable in cases where user manipulation is difficult. In contrast, we propose to embed the topological constraint into the discrete min-cut/max-flow algorithm, which leads to a new and efficient way of considering global topology information for the general problem of topology preservation.

Our work is inspired by the topology preserving level set method of [14]. This algorithm makes use of digital topology theory for N-D images [2] to detect topology changes during the evolution of level sets. Taking advantage of the fact that the level set functions are solved in a gradient descent manner, and assuming that the change of sign for the pixels only occurs one pixel at a time on the boundary of the evolving objects, the topology of the object can be easily controlled.

However, transferring the idea of topology preserving evolution from the continuous level sets algorithm to the discrete graph-based algorithm is not straightforward. The main difficulty lies in the fact that previous graph-cut implementations [1, 5, 15] are inherently topology-free and thus not conductive to topology considerations during the search of max-flow. To make the consideration possible for the discrete graph-based algorithm, we introduce the following elements.

1. An F/B *label* attribute is introduced to explicitly handle the topology property in the image. This resolves an ambiguity in the existing graph cuts algorithms, *i.e.*, it is possible that the labels for a subset of the graph's nodes can be changed without changing the optimal solution (multiple solutions for the energy minimization problem). Existing algorithms set these nodes' labels to a default label, which unavoidably leads to topological errors.
2. An *initialization* step is used to provide the graph with initial topology information.
3. The computation of max-flow is divided into *inter*-label and *intra*-label stages, to facilitate the propagation of topology information during the search for the minimum of the energy function.
4. A *distance map* (function) which keeps track of the nodes that are closest to the current boundary between the different label sets is set in the beginning and is updated during the computation.
5. To efficiently insert and extract nodes on the current evolving boundary (the level set of the distance map), we use the *bucket priority queue* data structure [9, 11], which only requires time of $O(1)$ complexity for each insertion and extraction operation. Hence, there is no loss of efficiency compared to the previous graph cuts algorithms. Our algorithm shares the same complexity with the widely used graph-cut implementation [5], and in practice it runs in comparable speed.

The contributions of this paper can be summarized as follows:

- To the best of our knowledge, this is the first work that incorporates the global topology prior into the design of the discrete graph-based min-cut/max-flow algorithms.
- We prove that considering topology constraints in the MRF framework is NP-hard.
- In the design of our algorithm, we combine concepts from the level sets literature (such as distance maps and level set evolution [20, 21]) into the efficient discrete graph-based algorithm. The techniques we use here are general and define a new way of incorporating geometric prior knowledge into the existing graph-cut models/algorithms such as curvature or shape priors.

Additionally, our new algorithm is suitable for the concept of multilevel banded graph cuts [19] to speedup the computation. In experiments, we show that our algorithm achieves more meaningful and visually better results compared with graph cuts for problems where topology information is available, *e.g.*, image segmentation and object tracking.

Organization of this Paper: In Section 2, we review the essential background for describing our new algorithm. Section 4 gives the formulation of topology cuts problem. Section 5 explains the design of our new algorithm. Section 7 analyzes our algorithm in different aspects. The experimental results are presented in Section 8 . Finally, we conclude our work and outline the future work in Section 9.

2. Preliminaries

2.1. Digital topology

Here we discuss two key concepts in the topology of digital images[2]: connectivity and simple point.

To account for the topology of the objects in the digital image, the *connectivity* of the foreground and the background can not be defined arbitrarily, or the topological paradox problem may arise. For 2D images, valid (foreground, background) connectivity pairs are $(4, 8)$ and $(8, 4)$; for 3D images, the valid pairs are $(6, 18)$, $(6, 26)$, $(18, 6)$ and $(26, 6)$.

With the clarification of connectivity in the digital image discussed above, a *simple point* is defined as a point whose change from foreground to background or vice versa, does not change the number of connected components of both the foreground and background. A simple point can be efficiently computed using the concept of topological number [2]. In this paper, we adopt the basic definition of simple points discussed above for detecting topology change.

2.2. Energy minimization and min-cut/max-flow

The MRF energy function [13] solved by graph cuts can be formulated as:

$$\inf_{x_p, p \in \mathcal{V}} \{ \sum_{p \in \mathcal{V}} D(x_p) + \sum_{(p,q) \in \mathcal{E}} V_{pq}(x_p, x_q) \}, \qquad (1)$$
$$\text{with } x_p \in \{0, 1\}, p \in \mathcal{V}.$$

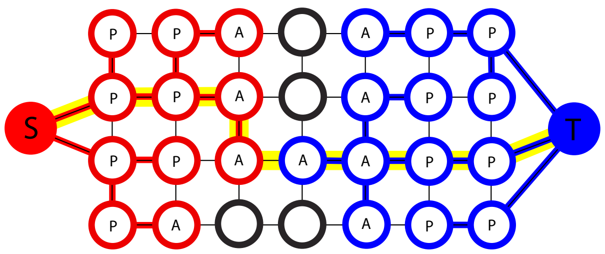


Figure 1. The dynamic tree implementation of the *s*/*t* cut

Here $\mathcal{V}$ and $\mathcal{E}$ usually denote the image pixels and their pairwise relationships respectively. If the terms of the MRF energy function have the form $D(x_p) = D_p^t(1 - x_p) + D_p^s x_p$ and $V_{pq}(x_p, x_q) = w_{pq}((1 - x_p)x_q + (1 - x_q)x_p)$ with $w_{pq} \geq 0$ (submodular condition), we can define a graph $\mathcal{G}$ with a source terminal s, a sink terminal t, and nodes $\{p|p \in \mathcal{V}\}$. The capacity from s to each node p is defined as D_p^s, the capacity from each p to t is defined as D_p^t, the capacity between neighboring nodes p, q is defined as w_{pq}.

Adopting the notation in [5], a *s*/*t* cut of a graph $\mathcal{G}$ is a partition of the nodes and terminals into two disjoint subsets S and T with $s \in S$ and $t \in T$. Also, the cost of a cut $C = \{S, T\}$ is the sum of the costs of all the edges (p, q) where $p \in S$ and $q \in T$. It has been proven [1] that the optimal solution of the binary energy function 1 corresponds to a *min-cut* of the graph $\mathcal{G}$, which is the minimal cut of all the cuts in the graph.

Finding the min-cut of a graph is equivalent to computing a maximum flow from s to t [1]. In general, algorithms for solving this min-cut/max-flow problem fall into two groups: augmenting paths and push-relabel [1] techniques. Here we review a popular algorithm based on the augmenting paths that is closely related to our method [5].

2.3. A dynamic tree implementation of the s/t cut

Generally speaking, the idea of augmenting paths algorithm [1] is to iteratively search a non-saturated path from s to t and push the maximal possible flow along this path. When no more such paths can be found, the maximum flow has been reached.

There are a number of ways to search non-saturated paths between two terminals. The efficient algorithm in [5] searches non-saturated paths by growing two trees from both the source and the sink. This idea can be efficiently implemented using the dynamic tree data structure.

A dynamic tree grows by adding non-saturated edges dynamically. As Figure 1 shows, two non-overlapping dynamic trees S and T are maintained. Each node either belongs to one of the two trees or is "free". A node that belongs to a tree can have either "active" or "passive" state. An active node is at the border of the tree while a passive node is inside the tree.

To find a non-saturated path in the graph, three stages are iteratively repeated:

- "growth": grow trees S and T until they meet in the middle, giving a non-saturated $s \to t$ path.
- "augmentation": push the maximum possible flow along this path, breaking the trees into a forest.
- "adoption": restore the single tree structure of the two trees by finding a new parent for the isolated parts. If no such parent can be found, they become free nodes.

The algorithm stops when there are not active nodes in the two trees.

3. Tree Membership and the Primal-dual Solution to the s/t Cut

If we relax the variables of the discrete optimization problem 1 to be continuous, its duality can be formulated as [26]

$$
\begin{aligned}
\max \quad & f_{ts} && && (2)\\
\text{s.t.} \quad & f_{pq} \leq w_{pq}, && (p, q) \in \mathcal{E} \\
& \sum_{p:(p,q)\in\mathcal{E}} f_{pq} - \sum_{p:(q,p)\in\mathcal{E}} f_{qp} \leq 0, && q \in \mathcal{V} \\
& f_{pq} \geq 0 && (p, q) \in \mathcal{E}.
\end{aligned}
$$

It can be shown that for any feasible solution of Equation 2, $f_{ts} \leq c_{opt}$ where c_{opt} is the optimal solution of the primal problem 1. Thus finding max-flow of the graph corresponds to finding a lower bound of the primal problem. In the ideal situation this lower bound can reach its primal solution and obtains a global optimal of the primal problem. However, for many cases, *e.g.*, optimizing energy 1 with additional constraints, this lower bound can not reach its primal solution. Thus it can only be used as a guidance to the approximation of the primal problem.

In the standard implementations of graph cuts, the label of each node (foreground or background) is normally determined by whether there is a non-saturated path from it to s or t when the max-flow is reached, namely, these algorithms determine the label of each node by its *tree membership*. Using tree membership to determine the label of each node makes it impossible to consider topology properties during the max-flow computation. This is because the tree membership of each node is updated in an irregular order and thus can not contain any topological information (graph cuts are inherently topology-free).

With the above discussion of the primal-dual relation, our new algorithm can be regarded as solving the the 0/1 optimization in its dual space (namely, finding max-flow), while making sure that its intermediate primal solution, which is represented by tree-membership, conforms to topology constraint.

4. The Topology Cuts Problem

The energy function that combines the MRF formulation and digital topology on the image grid is

$$\inf_{x_p, p\in\mathcal{V}} \{\sum_{p\in\mathcal{V}} D(x_p) + \sum_{(p,q)\in\mathcal{E}} V_{pq}(x_p, x_q)\}, \quad (3)$$

$$\text{s.t.} \qquad \mathcal{T} = \mathcal{T}_{init}$$

$$\text{with } x_p \in \{0,1\}, p \in \mathcal{V}.$$

Here $\mathcal{T}$ denotes the topology of the $0/1$ labeled image as defined in [2]. $\mathcal{T}_{init}$ is the initial topology information that is assigned to the image either interactively, or automatically as shown in our tracking example. The meanings of the other notations are the same as in Equation 1. Note that here we only consider the hard-constraints on the topology of the image. However, soft-constraints can be conveniently introduced by considering alternative definitions of simple point [25].

Theorem. *The topology cuts (TP-Cut) problem 3 is NP-hard.*

Proof. To prove the NP-hardness of the TP-CUT problem, we reduce from the connected vertex-covering (CVC) problem [24]. We provide a brief sketch here and the complete proof in the supplemental matterials.

Definition. (**Connected Vertex-Covering problem**) *Given a planar graph $\mathcal{H} = (V_h, E_h)$ with maximum degree 4 for each node and an integer K (≥ 1), is there a connected subset $V' \subset V_h$ such that $|V'| \leq K$ and for each edge $e \in E_h$, at least one of its two end nodes is in V' (a vertex cover)?*

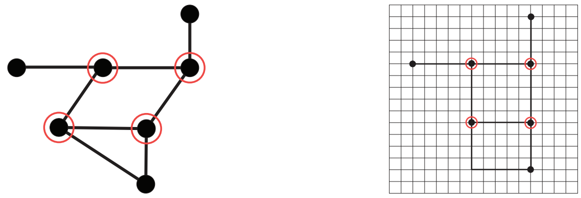
The CVC problem have been proven to be NP-hard in [24]. To reduce from the CVC problem to our TP-CUT problem, we use the following techniques.

1. A planar graph with maximum degree 4 can be embedded into the image grid domain $\mathcal{G}$ [24].

2. The weight of the edges for neighboring pixels is set to be zero, so we only have to consider the first term in 3.

3. The 0/1 label of a node on the image $\mathcal{G}$ can be "fixed" by setting a sufficiently large weight to the edge which links it to the source (*source edge*) or the sink nodes (*sink edge*). This means if we are to change the the label of the node, a very large penalty would be added to the energy function (3) (Figure 3). We call such a large weight *infinite* weight.

4. We may give some nodes on the image the "freedom" to change their label by setting the weight of the source/sink edge to be a specifically designed value. The weight of the sink edges for the nodes corresponding to vertices in $\mathcal{H}$ (*vertex node*) is set to be $|E_h| + 1$, where $|E_h|$ is the number of edges of $\mathcal{H}$ ((Figure 3(b)). The weights of the sink edge of all the other nodes (grid) who does not lie on the embedded graph are set to be infinite (Figure 3(c)).



(a) $\mathcal{H}$ (b) $\mathcal{G}$

Figure 2. A planar graph $\mathcal{H}$ with its connected vertex covering (a), and its embedding into the grid image domain $\mathcal{G}$ (b).

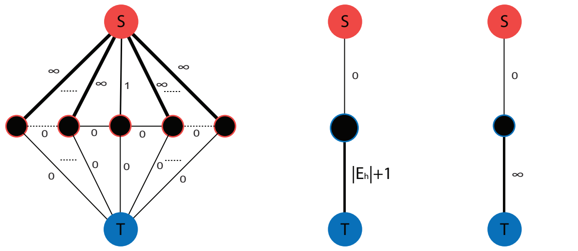
5. To respect topology constraints, we pick one node for each embedded edge in $\mathcal{G}$ and set the weight of its source edge to be a small value (Figure 3(a)). Intuitively, this constructs a "door" on each edge to allow different closed regions in the original planar graph connected to each other (we call such nodes *door nodes*). Such door nodes can be labeled 1 only when the connectivity of the 0-labeled foreground would not be broken. In other words, using these nodes, it is always possible to connect an enclosed 1-labeled region (by a 0-labeled "wall") to the outside of the wall by opening one door on the wall without separating the wall into two parts. When such a door is opened, a small penalty is added to the total energy function (Figure 3(a)).

6. With the above construction, a CVC problem has a vertex cover of size no greater than K if and only if its corresponding TP-CUT problem with the constraint that the all the foreground/background nodes are connected, has an optimal solution whose energy is less than $(K+1)(|E_h|+1)$. In this way a one-to-one correspondence between the CVC problem and the TP-CUT problem is established.

An example of the reduction is shown in Fig 4.

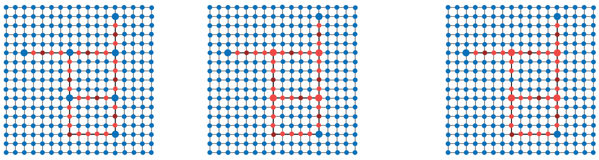
The reduction is constructed as follows. a) A planar graph $\mathcal{H}$ is embedded into an image $\mathcal{G}$ with two terminals (source/sink). b) The weight of the edges for neighboring pixels is set to be zero. The other edges in $\mathcal{G}$ are those edges linking the nodes to the two terminals (source/sink edges). c) The label of nodes on the image that do not lie on the embedded graph $\mathcal{H}$ is "fixed" to be background by setting a sufficiently large ("infinite") weight to each node's sink edge. d) To respect topology constraints, we pick one node for each embedded edge and set the weight of its source edge to be one. We also fix the label of the other nodes on these edges to be foreground by setting an infinite weight for their source edges. e) The weight of the sink edges for the nodes corresponding to vertices in $\mathcal{H}$ is set to be $|E_h| + 1$, where $|E_h|$ is the number of edges of $\mathcal{H}$; With the above construction, a CVC problem has a vertex cover of size no greater than K if and only if its corresponding TP-CUT problem with the constraint that the all the foreground/background nodes are connected, has an optimal solution whose energy is less than $(K + 1)(|E_h| + 1)$.

□



(a) Edge nodes and door nodes (middle) (b) Vertex node (c) Grid node

Figure 3. Weight configuration for (a) edge nodes, (b) vertex node and (c) grid node (c).



(a) Default labeling. (b) A cover with topology error. (c) One solution of TP-CUT .

Figure 4. (Best viewed in color) A default labeling of the embedding graph (a) corresponds to an optimal solution (a minimum cut of 0) of the energy function (3) without considering topology constraints. By considering the topology constraints, *i.e.*, the foreground (red) and the background (blue) should be one connected component, the label of some door nodes and edge nodes should be changed (b)(c). These changes add additional penalties to the energy function (3).

Since the TP-CUT problem is NP-hard, our goal in this paper is to design an efficient algorithm that finds a local optimum. We update the binary partition of the image by solving the above energy function using the standard min-cut/max-flow algorithm while making sure that each update does not violate the topology constraint. Our algorithm is able to handle all the energy functions that can be solved by graph cuts [17] with an additional topology constraint. In the rest of this paper we discuss our algorithm in detail.

5. Design of the Topology Cuts Algorithm

In this section, we discuss the novel aspects of our algorithm. The whole algorithm can be found in Table 1.

5.1. Explicit F/B labeling

The *s/t* partition does not guarantee segmentation with topology constraints, hence we need to partition based on a different attribute; we propose to explicitly add an F/B *label* attribute to each node, which is set in the beginning of segmentation. We specify that 0 is associated to F and 1 is associated to B in the energy function 1. If there is no topology constraint, partition according to the F/B label is identical to the *s/t* partition described in Section 2.3, *i.e.*, all the nodes in tree S are labeled F, and all the nodes in tree T are labeled B, thus the optimal solution of the energy function 1 is reached at the end of the computation of max-flow (by definition F is *associated* to tree S and B is *associated* to tree T). Since we are motivated by topology constraints, our algorithm must ensure that during max-flow computation, each update of the F/B label not only goes to its associated tree, *i.e.*, to achieve lower energy of Equation 1, but also conforms to the topology constraints. The label initialization is discussed in Section 5.2.

Such an explicit label attribute also resolves an inherent ambiguity in the graph cuts algorithm. When the max-flow is reached, it is possible that some nodes are isolated from both the source and sink terminals. As an example, in the image segmentation context, a node can belong either to the foreground or the background without changing the optimal solution (multi-optimal solutions). In this situation, the default assumption in [5] is: if a node does not belong to tree S, then it is assigned to tree T[1], *i.e.*, $T = \mathcal{V} - S$. However, a region of such isolated nodes may be inside the foreground object. By default, these nodes would be labeled background, leading to undesirable holes in the object. With our F/B label attribute, the foreground/background assignment is decoupled from the source/sink tree assignment.

Each node of the tree S and T may have one of two different labels, F or B. Thus the trees S and T are divided into four subtrees: S_F, S_B, T_F and T_B (Figure 5).

5.2. Initialization

Next we need to initialize the F/B label for each node to provide the topological prior information of the target segmentation. There are several ways to assign an initial label for each node, *e.g.*, we may interactively draw some seeds to specify different connected regions and propagate them to give an initial labeling of the whole image. Alternatively, we can integrate the sampling with the segmentation, as Grabcut [23] or graph cuts for level set segmentation [6] did.

As noted in [15], the concept of initialization is generally not used in the standard min-cut/max-flow algorithms, because the label of each node is not known until the min-cut is found. However, because the min-cut can not be found in a single step, any *s/t* cut algorithm must start from one initial state to carry on its computation. In the case of the implementation in [5], each node is initialized by saturating one of its edges to the source or sink, *e.g.*, if its edge to the source is not saturated, then its state is set to be active and it belongs to the S tree; if both edges are saturated or there's no such edge at all, its state is set to be free. In our algorithm, we use the same technique as in [5] to initialize the active sets for each of the four subtrees S_F, S_B, T_F and T_B.

[1]This is based on the publicly available source code implementing [5].

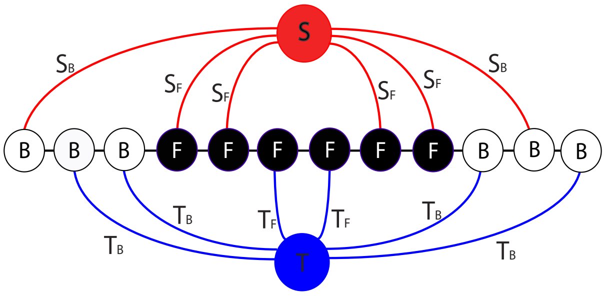


Figure 5. Organizing the nodes by four subtrees.

5.3. Inter/intra-label maximum flows

The computation of augmenting paths results in the change of tree membership of each node. When changes happen, we also need to update the F/B label of nodes. However, the change of tree membership does not necessarily occur along an evolving boundary between the label sets (similar to a level set evolving boundary). This makes it difficult to update the F/B label without violating the topology constraint.

If we were able to ensure that the change of tree memberships during the computation of max-flow, occurs along the boundary between the F/B labels, we would ensure that the label update is in accordance to the topology constraints. To achieve this, we give high priorities to the growing of tree nodes with different labels. That means the augmenting paths between S_F and T_B, S_B and T_F should be searched first. We call this stage *inter-label maximum flow*. After all paths between the above two pairs are saturated, there may still be non-saturated paths between nodes with the same label, namely, S_F and T_F, S_B and T_B. Thus, a second stage– *intra-label maximum flow* is employed to saturate all paths between the two subtrees pairs S_F and T_F, S_B and T_B. The details on the label updates are described in Section 6.

5.4. Organizing the search of augmenting paths using distance maps

In the computation of max-flow, in order to localize the occurrence of the changes of tree memberships along the boundary between two label sets, those active nodes close to the boundary should be handled first in the tree growing stage. Hence, an additional attribute for each node, *DIST*, that records its distance to the boundary, is employed.

Typically, the l^k distance between two 2D points $p = (x_p, y_p)$ and $q = (x_q, y_q)$ is defined as $\|p-q\|_k = (|x_p - x_q|^k + |y_p - y_q|^k)^{\frac{1}{k}}$. Note that in the image grid, only the l^1 distance leads to integer value. The computation of the distance map with L^2 distance requires a complexity of $O(n \log n)$, whereas the computation of L^1 distance needs only $O(n)$ complexity[20]. In our algorithm, we are mostly concerned with those nodes that are closest to the boundary, which often correspond to the boundary. Here we adopt an l^1 distance map, which requires only $O(n)$ computational complexity where n is the number of nodes in the image.

In the level sets approach, the maintenance of a (signed) distance map can be costly [20]. It often requires re-initialization to make sure that the distance map is valid. Because we are only interested in those nodes that have the smallest distance values (thus they are the closest ones to the boundary), the real distance from each pixel to the boundary is not important. Thus, re-initialization is not required. The new distance value for a newly updated node can be computed as the smallest distance value among its neighboring nodes with the new label minus one.

$$DIST(n_i) = \min_{n_j,(n_i,n_j)\in\mathcal{E},Label(n_i)=Label(n_j)} DIST(n_j) - 1$$

Hence we only need to initialize the distance map one time in the beginning. The subsequent updates happen only when a node's label is changed, which requires an $O(1)$ computation for each label update.

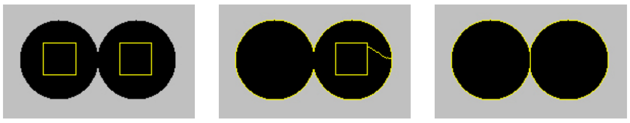
5.5. Controlling the label propagation using the bucket priority queue data structure

At first glance, the priority queue data structure is suitable for keeping track of nodes that are closest to the boundary. However, two problems arise if we use a priority queue. First, it requires a complexity of $O(\log n)$ (n is the number of nodes in the priority queue) to extract a node, which increases the computation cost at the tree growing stage. Second and more importantly, in our topology cuts problem, those nodes that have the smallest distance value usually represent the boundary between the foreground and the background. A node on the boundary grows the tree by recruiting a new child node from its neighbors. Since the newly recruited node now has the smallest distance value, it is the first to be considered in the next stage of growing the trees by using a priority queue. This will lead to an *inhomogeneous propagation* of the boundary (evolution of the boundary from one point on the boundary as in Figure 6 (b)), in contrast to the active contour's *homogeneous propagation* (evolution of every point on the boundary).

To reduce the cost of computation, we adopt the idea of bucket sort [9]. The range of the distance value must be within $[-m-n, m+n]$ for an $m \times n$ image with an l^1 distance map. Thus, we may allocate an array of size $2(m+n)+1$ with each entry recording the nodes with the same distance. We also use a variable to record the current smallest distance. Because the deletion of a node can be efficiently implemented by using the node pointer, the complexity for extracting the next smallest distance node is only $O(1)$.

As for the problem of inhomogeneous propagation, we use an additional pointer to record the currently evolving boundary's distance value, which is actually a level set of the distance map. Furthermore, we restrict this pointer to not point to the entry with the smallest distance until all the nodes in the entry it currently points to have been extracted. This ensures that the boundary will evolve homogeneously (Figure 6 (c)). This makes sure that we obtain a balanced result when multiple regions (foregrounds) belong to the same tree (source or sink).

The above consideration can be efficiently implemented using the *bucket priority queue* (BPQ) data structure as first introduced in [9, 11]. (Figure 7). shows the structure of the layered priority queue. Note that our new data structure can also handle other distance metric, we only need to change each entry to represent an interval instead one distance.



(a) Initialization (b) Without BPQ (c) With BPQ

Figure 6. A synthetic example illustrates the importance of using our BPQ data structure in organizing the search of the maximum flow.

1. Assign an initial label for each node.
2. Set the parameters of the MRF energy function 1.
3. Compute the distance map based on the initial label map.
4. Construct two trees S and T, and eight bucket priority queues:
   Act$\{S_F\}$, Act$\{S_B\}$, Act$\{T_F\}$, Act$\{T_B\}$
   Pas$\{S_F\}$, Pas$\{S_B\}$, Pas$\{T_F\}$, Pas$\{T_B\}$.
5. Initialize four active sets:
   Act$\{S_F\}$, Act$\{S_B\}$, Act$\{T_F\}$, Act$\{T_B\}$,
   by saturating direct edges between each node to s and t.
6. Topology-preserving min-cut/max-flow.
   while Act$\{S_F\}\cup$Act$\{S_B\}\cup$Act$\{T_F\}\cup$Act$\{T_B\}$ is not empty
   6.1 **Inter-label maximum flow**:
      Find all non-saturated paths between the subtrees:
      $S_F \Leftrightarrow T_B$, $S_B \Leftrightarrow T_F$, until all paths are saturated.
   6.2 **Intra-label maximum flow**:
      Act$\{S_F\}\leftarrow$Pas$\{S_F\}$, Act$\{S_B\}\leftarrow$Pas$\{S_B\}$,
      Act$\{T_F\}\leftarrow$Pas$\{T_F\}$, Act$\{T_B\}\leftarrow$Pas$\{T_B\}$.
      Find all non-saturated paths between the subtrees:
      $S_F \Leftrightarrow T_F$, $S_B \Leftrightarrow T_B$, until all paths are saturated.

Table 1. The topology cuts algorithm

6. Implementation Details

The overall guideline in implementing the topology cuts algorithm is to reduce the energy function while maintaining the topology constraint. Observe that if after computing the maximum flow, all nodes with label F only belong to the source tree or be free and all nodes with label B only belong to the sink tree or be free, then the energy function is minimized. Thus, during the maximum flow computation, we need to update the label of each node according to which tree it belongs to (favors).

6.1. Inter-label maximum flow

The goal of inter-label maximum flow is to quickly evolve the boundary between the F-labeled and B-labeled regions under topology constraints. In this stage, all the augmenting paths between the two subtree pairs, (S_F, T_B) and (S_B, T_F), are searched. In each search step, we either find an augmenting path or grow these subtrees by setting each active node's neighbors with non-saturated edges as its new children, if the conditions stated below are met. The distance value and the label of a new child q are updated if q's original label is different from that of its parent.

The conditions for recruiting a new neighbor q by active node p are:

1. If the label of q is the same as that of p, then q is recruited if and only if q is free.
2. If q has a different label and q is a simple point, then q is recruited if either of the following conditions is met:
   (a) q is free, or
   (b) p is associated to the tree that q belongs to (Section 5.1).

The above conditions ensure that the F/B labels are updated with respect to the topological constraints while minimizing the MRF energy function.

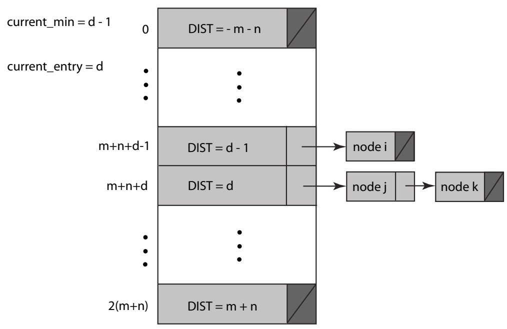


Figure 7. The bucket priority queue data structure.

6.2. Intra-label maximum flow

The goal of the intra-label maximum flow is to saturate all the single-label paths between tree S and tree T, and change the label of any node for which the following conditions are met.

The label of a node p is changed only when (1) p's opposite label is associated to the tree that p belongs to (Section 5.1), and (2) p is a simple point. Note that because we do not know if p's tree membership will change in the subsequent computations, we should not change the label of p until the end of this stage. Thus, we simply record all the nodes that meet the above two conditions as candidate nodes and finalize label changes after the computation of max-flow in this stage. The labels of these candidate nodes that still satisfy the above conditions will be changed and inserted into their corresponding active sets for the inter-label maximum flow stage in the next iteration.

7. Analysis of the Topology Cuts Algorithm

Convergence: Convergence of our algorithm is ensured by 1) all augmenting paths between s and t are guaranteed to be saturated, and 2) the algorithm will stop within two iterations. This guarantees that our algorithm obtains a local optimal solution of the energy function 3.

To verify the first claim, observe that in the first iteration, all augmenting paths between the underlying four subtrees with two different label sets are saturated after finding the inter-label maximum flow. Likewise, between the stages of searching the intra-label maximum flow and changing labels, all augmenting paths with the same labels are also saturated. The only non-saturated paths left are those between the current S and T trees with different label sets. Then at least one of these nodes along such a path should reside on the boundary. According to our intra-label flow algorithm, this node must be recorded and the path will be found and saturated in the next iteration.

For the second claim, note that the remaining non-saturated paths are those crossing two different label sets at the end of the first iteration. According to our inter-label maximum flow implementation, all such paths must be saturated in the next iteration. There does not exist any other non-saturated path after the second iteration and hence the whole algorithm must stop within two iterations.

Note that without considering the topology constraint (updating the labels without considering the simplicity of the nodes), our algorithm is actually another implementation of the min-cut/max-flow algorithm.

Topology preservation. This can be easily verified by looking into our algorithm to see that, before each update of the label, the simple point condition is always checked, *i.e.*, the label of a node is changed only if its change does not affect the global topology of the image.

Time complexity. The time complexity can be shown by comparing the differences between our algorithm and the implementation of graph cuts in [5]. Both algorithms search the augmenting paths by growing two trees from s and t respectively. The main difference here is that we divide the search into two stages. Since our algorithm stops within two iterations, each node is traversed at most four times (one time for each stage). The bucket priority queue data structure ensures that the selection of active nodes needs an $O(1)$ operation. The update of the distance value for a node also needs an $O(1)$ operation. In addition, an initialization of the distance map is computed in the beginning, which requires an $O(n)$ operation where n is the number of nodes (pixels) in the image. In total, our algorithm only adds a constant factor to the complexity of the original algorithm in the worst case. In practice, our algorithm works sufficiently fast since the number of active nodes is significantly reduced in the second iteration.

8. Experimental Results

We apply our topology cuts algorithm to two problems: image segmentation and object tracking[2]. All results were run on a PC equipped with an Intel Pentium M 2.0G Hz processor and 1.5G memory.

8.1. Results for image segmentation

To verify our algorithm, we use the discrete piecewise Mumford-Shah style energy function [3, 10]. The Mumford-Shah model allows us to use a level set style initialization that integrates topology initialization and seed assignment. The parameters of the MRF energy function 1 are defined as follows:

$$\begin{aligned} D(x_p) &= \lambda((u_p - c^B)^2 x_p + (u_p - c^F)^2(1 - x_p)) \\ V_{pq}(x_p, x_q) &= x_p(1 - x_q) + (1 - x_p)x_q \end{aligned} \tag{4}$$

Here u_p denotes the gray value of the image at p, c^F and c^B denote the mean gray value of the pixels with the label F and B respectively (the initial labeling is assigned by users). We use $\lambda = 10$ in our experiments, and solve the energy function in one step instead of iteratively estimating the mean gray value of the foreground and background.

Traditionally, graph cuts algorithms only take account of color and coherence between neighboring pixels. Whereas, solely exploiting the color similarity can not fully guarantee meaningful segmentation results for medical applications. An immediate example is demonstrated in Figure 8. Our algorithm gives a result (Figure 8 (c)) that faithfully conforms to the initialization, which is more meaningful than that of the standard graph cuts algorithm (Figure 8 (b)).

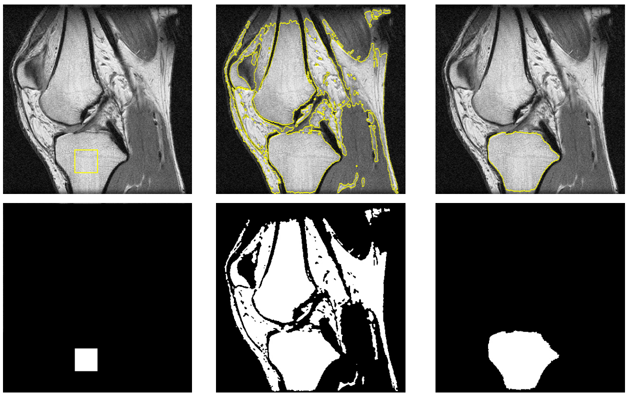
Table 2 shows the comparison between the graph cuts implementation [5] and our topology cuts algorithm[3].

8.2. Results for interactive object cutout

Natural images often contain richly textured parts. Modeling them using the pairwise MRF model is insufficient since it only considers local interactions of the pixels.

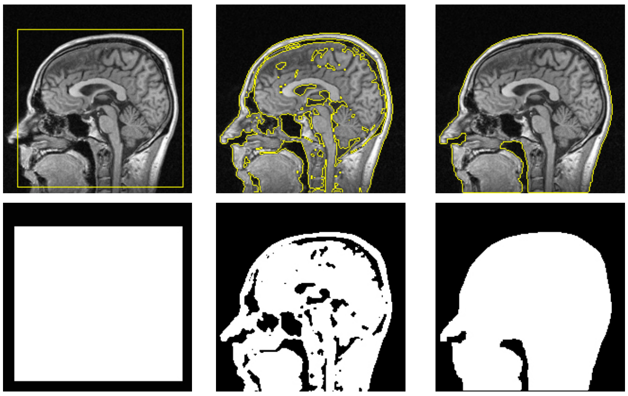
[2]The source code, the executable, more results and comparisons are submitted as supplementary materials.

[3]The examples of Brain and Ventricle are shown in the submitted video.



(a) Initialization (b) Result by graph cuts (c) Result by topology cuts

Figure 8. (Best viewed in color) An example of medical image segmentation illustrates the advantage of our algorithm over graph cuts. Topology-free segmentation is usually not desirable for medical applications. The second row shows the corresponding label maps of the results in the first row.



(a) Initialization (b) Result by graph cuts (c) Result by topology cuts

Figure 9. (Best viewed in color) An example of medical image segmentation illustrates the advantage of our algorithm over graph cuts. Topology-free segmentation is usually not desirable for medical applications. The second row shows the corresponding label maps of the results in the first row.

Table 2. Performance comparison between graph cuts and topology cuts with the same initialization and parameter values.

Image	Size	Graph Cuts	Topology Cuts
Synthetic	250×180	16 ms	16 ms
Brain	200×200	20 ms	31 ms
Ventricle	256×256	16 ms	47 ms
Knee	400×400	63 ms	109 ms

Moreover, its result tends to be sensitive to the choice of parameters, *i.e.*, the weights balancing the data term (the first term of 1) and the smoothness term (the second term of 1) are often difficult to determine. With a large weight for the smoothness term, a large part of the background may be segmented. And with a large weight for the data term, holes and outliers may easily be generated in the segmented object. To account for this, some global properties should be introduced to appropriately model the natural image, such as global topology information which is often available in many applications.

Fig 10 illustrates the result of applying our topology cuts algorithm for interactive object cutout. We use small values for the weight that balances the two terms, *e.g.*, $\gamma = 10$. From Figure 10 (b)(c) we can see that holes and small outliers are generated inside the object by using the standard graph cuts algorithm. Instead, by applying our topology cuts algorithm, we obtain a complete result as shown in Figure 10 (d)(e). Our method might still suffer if outliers are large and conform with the topology prior. However, our algorithm significantly reduces sensitivity to weight selection.

8.3. Results for object tracking

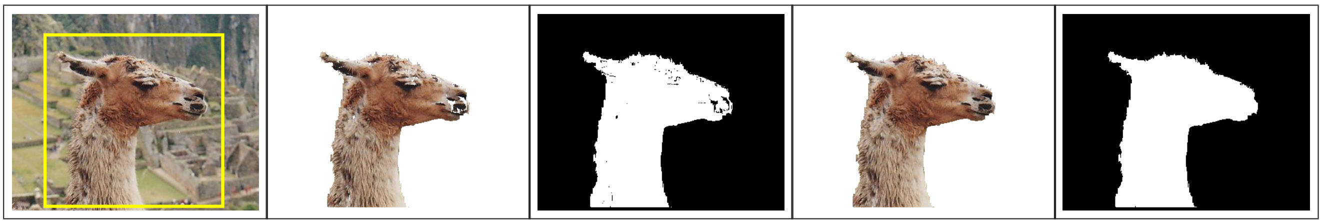
Our topology cuts algorithm can be applied to topology-preserving object tracking. The user assigns the initial contours (or they can be automatically located) containing the topology prior information for the first frame. For all the subsequent frames, the segmentation result from the previous frame is used as the initialization before the topology cuts algorithm is applied, thus the topology information is propagated from the first frame to the other frames. Figure 11 illustrates the results of a simple implementation for a hand tracking case. Topology is correctly preserved as shown in Fig 11 (b)(e)(f), even when the two hands meet [4].

9. Conclusion and Future Work

We proposed a new algorithm for solving a subset of MRF functions that can be addressed by graph cuts while respecting topological constraints. It combines certain advantages of level sets and graph cuts. The idea of boundary evolution is introduced into the graph cuts framework by using the explicit F/B label attribute. Rather than evolving the boundary in a gradient descent manner to update the distance function as level sets do, the boundary evolution of the F/B label set is driven by the computation of max-flow, which is fast and stable. The bucket priority queue data structure ensures that there is no increase in computational complexity compared with the existing graph cuts algorithms.

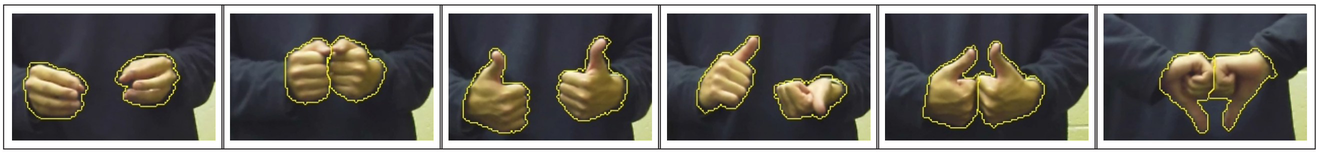
In the near future, we plan to extend the topology cuts algorithm to allow soft constraints. We would also like to apply our new algorithm to other vision problems such as stereo, 3D reconstruction. One promising direction is the incorporation of other prior knowledge into the min-cut/max-flow algorithm, *e.g.*, the curvature of the boundary can be approximately encoded into the distance function.

[4]The tracked video sequence is in the supplemental materials.



(a) Initialization (b) Grabcut result (c) Labels of (b) (d) Topology cuts result (e) Labels of (d)

Figure 10. Example of using topology cuts algorithm for interactive object cutout (without border matting).



(a) Frame 1 (b) Frame 93 (c) Frame 151 (d) Frame 211 (e) Frame 265 (f) Frame 344

Figure 11. (Best viewed in color) Results for topology-preserving hand tracking (also see our video).

With our framework we would be able to design a new cut algorithm that considers the smoothness of the boundary as well.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993. 2, 3

[2] G. Bertrand. Simple points, topological numbers and geodesic neighborhoods in cubic grids. *Pattern Recognition Letters*, 15(10):1003–1011, 1994. 2, 4

[3] Y. Boykov and G. Funka-Lea. Graph cuts and efficient n-d image segmentation. *IJCV*, 70(2):109–131, 2006. 1, 8

[4] Y. Boykov and V. Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *ICCV '03*, pages 26–33, 2003. 1

[5] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 26(9):1124–1137, 2004. 2, 3, 5, 8

[6] Y. Boykov, V. Kolmogorov, D. Cremers, and A. Delong. An integral solution to surface evolution PDEs via geo-cuts. In *ECCV '06*, pages 409–422, 2006. 1, 5

[7] Y. Boykov and O. Veksler. Graph cuts in vision and graphics: Theories and applications. In: N. Paragios, Y. Chen, and O. Faugeras, *The Handbook of Mathematical Models in Computer Vision*, Springer-Verlag. pages 79–96, 2006. 1

[8] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 23(11):1222–1239, 2001. 1

[9] R. Brown. Calendar queues: a fast 0(1) priority queue implementation for the simulation event set problem. *Commun. ACM*, 31(10):1220–1227, 1988. 2, 6, 7

[10] T. Chan and L. Vese. Active contour model without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001. 8

[11] B. V. Cherkassky and A. V. Goldberg. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 19(4):390–410, 1997. 2, 7

[12] D. Freedman and T. Zhang. Interactive graph cut based segmentation with shape priors. In *CVPR '05*, pages 755–762, 2005. 1

[13] S. German and D. German. *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images*. MIT Press, Cambridge, MA, USA, 1988. 1, 2

[14] X. Han, C. Xu, and J. L. Prince. A topology preserving level set method for geometric deformable models. *PAMI*, 25(6):755–768, 2003. 1

[15] O. Juan and Y. Boykov. Active graph cuts. In *CVPR '06*, pages 1023–1029, 2006. 2, 5

[16] V. Kolmogorov and Y. Boykov. What metrics can be approximated by geo-cuts, or global optimization of length/area and flux. In *ICCV '05*, pages 564–571, 2005. 1

[17] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *PAMI*, 26(2):147–159, 2004. 5

[18] M. P. Kumar, P. H. S. Torr, and A. Zisserman. OBJ CUT. In *CVPR '05*, pages 18–25, 2005. 1

[19] H. Lombaert, Y. Sun, L. Grady, and C. Xu. A multilevel banded graph cuts method for fast image segmentation. In *ICCV '05*, volume 1, pages 259–265, 2005. 2

[20] S. Osher and R. Fedkiw, editors. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 1998. 2, 6

[21] S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed : Algorithms based on the Hamilton-Jacobi formulation. *Journal of Computational Physics*, 79(2):12–49, 1988. 1, 2

[22] N. Paragios, Y. Chen, and O. Faugeras, editors. *The Handbook of Mathematical Models in Computer Vision*. Springer-Verlag, 2005. 1

[23] C. Rother, V. Kolmogorov, and A. Blake. "Grabcut": interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3):309–314, 2004. 5

[24] W. Shi and C. Su. The rectilinear steiner arborescence problem is np-complete. In *SODA '00*, pages 780–787, 2000. 4

[25] Y. Shi and W. C. Karl. Real-time tracking using level sets. In *CVPR '05*, pages 34–41, 2005. 4

[26] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001. 3